Workshops on Extremely Large Databases

Tamr: Data Curation at Scale

Information Services Case Study
Tamr gets Smarter with Each Engagement
A Need to Replace Manual Curation
Massive Improvements with Tamr
Tame Your Curation Challenge Today

Tamr: Technical White Paper

About Data Curation

Figure 1: Tamr Architecture

Machine Driven

Figure 2. The Tamr Machine Learning Information Stack

Data
Features
Evidence
Local Resolution
Global Resolution

Human Guided

Figure 3. The Tamr Expert Sourcing Information Stack

Questions
Judgments
Recommendations
Decisions

Conclusion

Data Curation at Scale: The Data Tamer System

Abstract

1. Introduction

2. Example Applications

2.1 A Web Aggregator

2.2 A Biology Application
2.3 A Health Services Application

3. Data Tamer Semantic Model
   3.1 Human Roles
   3.2 Sites and Schemas
   3.3 Other Information
   3.4 Management Console and Data Tamer Actions
   3.5 Training Data
   3.6 Data Source Update

4. Data Tamer Components
   Figure 1: The Data Tamer Architecture
   4.1 Schema Integration
      4.1.1 Suggesting Attribute Mappings
   4.2 Entity Consolidation
      4.2.1 Bootstrapping the Training Process
      4.2.2 Categorization of Records
      4.2.3 Learning Deduplication Rules
      4.2.4 Similarity Join
      4.2.5 Record Clustering and Consolidation
   4.3 Human Interface
      4.3.1 Manual Mode
      4.3.2 Crowd Sourcing Mode
   4.4 Visualization Component

5. Experimental Validation
   Figure 2: Quality results of entity consolidation for the web aggregator data
   Figure 3: Quality results of entity consolidation for Verisk data

6. Future Enhancements

7. Conclusion

8. References
   [1]
   [2]
   [3]
   [4]
   [5]
   [6]
   [7]
   [8]
   [9]
   [10]
Appendix A. Demo Proposal

Slides

Slide 1 The State of the Art in Supporting "Big Data"
Slide 2 What is "Big Data"
Slide 3 Too Much Data-The Data Warehouse World
Slide 4 Too Much Data-The Hadoop/Hive World
Slide 5 Too Much Data-The Data Scientist World
Slide 6 Too Fast 1
Slide 7 Too Fast 2
Slide 8 Too Many Places 1
Slide 9 Too Many Places 2
Slide 10 DBMS Security
Slide 11 Encryption
Slide 12 Leaks
Slide 13 However

Slides

Slide 1 Big Data Means at Least Three Different Things....
Slide 2 The Meaning of Big Data - 3 V's
Slide 3 Big Volume - Little Analytics
Slide 4 In My Opinion....
Slide 5 Big Data - Big Analytics
Slide 6 Big Analytics on Array Data – An Accessible Example
Slide 7 Now Make It Interesting ...
Slide 8 Array Answer
Slide 9 DBMS Requirements
Slide 10 These Requirements Arise in Many Other Domains
Slide 11 In My Opinion....
Slide 12 Solution Options R, SAS, MATLAB, et. al.
Slide 13 Solution Options RDBMS alone
Slide 14 Solution Options R + RDBMS
Spotfire Dashboard
Research Notes

2012: Report from the 5th Workshop on Extremely Large Databases

**ABSTRACT**

1 EXECUTIVE SUMMARY
2 ABOUT THE WORKSHOP
   2.1 Participation
   2.2 Structure
3 NEW COMMUNITIES: HEALTH CARE AND GENOMICS
   Fragmented, small-scale approach to data
   Problems driven by technological advances
   Future directions
4 FROM SPREADSHEETS TO LARGE SCALE ANALYSIS
5 STATISTICS AT SCALE
6 MACHINE LEARNING
7 OTHER TOPICS
8 NEXT STEPS
9 ACKNOWLEDGEMENTS
10 GLOSSARY

2010: Report from the 4th Workshop on Extremely Large Databases

ABSTRACT

1 EXECUTIVE SUMMARY

2 ABOUT THE WORKSHOP
   2.1 Participation
   2.2 Structure

3 USER COMMUNITIES' PERSPECTIVES
   3.1 Oil / Gas
   3.2 Finance
   3.3 Medical / Bioinformatics

4 SCIENCE BENCHMARK

5 APPROACHES TO BIG DATA STATISTICAL ANALYTICS

6 EMERGING HARDWARE TECHNOLOGIES

7 WRITABLE EXTREME SCALE DATABASES

8 NEXT STEPS

ACKNOWLEDGMENTS

GLOSSARY

FOOTNOTES

1
2
3
4
5
6
7
8
9

2009: Report from the 3rd Workshop on Extremely Large Databases

ABSTRACT

1 EXECUTIVE SUMMARY

2 ABOUT THE WORKSHOP
   2.1 Participation
   2.2 Structure

3 DATA USER COMMUNITY PERSPECTIVE
   3.1 Specific user communities
      3.1.1 Radio astronomy
      3.1.2 Geoscience
3.1.3 Biology
3.1.4 HEP/LHC
3.1.5 Nokia

3.2 Data distribution architecture
   3.2.1 Distribution models
   3.2.2 Problems
   3.2.3 Practice

3.3 Data formats and models
3.4 Data integration
3.5 Data calibration and metadata
3.6 Data consistency and quality
3.7 Data preservation
3.8 Custom software
3.9 Large-scale process flow
3.10 80/20 rule
3.11 Importance of raw data
3.12 Data ownership and usage
3.13 Funding
3.14 Inertia
3.15 Other notes
   3.15.1 Imbalanced systems
   3.15.2 Cloud computing
   3.15.3 Self-management
   3.15.4 Append-only
   3.15.5 Green computing

4 RDBMS VS. MAP/REDUCE
   4.1 Key differences
      4.1.1 Procedural steps vs. monolithic query
      4.1.2 Checkpointing vs. performance
      4.1.3 Flexibility and unstructured data support
      4.1.4 Cost
   4.2 Convergence

5 SOLUTION PROVIDERS
   5.1 MonetDB
   5.2 Cloudera
   5.3 Teradata
   5.4 Greenplum
   5.5 Astro-WISE
Story

Science Database Pioneers: Michael Stonebraker and Kirk Borne

The Data Science Journal chronicles the history of the Extremely Large Databases and compliments the recent NAS report on Frontiers in Massive Data Analysis. Michael Stonebracker and Kirk Borne were major forces in the history of Extremely Large Databases as chronicled below in six journal articles.

At the recent Whitehouse - MIT Big Data Privacy Workshop, Mike Stonebraker, Adjunct Professor, MIT CSAIL, presented the “State of the Art of Big Data Technology”, Watch video (8 minutes) Download PowerPoint slides (PPT).

He said: "Where I do see a problem, an Achilles heal, it is going to be this: if your data is coming at you from too many places and formats, there is a mature technology that has been used for 20 years in getting data into data warehouses that scales to 20 or 30, or I'll give you maybe 50 data sources. But if you want to integrate a lot of data sources think about Data.gov, it is a zillion data sets, each relatively dirty and not integrated with anything else, so if you want to integrate Data.gov into a single, coherent data system you have a big problem.. So how do you integrate 1000's of data sources? Lots of people want to do this."

Interestingly, he alludes to other technologies that are not mature enough to be mentioned. Maybe that is graph databases.

The 2008: Report from the SciDB Workshop concluded:
A mini-workshop with representatives from the data-driven science and database research communities was organized in response to suggestions at the first XLDB Workshop. The goal was to develop common requirements and primitives for a next-generation database management system that scientists would use, including those from high-energy physics, astronomy, biology, geoscience and fusion, in order to stimulate research and advance technology. These requirements were thought by the database researchers to be novel and unlikely to be fully met by current commercial vendors. The two groups accordingly decided to explore building a new open source DBMS. This paper is the final report of the discussions and activities at the workshop.

Some highlights extracted from the 5 Workshops on Extremely Large Databases were:

On the other hand, other sciences are poised to adopt large scientific databases in the near future. Astronomy, in particular the LSST project, perhaps presents the most significant immediate opportunity. While other fields are rapidly approaching similar scales, the requirements and needs of LSST appear to be a superset of those of other scientific communities.

The “reference cases” were particularly highly demanded; participants envisioned an in-depth examination of at least two concrete examples of built systems, one where data is federated and one where data is kept in a single instance.

The relational model is still relevant for organizing these extremely large databases, although industry is stretching it and science is struggling to fit its complex data structures into it.

The longevity of large scientific projects, typically measured in decades, forces scientists to introduce extra layers in order to isolate different components and ease often unavoidable migrations, adding to system complexity.

So Michael Stonebraker continues to work on SciDB Systems and Kirk Borne continue to work on LSST. More specifically Michael Stonbraker is working on Paradigm4 and Tamr and Kirk Borne is blogging and consulting for MapR (also see Glossary for letter Y).

The federal Big Data Working Group is already collaborating with Kirk Borne and so we need to collaborate with Michael Stonebraker as follows:

Michael, I recently rediscovered your 2012 NIST BIG DATA Workshop Slides you sent me, and that you and Kirk Borne, who I work with, were pioneers in the CODATA Extremely Large Data Base Workshops:

Would you be able to present (remotely) to our Federal Big Data Working Group Meetup sometime this summer?

I forgot to mention that we would like to partner/collaborate on Data Tamer, if you are still doing that (see slides 32 and 34 below).

Brand, Collaboration: there is now a Data Tamer company. Depending on what you are interested in, we could discuss who you wanted to work with -- MIT or the company.
Michael, So I have looked at SCIDB.org, and it takes me to Paradigm4, the company that develops, supports and builds SciDB: and SciDB Community Forum where one can download the open-source Community Edition of SciDB and interact with other SciDB users and developers:

I also look at Wikipedia for: SciDB, Michael Stonebraker, and your interview and finally Tamr, and would recommend the following:

Our 200+ members are interested in both academic (NSF, NIH, etc. funding) and consulting activities (Federal Government and Private Industry) which suggests that you could provide an overview and context for all the above and then some specific examples on say June 30th or July 7th, 6:30-9 p.m. We would send you a WebEx email unless you are able to present in person.

Maybe he or someone he can suggest can also tell us something about the MIT Big Data Initiative: bigdata@CAIL and the new Intel Science and Technology Center for Big Data. And he did! See our June 20th Federal Big Data Working Group Meetup.

The Tamr Web Site says:

Tamr, based in Cambridge, Massachusetts, was founded in 2013 by database industry veterans Andy Palmer, Mike Stonebraker and Ihab Ilyas with George Beskales, Daniel Bruckner and Alex Pagan.

Following the success of initial research at MIT CSAIL, the Tamr team began building a commercial-grade solution designed to tackle the challenge of connecting and enriching diverse data at scale, quickly and cost effectively.

Launched in the spring of 2014 and backed by a series of investors, such as Google Ventures and New Enterprise Associates, Tamr is deployed in production at a variety organizations, including information services providers, pharmaceutical firms and retailers.

Their Tamr: Data Curation at Scale, Tamr: Technical White Paper, and Data Curation at Scale: The Data Tamer System papers are found below.

MORE IN PROGRESS

**Tamr: Data Curation at Scale**


**Information Services Case Study**

“Tamr proved that fast and accurate data integration results in tremendous benefits. By combining the system’s machine learning with the knowledge of our data experts, we can dramatically improve the quality of our services.” — VP of Data Services
Information Services Leader Uses Tamr to Automate Data Curation, Slashes Manual Reviews by 90% and Cuts Process Time by Months

A multinational media and information company faced challenges maintaining critical, accurate data. It had outgrown its manual curation processes and looked to Tamr to provide a better solution. Using Tamr, one project, estimated to take six months, was completed in only two weeks, requiring just forty man hours of manual review time—a 12x improvement over the manual process. The number of records requiring manual review shrunk from 30% to 5%, and the number of identified matches across data sources increased by 80%—all while clearing the company’s 95% precision benchmark. The disambiguation rate—the rate of resolving conflicts—rose from 70% to 95%. Furthermore, the knowledge Tamr gleaned from its machine learning activities means that future data integration will take even less time per source.

<table>
<thead>
<tr>
<th></th>
<th>Existing process (estimated)</th>
<th>Tamr Results (actual)</th>
<th>Benefits of using Tamr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Time</td>
<td>8 Months</td>
<td>2 Weeks</td>
<td>92% Reduction in time required for the project</td>
</tr>
<tr>
<td>Automated Matches</td>
<td>219,000</td>
<td>375,000</td>
<td>79% Increase in matches</td>
</tr>
<tr>
<td>Disambiguation Rate</td>
<td>70%</td>
<td>89%</td>
<td>35% Increase in resolution of matches or mismatches</td>
</tr>
<tr>
<td>Records Requiring Review</td>
<td>30%</td>
<td>6%</td>
<td>96% Reduction in time required for the project</td>
</tr>
<tr>
<td>Manual Review Time</td>
<td>&gt;60 Days</td>
<td>6 Days</td>
<td>&gt;90% Reduction in time to review</td>
</tr>
</tbody>
</table>

Note: Based upon 5.4 million records analyzed

Tamr gets Smarter with Each Engagement

Tamr has developed an intuitive user interface that allows data analysts to easily send curation tasks to users who know and people who produce the data—and to build that step into a company’s workflow. As machine learning algorithms understand more about users and data environments, organizations spend less time preparing data—even as they scale up the number of sources to hundreds or thousands—and more time competing on analytics.

A Need to Replace Manual Curation

An information services company’s reputation, brand and market position depend on the breadth and quality of its data. With a huge customer base—media organizations around the world, 80% of Fortune 500 companies and over 400,000 end users—the company’s data challenges limited the efficiency of its business operations and restricted its ability to capture additional market share.

As the number of available data sources grew from tens to hundreds to thousands, several issues became increasingly clear:

- Manual process limited scalability.
  - Integrating just a few data sources required extensive manual efforts and often relied on people conducting manual curation—essentially relying on spreadsheets to try and solve the problem. These slow, manual processes also left many crucial relationships between datum unmapped.
• Curation did not include all data experts and owners.
  ◦ The people most qualified to understand and make authoritative decisions about data were spread across the company, and the curation processes did not have a way to consistently involve them and leverage their knowledge of the data.

• Low-quality and the slow pace of curation impacted customers.
  ◦ Using its existing curation approaches, the amount and the speed of incoming data made it hard for the company to meet the contractually-obligated service level agreements of its customers. Manual curation had become a governor on growth.

Massive Improvements with Tamr

Company executives agreed that their business goals were not achievable without a new approach to integrating and curating data sources. Understanding that the ability to quickly integrate new sources at incremental or sub-linear cost could create tremendous and sustainable competitive advantage, they looked to Tamr to help them with two short-term goals: improve the quality of their data curation and achieve radically better scalability.

The company wanted to focus on integrating three of its core data sources—factual data on millions of organizations with more than 5.4 million records. Previous in-house curation efforts, relying on a handful of data analysts, found that 30%-60% of entities required manual review. The company estimated that if the project was completed in the existing, manual manner, it would require two months of man-hours to fully ameliorate the sources. Additionally, it was thought that the old process would identify 95% of duplicate matches (precision) and 95% of suggested matches that were, in fact, different (recall). Overall, the best guess for completing the activity with the manual process was six months.

Tamr kicked off the project by converting the company’s XML files to CSVs. Next, Tamr ingested the three sources to de-duplicate the records and find suggested matches, with a goal of achieving high accuracy rates while reducing the number of records requiring review. In order to scale the effort and improve accuracy, Tamr applied machine learning algorithms to a small training set of data.

Completed in two weeks, the project resulted in better matching results and dramatically less human intervention. The company, impressed with the significant increase in results and substantial decrease in required man-hours, is expanding its use of Tamr, integrating even more data sources. As the new, Tamr-driven curation processes expands in the organization, the benefits increase as the system continually learns and improves.

Tame Your Curation Challenge Today

To learn how Tamr can dramatically lower the cost, improve the quality and boost the speed of your data integration activities, call 617-413-6551.

Tamr: Technical White Paper

Tamr lets organizations connect and enrich all of their data sources, both internal and from partners and third parties. It overcomes the traditional problems of manual data curation and allows for continuous, cost-effective and timely connectivity of hundreds and thousands of sources.

About Data Curation

Data curation is an end-to-end process consisting of:

- Identifying data sets of interest (whether from inside the enterprise or external),
- Exploring the data (to form an initial understanding),
- Cleaning the incoming data (for example, 99999 is not a valid zip code),
- Transforming the data (for example, to remove phone number formatting),
- Integrating it with other data of interest (into a composite whole), and
- Deduplicating the resulting composite.

There are four characteristics required of a data curation system that aims to connect all of the myriad data that exists throughout the enterprise. These are:

1. **Automation.** The effort involved in connecting hundreds or thousands of disparate data sets precludes any solution built around human effort. A system that works at this scale must automate most decision making, using human input only when absolutely necessary.

2. **Direct engagement of data experts.** When human input inevitably does become necessary, the people who are experts on the data must be directly engaged in making decisions, using a non-programmer interface that matches their skills. This is the only practical way to get the expert input required to make timely progress.

3. **Accommodate independently constructed data.** Though some enterprise data is carefully structured and conformed to standardized dimensions, most enterprise data is constructed partially or wholly independently of enterprise standards. The only practical way to connect this large volume of independent data is to accommodate the data as-is, rather than first clean the data.

4. **Continuous, incremental operation.** New data is constantly being produced by and brought into the enterprise. This data contains an ever-changing variety of semantics and structure, so there is no point at which the task of describing this data is finished. To achieve the goal of comprehensive connection, data must be connected continuously and incrementally, as it arrives.

Figure 1: Tamr Architecture
Tamr’s architecture (Figure 1) delivers each of these characteristics. The core of the Tamr solution is the pairing of a flexible, automated workflow, which automatically performs most of the curation effort, with an expert involvement module, which engages the line-of-business people best able to provide necessary input. In the following sections, we describe how the Tamr system accomplishes this crucial blend of being machine driven, yet human guided.

Machine Driven

The driving force behind the Tamr system is an automated data curation workflow. Within this context, decisions are made by a self-contained, modular machine learning system that is specifically tailored to data integration and curation. This system starts with data, then, through a series of refinements, ultimately produces global resolution -- either at the attribute level or at the record level, depending on the context in which it is run. The stages of refinement are outlined in Figure 2 and explained in more detail in the following sections.

Figure 2. The Tamr Machine Learning Information Stack
Data

A system can only provide assistance with data it can actually accommodate, so it is critical that the Tamr system be built on a flexible repository for sampled data and metadata. This repository is able to accommodate arbitrary content, in any shape. This data store needs to accommodate a wide variety of independently constructed data, as well as schema-less data, dirty or missing data, data with hierarchical or irregular structure, poorly formatted data, and data with a high degree of redundancy. This flexible data store allows the very first stages of curation to occur within the system, rather than pushing them out to other data preparation systems. From this potentially very messy data store, the process of machine-driven curation can begin.

Features

In the domain of Information Retrieval, “Feature Extraction” refers to the process of identifying possibly obscured features in data, and separating them from other data, thereby making them available to downstream processing. Tamr implements an automated feature extraction system to enhance both the precision and recall of our machine learning algorithms. Tamr’s feature extraction system is modular and extensible, allowing different features to be used in different contexts. Our schema mapping algorithms take advantage of metadata features such as attribute name, data type, and constraints; as well as data features such as statistical profiles of values or value length, or TF*IDF-weighted tokenized terms. Our record deduplication algorithms take advantage of such data features as unweighted or weighted attribute values, TF*IDF-weighted tokenized terms, or geographic location. The modular structure of the system makes it straightforward to extend with additional features.

For those cases where the automatic feature extraction system does not have an existing method of identifying and extracting a feature, Tamr also supports manual transformation, allowing users to create new attributes that expose information that would otherwise be difficult to access. Transformations include things such as name or address standardization, extracting keywords from descriptive text, removing flag values from measurements, and parsing separate values out of composite fields. Tamr’s transformation system is built to work with the flexible data repository, and retains detailed provenance information to ensure that values can be traced back to their source.

Evidence

Comparing two data objects starts with comparing their features; this generates evidence supporting a connection between the two. Different features require different comparisons, so this subsystem is also modular and extensible. For example, a Z-test is a good way to compare statistical profiles of values, whereas Jaccard or cosine similarity are good ways to compare tokenized values, and geographic distance is good for comparing locations. Tamr’s collection of comparators evolves along with the collection of features.

Traditional analysis methods use blocking to partition data into discrete regions in order to reduce the number of comparisons that must be performed. We have found that traditional blocking methods don’t work well, both because their expressiveness is insufficient to find good blocking criteria on large, heterogeneous data, and because their decoupling from machine learning training prevents them from adapting to an ever-changing data landscape. Instead, we have developed a closely-coupled, two-stage machine learning system, where the first stage provides an automatic, adaptive alternative to blocking that allows comparisons to be performed on-demand.
Local Resolution

Within a given business domain, the Tamr system learns how to accumulate all the evidence supporting connections between data objects into local resolution decisions - decisions that consider just the objects being compared. The typical way to accomplish this task is to build a classifier using training data specific to the domain. Tamr uses a single set of training data to build a two-stage classifier. The first stage is designed to generate candidate connections, delivering the best possible recall while aggressively eliminating comparisons that are guaranteed to be unnecessary. The second stage is a more traditional classifier aimed at achieving the desired precision and recall; this second stage only needs to consider the candidates retrieved by the first stage. Each stage uses techniques such as kernel density estimation to handle different data distributions, discretization to identify uniform data regions, and prefix filtering to prune the search space.

There are many techniques for training classifiers, spanning unsupervised, supervised and active learning. We use a combination of unsupervised and active learning techniques to rapidly converge on a very high-quality classification model without requiring a technician to oversee the training process. The training process takes advantage of such information as externally-defined rules, bulk training data, and external knowledge of attribute weights, but does not require them. To ensure that a robust classifier is developed, the training process must ensure that the training data represents a complete and unbiased sample from the overall population; therefore, even if bulk training data is provided, the active learning process will generate training questions to be answered by humans. These questions are generated using stratified sampling to ensure good coverage of data variety while minimizing human effort, and our expert sourcing subsystem is used to engage data experts directly in answering these questions. The Tamr system continually monitors the quality of the classifier, and will perform additional training to re-tune it when necessary.

Global Resolution

Local resolution decisions may contain conflicts and uncertainty, so further refinement is necessary to turn it into global resolution of objects and entities. Tamr is able to automatically resolve many types of conflict and uncertainty using clustering analysis. We have found that a greedy clustering algorithm based on weighted network correlation is both extremely efficient and extremely effective in forming very high quality clusters. We have developed an incremental global entity resolution methodology, such that changes in local resolution can be accommodated with minimal re-calculation.

No automated machine learning system is 100% accurate, so there must be a way to incorporate expert review into global resolution decisions. The Tamr system engages humans both to address the areas where the automated system can not reach a high-confidence decision, and to spot-check high-confidence results to ensure that the final result meets the desired quality objectives. Our expert sourcing subsystem is applied in both cases to engage multiple data experts directly in each of these tasks. This coupling of automated processing with expert input provides a practical way to to achieve high-quality, global resolution at scale.
**Human Guided**

The machine-driven nature of the Tamr system allows it to make constant progress, but human guidance is required to ensure that it is making progress in the right direction. The core of Tamr’s method for engaging humans is an Expert Sourcing system that enables the automated system to determine when and how to engage human experts. This system starts by generating questions from a specific context, then reaches out to multiple human experts to make judgments about those questions, aggregating those judgments into recommendations, which can be used by a human or the automated system to make decisions. These stages are outlined in Figure 3, and explained in more detail in the following sections.

**Figure 3. The Tamr Expert Sourcing Information Stack**

```
Fig 3: The Tamr Expert Sourcing Information Stack

   Questions
   | Judgments
   | Recommendations
   | Decisions
```

**Questions**

Engaging human experts starts with asking the right questions. For each of the different contexts in which Tamr may want to engage human experts, it employs a question generator to determine the best question to ask. The goodness of a question depends on two things: how well an expert will be able to answer it, and how much benefit the system will get out of the answer. Since Tamr seeks to engage data experts, the questions must be about data, not about data processing. We have found that an effective structure for questions is to present one or several possible decisions - e.g. whether two objects should be connected, or whether a transformed view of the data is correct - and ask which, if any, is correct. The questions presented to experts must show that the system respects their time and effort; to achieve this, the generators use the results of the automated system to ensure that the decisions around which the questions are composed are likely but non-obvious, or make it clear that the expert is being asked to verify an automated decision. To aid the experts in addressing questions, they are composed with rich contextual information to minimize the need to consult other systems.

To minimize the number of questions asked, the question generators search for a minimal set of questions that will maximize the impact on the automated system, for example, using stratified sampling to ensure good coverage of data variety. Finally, since the questions are being asked of humans, they must be presented in a form that is carefully constructed to counteract known sources of bias, such as the serial position effect or confirmation bias.
Judgments

Once a question or set of questions has been generated, it needs to be sent to one or more experts so that they can render judgment on it. There are many varieties of expert that can effectively address Tamr’s questions, such as source owners, data engineers, data stewards, data curators, data architects and data scientists. These are all people who know the data and the business context in which the it is used. Given that there may be many experts able to answer a given question, the system performs intelligent load-leveling to balance the overall workload fairly across the entire pool of experts. This load leveling takes into account each expert's current and recent workload, recent responsiveness, and efficiency when answering a series of closely related questions.

Human experts are not perfect, and the Tamr system tracks this using levels of expertise - a model of the accuracy of an expert’s judgments on questions sent out by the system. Furthermore, a given expert can not judge all questions with the same accuracy; for this, the Tamr system uses knowledge domains. Within a single knowledge domain, an expert has a single level of expertise, but can have different levels of expertise in other domains. This allows expert responses to questions to contain uncertainty; for a given question, the Tamr system weighs a judgment by the expert’s level of expertise in that question’s knowledge domain.

Recognizing that an expert’s level of expertise in a single domain will vary over time, the Tamr system uses adaptive expertise - expertise that changes over time in response to observed performance. To continually assess an expert’s level of expertise, the Tamr system will intersperse assessment questions with known answers into an expert’s workload. We can then use windowing and Bayesian inference to model that expert's changing level of expertise over time.

Recommendations

If judgment rendered by a single expert does not provide sufficient confidence for the automated system to take action on it, the system can present the question to multiple experts, and use their potentially conflicting judgments to build a recommendation with a higher aggregate confidence. This recommendation is for a decision - e.g. whether data objects should be linked, or whether a transformed view of data is correct.

To determine how many experts to send the question to, the system models the expected confidence of their combined responses. The system can select groups optimized for different metrics, such as the cost to achieve the desired confidence, or the number of experts consulted. This, in turn, is used as input to the load-leveling mechanism. The experts' responses are integrated into an overall recommendation using Bayesian inference, which also produces a confidence score.

Decisions

A recommendation coming from the expert sourcing system can be acted upon either by a human data steward or by the automated curation process. Acting on a recommendation is making a decision - examples of decisions are that data should or should not be linked, or that a transformation should or should not be made, or that two objects do or do not have a particular relationship. All human decisions feed forward into the automated process, to ensure that it continues in the right direction. Expert feedback gathered as part of a training process has a natural place in the automated
process, but that gathered outside the context of training needs to be carefully weighed before feeding into the automated process, as it is part of a biased sample - it is biased towards those areas where existing models are uncertain. Data steward decisions can also be used as “gold standard” responses to feed into the system that manages adaptive expertise.

In all cases, the Tamr system retains a complete audit trail, allowing the provenance of every decision to be examined in detail. Decisions can be reviewed, commented upon, revised, and retracted, and these follow-on decisions can feed forward in the same way as the original.

Conclusion

The machine driven but human guided nature of the Tamr system delivers a high degree of automation, while directly engaging data experts. It readily accommodates independently constructed data into a continuous, incremental curation process. This closely-coupled design enables practical, end-to-end curation at the scale of of hundreds to thousands of disparate data sets. It is the only system designed from the bottom up to scale across the entire enterprise.

Data Curation at Scale: The Data Tamer System


Michael Stonebraker MIT stonebraker@csail.mit.edu
Daniel Bruckner UC Berkeley bruckner@cs.berkeley.edu
Ihab F. Ilyas QCRI ikaldas@qf.org.qa
George Beskales QCRI gbeskales@qf.org.qa
Mitch Cherniack Brandeis University mfc@brandeis.edu
Stan Zdonik Brown University sbz@cs.brown.edu
Alexander Pagan MIT apagan@csail.mit.edu
Shan Xu Verisk Analytics sxu@veriskhealth.com

Abstract

Data curation is the act of discovering a data source(s) of interest, cleaning and transforming the new data, semantically integrating it with other local data sources, and deduplicating the resulting composite. There has been much research on the various components of curation (especially data integration and deduplication). However, there has been little work on collecting all of the curation components into an integrated end-to-end system.

In addition, most of the previous work will not scale to the sizes of problems that we are finding in the field. For example, one web aggregator requires the curation of 80,000 URLs and a second biotech company has the problem of curating 8000 spreadsheets. At this scale, data curation cannot be a manual (human) effort, but must entail machine learning approaches with a human assist only when necessary.
This paper describes Data Tamer, an end-to-end curation system we have built at M.I.T., Brandeis, and Qatar Computing Research Institute (QCRI). It expects as input a sequence of data sources to add to a composite being constructed over time. A new source is subjected to machine learning algorithms to perform attribute identification, grouping of attributes into tables, transformation of incoming data and deduplication. When necessary, a human can be asked for guidance. Also, Data Tamer includes a data visualization component so a human can examine a data source at will and specify manual transformations.

We have run Data Tamer on three real world enterprise curation problems, and it has been shown to lower curation cost by about 90%, relative to the currently deployed production software.

This article is published under a Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), which permits distribution and reproduction in any medium as well allowing derivative works, provided that you attribute the original work to the author(s) and CIDR 2013. 6th Biennial Conference on Innovative Data Systems Research (CIDR ’13) January 6-9, 2013, Asilomar, California, USA.

1. Introduction

There has been considerable work on data integration, especially in Extract, Transform and Load (ETL) systems [4, 5], data federators [2, 3], data cleaning [12, 18], schema integration [10, 16] and entity deduplication [9, 11]. However, there are four characteristics, typically absent in current approaches that we believe future system will require. These are:

Scalability through automation. The size of the integration problems we are encountering precludes a human-centric solution. Next generation systems will have to move to automated algorithms with human help only when necessary. In addition, advances in machine learning and the application of statistical techniques can be used to make many of the easier decisions automatically.

Data cleaning. Enterprise data sources are inevitably quite dirty. Attribute data may be incorrect, inaccurate or missing. Again, the scale of future problems requires an automated solution with human help only when necessary.

Non-programmer orientation. Current Extract, Transform and Load (ETL) systems have scripting languages that are appropriate for professional programmers. The scale of next generation problems requires that less skilled employees be able to perform integration tasks.

Incremental. New data sources must be integrated incrementally as they are uncovered. There is never a notion of the integration task being finished.

These four issues should be addressed in a single coherent architecture, which we call a data curation system. The purpose of this paper is to describe Data Tamer, a data curation system motivated by these requirements. In Section 2, we begin with a brief description of three example data curation problems, which Data Tamer is designed to solve. Then, Section 3 continues with the semantic model that Data Tamer implements, followed in Section 4 by a description of the main components of the system. Lastly, Section 5 presents a collection of experiments on real world curation problems. We conclude in Section 6 with a collection of possible future enhancements.

2. Example Applications
2.1 A Web Aggregator

This aggregator integrates about 80,000 URLs, collecting information on “things to do” and events. Events include lectures, concerts, and live music at bars. “Things to do” means hiking trails, balloon rides, snowmobile rentals, etc. A category hierarchy of concepts is used to organize this space, and all information is placed somewhere in this hierarchy.

The decision to collect the data from a specific URL is made with a combination of manual and automatic processes that are not relevant to this paper. Once decided, an offshore “wrapper foundry” writes code to extract data from the URL. For each entity that it finds at a given URL, the wrapper outputs a collection of key-value pairs, e.g., (key1-name, value-1), (key2-name, value-2), …, (keyK-name, value-K). Unfortunately, the source data is rarely web tables, but is usually in pull-down menus, text fields and the like. Hence, site wrappers are non-trivial.

This aggregator needs to federate these 80,000 data sources into a semantically cohesive collection of facts. The 80,000 data sources contain approximately 13M local records with about 200K local attribute names. In addition, local information may be inconsistent, overlapping and sometimes incorrect. Hence, this aggregator faces a difficult data curation problem, which they solve using a collection of ad-hoc and human-centric techniques. The purpose of Data Tamer is to do a better job on this sort of problem than the currently deployed solution at substantially lower cost.

2.2 A Biology Application

A major drug company has 8,000 bench biologists and chemists doing lab experiments. Each maintains a “lab notebook”, typically as a spreadsheet where he records his data and observations. Most of the scientists are using different techniques and collecting experiment-specific data such as concentration and density. However, some of these 8000 scientists might well be studying the same reaction or be starting with the same molecule. There is great value to integrating these 8000 sources, so scientists can get a better picture of the reactions they are studying.

Unfortunately, there are no standards for attribute names, no standards for measurement units, and not even a standard for the language for text (English, German, etc…)

These 8000 spreadsheets contain about 1M rows, with a total of 100K attribute names. Again, the scale of the problem makes current data integration tools prohibitively expensive. The goal of Data Tamer is to do a better job at lower cost than current software.

2.3 A Health Services Application

Verisk Health does data integration for the claims records for a collection of 300 insurance carriers. They have manually constructed a global schema for these sources, and are looking to replace their manual processes with more automated ones. In addition, their integrated database contains 20M records, and they wish to consolidate claims data by medical provider. In other words, they want to aggregate all of the claims records, group by provider. In effect, they want to dedup their database, using a subset of the fields. They are currently doing this task with a lot of manual intervention, and looking for a lower cost, more automated solution. The purpose of Data Tamer is better results at lower cost than their current solution.

3. Data Tamer Semantic Model

http://semanticommunity.info/Data_Science/Big_Data_Science_for_CODATA/Workshops_on_Extremely_Large_Databases
Updated: Sat, 19 Sep 2015 00:38:45 GMT
Powered by mindtouch™
3.1 Human Roles

Data Tamer assigns roles for the following humans:

A Data Tamer administrator (DTA). This role is analogous to a traditional database administrator. Hence, the DTA is in charge of assigning roles to the other humans and in deciding what actions to take during the curation process. Specifically, the DTA specifies the collection of data sources that Data Tamer must try to curate.

One or more Domain Experts (DE). These are human domain experts that can be called on to answer questions that arise during the curation process. Each DE has one or more areas of expertise and they are organized into an innovation crowd-sourcing organization as will be explained in Section 4.3.

3.2 Sites and Schemas

Data Tamer assumes that the DTA specifies sites, indicated by a URL or file name. Each site is assumed to be a collection of records, each containing one or more key-value pairs. An upstream wrapper may be required to construct this format from what the site actually stores. At the present time, Data Tamer is not focused on lowering the cost of such wrappers.

Data Tamer assumes each local data source has information about one entity. Hence, if a source is multi-faceted, then two or more wrappers must be used to ensure that each source contains data about only one entity. If every site describes a different entity, then there is no integration problem. Hence, the goal of Data Tamer is to group local sites into classes, which describe the same entity. In Version 1, there has been no effort to identify relationships between entities (such as might be present in foreign keys in an RDBMS) or to deal with other integrity constraints. These extensions are left for future research.

For each class of entity, there are three possible levels of information available. These depend on whether curation is proceeding in a top-down fashion or a bottom-up fashion. In a top-down model the DTA has information about the schema he is trying to achieve. Hence, the goal is partially or completely specified. In a bottom-up model, such global knowledge is missing and the global schema is pieced together from the local data sources, with perhaps hints available from the DTA. The fact that either model may be used for a given class leads to the following three levels of information.

- **Level 3: Complete knowledge.** In this case, the complete global schema for a given class of entities has been specified by the DTA using a top-down methodology. Generally, the DTA also maps each local data source to a specific class. However, if not done, Data Tamer includes algorithms to perform this task automatically. Although this level of knowledge is available in the Verisk application, we have found level 3 to be quite rare in practice.

- **Level 1: No knowledge available.** In this case, nothing is known about the structure of the classes of information, and a bottom-up integration is utilized. This level of detail might be true, for example, about random HTML tables that were scraped from the web. This is the world of systems like Web tables [8]. Although, this is the level of knowledge in the biology application, we believe it is also uncommon in practice.

- **Level 2: Partial information available.** Using either a top-down or bottom-up methodology, there may be partial information available. There may be specific attributes that are known to be present for some class of entities. This is the case for the Web aggregator, as it requires specific attributes to be present for each node in its classification hierarchy. Alternately, there may be templates available. A template is a collection of attributes that are likely to appear together in one of more classes of entities. For example, a template for a US address might be (number, street, city, state, ZIP code). Note that a template is simply a compound type, i.e., a collection of attributes that usually appear together. Templates may be specified by the DTA as a 'hint' or they may be identified via machine learning as noted in Section 4.
3.3 Other Information

In addition, in many domains, there are standard dictionaries, which should be used by Data Tamer. A dictionary is a list of data values of some data type that can populate an attribute in some data source. For example, there are 50 states in the United States, about 30,000 cities in the USA, etc. A dictionary is used to specify the name and legal values of such attributes. There are as many dictionaries as the DTA wishes to specify.

Dictionaries are generalized to authoritative tables. These are auxiliary data tables that are known to have correct information. For example, a list of (city-name, airportname, airport-code) could be an authoritative table with three columns.

Furthermore, Data Tamer accommodates synonyms of the form XXX is a YYY. For example, \wages\" is a \salary\" or \town\" is a \city\". In future versions, we may extend this capability into more general ontologies.

3.4 Management Console and Data Tamer Actions

Sites, categories, templates, dictionaries, authoritative tables, and synonyms can be specified through a DTA management console, which is a fairly traditional-looking GUI. A portion of this console is dedicated to allowing the DTA to specify actions for Data Tamer to take. These actions are:

- Ingest a new data source, and store the incoming data in a Postgres database. In the current version of Data Tamer, this database exists at a single node; however, it is straightforward to partition this database onto multiple nodes and parallelize the algorithms to be described.
- Perform attribute identification on data source-i, as discussed in Section 4.1.
- Perform entity consolidation on data source-i, as discussed in Section 4.2

At any time during attribute identification or entity consolidation, Data Tamer may request human assistance from a DE, as discussed in Section 4.3. Lastly, as noted in Section 4.4, any human may visualize any data set currently using a Data Tamer specific interface. We may switch to the more sophisticated Data Wrangler visualization system [1], thereby supporting the manual transformations possible in that system. This issue is further discussed in Section 4.4.

At any time, the DTA may request that attribute identification and/or entity consolidation be redone on all sites. Obviously, this becomes a time consuming task as more sites are present in the Data Tamer system. However, better decisions may be available based on the greater amount of information present. Hence, if source-i or source-j are not specified above, then Data Tamer should run the algorithms to be described on all data sources.

Lastly, Data Tamer keeps a history of all operations performed, and the DTA may \snap\ the curation process backwards to any past historical point. This is implemented using a no-overwrite update strategy.

3.5 Training Data

In our conversations with people who have enterprise data curation problems, we have seen two very different scenarios for Data Tamer use. The first one is appropriate for cases with minimal or no advance knowledge (i.e., levels 1 and 2 above). In this case, Data Tamer simply commences operation. Initially it is quite dumb and must ask a human for assistance regularly. As it gets smarter, it asks less often. Moreover, with the benefit of added knowledge, it will often makes sense to go back and run attribute identification and entity resolution again on already processed sites.
presumably making better decisions with the benefit of added knowledge. As such, training data is simply accumulated over time by the crowd-sourcing component of Data Tamer.

The second scenario deals with applications where more information is known (level 3 above). In this case, we have observed that real-world applications often have training data available. Specifically, they have collections of entities and/or attributes that are \textit{known duplicates"}. In other words, they have a collection of pairs of local identifiers that are known to match. There is no guarantee that they have found all of the matches. Hence, they provide a collection of matching attribute names or entities with no false positives. We have observed that the danger of providing a false positive is high in real curation problems, so real world DTAs are very cautious. Therefore, they provide hand-curated collections of known matches.

In the first scenario we start running the Data Tamer system, asking a human for help as appropriate. In the second scenario, we make use of known duplicates as initial training data. We provide more details in Section 4.2.

3.6 Data Source Update

Finally, some data sources may be dynamic, and be continually updated. In this case, Data Tamer can make a new snapshot of a previous data source-k. In this situation, it makes sense to reprocess the data source, since the information has changed. In Version 1, there is no notion of accepting a live data feed. Such an extension is left to a future release.

4. Data Tamer Components

A block diagram of Data Tamer is shown in Figure 1. Indicated in the diagram are the management console and components for schema integration, entity consolidation, DE support and human transformation. These four subsystems are described in this section. Most of the features described in this section are currently operational.

**Figure 1: The Data Tamer Architecture**
4.1 Schema Integration

The basic inner loop in schema integration is to ingest an attribute, $A_i$ from a data source and compare it to a collection of other attributes in a pairwise fashion. For each pair, we have available both the attribute name and the collection of values. Our approach is to use a collection of algorithms, which we term experts, each returning a score between 0 and 1. Afterwards, the scores are consolidated with a set of weights to produce a composite value. Data Tamer comes with the following four built-in experts, and additional experts may be plugged in via a simple API.

- Expert-1. Perform fuzzy string comparisons over attribute names using trigram cosine similarity.
- Expert-2. Treats a column of data as a document and tokenizes its values with a standard full text parser. Then, measures TF-IDF cosine similarity between columns.
- Expert-3. This expert uses an approach called minimum description length (MDL) that uses a measure similar to Jaccard similarity to compare two attributes \[ \text{MDL} = \frac{|A \cap B|}{|A \cup B|} \]. This measure computes the ratio of the size of the intersection of two columns’ data to the size of their union. Because it relies on exact matching values between attributes, it is well suited for categorical and finite domain data.
- Expert-4. The final expert computes Welch’s t-test for a pair of columns that contain numeric values. Given the columns’ means and variances, the t-test gives the probability the columns were drawn from the same distribution.

The attributes to be compared depend on which level of information is available to Data Tamer, as described in the next section. Moreover, the DTA can set a threshold for suggested mappings, so that high confidence suggestions can be automatically adopted, while low confidence mappings enter a queue for human review.

4.1.1 Suggesting Attribute Mappings

The attribute mappings to be considered by Data Tamer depend on what information is available for the curation problem at hand, as noted in Section 3.2. Depending on which level is being examined, different tactics are used.

Level 3. In this case, Data Tamer knows the global schema, i.e., all the classes of entities and their associated attributes. Sometimes Data Tamer is told the class to which the incoming data source belongs. In this case, it must merely match the two collections of attributes. The result of running the schema integration component is a pairing of incoming attributes to the elements of the class in the global schema. If Data Tamer is unsure of a pairing, i.e., the matching score is less than a threshold, then a human is involved as noted in Section 4.3.

In other cases, the class to which the incoming entities belong must be identified. In this case, Data Tamer runs the algorithm above on all classes, and computes an aggregate matching score for the attributes. It then picks the best one. Of course, if no score is high enough or if there are two classes with similar scores, then a human is involved in the decision process. It should be noted for each incoming attribute that this algorithm is linear in the number of attributes seen so far. Hence, the total complexity is quadratic. We discuss scalability issues later in this section.

Level 2. In this case, Data Tamer may have certainty on a subset of the attributes. If so, it runs the above algorithms. If an attribute fails to match, it is added to the schema for the class that is specified by the DTA or identified algorithmically. Future data sources can then match a larger collection of attributes. The complexity is the same as for Level 3.

If templates are available, then consider the set, $S$, of all attributes in any template, plus any dictionary names and synonyms. Data Tamer employs a two-pass algorithm. First, it matches all incoming attributes against the members of
S, keeping only the highest scoring one. In a second pass, the score of an incoming attribute is adjusted upward if other attributes match other attributes in the template selected. Then, the match is kept if it is above a threshold.

In addition, Data Tamer watches the incoming sites for collections of attributes that typically appear together. If so, it automatically defines the collection as a new template, and adds the newcomer to the template dictionary.

Level 1. Each incoming attribute is compared against all previously seen attributes, synonyms and dictionaries.

For all levels, the worst case complexity is quadratic in the total number of attributes. The rst expert is very cheap to run on pairs of attributes, since it does not look at the data. The other three experts must inspect the data columns and are much more expensive. So far, running time of our attribute identification algorithms has not been a show-stopper, since they are run\o line". Our first improvement will be to parallelize them over multiple nodes in a computer network, by replicating the Postgres database and then \sharding" the incoming attributes. This will yield an improvement of a couple of orders of magnitude. After that, further running time improvements will require a two-step process with a cheap first pass and a more expensive second pass on a subset of the attributes.

This two-pass approach will introduce additional experts whose running times are independent of the size of attributes' data sets. The current rst expert, which compares attribute names, is an example. Other experts will base comparisons either on attribute metadata or on samples derived from attribute data. Available metadata includes explicit fields in the data set like types and description strings. Although explicit metadata may not be available, useful attribute properties can always be computed and stored in a statistics table. Useful derived metadata include counts and histograms of distinct values, inferred data type, and so on. These statistics are also useful for constructing samples for other experts, for example, an expert that computes the Jaccard similarity of the sets of top-k most common values of two attributes. These rst-pass experts will act as a high-pass filter for the more expensive second-pass, and will save wasted effort making detailed comparisons between fields with little in common.

4.2 Entity Consolidation

Entity consolidation is effectively modeled as duplicate elimination. The goal is to nd entities that are similar enough to be considered duplicates. This module receives a collection of records, R1; : : : ;Rn, from one or more local data sources that arrive incrementally. We assume that attribute identification has been previously performed. Hence, all records have attributes values from a collection of attributes A1; : : : ;Am. In general, the data may well be noisy and sparse. The deduplication process is divided into multiple tasks as we show in the following.

4.2.1 Bootstrapping the Training Process

Initially, the system starts with zero knowledge about the deduplication rules. We learn deduplication rules from a training set of known duplicates and non-duplicates. We assume that duplicate tuples usually have at least one attribute with similar values. We obtain a set of tuple pairs that are potentially duplicates to be presented to expert users as follows. Let Simi indicates a similarity measure for values of attribute Ai. For each attribute Ai, we partition the range of Simi into a number of equal-width bins, and for each bin we select a sample of tuple pairs with Simi belonging to this bin. The obtained pairs, ordered by attribute similarities, are then presented to expert users for labeling. Since the presented pairs are sorted by the attribute similarity in a descending order, the experts could choose to stop labeling pairs below a
certain similarity threshold, and declare the remaining unseen pairs as non-duplicates. We denote by TP the set of pairs labeled as duplicates, and we denote by TN the set of pairs labeled as non-duplicates.

In order to increase the expected number of found duplicates in the candidate pairs, we only consider the attributes that have relatively large numbers of distinct values when obtaining the candidates (e.g., Title, Address and Phone) while discarding other attributes that are less distinctive (e.g., City and State). The reason is that high similarity on non-distinctive attributes does not increase the chances of being duplicates very much.

Another important source of training data is known duplicates, available with the data sets, as mentioned earlier. In addition, the web aggregator has specified several handcrafted rules that it uses to identify duplicates with high precision. Again, this is a source of known duplicates. We use the existing information as positive training data (i.e., TP). Negative training data (TN) is easier to find since non-duplicates are very frequent. Given a random set of tuple pairs, expert users need only to filter out any non-matching pairs that are highly similar, resulting in negative training data (TN).

4.2.2 Categorization of Records

Records are classified into multiple categories such that each category represents a set of homogenous entities that have similar non-null attributes and similar attribute values. This might occur, for example, if western ski areas look different than eastern one. For example, vertical drop and base elevation are noticeably different in the two classes of records. In addition, closure because of high winds may be commonly reported for one class and not the other. The benefit of record categorization is twofold: rst, by learning deduplication rules that are specific to each category, we achieve higher quality rules that can accurately detect duplicate tuples. Second, we use tuple categorization to reduce the number of tuple pairs that need to be considered in our duplicate detection algorithm. The performance gain is similar to that obtained by blocking techniques (e.g., [7, 14]) currently used in entity resolution in large datasets.

Categorization of records can be done using classifiers. In Data Tamer, categorization is performed in two steps. In the first step, we obtain a set of representative features that characterize each category. We obtain such features by clustering a sample of the tuples from the available sources. We use a centroid-based algorithm such as k-means++ [6]. The number of categories is determined with the help of the duplicates in training data TP that was obtained in the bootstrapping stage (Section 4.2.1). In the second step, we assign each tuple to the closest category (w.r.t. to some distance function such as cosine similarity). While similar to blocking in the achieved performance gain, this two-phase categorization is substantially different from previously proposed blocking algorithm, which are usually performed by clustering, indexing, or sorting the entire data set; these are very expensive operations, which we avoid in our categorization algorithm.

Categorization of tuples might change over time when new data sets become available. We maintain the categorization by adding new categories and/or merging/splitting the existing categories when needed. For example, consider the case where the radius of a given category (measured by the maximum distance between the representative features of the category and the category's members) becomes very large compared to the other categories. In this case, we split the category into two or more smaller categories. Efficient incremental categorization is one of our current research directions.
4.2.3 Learning Deduplication Rules

The deduplication rules are divided into two types: (1) cut-off thresholds on attribute similarities, which help pruning a large number of tuple pairs as we show in Section 4.2.4; and (2) probability distributions of attribute similarities for duplicate and non-duplicate tuple pairs. We learn such rules from the collected training data TP and TN. For example, one rule indicates that the probability of having two tuples with similar values of `Title', given that they are duplicates, is close to one. Another rule indicates that having different values for attribute `State' among duplicates is almost zero. Note that the learning module will choose to neglect a number of attributes that are not useful in learning the probability of being duplicates (e.g., `user rating' in data collected by the web aggregator). Also, since they are semantically different, the deduplication rules distinguish between missing attribute values and dissimilar attribute values, and we learn the probability of each event separately.

We use a Naive Bayes classifier [15] to obtain the probability that a tuple pair is a duplicate given the similarities between their attributes. This classifier aggregates the conditional probabilities for all attributes to obtain the marginal probability of being duplicates (assuming conditional independence across attributes).

4.2.4 Similarity Join

The goal of the similarity join between two data sets is retrieving all duplicate tuple pairs. Once the deduplication rules are obtained as shown in Section 4.2.3, we perform similarity join as follows. We obtain all candidate tuple pairs, where each pair belong to the same category and at least one attribute has a similarity above its learned threshold. Then, we compute the attribute similarities of the candidate pairs and we use these similarities to identify duplicate records according to the classifier learned in Section 4.2.3.

Similarity join is performed incrementally to accommodate new data sources that are continuously added. For each new source, we first categorize tuples within the new source, perform self-similarity-join on the new tuples, and perform similarity join between tuples in the new source and tuples in exiting sources. When new training data is added as a result of asking humans for help resolving ambiguous cases, we update the deduplication rules, efficiently identify which tuples are affected by such changes and re-classify them.

4.2.5 Record Clustering and Consolidation

Once we obtain a list of tuple pairs that are believed to be duplicates, we need to obtain a clustering of tuples such that each cluster represents a distinct real-world entity. Clustering of tuples ensures that the nal deduplication results are transitive (otherwise, deduplication results would be inconsistent, e.g., declaring (t1; t2) and (t2; t3) as duplicate pairs while declaring (t1; t3) as non-duplicate). We depend on a modified version of a correlation clustering algorithm introduced in [13]. Given a similarity graph whose nodes represent tuples and whose edges connect duplicate pairs of tuples, we perform clustering as follows. The algorithm starts with all singleton clusters, and repeatedly merges randomly-selected clusters that have a `connection strength" above a certain threshold. We quantify the connection strength between two clusters as the number of edges across the two clusters over the total number of possible edges (i.e., the Cartesian product of the two clusters). The algorithm terminates when no more clusters could be merged.

When the underlying similarity graph changes (i.e., new edges are added and/or existing edges are removed), we update the clustering as follows. We identify all nodes in the graph that are incident to any modified edges. Clusters that
include any of these nodes are split into singleton clusters. Then, we reapply the same merging operation on the new singleton clusters.

Tuples in each cluster are consolidated using user-defined rules. Null values are first discarded, and then we use standard aggregation methods such as Most-Frequent, Average, Median, and Longest-String to combine the attribute values of tuples within each cluster.

4.3 Human Interface

In both the attribute identification phase and the entity consolidation phase, a human DE may be asked by a DTA to provide curation input. In the case of attribute identification, the task is to decide if two attributes are the same thing or not. In the case of entity resolution, the task is to decide if two entities are duplicates or not. There are two cases that Data Tamer deals with as noted in the next two subsections.

4.3.1 Manual Mode

If the tasks requiring human intervention are few or if there are only a few DEs, then the DTA can manually assign human tasks to DEs. He does so by using a collection of rules that specify the classes of issues that should go to specific individuals. Alternatively, he can just assign tasks manually. In either case, routing of human requests is done by the DTA with minimal infrastructure.

However, if there are many issues, or if DEs vary in the amount of time they have to operate, or if there are many DEs to deal with a large volume of issues, then a more sophisticated crowdsourcing model should be employed. Data Tamer implements the model discussed in the next section.

4.3.2 Crowd Sourcing Mode

Large-scale data curation may require enlisting additional DEs with lesser expertise to help with the workload. Nonguru DEs can be asked to complete easier tasks, or they could be crowdsourced to produce results with higher confidence of correctness than could be assumed of any of them individually. But with a large and diverse DE population come the following issues that must be addressed:

- Determination of Response Quality. When a task requiring human intervention is forwarded to a single guru, the resulting response can be assumed to be accurate. But a task addressed by multiple DEs with variable expertise is likely to result in multiple responses of variable quality. Therefore, the set of distinct responses returned by a set of DEs should be accompanied by a probability distribution that reflects the aggregate confidence in each response being correct.

- Determination of DE Domain Expertise. The confidence of each distinct response to a task must be a function of the expertise ratings in the given task’s domain of those DEs who gave that response. This introduces a challenge for how to characterize and determine domain expertise.

- Motivation of Helpful and Timely DE Responses. Given a diverse DE population, individual DEs will differ by the responsiveness and effort put into their responses. The issue here lies in how to incent DEs to be good citizens in responding to tasks.

- Management of DE Workloads. DEs not only have variable domain expertise but also variable availability to respond to tasks. Thus, it is necessary to manage workloads in such a way that individual DEs are neither overtaxed nor underutilized given their workload constraints.
We have built a tool (the Data Tamer Exchange, or DTX) which acts as a market-based expert exchange by helping to match a task requiring human input to an individual or crowdsourced set of DEs who can provide it. DTX assumes a collection of attribute identification or entity resolution problems, which must be validated or refuted. For each such task, the tool shows a DTA how many DEs are available in each of some number of fixed Expert Classes associated with the task domain, and how the cost and quality of response will vary according to how many DEs from each class are asked to respond.

The key features of the DTX are the following:

- **Confidence-Based Metrics for DEs and Responses.** The DTX maintains a vector of confidence-based expertise ratings for each DE reflecting their degree of expertise in each of a set of domains specified by a DTA. Each rating is a value between 0 and 1 that denotes the probability that the DE produces a correct response to tasks from the associated domain. A DE's expertise rating for a given domain is calculated from reviews made of each of his responses in that domain by other, more expert DEs and from the DTA who requested the response. A similar confidence-based metric is used to measure the quality of a response (i.e., the probability that the response is correct). A response solicited from a single DE has a quality rating equivalent to the expertise rating of the DE. Crowdsourcing produces a collection of responses (TRUE or FALSE), and the quality rating of each vote is aggregated using Bayesian Evidence Gathering from the expertise ratings of the responders who voted for that choice. More specifically, given a question with n responders with expertise ratings \( \sim E \) and responses \( \sim R \), the cumulative confidence of a given response \( b \) is:

\[
\frac{P(\tilde{R}|b) \cdot P(b)}{P(\tilde{R}|b) \cdot P(b) + P(\tilde{R}|\bar{b}) \cdot P(\bar{b})}
\]

such that \( B \) and \( \bar{B} \) are random variables denoting events whereby response \( b \) is a correct and incorrect answer, respectively, and for either random variable \( X \),

\[
P(\tilde{R}|X) = \prod_{R_i = X} E_i \cdot \prod_{R_i \neq X} (1 - E_i)
\]

such that the subproducts above are the combined probabilities of the correct responders being correct \( (R_i = X) \) and the incorrect responders being incorrect \( (R_i \neq X) \).

- **Expert Classes.** The DTX dynamically clusters DEs into domain-specific expert classes according to the DE's expertise rating in that domain. For example, the most expert DEs in a given domain might be assigned to expert class #1 on the basis of having an expertise rating of 0.9 or higher. When a task is submitted to DTX, the tool responds by presenting statistics about each expert class in the task domain, including the number of DEs, the cost per DE response (classes with higher expertise ratings charge more per response) and the minimum expertise rating of DEs within the class. On this basis, a DTA decides for each class, how many (if any) DEs he will pay to respond to a task in this class.

- **Economic Incentives for Good Citizenship.** The DTX assumes an economic model whereby the payments earned by a DE for a response is commensurate with his expert class. Payment for responses comes from the accounts of DTAs who are provided a budget with which to complete their tasks. Payment for reviews of DE responses is provided by the system to DEs (at the same rate as responses) and to DTAs (whose pay is added to their budgets).

- **Dynamic Pricing to Manage Workload.** The DTX dynamically adjusts the prices paid per response at each expertise level so as to encourage the selection of underutilized DEs and discourage the selection of overtaxed DEs.
The economic model helps to address two issues with large-scale data curation:

1. Given that responses may be reviewed, DEs are incented to produce helpful and timely responses so as to achieve higher expertise ratings and therefore to earn higher pay.

2. Given that DTAs are assigned a fixed budget with which to complete all of their tasks, they are incented to spend as little as possible for DE responses. This helps to offload the responsibilities of guru DEs by encouraging DTAs to classify tasks according to their difficulty and solicit responses from the least expert (and hence cheapest) DEs whose responses have minimally acceptable confidence.

Further, the payment for reviews incent this crucial input, helping to ensure the accuracy of the confidence-based ratings of DE expertise and response.

4.4 Visualization Component

At any time, the DTA or a DE can call our visualization system and pass a local data source to that system. It displays the data source (or a sample of the source) as a table on the screen. The human can inspect the data source for insight.

It is possible that we will switch the visualization system to Data Wrangler to access their Extract, Transform and Load (ETL) operations. In this way a Data Tamer user could apply Data Wrangler transformations manually and convert data types and formats. In this case, we can remember the transformations applied and put them into a graph of data types. If we see the data types in the future, we can apply the transformations automatically. To allow the DTA to "snap" the data, we always employ a no-overwrite update strategy for data in the Postgres database.

Alternatively, we could implement a Data Tamer-specific visualization interface. In this way, the screen component can be tailored to Data Tamer needs. For example, the entity consolidation system wants to display potentially matching clusters and the schema matcher wants to show columns from multiple local data sources which might match an attribute of interest. Neither capability is included in Data Wrangler.

5. Experimental Validation

We ran the Data Tamer system on the data used by the web aggregator described in Section 2. After modest training on fewer than 50 manually labeled sources, our automatic system successfully identified the correct attribute mapping 90% of the time. This overall success depends on combining results from the individual experts. On their own, attribute name matching is successful 80% of the time, MDL 65%, and fuzzy value matching 55%. Because the aggregator has few numeric columns (postal codes, and latitude/longitude), the T-test expert only provides results for fewer than 6% of attributes, but its results are correct in 65% of cases. The experts compliment each other well: at least one expert identified the correct mapping for 95% of the attributes.

We evaluated our entity consolidation module using a set of 50 data sources. On average, each data source contains 4000 records, which took 160 seconds to be deduplicated and integrated into the central database (using a single machine). Statistics about the found duplicate pairs in the 50 data source are summarized in Figure 2. We compared our results to the duplicate pairs found by the current deduplication algorithm used by the web aggregator. The total number of records in the 50 data sources is 146690. Data tamer has reported 180445 duplicate pairs, while the aggregator's
algorithm only reported 7668 duplicate pairs. The number of common pairs reported by both the aggregator's algorithm and Data Tamer is 5437.

**Figure 2: Quality results of entity consolidation for the web aggregator data**

<table>
<thead>
<tr>
<th></th>
<th>Aggregator</th>
<th>Data Tamer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total records</td>
<td>146690</td>
<td>180445</td>
</tr>
<tr>
<td>Pairs reported as duplicates</td>
<td>7668</td>
<td>180445</td>
</tr>
<tr>
<td>Common reported pairs</td>
<td>5437</td>
<td>5437</td>
</tr>
<tr>
<td>Total number of true duplicates (estimated)</td>
<td>182453</td>
<td>182453</td>
</tr>
<tr>
<td>Reported true duplicates (estimated)</td>
<td>7444</td>
<td>180445</td>
</tr>
<tr>
<td>Precision</td>
<td>97%</td>
<td>100%</td>
</tr>
<tr>
<td>Recall</td>
<td>4%</td>
<td>98.5%</td>
</tr>
</tbody>
</table>

**Figure 2: Quality results of entity consolidation for the web aggregator data**

We assume that common pairs are true duplicates. Also, we assume that pairs that were not reported by either algorithm are true non-duplicates. We evaluated the accuracy of the remaining pairs (i.e., pairs reported by one algorithm but not the other) by asking a domain expert to examine a sample of 100 pairs. Based on the expert feedback, 90% of the pairs reported by the web aggregator but not reported by Data Tamer are true duplicates. Also, 100% of the pairs reported by Data Tamer but not by the aggregator were labeled as true duplicates. It follows that the (estimated) number of true duplicates reported by the aggregator is 5437+ (7668-5437)*0.9 = 7444. The number of true duplicates reported by Data Tamer is 5437 + (180445-5437) * 1.0 = 180445. The total number of true duplicates in the data set is 5437+(180445-5437) * 1.0 + (7668-5437)*0.9 = 182453. The precision of Data Tamer is 180445/180445=100%, while the precision of the aggregator is 7444 / 7668 = 97%. The recall of Data Tamer is 180445/182453 = 98.9%, while the recall of the aggregator is 7444/182453= 4%. These results clearly show that our entity consolidation module is capable of significantly increasing the recall of existing deduplication algorithms, while maintaining the same level of precision.

We also ran the schema identification system on the biology problem discussed in Section 2.2. Data Tamer successfully mapped 86% of the attributes.

Lastly, we ran the entity consolidation module on the Verisk medical claims data set. Figure 3 shows the quality of the resulting record pairs for various cut-o thresholds on pairwise similarity. We divided the threshold range [0,1] into 10 equiwidth subranges, and we obtain a sample of 15 pairs from each subrange. We relied on a domain expert to classify the sampled pairs into true duplicates and true non-duplicates. We computed the precision, the recall, and the F1-measure for each similarity threshold (Figure 3). To put these results into perspective, we computed the accuracy of the current deduplication algorithm used at Verisk. The precision of this algorithm is 12%, the recall is 30% and the F-score is 17%. On the other hand, our algorithm archives an F-score of 65% at threshold of 0.8.
To gauge user acceptance of our crowd-sourcing exchange, we are executing a two-step evaluation with the biology data mentioned earlier. The company plans to run Data Tamer on their entire biology problem, with several hundred experts as their crowd sourcing component. As a first step, they wanted to do a “dry run” to make sure the system worked correctly. Hence, they asked 33 domain experts to participate in a beta test of the system. We used Data Tamer to perform schema mapping on a portion of their integration problem. To verify Data Tamer's mappings, we used DTX's expert worker allocation algorithms to assign the 33 experts a total of 236 schema-matching tasks. In each task, the user was asked to mark Data Tamer's suggested match as True or False, and if False, to suggest an alternate match. On average, each task was redundantly assigned to 2 experts. This resulted in an average of 7 task assignments per user.

No economic incentives were given to users in this test and participation was voluntary. Of the 33 experts that we contacted, 18 (54%) logged into the system. Each user who logged in performed all of the tasks assigned. In total, 113 of the 236 task assignments were completed and 64% of the tasks received at least one response. The low voluntary response rate suggests the need for economic incentives that reward timely responses. After completion of the assigned tasks, we asked each participant to rate the system’s usability on a scale of 1 to 3. The average score given by users was 2.6.

The company is now proceeding with a full-scale test of the system using hundreds of domain experts. We plan to cluster data sources by domain, and then leverage each user's data entry history over those sources to determine appropriate initial domain expertise levels. For example, if a user is the creator of a data source in a certain domain, then that user's response can be weighted more heavily in determining the correct answer for a task than responses from users who have not entered data in that domain. We expect to report on this study in the near future.

6. Future Enhancements

In the body of the paper we have indicated assorted future enhancements, which we discuss in this section. First, we expect to parallelize all of the Data Tamer algorithms, so they can be run against a sharded and/or replicated Postgres.
database or against one of the parallel SQL DBMSs. Since many of the algorithms are implemented in SQL or as user-defined functions, this extension is straightforward.

The schema integration module may need to be accelerated by making it a two-step algorithm as noted earlier. In addition, we have yet to consider mechanisms to efficiently redo schema integration. Since our schema integration algorithms are inherently order sensitive, it is entirely possible that a different outcome would be observed with a different ordering of sites. As such an efficient redo will be a required feature.

Our entity consolidation scheme needs to be made incremental, so all of the subtasks can be efficiently run as each new data source is integrated and/or when new training data is added. Moreover, parallelization of this resource intensive module will be especially desirable.

So far, the only data cleaning operations in Data Tamer are in the entity consolidation system. Whenever there are multiple records that correspond to an entity, we can generate a clean result, either automatically or with human assistance. Although this is a valuable service, we need to implement a specific cleaning component. Unfortunately, data cleaning often relies on outlier detection or on a collection of cleaning rules. Outliers are not necessarily errors; for example -99 is often used to indicate that data is missing. Hence, finding an outlier is equivalent in this case to finding a missing value. Also, most cleaning rules are very complex, if they are to be useful. Although it is easy to state that ages and salaries must be non-negative, it is much more difficult to state that temperature near a window should be lower than temperature next to a heat vent, if it is winter. We expect to work on this component in the near future, guided by what users actually find useful in practice.

Similarly, we have not yet systematically addressed data transformations, for example to convert local data to have the same representation, to convert units, or to convert attributes to a common meaning (price without sales tax to total price, for example). Our approach is to maintain a graph of Data Tamer data types. Whenever a user exercises the visualization system to make a transformation, we plan to remember this as an arc in the graph. Obviously, a user should be able to add arcs to the graph with corresponding code to implement the transformation. This graph could then be used to suggest transformations in the visualization engine.

7. Conclusion

This paper has described the main features of Data Tamer, namely a schema integration component, an entity consolidation component, a crowd-sourcing module to organize domain experts, and a visualization component. In the future, we will add more modules to perform data cleaning and reusable transformations.

The system has already been shown to be valuable to three enterprises. At the current time, the code is being adopted by all three companies for production use.

8. References

[1]

http://vis.stanford.edu/wrangler/.


Appendix A. Demo Proposal

At the present time we have validated Data Tamer on three enterprise data curation problems:

- The web aggregator mentioned in Section 2.1
- The lab notebooks from the "Big Pharma" company mentioned in Section 2.2
- The health claims records mentioned in Section 2.3.

In the demo, we pick the web aggregator application, and we go through the end-to-end steps of data curation, namely:

1. Bootstrap the system by adding a few data sources, and providing initial training data for attribute matching and deduplication modules.
2. Ingest a new data source.
3. Examine the data source visually.
4. Perform schema integration with the composite source assembled so far, asking for human help through our crowdsourcing module.
5. Perform entity consolidation with the composite source assembled so far, again asking for human help in ambiguous cases.
6. We compare our solution with the one currently in production use, and show the superiority of our algorithms.
7. We also illustrate the scaling problem faced by this aggregator, and show why traditional human-directed solutions are problematic.
Slide 1 The State of the Art in Supporting "Big Data"

The State of the Art in Supporting “Big Data”

by

Michael Stonebraker

Slide 2 What is "Big Data"

What is “Big Data”

- Too much Volume (I have too much data)
- Too much Velocity (It’s coming at me too fast)
- Too much Variety (It’s coming at me from too many places in too many formats)
Slide 3 Too Much Data-The Data Warehouse World

Too Much Data -- The Data Warehouse World (structured data)

- Mature (and large) commercial market with several well-regarded vendors
- I know of a couple dozen of these in production use on petabytes of data
  - E.g., Zynga, E-Bay
  - That is about 20 Mbytes for every person in the US!
- No reason why this technology won’t scale as customers want larger installations
  - Expect data warehouses to get larger by 1-2 orders of magnitude over the rest of this decade.

Slide 4 Too Much Data-The Hadoop/Hive World

Too Much Data -- The Hadoop/Hive World (semi-structured data)

- I know over another 20 or so petascale Hadoop installations
  - E.g., Facebook
- No reason this technology won’t continue to scale
- And probably converge with the data warehouse world
Slide 5 Too Much Data-The Data Scientist World

Too Much Data --
The Data Scientist World

- Predictive Modelling, data mining, data clustering, recommendation engines, ...
- Complex analytics - not in SQL
- Not well understood
  - World of research, start-ups, ...
- My prediction:
  - As the world moves from simple analytics to complex analytics, the server side technology will mature to meet the need

Slide 6 Too Fast

Too Fast

- Often a legacy problem
  - Rise in stock market volume breaking the legacy real-time infrastructure of investment banks
- Usually solvable by throwing money/hardware at the problem
- Usually amenable to aggregation in the sensor network to knock down the velocity
  - E.g. car insurance sensors
Slide 7 Too Fast 2

Too Fast

- Some problems yet to be solved (query languages, integration of storage with “on-the-wire processing”)
  - But I see no showstoppers here
- Technology is capable of handling “the firehose” that will result from “the internet of things”

Slide 8 Too Many Places 1

Too Many Places

- Mature technology for integrating 20 data sources
  - Extract-Transform and Load (ETL) vendors
- But how to integrate 10,000?
  - Novartis has 10,000 bench chemists and biologists, each with an (independently constructed) data set of experimental results
  - Company wants to integrate these 10,000 data sources
  - And add additional ones from the public web
Slide 9 Too Many Places 2

Too Many Places

- Research problem!
  - Killing most CIO’s that I know
- Very active area of investigation
- Startups in this space

- If there is any achilles heel in big data, this is it!

Slide 10 DBMS Security

DBMS Security

- Works well
  - i.e. I have never heard of the DBMS screwing up in this area.
Slide 11 Encryption

Encryption

- Can be entrusted to the DBMS
  - Appropriate when there are many clients sharing data
  - Don’t want the encryption key to be on 500 desktops
- Can be entrusted to the client
  - Appropriate when there is personal (single user) data
  - See Nickolai’s talk this afternoon

Slide 12 Leaks

Leaks

- Usually insiders (think Edward Snowdon)
- Or unguarded desktops (my password on a post-it note on my PC)
- No possible way for the DBMS to prevent this
Slide 13 However

However

- DBMS can write a “command log” (everything everybody did)
  - Enables after-the-fact auditing
  - Sniff the log for suspicious behavior (unusual activity)
  - Would be a nice DBMS add-on
- But it is a human management problem to actually use it!!!

Slide 1 Big Data Means at Least Three Different Things....

Big Data Means at Least Three Different Things....

Michael Stonebraker
Slide 2 The Meaning of Big Data - 3 V's

The Meaning of Big Data - 3 V's

- Big Volume
  - With simple (SQL) analytics
  - With complex (non-SQL) analytics
- Big Velocity
  - Drink from a fire hose
- Big Variety
  - Large number of diverse data sources to integrate

Slide 3 Big Volume - Little Analytics

Big Volume - Little Analytics

- Well addressed by data warehouse crowd
- Who are pretty good at SQL analytics on
  - Hundreds of nodes
  - Petabytes of data
Slide 4 In My Opinion…..

In My Opinion…..

- Column stores will win
- Factor of 50 or so faster than row stores

Slide 5 Big Data - Big Analytics

Big Data - Big Analytics

- Complex math operations (machine learning, clustering, trend detection, ...)  
  - the world of the “quants”  
  - Mostly specified as linear algebra on array data

- A dozen or so common ‘inner loops’  
  - Matrix multiply  
  - QR decomposition  
  - SVD decomposition  
  - Linear regression
Slide 6 Big Analytics on Array Data – An Accessible Example

**Big Analytics on Array Data - An Accessible Example**

- Consider the closing price on all trading days for the last 10 years for two stocks A and B
- What is the covariance between the two time-series?
  
  \[(1/N) \times \text{sum}(A_i - \text{mean}(A)) \times (B_i - \text{mean}(B))\]

Slide 7 Now Make It Interesting …

**Now Make It Interesting …**

- Do this for all pairs of 4000 stocks
  - The data is the following 4000 x 2000 matrix

<table>
<thead>
<tr>
<th>Stock</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
<th>...</th>
<th>t2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_4000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hourly data? All securities?
Array Answer

- Ignoring the (1/N) and subtracting off the means ....

\[
\text{Stock} \cdot \text{Stock}^T
\]

DBMS Requirements

- Complex analytics
  - Covariance is just the start
  - Defined on arrays
- Data management
  - Leave out outliers
  - Just on securities with a market cap over $10B
Slide 10 These Requirements Arise in Many Other Domains

These Requirements Arise in Many Other Domains

- Auto insurance
  - Sensor in your car (driving behavior and location)
  - Reward safe driving (no jackrabbit stops, stay out of bad neighborhoods)
- Ad placement on the web
  - Cluster customer sessions
- Lots of science apps
  - Genomics, satellite imagery, astronomy, weather, ....

Slide 11 In My Opinion....

In My Opinion....

- The focus will shift quickly from “small math” to “big math” in many domains
- I.e. this stuff will become main stream....
Slide 12 Solution Options R, SAS, MATLAB, et. al.

Solution Options  
R, SAS, MATLAB, et. al.

- Weak or non-existent data management
- File system storage
- R doesn’t scale and is not a parallel system
  - Revolution does a bit better

Slide 13 Solution Options RDBMS alone

Solution Options  
RDBMS alone

- SQL simulator (MadLib) is sloooow (analytics * .01)  
  - And only does some of the required operations
- Coding operations as UDFs still requires you to simulate arrays on top of tables --- sloooow  
  - And current UDF model not powerful enough to support iteration
Solution Options
R + RDBMS

- Have to extract and transform the data from RDBMS table to R data format
- ‘move the world’ nightmare
- Need to learn 2 systems
- And R still doesn’t scale and is not a parallel system

Solution Options
Hadoop

- Analytics * .01
- Data management * .01

- Because
  - No state
  - No “sticky” computation
  - No point-to-point messaging

- Only viable if you don’t care about performance
Solution Options

- New Array DBMS designed with this market in mind

An Example Array Engine DB
SciDB (SciDB.org)

- All-in-one:
  - data management on arrays
  - massively scalable advanced analytics
- Data is updated via time-travel; not overwritten
  - Supports reproducibility for research and compliance
- Supports uncertain data, provenance
- Open source
- Hardware agnostic
Slide 18 Big Velocity

Big Velocity

• Trading volumes going through the roof on Wall Street - breaking infrastructure
• Sensor tagging of {cars, people, ...} creates a firehose to ingest
• The web empowers end users to submit transactions - sending volume through the roof
• PDAs lets them submit transactions from anywhere....

Slide 19 Two Different Solutions 1

Two Different Solutions

• Big pattern - little state (electronic trading)
  – Find me a ‘strawberry’ followed within 100 msec by a ‘banana’

• Complex event processing (CEP) is focused on this problem
  – Patterns in a firehose

P.S. I started StreamBase but I have no current relationship with the company
Slide 20 Two Different Solutions 2

**Two Different Solutions**

- Big state - little pattern
  - For every security, assemble my real-time global position
  - And alert me if my exposure is greater than X

- Looks like high performance OLTP
  - Want to update a database at very high speed

---

Slide 21 My Suspicion

**My Suspicion**

- Your have 3-4 Big state - little pattern problems for every one Big pattern - little state problem
**Slide 22 Solution Choices**

**Solution Choices**

- Old SQL
  - The elephants
- No SQL
  - 75 or so vendors giving up both SQL and ACID
- New SQL
  - Retain SQL and ACID but go fast with a new architecture

**Slide 23 Why Not Use Old SQL?**

**Why Not Use Old SQL?**

- Slooow
  - By a couple orders of magnitude
- Because of
  - Disk
  - Heavy-weight transactions
  - Multi-threading
- See "Through the OLTP Looking Glass"
  - VLDB 2007
Slide 24 No SQL

No SQL

- Give up SQL
  - Interesting to note that Cassandra and Mongo are moving to (yup) SQL
- Give up ACID
  - If you need ACID, this is a decision to tear your hair out by doing it in user code
  - Can you guarantee you won’t need ACID tomorrow?

Slide 25 VoltDB: an example of New SQL

VoltDB: an example of New SQL

- A main memory SQL engine
- Open source
- Shared nothing, Linux, TCP/IP on jelly beans
- Light-weight transactions
  - Run-to-completion with no locking
- Single-threaded
  - Multi-core by splitting main memory
- About 100x RDBMS on TPC-C
Slide 26 In My Opinion

In My Opinion

- ACID is good
- High level languages are good
- Standards (i.e. SQL) are good

Slide 27 Big Variety

Big Variety

- Typical enterprise has 5000 operational systems
  - Only a few get into the data warehouse
  - What about the rest?
- And what about all the rest of your data?
  - Spreadsheets
  - Access data bases
  - Web pages
- And public data from the web?
Slide 28 The World of Data Integration

The World of Data Integration

- the rest of your data
- enterprise
- data warehouse
- text

Slide 29 Summary

Summary

- The rest of your data (public and private)
  - Is a treasure trove of incredibly valuable information
  - Largely untapped
Slide 30 Data Tamer

Data Tamer

- Goal: integrate the rest of your data
- Has to
  - Be scalable to 1000s of sites
  - Deal with incomplete, conflicting, and incorrect data
  - Be incremental
    - Task is never done

Slide 31 Data Tamer in a Nutshell

Data Tamer in a Nutshell

- Apply machine learning and statistics to perform automatic:
  - Discovery of structure
  - Entity resolution
  - Transformation

- With a human assist if necessary
  - WYSIWYG tool (Data Wrangler)
Data Tamer

- MIT research project
- Looking for more integration problems
  - Wanna partner?

Take away

- One size does not fit all
- Plan on (say) 6 DBMS architectures
  - Use the right tool for the job
- Elephants are not competitive
  - At anything
  - Have a bad ‘innovator’s dilemma’ problem
The two groups accordingly decided to explore building a new open source DBMS. On the other hand, other sciences are poised to adopt large scientific databases in the near future. Astronomy, in particular the LSST project, perhaps presents the most significant immediate opportunity. While other fields are rapidly approaching similar scales, the requirements and needs of LSST appear to be a superset of those of other scientific communities.

The “reference cases” were particularly highly demanded; participants envisioned an in-depth examination of at least two concrete examples of built systems, one where data is federated and one where data is kept in a single instance.

The relational model is still relevant for organizing these extremely large databases, although industry is stretching it and science is struggling to fit its complex data structures into it.

The longevity of large scientific projects, typically measured in decades, forces scientists to introduce extra layers in order to isolate different components and ease often unavoidable migrations, adding to system complexity.

Source: https://www.jstage.jst.go.jp/article...1_012-010/_pdf (PDF)
ABSTRACT

The 5th XLDB workshop brought together scientific and industrial users, developers, and researchers of extremely large data and focused on emerging challenges in the healthcare and genomics communities, spreadsheet-based large scale analysis, and challenges in applying statistics to large scale analysis, including machine learning. Major problems discussed were the lack of scalable applications, the lack of expertise in developing solutions, the lack of respect for or attention to big data problems, data volume growth exceeding Moore's Law, poorly scaling algorithms, and poor data quality and integration. More communication between users, developers, and researchers is sorely needed. A variety of future work to help all three groups was discussed, ranging from collecting challenge problems to connecting with particular industrial or academic sectors.

Keywords: Analytics, Database, Petascale, Exascale, VLDB, XLDB

1 EXECUTIVE SUMMARY

The 5th XLDB (XLDB-2011) workshop focused on emerging challenges in the healthcare and genomics communities, spreadsheet-based large scale analysis, and the challenges of applying statistics and machine learning at large scales.

XLDB-2011 clarified the data-related problems in health-care and genomics. Some problems are general--conceptually-duplicated yet incompatible software, data formats, and usage models. Usage practices are not significantly converging because of a culture resistant to change. Stakeholders maintain a data-scarce mentality in a now data-rich world though some (e.g., those in attendance) have begun to realize the problem. Rapid growth in data scale from new machines and new technologies (DNA sequencing, medical imaging) caught them off-guard but are useful to highlight the lack of scalable tools and the need for stronger, more scalable data management. Unfortunately, funding sources are reluctant to pay for computation.

Spreadsheets in the context of big data were discussed at XLDB, following interest from the past year's workshop. Spreadsheets are individually small, but so popular, numerous, and ubiquitous (esp. in business) that they have become a large problem. Spreadsheets, due to their intuitive interface, are unlikely to be replaced, despite their problems in data quality. They are more like raw data, with no quality-enforcing mechanisms, such as schema, data typing, integrity, and authenticity, and thus are difficult to archive and maintain. The lack of strictness facilitates ease-of-use and reduces friction when exploring and recording ideas. Thus approaches to deal with spreadsheet problems focus on providing...
spreadsheet interfaces to other technologies better adapted to scale to large data sets, such as Hadoop or parallel RDBMS.

Statistics at large scales are a generally unsolved problem although some specific solutions exist. Statistics software packages do not scale but are used to prototype algorithms before building custom scalable code. Some participants noted that a single software solution balancing usability and scalability is just not practical, but others claimed that enough scaling problems can be addressed to produce an 80% solution. Some common algorithms are difficult to scale due to their computational cost (e.g., supra-linear scaling), so new, cleverer algorithms or more aggressive approximations are needed. Poor communication between statisticians and technologists is a big problem, with statisticians viewing it difficult to find solutions even when they exist, and technologists viewing statisticians uncooperative in describing their needs and problems. XLDB participants were optimistic about future collaboration and agreed to work towards collecting and curating problem descriptions from statisticians both to help statisticians cooperate among themselves and to help technologists build solutions.

State-of-the-art machine-learning (ML) practice is the extraction of data from their homes in data warehouses, archives, or managed data stores and subsequent feeding of them to specific algorithms. There are three primary approaches for scaling machine-learning. The first is to push logic into databases in order to leverage database optimizations and scalability. Unfortunately, not all logic can be pushed, and the resulting split is inconsistent, messy, and difficult to maintain. Yet databases should be part of the solution; the data cleanup and preparation enforced by databases and their quality controls are important. The second approach is to favor empirical heuristics and avoid sophisticated machine-learning models. This approach argues that existing ML research is sufficient for today's and tomorrow's problems. This last approach echoes the statistics community--prototype at the small scale and building customized code for particular large-scale conditions.

The small, informal atmosphere of XLDB workshops stimulated impromptu discussion of unplanned topics. Interest in free software is growing quickly, but larger organizations balk when commercial support is missing. Service computing architectures are attractive but too expensive when carrier-grade reliability and availability are unnecessary. In discussing communication gaps, we found the gap between SQL and non-SQL enthusiasts is wide and deep, with differences in culture (suits vs. hackers) and in approaches (rigid, well-defined vs. flexible, ad-hoc). The next steps for XLDB are to reach out to health-care again, to discuss data-integration problems, and to collaborate more with the high-performance computing (HPC) community. XLDB-2012 will be in the San Francisco Bay Area, but another satellite XLDB gathering is likely. Peer-reviewed papers are not being considered for the next XLDB.

2 ABOUT THE WORKSHOP

My Note: The first URL does not work and the second URL does and see: http://www-conf.slac.stanford.edu/xl...11/Program.asp

The Extremely Large Databases (XLDB) workshops provide a forum for topics related to databases of terabyte through petabyte to exabyte scale. The 5th workshop (XLDB-2011) in this series (workshop website: http://wwwconf.slac.stanford.edu/xld...1/Workshop.asp) was held at SLAC in Menlo Park, CA on 20 October 2011. The main goals of the workshop were to:

- reach out to the health care and genomics communities, which were not represented in the past,
• review statistics and machine learning as special topics in big data analytics, and
• discuss spreadsheet-based analysis.

This XLDB workshop followed a 2-day open conference, which was attended by 280 people. This report covers only the workshop portion. Information about the conference sessions, including the presentations, can be found at the conference website (http://www-conf.slac.stanford.edu/xldb2011/ - Jacek Becla, Kian-Tat Lim, and Daniel L. Wang. Facts about XLDB-2011. Technical Note SLAC-TN-12-001, SLAC National Accelerator Laboratory, February 2012).

2.1 Participation

Like its predecessors, XLDB-2011 was invitational in order to keep the group both small enough for interactive discussions and balanced for good representation among communities. XLDB-2011’s attendance numbered 56 people representing science and industry database users, academic database researchers, and database vendors. Industrial user representation continues to grow. Further attendance details are on the website.

2.2 Structure

Continuing the XLDB tradition, XLDB-2011 was composed of interactive discussions. It began with panel discussions on new communities: health care and genomics. Next were discussions focused on spreadsheet-based large-scale analysis, followed by discussions on statistics at scale and machine learning. The concluding discussions reviewed the plans for the next XLDB.

3 NEW COMMUNITIES: HEALTH CARE AND GENOMICS

The XLDB-2011 workshop engaged two “new” user communities, genomics and health care, via two representatives from the National Institutes of Health (NIH) and one from GNS Healthcare. Workshop attendees discussed data management and analytics in these communities: the current practice, the biggest problems, the barriers to solutions, and how they and the larger XLDB community could make progress.

Fragmented, small-scale approach to data

Genomics and healthcare communities are very fractured, with no consensus among many groups producing and managing data. Both communities have a pragmatic perspective of computing as a necessary but periphery expense. With little incentive for standardization and unification, data-producing equipment and data analyzing practices vary widely. Commonalities in language, definitions, and practices are scarce, making collaboration difficult. For example, sequencing machines have inconsistent resolutions, file formats, and interfaces that sometimes vary even between releases of a particular machine. The resulting “mess of data” is difficult to work with and divisive to the community. The good news is that people are beginning to notice this “horrible fragmentation.”

Some effort is being made to decrease in-house development in favor of more off-the-shelf software (possibly increasing interoperability). The genomics community welcome both not-too-expensive commercial and open source software but find that the commercial systems are “too expensive” and open-source is “not there yet” and “needs time to mature.” Thus the community continues to develop their own solutions. In-house solutions are also often developed because requirements and specifications are rarely known ahead of time—by the time they are finally known, a custom, non-elegant, half-baked solution is usually ready for deployment.
The healthcare industry frequently purchases commercial software, such as analytics software, and this results in “huge expenditures,” some of which are unfortunately “wasted.” Some companies are both users and providers, such as GNS Healthcare who specialize in building and commercializing custom solutions. Industrial users value commercial support for open source because they can externalize the associated liability to an external company. Somewhat less fragmentation exists in programming languages. Both communities use Java, R, and various scripting languages. Though not popular, SQL is also an acceptable language. R, a statistics package, is ubiquitous in genomics and used for many popular projects (e.g., the Bioconductor, a framework for analysis and comprehension of high-throughput genomics data). R is broadly accepted and appreciated but understood to have poor scalability. The community is accustomed to working around R’s limitations, needing more scalable analytical tools but not aware of any better alternatives.

**Problems driven by technological advances**

The genomics community has an immediate and desperate data explosion problem to solve within 1–1.5 years. The data explosion itself was caused, unsurprisingly, by technological advances. Higher resolutions and higher performance of instruments that are now cheaper by orders-of-magnitude are causing a growth in data production well beyond Moore's law. At the time of the workshop, NIH had collective capacity to produce 1 petabyte per year. The main problem is cultural and human rather than technological. The biology community has been slow to accept computing as an important part of research. Biologists are not used to accounting for computing and analysis in their budgets. Historically, sequencing has been expensive and its data scarce, meaning that the cost of storing and analyzing data was negligible. Conditions have changed drastically—the National Human Genome Research Institute (NHGRI) reported that it cost $10M to sequence a human genome in 2007 but just under $10K in 2011 (Wetterstrand, KA. DNA Sequencing Costs: Data from the NHGRI Large-Scale Genome Sequencing Program, available at: www.genome.gov/sequencingcosts). Hardware infrastructure for biologists has not kept up, especially for most biologists: an estimated half of all grants are forced to use insufficient, non-scalable, fixed compute infrastructure.

Another aspect of the human problem is that both communities lack people capable of grasping the “big data” challenges: one participant estimated that for some 1.4 million "data scientists" in health care, there were only 200 thousand “big data” people. The lack of know-how of existing tools and technologies is a problem as well. A lot of things that were obvious to the XLDB community were completely foreign and unknown to biologists. New technologies may be needed, but the lack of awareness of and expertise in existing computing is a more immediate problem. Most of the community does not possess the skills to write custom code and integrate off-the-shelf software. Among those with the skills, knowledge is typically shallow and ignorant of deeper system architecture and software construction principles. Sometimes the superficial knowledge is detrimental—one participant quipped “A little knowledge is a dangerous thing.”

The biologists who do invest significant effort in computing (e.g., designing data layout in databases) are often dismissed in their community (“structuring data is not science”, “writing code is not biology”). Programmers are "second-class citizens." Hospitals seem particularly unappreciative of IT professionals— one attendee told of a sad exodus of talented IT staff from a hospital where they felt particularly unappreciated. One commercial vendor argued that there was “too much democratization” of data analytics tools, claiming that better, commercial options are neglected. The practice of medicine was said to be still largely an “oral tradition” that lacks computational methods.
One new interesting practice not mentioned at past XLDB events was related to the determination of causality from data. Health care users particularly need to analyze data causality, that is, which data affect other data and in what ways. They noted that there are too many “known” causalities that are untrue in practice.

**Future directions**

Attendees were optimistic that the cultural problems can be solved. Collaboration should be increased between software engineers and biologists. Bridges are needed between science, computer science, academia, and industry. More partnerships are needed between hospitals and solution providers. One idea mentioned involved implanting engineers in projects as a way of changing the culture. Although one participant was skeptical that biologists would accept this, another cited a Dutch science foundation that required computer science students to work directly on a science problem as a degree requirement. Greater collaboration would reduce investment in the wrong things. For example, IBM invested heavily with a genomics company, but focused too much on the business buyer instead of the scientist, and ended up building models and demonstrating impressive computation while not significantly improving science. Vendor representatives suggested that collaboration could happen on an institutional level so that computing experts do not get pigeon-holed by biologists into system and database administration.

To address the community’s lack of expertise, solutions can be delivered as services rather than software and hardware that would require (greater) customer integration. In this way, the community can outsource its computing needs to experts and potentially reduce the need for in-house development. It was unclear, however, whether their data analytics can be met in this fashion. Another proposed way to reduce software integration and solution effort is to promulgate a common software stack for science much in the way that web companies have standardized on a LAMP (Linux-Apache-MySQL-Perl/PHP/Python, a popular foundation for application servers) stack. This tactic of raising the degree of shared commonality is used interdepartmentally at Indiana University.

The attendees recognized that XLDB facilitated solutions to a lot of the above problems. The workshop and (more recently) the conference were founded to facilitate communication and advocate development of solutions for large data problems. XLDB has built a human network that can collect use cases, put up a collection of well-chosen cases (“lighthouses”), find common needs across disciplines, match the technical strengths of available solutions with the requirements, produce recommendations, and package them into stacks (“foundation services”) usable by multiple disciplines. XLDB has already made a visible impact in technology use in some places (e.g., Exxon Mobil, according to a company representative), and it has the strong potential to influence culture elsewhere.

4 FROM SPREADSHEETS TO LARGE SCALE ANALYSIS

Although individual spreadsheets are not “big data,” they contain vast amounts of data in aggregate. The previous XLDB workshop’s attendees sought help in managing this data—their widespread usage in storing critical fragments of data could no longer be ignored. This session brought clarity to the problem of spreadsheets in the context of large scale data management and analysis.

While the usage and data volume of spreadsheets is not well known, one attendee estimated that 90% of all business data resides in spreadsheets. Used by practically every computer user, their ubiquity is undisputed. They are often used for critical multi-billion dollar business decisions, claimed one participant. Some popular uses for spreadsheets include:

- authoritative storage for field data collection,
• tools for computing or presenting summary statistics,
• simple ways of data visualization,
• places for ad-hoc integration of multiple data sources,
• input forms for data entry,
• scratch pads, and
• sandboxes for prototyping analysis techniques.

One participant declared that “the human mind thinks in rows of tables.” Hence, the spreadsheet interface model will never be replaced, despite the problems with the way they are used and the way most spreadsheet software operates. The tabular interface of spreadsheets for editing, visualizing, and manipulating is intuitive and powerful. Powerful functionality is usually included by default and more specialized functionality, such as text analytics or statistical processing with R, is well-integrated. Registered Microsoft Excel users were estimated to number 500 million (most of whom are not data professionals), and unregistered users equally as many. Some data sets, such as the Statistical Abstract published by the U.S. Census Bureau, are published in spreadsheet form. The simple row and column structure impose minimal constraints and do not interfere as users input, edit, manipulate, and interact with their data.

Spreadsheets make great playgrounds for exploring data, developing algorithms, and constructing models. (Algorithms developed by analysts inside spreadsheets are often completely rewritten by programmers and converted into “real”, repeatable pipelines to enable execution on large data sets. Large data sets are not stored in spreadsheets since their unified input/edit/visualize table interface is unwieldy beyond some number of rows and columns.)

Unfortunately, when the interface is loose, so is the data. Data types, units of measurement, and other semantic meanings are not directly stored, so data are re-interpreted according to user specifications for each new formula or chart. Spreadsheets do not specify or impose schema, as traditional databases do. Spreadsheet data are copied frequently, sometimes with transformations or tweaks, sometimes as a shim for integrating with other software, and sometimes for sharing. With proliferation of copies the authoritative, canonical version is often unclear. These problems of schema, data types, integrity, and authenticity are inherent to the spreadsheet model of computation. Capabilities difficult for databases, such as data provenance, security, and reproducibility, are doubly difficult with spreadsheets. All of the above make stewardship of data stored in spreadsheets very expensive or impossible. Participants agreed that solving these problems should preserve elements of the spreadsheet interface but that the computation and storage should be moved off the desktop. One approach is to integrate data from existing databases, data warehouses, and Hadoop clusters into a spreadsheet interface (the Datameer Analytics Solution is one such example). Another approach is implemented as a scalable cloud-based spreadsheet system (Google Fusion Tables (GFT) is an example of a cloud-based spreadsheet, allowing storage of tabular data for search, visualization, and collaboration). No approach seemed dominant, and more implementations are forthcoming. Neither of these approaches addresses the problem of data in existing spreadsheets, and while no solution is widely available, a participant from the University of Michigan demonstrated a sort of spreadsheet search engine, which is able to infer semantic meaning and structure from a large spreadsheet collection and answer free-text queries using the resulting index.

5 STATISTICS AT SCALE

The workshop participants agreed that the general problem of performing statistical analysis at large scale remains unsolved, despite reported solutions for specific sectors from SAS and other commercial vendors. Just as other computing applications have struggled to cope with growing data intensity, statistical analysis software packages like

http://semanticommunity.info/Data_Science/Big_Data_Science_for_CODATA/Workshops_on_Extremely_Large_Databases

Updated: Sat, 19 Sep 2015 00:38:45 GMT
Powered by mindtouch
MATLAB, R, and SAS are found to be ineffective or inapplicable at large scales. The paradigm suggested was that a statistical methodology should be developed using these tools on a data sample prior to re-implementing the solution on a more scalable platform like Hadoop, much as software is often prototyped using a slower but more dynamic programming language and then reimplemented using a faster, “production” language. Echoing this practice, John Chambers, creator of the S statistical programming language, pointed out that statistics packages were designed to allow statisticians to focus on the problem rather than details like scale and efficiency, which would be solved in a reimplementation anyway. Thus statistics software packages, like spreadsheet packages, should be used as a prototyping playground, with the heavy lifting to be done elsewhere. This also resonated well with a conclusion from previous XLDB workshops that “no single software system is a complete solution.”

Unfortunately, large-scale data-intensive computing is inherently non-trivial, and participants sought ways to integrate more scalable computing platforms with statistics software. SAS’s user-defined function (UDF) capability allows delegating functionality to a variety of database backends, including scalable parallel databases, such as Teradata. Similar computation-outsourcing plugins are available for R (noted in particular was the Rhadoop project [including rhbase, rhdfs, and rmr] of Revolution Analytics, a company founded to provide commercial support and development of R. The company was also developing a parallel implementation of R), but participants quickly noted that those are still young and need much work. Pushing processing to an external scalable backend is somewhat successful, but several claimed that making more and more computing resources available through statistics software, even if achieved, would not be a full solution.

A larger problem, claimed some, was the computation of algorithms with supra-linear time cost, i.e., those whose cost scaled with the square or cube of the data size or even greater. While more computational resources would speed their execution, their overall computational cost would remain unworkable. More work is needed to develop more efficient methods, either by novel implementation techniques such as Strassen’s algorithm for matrix multiplication or by using stochastic methods to reduce the computational data size. In praise of good approximations, a participant noted that the web search problem was essentially an eigenvalue problem and that Google’s PageRank provides efficient approximation of the largest eigenvalues.

Generally, the largest barrier to solving the scalable statistics problem is poor dissemination of knowledge of existing approaches. Statisticians were unaware of solutions that may already exist—R's software library is immense but often bewildering to most of its users. Software developers lacked enough details of statisticians' problems, making it difficult to identify and develop useful solutions. Some computing researchers complained that those with the problems are often reluctant or unable to release details due to security concerns (especially in medicine due to patient privacy) or competitive concerns (of both for-profit companies and academic researchers). A Microsoft representative noted that scientific communities have not made it clear what features are missing from existing tools. Some proposed the collection and curation of representative problems and case studies with sufficient detail that (a) statisticians can find problems (and accompanying solutions) similar to their own and (b) developers can evaluate their ideas and prototypes against concrete specifications. Finally, a benchmark or formal statement of a statistical “grand challenge” would spur new research, as the PennySort benchmark did (see: Performance / Price Sort and PennySort, by Jim Gray at al., MS-TR-98-45), push existing systems into addressing the yet-unsolved problems, and expose how rapidly vendors’ technologies can solve the statistics-at-scale challenges.
XLDB-2011 investigated the problems and considerations of machine learning (ML) in the context of large data volumes.

One interesting idea was that ML processing can take place entirely within databases, instead of extracting data from the database into an external statistics package. If ML primitives are to be implemented in the database layer, processing can leverage common database optimizations such as parallelism, caching, and improved I/O scheduling. The implementation of a data mining model in a relational database was described, mapping model specification to model-table creation, training to table loading, learning to querying, and prediction to a join operation between an input table and the model table. Yet the implementation showed that databases are poorly matched for performing all phases of ML processing, even though they can be used to accelerate significant portions. Participants agreed ML should exploit data within databases and thus leverage the considerable cleanup, normalization, and other preparation necessary to load data into strict schemas.

Another participant believed that the current data-abundant world is well-suited to empirically-trained models rather than sophisticated predictive algorithms and maintained that well-tuned, well-optimized heuristics are more effective in practical settings. LinkedIn's “People you may know” feature was cited as such an example of a collection of heuristics well-executed at scale. There was a sentiment that academic research in machine learning is already adequate to apply to general real-world problems, but real usage is uncommon and scattered—though there is evidence of successful ML usage in several places. One participant noted that heuristics are not grounded and their results cannot be used in diverse situations, but others felt they are usually adequate, citing extensive use in medicine.

Representing ML models was cited as a challenge. The Predictive Model Markup Language (PMML) is one of a few specifications, but no standards have been widely-adopted. Quality control was cited as another problem—one participant wished for quality measures for each step of processing to aid in understanding the result. Participants discussed more general considerations of deriving answers from data. The multi-hypothesis pitfall (given sufficient data, any arbitrary hypothesis can be supported) is a real danger in data-abundant environments. Another problem is that algorithms often operate on data sets assuming they represent the complete, closed world—a dangerous assumption because empirical data are usually more accurately considered a partial sampling. The practice of prototyping in one language or software environment and reimplementing in another is reiterated in the context of machine learning. Reimplementation is not only an opportunity to improve the algorithm but also an opportunity to introduce errors or misinterpretations. Algorithm creators and reimplementors are usually different groups of different specializations (i.e., statisticians and computer scientists) who communicate poorly due to differences in jargon, perspectives, and priorities. Most were not optimistic that prototype-and-reimplement can be eliminated in favor of a unified development process but suggested addressing the barriers between the different groups. Interdisciplinary groups were suggested over collaborating groups. A collection of data with use cases was again suggested as something that would steer computer scientists towards building better, more suitable tools while providing reference “best practices” for data scientists and other domain experts.

The XLDB community could help with enabling scalable ML by identifying and connecting with people who can represent their community with an understanding of both their domain and its computing. Also helpful would be the identification of a prototypical case, such as bi-clustering on the EXPO oncology data set (few GB) or analysis of the 200TB sequence data at google.org.
7 OTHER TOPICS

XLDB workshops foster intense discussions that often diverge toward unanticipated topics, and a sampling of these topics is provided below.

Free (libre) software was discussed as a solution tactic and as a development model. Representatives from larger industrial entities noted that commercial solutions have been strongly preferred historically, but that acceptance of free software is growing and not as frequently dismissed as immature. For the larger entities in non-computing industries, free software is only viable when accompanied by a commercial entity that provides support and absorbs liability.

Service computing is a promising method to satisfy storage and computation scalability needs, with some reservations. While current pricing is reasonable for commercial for-profit entities, another alternative is needed for academic usage where low cost is prioritized over high reliability and performance. Amazon Web Services is reportedly investigating such an alternative.

In discussing analytical models, participants recognized that a database computation model, where a question is posed in terms of a declarative query to an engine executing close to stored data, is effective for a large fraction of analytical processing. However, they also pointed out that iterative processing techniques, such as those that compute a result after some (possibly non-deterministic) number of repeated steps, are poorly supported by databases and require their own custom implementations.

Probabilistic answers to queries, that is, presenting a single answer as a probability mass function or probability density function, were reportedly desired by statisticians to guard against misinterpretation of results. However, no off-the-shelf solutions were mentioned, and one participant cautioned that implementing such an engine was much more difficult than one would expect. One approach was described in a poster at the XLDB conference: a prototype implementation that computed a probabilistic answer by duplicating database instances for each possible outcome.

The gap between communities using SQL and those not using seemed nearly unbridgeable to most participants. Hatred of SQL stems from the accompanying baggage accompanying most SQL-speaking database software, e.g., transactions and schema, but other problems are rigidity and inflexibility, e.g., the lack of support for fancy hacks in SQL, and the difficulty of embedding within C or other programming languages. Participants felt that while conversion and reconciliation are not likely, the communities can learn a lot from increased communication and knowledge exchange.

The use of data in high-performance computing simulations was split into two large categories. The first is concerned with monitoring large simulations' produced data streams for early error detection. The second is the "offline" analysis after such simulations complete and frequently not considered for execution on "big-iron" supercomputers.

8 NEXT STEPS

As in the past, a small portion of the workshop was devoted to future planning.

The next XLDB event should again reach out to the healthcare community. There is much more to explore and discover about large data problems in health care, whereas genomics and web-scale problems were well-covered this year. More general diversity was requested for the next event. Industrial communities suggested included mobile
telecommunications, manufacturing, in-flight data (e.g., Boeing), and national intelligence (e.g., DARPA). Hardware vendors, such as Intel, were requested so they could offer their long-term plans and perspective on emerging trends to the attending large data communities. Input from the venture capitalist perspective, perhaps via a panel discussion, was also requested.

The most-demanded topic was data integration. The data integration problem is one of the biggest unresolved challenges with few who understand the problem and even fewer who are working on solutions. Another topic was cloud computing in terms of costs and tradeoffs for data-intensive (not typical high-performance computing) usage. Other topics were database support for per-query schema (in response to schema-less computing in Hadoop) and array databases.

By far, the most highly-demanded new future activities for the XLDB community were:

1. collecting test cases (data and corresponding application software),
2. collecting use cases/challenges,
3. setting up a single repository to document/publish collected information (e.g. test cases and use cases from above), and
4. specifying “reference architectures” that detail particular hardware/software combinations for data-intensive solutions.

Attendees were interested in working more closely with the HPC community: many believed there are numerous lessons that the HPC and the XLDB communities could learn from each other. The best idea was for XLDB representatives to attend an HPC gathering such as one of the Supercomputing conferences. XLDB itself is too small to accommodate a large HPC contingent, and attendees felt that a small HPC contingent might feel uncomfortable or alienated.

The next conference should remain substantially similar to XLDB-2011 in overall length, balance between topic diversity and focus, and organization. Possible improvements suggested including an additional specialized workshop day and demonstrations (possibly during the pre-dinner reception). The community felt strongly that introducing peer-reviewed papers was not a good idea since they could dramatically reduce the presentation quality and suppress the open, uncensored dialog. Satellite workshops, such as XLDB-Europe in Edinburgh, were considered beneficial, but organization of a workshop located remotely (e.g., in Asia) could be difficult due to distance, language barrier, and visa issues (especially if organized in China). Given the limited financial and human resources of the XLDB core team, the tasks will need to be carefully chosen and balanced to maximize the benefits to future XLDB events and activities.

9 ACKNOWLEDGEMENTS

The XLDB-2011 workshop was sponsored by eBay, Vertica/HP, Chevron, SciDB&Paradigm4, MonetDB and IBM.

The XLDB-2011 was organized by a committee consisting of Anastasia Ailamaki (École Polytechnique Fédérale de Lausanne), Jacek Becla (SLAC, chair), Peter Breunig (Chevron), Bill Howe (University of Washington), David Konerding (Google), Samuel Madden (MIT), Jeff Rothchild (Facebook), and Daniel L. Wang (SLAC).

This work was supported by the U.S. Department of Energy under contract number DE-AC02-76SF00515.
2010: Report from the 4th Workshop on Extremely Large Databases

Source: https://www.jstage.jst.go.jp/article.../9_xldb10/_pdf (PDF)

(Article history: Received 1 February 2011, Accepted 1 February 2011, Available online 22 February 2011)

REPORT FROM THE 4th WORKSHOP ON EXTREMELY LARGE DATABASES
Jacek Becla*1, Kian-Tat Lim 2, Daniel Liwei Wang 3
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA
*1 Email: becla@slac.stanford.edu
2 Email: ktl@slac.stanford.edu
3 Email: danielw@slac.stanford.edu

ABSTRACT

Academic and industrial users are increasingly facing the challenge of petabytes of data, but managing and analyzing such large data sets still remains a daunting task. The 4th Extremely Large Databases workshop was organized to examine the needs of communities under-represented at the past workshops facing these issues. Approaches to big data statistical analytics as well as emerging opportunities related to emerging hardware technologies were also debated. Writable extreme scale databases and the science benchmark were discussed. This paper is the final report of the discussions and activities at this workshop.

Keywords: Analytics, Database, Petascale, Exascale, VLDB, XLDB
1 EXECUTIVE SUMMARY

The 4th XLDB workshop (XLDB4) focused on challenges and solutions in the oil/gas, finance, and medical/bioinformatics communities, as well as several cross-domain big data topics.

The three domain-specific panels expressed similar concerns about an explosion of data and limits of the current state of the art, despite having different applications and analyses. All three communities (and others present) were struggling with these challenges: integrating disparate data sets including unstructured or semi-structured data; noise and data cleansing; and building and deploying complex analytical models in rapidly changing environments. The oil/gas exploration and production business analyzed petascale seismic and sensor data using both proprietary rendering algorithms and common scientific techniques like curve fitting, usually with highly summarized data. The refining and chemicals business had terabyte, but growing, datasets. Most processing of historical financial transaction data was offline, highly parallelizable, and used relatively simple summarization algorithms although the results often fed into more complex models. Those models may then be applied, especially by credit card processors, to real-time transactions using extremely low-latency stream processing systems. High-throughput sequencing and other laboratory techniques as well as increasingly electronic medical records (including images) produced the large datasets in the medical/bioinformatics field. Applications here included shape searching, similarity finding, disease modeling, and fault diagnosis in drug production. The medical community was striking for its non-technical issues including strict regulation and minimal data sharing.

Progress was made on the science benchmark that was conceived at previous XLDB workshops. This benchmark was created to provide concrete examples of science needs for database providers and to drive solutions for current and emerging needs. Its specifications and details have now been published. The next iteration will go beyond processing of images and time series of images to include use cases from additional science domains.

Statistical analysis tools and techniques were reportedly insufficient for big, distributed data sets. First, statistical tools should be developed to scale efficiently to big data sizes. Second, approximating and sampling techniques should be used more often with large data sets since they can reduce the computational cost dramatically. Finally, existing statistical tools should be made easier to use by non-specialists.

New hardware developments have made big data computation more accessible though uncertain in some ways. Power is the biggest issue and one that will drive the future of hardware as well as analysis. Regarding performance, more evidence of the potential speedup from GPU computing was shown through examples of complex computations within SQL databases. Attendees were enthusiastic about new storage technologies like solid state disks but disagreed on whether they would displace older, proven technologies. They similarly disagreed on whether many-core processing (hundreds to thousands) would begin to replicate core+memory onchip rather than the current, simpler model.

True updates are rarely needed in extreme-scale databases since the vast majority of big data sets are immutable. Wherever data updates are necessary, attendees preferred data versioning and history over updating data in-place.

The next workshop is expected to convene in the fall of 2011. Reference case studies, high performance extreme-scale visualization, data simulation, and cloud computing were among most demanded topics.

http://semanticommunity.info/Data_Science/Big_Data_Science_for_CODATA/Workshops_on_Extremely_Large_Databases
Updated: Sat, 19 Sep 2015 00:38:45 GMT
Powered by mindtouch™
2 ABOUT THE WORKSHOP

The Extremely Large Databases (XLDB) Workshops provide a forum for topics related to databases of terabyte through petabyte to exabyte scale. The 4th workshop (XLDB4) in this series was held at SLAC in Menlo Park, CA, on October 5, 2010. The main goals of the workshop were to:

- reach out to the communities under-represented at the past workshops, in particular oil/gas, finance, and medical/bioinformatics,
- review special topics in big data analytics: approaches to big data statistical analytics, opportunities from emerging hardware technologies, and writable extreme-scale databases, and
- discuss the advancement of the state of the art of XLDB.

This XLDB workshop was followed by a 1.5 day open conference, which was attended by 150 people. This report covers only the workshop portion. Information about the conference sessions, including the presentations, can be found at the conference website.

2.1 Participation

Like its predecessors, XLDB4 was invitational in order to keep the group both small enough for interactive discussions and balanced for good representation among communities. XLDB4’s attendance numbered 55 people representing science and industry database users, academic database researchers, and database vendors. Industrial user representation was greater compared to past workshops. Further attendance details are on the website.

2.2 Structure

Continuing the XLDB tradition, XLDB4 was composed of interactive discussions. The first sets were panel discussions on domain-specific challenges and solutions. Next were discussions focused on specific crossdomain big data topics. The concluding discussions reviewed the science benchmark and plans for the next XLDB.

3 USER COMMUNITIES’ PERSPECTIVES

XLDB4 involved several new user communities in the areas of oil/gas, finance, and medical/bioinformatics. The first two of these three had never been represented or discussed at past workshops. Bioinformatics was discussed at XLDB3, but the topic was expanded to include new areas such as medical informatics and biosecurity.

Many similarities to other domains raised in the past were noted, including needs for incremental scalability, full automation of operations, fault tolerance, approximate results for queries, and software simplicity. Regarding the issue of using a database, particularly a relational one, versus analysis with a map/reduce framework, it was noted that adding SQL-like features to non-database solutions such as Hadoop increases users’ interest in these solutions, in some cases by as much as a hundredfold.

3.1 Oil / Gas

The oil/gas panel consisted of representatives from two large multinational corporations — Exxon Mobil and Chevron — and TechnoImaging, a company recently spun-off from the University of Utah's CEMI3. Some panel members were from
the upstream, or oil exploration and production, portion of the business and others came from the downstream, or oil refinement and chemistry portion.

The industry has had petascale data for some time now, having "more bits than barrels" as one panelist put it. The largest data sets are from upstream users and consist of long-term seismic measurements. They are used to build 3-D earth models at the highest possible resolution. Smaller, but rapidly growing, data sets come from instrumented wells, production facilities, refineries, and chemical plants. These sensor readings have typical sampling rates on the order of 1 per minute except when detection of certain transient features requires higher rates (~100Hz); their total data volumes may be in the dozens of terabytes. The growth in data volumes comes from faster and cheaper sensors and more wells that need monitoring. In some cases, e.g., passive sensing, data are unavoidably noisy and their sources poorly known, and a large amount of metadata is required to make them useful. Recorded data are typically de-sampled and compressed immediately after collection.

Common analyses include stacking multiple images together to achieve higher resolution, creating depth images through reflection coefficients, and curve fitting. These are similar to techniques used in astronomy and other sciences. Pattern-matching is used to explore new regions by comparing them to well-understood regions. Seismic data rendering is reportedly similar to movie industry rendering done by Pixar or DreamWorks, but these are difficult to compare since algorithms in both industries are proprietary. Most complex analyses on larger data sets are hard-coded as fixed pipelines; ad-hoc querying is highly desired but thought to be too difficult.

Industry insiders considered their data analytics practices to be "stone age" (in some cases dating back to the 1980s). For example, a disk is often still viewed as a precious resource, so data analyzers receive only very highly summarized data. Existing tools make it hard to extract, analyze, and visualize data. The community relies mostly on off-the-shelf software; among the tools mentioned were Paradigm 4, Schlumberger's Petrel 5, Halliburton's OpenWorks 6, Spotfire 7, Apache Tomcat Application Server, Matlab, Oracle, and SQL Server. Legal considerations are strong barriers to evaluating, let alone deploying, new software.

The oil/gas community reported its biggest problem to be poor data integration. Data sets originate from multiple sources world-wide and often cannot leave their countries of origin. Data sources and schemas are typically completely disjointed (e.g., brought in through acquisitions and never properly merged). However, upper management is pushing for data to be cleaned and better integrated since the cost of building new wells is increasing and the payoff of more educated decisions is apparent. Another reason for better data standardization is data exchange (driven by cost-cutting) between contract parties, like government agencies (such as USGS) and the oil/gas companies.

Other problems are poorly or inconsistently formatted data, difficulties in synthesizing different types of data, poor tools, and poor tracking of past work. Important data is often unstructured and/or in forms difficult to analyze, such as PowerPoint or XML, and analysis involves "art in interpretation." Analysis is further complicated by the variety of available aerial models (electric, acoustic, atomic, magnetic, seismic, and others). Additionally, the lack of appropriate tools and procedures to capture, preserve, and query data provenance results in unnecessary repeats of similar experiments.

3.2 Finance

The finance community was represented by users from JP Morgan Chase and VISA. They revealed different processing models, one dominated by low-latency lightweight transaction processing (credit card processing)
and another dominated by big offline data analysis (banking). Petascale data sets are not uncommon, especially where historical data must be kept for regulatory reasons (e.g., within banks). Data sets are naturally divisible, typically into geographical “zones” containing ~1 petabyte each. Zones are further subdivided into ~20 terabyte pieces to bypass limitations of existing off-the-shelf systems used for analyzing the data.

Computation at credit card processors was dominated by extremely low-latency stream processing on many small pieces of data. In these systems, each transaction performs ~400 jobs (“joins”) that process encrypted data (including hashes of names and account numbers) against recent (within 1 year) customer data without a database. Each transaction operates in a stream that processes up to a thousand transactions per second, and streams are typically grouped in 100 stream units. Because of tight latency requirements, these streams operate only on a terabyte, keeping everything in RAM. Card processors also performed offline analysis on larger, petabyte data sets that include longer time ranges (up to 10 years) for tasks like building neural nets and risk models, but “anything that makes money” is a stream process. Processing at banks, on the other hand, was dominated by heavy offline, pro-active analysis on cumulative historical data: continuous risk calculation, fraud detection, and pattern analysis. Compared to card processors, banks kept more data of a larger number of different types (e.g., binaries like check images).

Offline analysis is highly parallelizable — simultaneous runs of >100 streams are typical. Time-based analyses (monthly, daily, 10 min, 30 sec batches) are the most common. A typical process relies on basic commands such as cat, grep, awk, or sort. 80% of processing can be classified as simple grouping and sorting, independent of the type of analyses run.

The finance community reported that the required performance is usually achieved through expensive brute force in both hardware (e.g., hardware accelerators) and software (high end, vendor managed). For security reasons only private clouds tightly sealed with firewalls are used. Banking data centers tend to be as large as those run by web companies (O(100K) nodes) although the number of them is much smaller.

A typical analysis software stack includes dozens of different off-the-shelf programs ranging from Hadoop, through R, to Oracle, DB2, and Teradata. Custom C++ code is also prevalent, although it is gradually being replaced by Python equivalents (observing a 40:1 reduction in lines of code). R is used primarily with smaller (few gigabytes) data sets because of its complexity when used in full-stream processing. Different geographical areas are analyzed independently and never merged together.

The main problem cited by banking was just accessing data quickly—not just large scale data, but data buried in spreadsheets too. Another tough challenge cited was the construction and deployment of reliable models, e.g., those for fraud detection. Models are manually built in-house by modelers who typically do not truly understand what is modeled and how to model it. Their deployment is complicated when models exhibit anomalous behavior when using live versus test data. Models help the system do the right thing despite unique conditions where the best customers look like the worst customers — frequent travelers trigger many false fraud alerts but use credit cards the most. Other important needs included high availability, hot-hot failover, as well as role-based and row-level access.

3.3 Medical / Bioinformatics

The medical/bioinformatics panel was represented by institutional users from NIH (molecule screening), two children’s hospitals (medical records, analyzing proteins), U.S. Department of Energy (cybersecurity, genomics), and NASA (early disease detection). The medical and bioinformatics communities reported a data explosion similar to other domains,
caused by similar reasons: cheaper and higher-resolution instruments. All reported some degree of unpreparedness for the scope and scale of emerging data volumes.

Examples of medical and bioinformatics analytics include shape searching, similarity finding, disease modeling and analysis, and fault diagnosis in drug production. The last requires detailed provenance tracking of data from highly distributed sources and was reported as the most demanding provenance-related use case so far.

The most striking difference between these two communities and the rest is (lack of) research cohesion. Both communities reported wide dispersion and fragmentation, with many small groups competing instead of collaborating. Data is rarely shared due to ethical concerns and extreme regulations, as well as due to a desire to protect research that might yield valuable publications. As some noted, most problems are related to humans rather than software. They complained of the culture of cutthroat competition and non-sharing rather than collaboration but did not blame the researchers, agreeing that it seemed necessary for survival in their funding and leadership structures.

Data quality is a big problem. Because hardware and processing practices at wet-laboratories change frequently, sometimes from one month to the next, there is little time to achieve production and process stability. The arrival of new hardware means new proprietary formats and changes to the nature and form of collected data. Even if the instruments and formats were stable, there is considerable variation in its collection. In many cases, in particular when medical records are involved, data is observational in nature — it is noisy and collected inconsistently without any enforced standards. Thus data are not recorded with science and analytics in mind. Later analysis is complicated by missing data that are biased in statistically significant ways — for example, a doctor might determine that patient is not sick enough and not collect certain data. “Negative results” are often not collected because they are useless for publication, even though they would be valuable for future statistical analysis.

4 SCIENCE BENCHMARK

The concept of a science benchmark was first introduced at XLDB2. The idea behind it is to capture the essence of science data processing and analysis, including not only querying processed data but also the processing or “cooking” of raw data itself. The benchmark would highlight areas that are not well-served by traditional RDBMSes and would serve both as a repository of abstract, general, multidisciplinary use cases and as a spur to database developers to provide features that are useful to science and science-like industry.

The current version of the benchmark covers one such area: processing of images and time series of images. The data and queries are based on astronomy and in particular the LSST project, but they are designed to be general enough to represent the needs of similar domains such as geoscience and medical imaging although the exact alignment between the benchmark and those domains needs to be determined and may require adjustment. Example queries include detecting objects in images, resampling an image onto a different pixel grid, and finding intersections of object trajectories with regions of space and time. The benchmark can be scaled to different levels of computational difficulty and data size, enabling measurement on systems from a single computer up to a large cluster.

The team working on the benchmark submitted a paper to ICDE’11, and the paper was made publicly available shortly after the workshop through the XLDB4 website. The data generator and a sample implementation on top of MySQL are also available on the XLDB wiki.

http://semanticommunity.info/Data_Science/Big_Data_Science_for_CODATA/Workshops_on_Extremely_Large_Databases
Updated: Sat, 19 Sep 2015 00:38:45 GMT
Powered by mindtouch
At least three providers (SciDB, Greenplum, MonetDB) expressed interest in trying to run the benchmark. Broadening the buy-in from science and engaging more science disciplines were discussed as the most important steps to make the benchmark more useful. Other possibilities included a text-oriented benchmark emphasizing UDFs and including more of the overall data management process. Building a TPC-like organization that would coordinate the benchmark effort was considered but was deemed premature at the current level of momentum.

5 APPROACHES TO BIG DATA STATISTICAL ANALYTICS

During this session, attendees discussed statistical analytics in a big data context, discussing the main problems, current practices, and future directions. Statistical methods are widely used in many areas like forecasting, biodefense, and web user modeling. A few truths seemed obvious: (a) the current methods should be more scalable, simpler, and more accessible to non-experts; (b) use of analytics is widespread and becoming more so; and (c) the data volumes for analytics have long grown past the point where simple hardware upgrades are sufficient for large data sizes — new techniques must be used.

Computational statistics, like most computational applications, is still in transition to software that can scale and deal with the practicalities of big data. Most methods assume that data sets fit in a machine’s main memory and are only applicable to big, distributed data sets through much pain. Success requires an awareness (perhaps an intimate one) of implementation details like data partitioning schemes (big data sets are invariably partitioned), infrastructure details (e.g., topology, memory size), and runtime hardware failure. Few who have such an awareness/skill are also statisticians, and it does not seem practical in the long term for statisticians to worry about implementation details. Good solutions must be built, and in the near term, statisticians need to get involved in new areas, such as databases, visualization, or exotic hardware technologies. Solutions for some hard-to-parallelize algorithms, like machine learning, will be difficult to build, but solutions for most algorithms should be tractable.

The off-the-shelf statistics software systems used by statisticians, in general, have not embraced big data and are difficult if not impossible to use with large data sets. Attendees wished for more statistics capabilities in languages familiar to non-statisticians (like SQL) but cautioned that education was necessary because powerful tools are “dangerous” in the wrong hands. Still, statisticians are reluctant to learn new, more scalable methods because they are “stuck” in software systems such as R, SAS, and MATLAB that took extraordinary effort to master but that are extremely productive on desktop-sized datasets. To save human time, analytics need to be as automated as possible, and statistics functions need to be more widely available (e.g., in scalable tools). Attendees repeatedly called for merging the power of statistical tools with the scalability of Hadoop.

The use of analytics is so widespread that large organizations (especially in industry) now perform “analytics of analytics” to share knowledge, avoid duplication of effort, optimize resource usage (“avoid 15 identical jobs each touching the same petabyte of data”), and connect clusters of people doing similar analytics, in internal LinkedIn-style social networks.

Some have dealt with big data volumes by not persisting them. Instead, they perform continuous analytics on live data streams and visualize the results directly without persisting them. Certain data characteristics — variability, for example — are difficult to visualize, however.
To reduce the data and computational intensity, some participants pushed for more exploration of approximate results because they can be computed so much more quickly and because perfect results are nearly impossible in the presence of faults and “messy data.” Others cautioned that approximate results (from probabilistic algorithms or sampling strategies) can be misleading and could easily be interpreted incorrectly, warning of a “slippery slope.” Yet it is clear that, at least in some cases, computational costs can be reduced by using simpler algorithms, especially with bigger data volumes. Attendees cited anecdotal evidence that simpler models generalize and produce better results, noting that real data is messy and additional variables add big human costs to understanding. Research into new models and algorithms is hampered, however, by the limited availability of large, freely-distributable data sets.

6 EMERGING HARDWARE TECHNOLOGIES

Undeniably, the appetite for data is “growing faster than memory gets cheaper.” Flash memory-based solid state disks (SSDs), with their fast random access and potential for high bandwidth at a relatively low price, are attempting to satisfy the need for increased performance while keeping up with the hunger for larger sizes. They are quickly appearing in production systems, but opinions are highly divided on whether or not flash memory or other storage technologies, like memristors, can “change the curve” of computing by displacing old, cheap, proven technologies. Opinions are also divided regarding the future direction of general-purpose multi-core processing — many argue that replicating units of cores and memory is more likely than continuing the expansion of the number of cores per physical CPU package.

After demonstrations of their use and effectiveness by several disciplines, GPUs are now commonly considered as a means to accelerate data processing and analysis. Dividing tasks into small, highly-parallel units executed on GPUs has the potential to drastically speed up many applications, including complex computations within SQL databases, as a team from JHU demonstrated. In their case, moving computation to GPUs meant the use of different algorithms; they pointed out that the tree-based algorithms commonly found in database processing were inefficient at bin sizes small enough to fit in GPU implementations.

Some claimed that increasing network bandwidth speeds would be a game-changer for analytics, as the currently available 10Gbps (or emerging 40Gbps) bandwidth is close to the speed of local storage. Fast networks will certainly enable better virtualization and streaming.

The biggest issue for everyone was power. The most common techniques mentioned to limit consumption were: (a) eliminating computing (thinking before computing, eliminating useless queries), (b) optimizing computing (building power-optimized software), and (c) optimizing hardware (lower-power CPUs, GPUs instead of CPUs, SSDs instead of spinning disks). Power considerations are the most likely to decide the shapes of future analysis and hardware.

7 WRITABLE EXTREME SCALE DATABASES

The session on writable extreme scale databases exposed a limited need for true updates on extreme scale databases. Participants noted that the vast majority of data was immutable, primarily because published results must always be reproducible — nobody dared to update raw data. Derived data products, however, often require updates. For example, LSST will need to update some portion of its derived data products daily while tracking fast-moving and fast-changing astronomical objects. In most cases where updates are necessary, projects are choosing to append and track lineage instead of updating data in-place.
All agreed that guaranteeing true consistency at a large scale is too hard and too expensive, and thus users have increasingly accepted weaker data consistency, relying on provenance to recover from the unexpected. In summary, the session underscored the needs for tracking versions, history, and provenance reliably and did not expose any new big challenges related to updating large data sets.

8 NEXT STEPS

As in the past, a small portion of the workshop was devoted to future planning.

The future of the science benchmark was discussed. The next steps include publishing the ICDE’11 submission (done immediately after the workshop), publishing the benchmark along with an explanation of how to synthesize the input data, incrementally improving the benchmark using community feedback, and aligning the benchmark with additional science disciplines. Finding similarities between the benchmark and industrial needs, in particular from big areas such as health care, was viewed as a positive step forward. Increasing awareness of the benchmark should encourage vendor competition to support scientific needs.

Participants were once again overwhelmingly satisfied with the value of the workshop and thought it should continue. They agreed that the next workshop should be held in the fall of 2011 in the San Francisco Bay Area and that the current format and length should be left unchanged. They suggested that XLDB5 cover reference cases, high performance visualization for extreme scale data, analytical (including extract-transform-load) tools, simulation data in science (e.g., climatology) and industry (e.g., automotive industry), and cloud computing. The “reference cases” were particularly highly demanded; participants envisioned an in-depth examination of at least two concrete examples of built systems, one where data is federated and one where data is kept in a single instance. They hope to learn about the architecture, fault tolerance and data replication strategies, and the tactics of getting daily analytical jobs done. The attendees suggested including representatives from health care, pharmaceutical research, the movie industry, the automotive industry, the census, and national intelligence.

ACKNOWLEDGMENTS

The XLDB4 workshop was sponsored by SciDB and Zetics, Greenplum, IBM, eBay, LSST, and Aster Data.

The XLDB4 was organized by a committee consisting of Magdalena Balazinska (University of Washington), Jacek Becla (SLAC, chair), Jeff Hammerbacher (Cloudera), Peter Fox (Tetherless World Constellation), Kian- Tat Lim (SLAC), Raghu Ramakrishnan (Yahoo!), and Arie Shoshani (LBNL).

GLOSSARY

CEMI – Consortium for Electromagnetic Modeling and Inversion  
ETL – extract-transform-load  
DOE – Department of Energy  
GPU – Graphics Processing Unit  
ICDE – International Conference on Data Engineering  
JHU – Johns Hopkins University  
NASA – National Aeronautics and Space
netCDF – Network Common Data Form
NIH – National Institutes of Health
RDBMS – Relational Data Base Management System
SKA – Square Kilometer Array
SSD – Solid State Disk
TPC – Transaction Processing Performance Council
UDF – User Defined Function
USGS – U.S. Geological Survey
XLDB – eXtremely Large Data Base

FOOTNOTES

1

2

3

4
http://www.pdgm.com

5
http://www.slb.com/services/software/geo/petrel.aspx

6
http://www.halliburton.com/ps/Default...210&pageid=852

7
http://spotfire.tibco.com/

8

9
https://confluence.slac.stanford.edu...S-DB+Benchmark
Academic and industrial users are increasingly facing the challenge of petabytes of data, but managing and analyzing such large data sets still remains a daunting task. Both the database and the map/reduce communities worldwide are working on addressing these issues. The 3rd Extremely Large Databases workshop was organized to examine the needs of scientific communities beginning to face these issues, to reach out to European communities working on extremely large scale data challenges, and to brainstorm possible solutions. The science benchmark that emerged from the 2nd workshop in this series was also debated. This paper is the final report of the discussions and activities at this workshop.

Keywords: Analytics, Database, Petascale, Exascale, VLDB, XLDB

1 EXECUTIVE SUMMARY

The 3rd XLDB workshop (XLDB3) focused on reaching out to communities underrepresented in the first two workshops, improving the science benchmark, and reviewing available large-scale solutions. The new communities brought to XLDB3 expressed strong needs for petascale solutions: radio astronomy's SKA survey is preparing to deal with a data set that might exceed an exabyte; geoscience needs solutions for integration and unification of their diverse data sets; and biology faces a data explosion in microscopic imaging and protein and genomic sequencing. LHC's largest headaches reside in managing the heavily-accessed metadata for their petascale data. Nokia reported that industrial users urgently need a robust petascale analytics platform, showing that their efforts (and funding) for analytics are strikingly large even when compared to efforts in large scientific projects.

To achieve extremely large scales, partitioning/chunking and distribution of data is required. Data distribution could be hierarchical, as in HEP's hub-and-spoke model, or non-hierarchical, as in geoscience and others. A significant fraction of scientific data is image-based and can be naturally represented as n-dimensional arrays. These data sets fit poorly into relational databases, which lack efficient support for the concepts of physical proximity and order, and so they are typically stored in array-friendly formats such as HDF5, netCDF, FITS, or casacore. Scientific data sets often need to be integrated from a large variety of sources, transformed, regridded, aligned, and calibrated before they can be analyzed. Performing QA and maintaining data consistency pose new challenges at the petascale level and often require using...
advanced techniques such as machine learning. Long term data preservation is hard but necessary and is being seriously investigated. No comprehensive, petascale, freely available database solutions exist yet, so most large-scale users continue to write custom software. Userdefined function (UDF) interfaces are helpful but provide only a minimal level of software reuse. Scientific process flows are highly optimized for maximum throughput and minimum cost, often at the expense of flexibility and response time. Both industrial and science users observed that the majority of their resources are consumed by a small number of sophisticated users. Despite the huge potential of mining raw data, such as transaction logs in industry or raw pixel data in science, all participants reported throwing away most of this valuable data, as it is too expensive to keep. Centralizing analysis requires a paradigm shift within the science community, as the desire for owning and controlling data by individual scientists stands in the way. Funding for scientific data management (SDM) is often misallocated: examples include overfunding data collection at the expense of underfunding analysis or cutting funds in favor of funding science directly. Finally, SDM appears to evolve at a much slower pace than its industrial peers, due to tight funds, legacy software, and inertia within large communities centered around big projects.

The map/reduce (MR) model, which has become popular in industry, was discussed. The ease of expressing queries through a procedural language and the availability of a free open-source system (Hadoop) were believed to be among the strongest points of MR. Frequent checkpointing of MR limits performance but is critical for handling failures that can wreak havoc with RDBMSes’ optimistic assumptions. Strict enforcement of data structures in RDBMSes has led users with poorly structured and highly complex data to avoid databases. Luckily, the RDBMS and MR communities are quickly learning from each other; each community is fixing its deficiencies and adding missing features. In practice, they appear to be rapidly converging.

Several solution providers presented their thoughts on terascale and petascale analytics. MonetDB presented a successful port of the SDSS multi-terabyte database. Cloudera discussed activities to support the Hadoop community. Teradata explained new techniques involving migration of data to appropriate (faster or slower) media based on frequency of access. Greenplum discussed dynamic re-mapping of a pool of servers to warehouses. Astro-WISE presented their system. SciDB demonstrated a from-scratch prototype supporting an ndimensional array data model, running in a shared-nothing environment.

A first draft of the science benchmark concept, introduced at the previous XLDB workshop, was discussed. The draft covers raw data processing and derived data analytics in the context of an array data model. The next steps include adding extra scaffolding, broadening the team, and expanding the scope to cover additional data models.

It was agreed that the next XLDB workshop, XLDB4, will be held in the fall of 2010 in Silicon Valley in the United States. It will attempt to reach out to remaining underrepresented communities, and industry presence will be increased. Biology, geoscience, and HEP will provide their use cases shortly after the XLDB3 workshop. Applying for funding from the European Commission for Europe-based XLDB and/or SciDB activities through "FP7" proposals will be considered.

2 ABOUT THE WORKSHOP

The Extremely Large Databases Workshops (XLDB) provide a forum for topics related to databases of terabyte through petabyte to exabyte scale. The 3rd workshop 1 in this series was held in Lyon, France, from August 28 to 29, 2009. The main goals were to:

- exchange information with communities outside the USA,
• invite and involve more science disciplines, especially those underrepresented in the past,
• review existing XLDB tools and solutions and discuss how to move the state of the art forward.

2.1 Participation

Like its predecessors, XLDB3 was invitation-only. This kept the attendance small enough to enable interactive discussions without microphones and ensured an appropriate balance of participants among the communities. Fifty-two people attended, including science and industry database users, academic database researchers, and database vendors. Compared to past XLDB workshops, a smaller fraction were industrial users, owing to difficulties in finding the appropriate people from European companies. Further attendance details are on the website.

2.2 Structure

Like past workshops, XLDB3 was chiefly composed of highly interactive discussions. These included sessions on complex analytics, scalable architectures, and various XLDB solutions. Particular focus was given to issues for scientific communities underrepresented in the past, including geoscience, radio astronomy, and biology, as well as high-energy physics (HEP), which was well-represented due to CERN's proximity.

3 DATA USER COMMUNITY PERSPECTIVE

XLDB3 involved several new user communities: geoscience, biology, radio astronomy, and a representative from the telecommunications equipment and service industry (Nokia). With the geographic proximity to CERN, it was natural to delve deeper into the needs of HEP, particularly the Large Hadron Collider (LHC) experiments. Although overtures were made, the workshop will need to continue to work to draw attendance from more communities such as chemistry and the oil and gas industry.

3.1 Specific user communities

Many science and industrial users already, or will soon, manage petabytes of data, some in a database but most outside. HEP reached the petascale with projects like BaBar, D0, and CDF. Optical astronomy will reach this data size as surveys such as PanSTARRS and LSST come on-line. More science domains are approaching the petabyte scale, and a few are even planning for exabytes of data. Approaches and solutions for smaller data scales, which are strained at current data volumes, will soon be non-functional and must be redesigned.

3.1.1 Radio astronomy

Radio astronomy’s Square Kilometer Array (SKA) survey, which has a planned construction start date of 2015, will collect data from over 3,000 dishes plus other receptor technologies covering an area of one square kilometer at a rate of 100s of terabytes per second. SKA’s current major challenges are to estimate costs for sponsors, to determine the feasibility of potential requirements, and to find ways to reduce the data volume. With the current level of reduction, SKA estimates its persisted data will exceed an exabyte over its 20+ year lifetime. The community is exploring ways to automate analysis, as the current manually-driven methods will not cope with the new data volumes.
3.1.2 Geoscience

Unlike other data intensive communities, geoscience has no “very large” projects. Instead, it is a diverse “system of systems” consisting of many disciplines such as solar/heliospheric physics, planetary sciences, geology, atmospheric and ocean sciences, tectonophysics, seismology, and hydrology. Data sets from individual observatories (surface, orbital, etc.) or instruments may be multi-terabyte-sized. An important challenge is the integration and unification of these diverse data sets to facilitate their access and analysis in larger, more coherent units.

3.1.3 Biology

Biological data sets, at a high level, are organized like those in geoscience – they consist of a large number of relatively small, independent data sets. However, two notable exceptions exist: (a) microscopic and other forms of imaging and (b) protein and genomic sequencing. Both are now approaching the petascale. For example, 20 samples of 20,000 genomes, each containing 3,000,000 bases, amount to 1.2 quadrillion floating point numbers. The growth in sequence data is fueled by rapid decreases (as much as a factor of five per year) in sequencing costs, and the resulting rapid expansion in sequencing machines from a few per city to a few per hospital. Controlling access to data in a way that complies with ethical and privacy regulations is a big problem.

3.1.4 HEP/LHC

LHC has now started data taking, and will soon generate some 15 petabytes of data per year, with a matching amount of data from simulations. The vast majority of this data is composed of events stored in files, with tens of terabytes of database storage used for auxiliary information, including calibrations, detector and accelerator conditions and configurations, metadata for event selection, provenance, catalogs, and file and job management metadata. The main database challenge for the LHC experiment teams lies in managing the metadata needed to select, understand, and process the event data, including substantial amounts of information that vary with the time of each event. While such data amounts to only tens of terabytes, ensuring scalable access to it for a large number of simultaneous, distributed jobs poses non-trivial challenges. Selections based upon event-level metadata are typically time-varying and combinatorial and so also tax the capabilities of traditional relational databases even at terabyte scales.

3.1.5 Nokia

Nokia was the only industrial user officially presenting at XLDB3. Business analytics at Nokia are typically performed using a combination of Teradata and Oracle, but the company is simultaneously trying many other approaches, including normalized and denormalized databases, in-memory databases, Hadoop, and home-grown software. The amount of effort (and funding) going towards analytics is strikingly large compared to efforts in large scientific projects. This effort is driven by the huge business value of having agile, flexible analytics. The system has to be capable of adjusting to rapidly changing needs; new data marts are built daily; and new sources of data are integrated weekly.

The company is already dealing with petabytes of data, and it expects to reach the level of a few tens of petabytes as soon as next year. The system is managed by a team of nine highly skilled US-based engineers, augmented by some 200 off-shore developers in Asia.
3.2 Data distribution architecture

3.2.1 Distribution models

Data can be distributed in three major models: (1) centralized, non-distributed, (2) hub and spoke, hierarchical, and (3) non-hierarchical, fully-distributed. The centralized model, where data is kept in a single place (data center), is technically the simplest for access and for backup. Data loss can be avoided by single site backups and offline storage. Funding concerns affect data distribution, as funding has been easiest to obtain for locally-controlled data repositories rather than centralized ones. In the hub-and-spoke model, there is a central warehouse with a master copy of less-processed data and spokes (data marts), which extract portions from the master and apply additional processing. The hub-and-spoke model is advantageous in politics/funding but can be problematic when access to products across multiple sites is needed or when data are updated. The fully distributed model, where data is produced independently at many sites, may be the most funding-friendly, but the lack of organization or standards (even de-facto standards) make data integration a serious problem. This model, prevalent in geoscience and much of biology, happens naturally when no major experiments or collaborations dominate. Industrial users reported use of all three models.

3.2.2 Problems

To manage large data sets, most data-intensive users “chunk” data into manageable pieces and distribute them across multiple data centers. Determining the right distribution scheme is non-trivial, as there are many important implications that need to be considered, ranging from data locality, latency and performance, through cost, to data recoverability. Excessive centralization leads to insufficient protection against failures, while excessive distribution is counter-productive: it leads to redundant data storage and expensive WAN transfers. In some cases, especially in industry, data distribution must be abstracted from end-users; e.g., a system must gracefully recover even from a failure of an entire data center. In science, this level of transparency is not usually required.

3.2.3 Practice

Industrial users tend to build geographically distributed data centers. In typical practice, data is replicated in at least two locations, and in case of a failure of a data center, another data center (or several) take over the traffic. The hub-and-spoke topology is frequently used, consisting of a few large centers and many smaller data marts.

Different science domains have adopted different models. Most common is a tiered approach, a form of hub-and-spoke pioneered by the HEP community. Data is organized into hierarchical data centers: a single tier 0 center (e.g., CERN), large analysis centers at tier 1, medium and small data centers at tier 2 and tier 3, and small tier 4 sites (which may be as small as a team of a professor and a few students). Each tier supports different roles and different forms of access. The tiered model is being considered by both optical and radio astronomy and seems to function especially well for large collaborations with distributed funding.

The distributed model is seen in the geoscience community, with a large number of different, largely independent sites. Services such as OPeNDAP 4 are built on top of these sites that aggregate, organize, and virtualize access, much as search engines organize the web. Such distribution has happened naturally due to highly distributed funding practices. The resulting diversity of data formats and access methods is a serious problem hindering wide data use. About 50
years ago there were attempts to introduce large data aggregation centers in geoscience, with only partial success: these centers became just archive centers instead of data access centers as originally planned. Though there are some new attempts to restart this effort now, the distributed nature of funding and the research itself make a unified effort hard.

Within the larger biology community, the genomics community requires that all raw data used to publish must be archived and publicly available, except where personally identifiable information is involved. This led to the establishment of three large data centers, each archiving and providing data access. This model seems to work well. One of the main “headaches” reported by these centers is a widely varying level of expertise on the receiving end: the end users range from a small number of experienced scientists to millions of untrained clinicians.

3.3 Data formats and models

A significant portion of scientific data is image-based and can be represented as n-dimensional arrays. This is true for almost all of the geoscientific data, most astronomical (both optical and radio) data, most biological data (especially the microscopic images that are experiencing the most growth), and medical images. In a large proportion of cases the images have spatial dimensionality (x/y, longitude/latitude, and right ascension/declination) and time dimensionality (a time series). Of the domains represented, only HEP and parts of biology reported significant volumes of non-image data. HEP deals with uncorrelated events, where an event is a complex structure including tracks of particles, and the non-image parts of biology deal with big graphs and sequences.

All sciences that deal with images store them as flat files, using popular formats such as HDF5, netCDF (geoscience, biology), FITS, and casacore (astronomy). These communities had considered or experimented with relational database management system (RDBMS) solutions for images but ended up reverting to flat files and using the RDBMS only for metadata. The two main difficulties with RDBMSes are: a) images fit poorly in a relational data model: their spatial and temporal dimensionality and the proximity and ordering of their pixels are essential and not easily represented in a set-oriented relational model; b) naïve image data chunking hampers parallelization: many operations (e.g., near neighbor search or regridding) require looking at adjacent pixels, which often are contained in an adjacent chunk that may be stored on a different node.

RDBMSes fit better for derived data distilled from images. Even there, however, the attendees noted that SQL is nice when the data are known to exist and to be computable, but that scientific discovery often operates where those are unknown.

The communities noted that data formats are usually driven by data producers (due to funding arrangements), causing data to be archived optimally for data storage but poorly for retrieval and analysis. Data is often stored without sufficient metadata, which makes it impossible to interpret the data. The typical non-uniformity of catalogs and inconsistent level of services make it difficult to find data. Finally, domain-specific data structures and formats further complicate data use.

3.4 Data integration

Data sets often need integration before they can be used. All represented communities noted this need in some capacity, with geoscience having the most sophisticated need.

Geoscientific data can come from a large variety of sources: ground observatories, mobile stations, sensor
networks, aerial observers, simulation models, etc. The set of data for a given location may have different resolutions, different sample rates, different perspectives, or different coordinate systems and therefore must be transformed, regridded, aligned, and otherwise unified before they can be analyzed. Current tools provide virtualization and aggregation services, but they are spotty and insufficient. As a result, sophisticated integration is generally rare, though the resultant value is tremendous.

3.5 Data calibration and metadata

One of the problems brought up by all scientific users was the issue of data calibration. The calibration parameters of an instrument are essential to correctly interpret its data. Calibration data usually fits the relational data model and is often heavily indexed to enable complex analysis; thus managing it is a database issue.

In HEP, the data set containing calibration and configuration information is the hardest to efficiently organize and provide efficient access to, despite the fact that it is less than 0.05% of the bulk data set.

The geoscience community strongly argued that their calibration data may be unreliable and evolve over time, forcing the community to preserve all raw data, some of which could have been discarded otherwise. An example was the initially incorrect image placement of hurricane Katrina which occurred due to coordinate misalignment.

In biology, the main problem is less related to “calibration” and more to lack of sufficient languages and tools that unambiguously describe metadata. There can be ambiguity even when using the same words to describe actions, areas, and states, such as “which part of an animal was cut out” or “what stage of a disease a given piece exhibits.” These natural-language descriptions also become cumbersome in large data volumes.

By contrast, in astronomy, dedicated calibration data are rarer. Instead, astronomical data are self-calibrated, where the data itself are used to do calibration. That is only possible because the sky is to a large extent “empty.” Scientific users also pointed out the ever-changing nature of metadata, noting that good research often leads to new metadata.

3.6 Data consistency and quality

Scientific and industrial users noticed new challenges in assessing data quality and maintaining data consistency at the petascale. Referential integrity was insufficient – data from multiple sources may be individually good but collectively inconsistent. While some users have attempted analysis of non-quality-checked data, the “slightly garbaged” results were unacceptable. Users also reported that postponing quality assurance in system development was always a bad idea and significantly increased cost: Nokia reported that not treating QA sufficiently seriously resulted in a cost increase of 40% plus delays of millions of dollars per month.

There is a tendency to abandon strict consistency checking and ACID enforcement in the petascale regime. The biology representatives reported that data often have no built-in methods (not even checksums or hashes) to verify integrity, and petascale data means that disk writes with extremely small error rates (e.g. 10-15) can still be untrustworthy.

Visual image inspection cannot be used for petabytes of data, so scientists must develop new solutions to fully automate data QA. Such solutions often require sophisticated techniques such as machine learning.
3.7 Data preservation

Data preservation is an important topic. Old data are needed to determine baselines and understand long term variability. In some cases, scientists have tried to go back as far as data stored on VHS tapes since they may contain scientifically important information. Industry users noted that old data are often useful for forensic analyses.

Often, the real value of a given data set is not known until much later. Insufficient funding often prevents analyzing all collected data – only about 0.5% of all collected geoscience data has been examined because the community does not have funding and appropriate tools to examine the rest.

Numerous efforts to solve the problem of long term data preservation are underway, such as various study groups (e.g. see http://www.dphep.org/), focused workshops, and dedicated funding for work towards sustainable digital data preservation (e.g. the NSF's DataNet).

Adequate standards are one of the key elements of successful data preservation. There are often too many data standards, as evidenced by the joke, “the good thing about standards is there are so many.” Sometimes interpretation of otherwise pristine data is impossible due to lack of metadata – this is the case for some data gathered about the moon, for example. Therefore, well-defined, relatively static schemas such as those inside databases or in structured files such as HDF5 or FITS are important for data preservation. Long-term preservation becomes more difficult in systems such as map/reduce that may lack official schemas and have data format logic scattered throughout the analysis code.

The rapid obsolescence of electronic formats and media is of significant concern. Some users reported falling back on paper-based archives – dumping HDF metadata in ASCII and printing it on paper whose lifetime is empirically known and longer than the design life of many electronic media.

Services such as Amazon’s Simple Storage Service could be used to preserve scientific data, but funding agencies often prefer to keep data on servers they have more control over.

3.8 Custom software

Most invitations for XLDB3 were to groups underrepresented in the previous workshops, and the resulting overlap in attendance with XLDB1 or XLDB2 was small. Yet the attendees repeated a message that had been loudly voiced in the past: everybody is writing custom software and reusing little. This is primarily because there are no comprehensive petascale solutions freely available to use or reuse. Custom software built by petascale users includes entire systems like ROOT, glue software (converters, services), storage resource managers, pipelines and workflows, provenance trackers, specialized indexes, and add-on features like progress indication/estimation and query suspension.

Some amount of reuse is facilitated by User-Defined Function (UDF) APIs in databases, which allow custom analytics to bypass SQL interfaces and be inserted into servers where they can operate close to data. UDFs can be expressed in more familiar procedural languages, which simplifies complex queries. The Sloan Digital Sky Survey (SDSS) depends heavily on UDFs in nearly all of its queries.

Porting code to UDF interfaces may not be easy, however. Algorithms are often already written to different interfaces and packaged into well-tested and documented, community-approved libraries for use with tools such as IRAF or MATLAB. Frequently these tools and libraries are run "next to" a database cluster, that is, they are run on machines with
high bandwidth connections to the DBMS. A similar approach considered by some industrial users is to run internal software "next to" a map/reduce cluster rather than migrate everything into map/reduce.

It is very unlikely that the collective XLDB community has simply overlooked available tools. The Nokia representative explained, "There is not an analytics tool that we would not try. RDBMS normalized, RDBMS denormalized, Hadoop, statistical tools – we tried them all." While industries can afford to "let a thousand flowers bloom," scientific users have more limited budgets and usually decide to build custom solutions rather than risk running into a tool's limits without the ability to fix them cheaply. The consensus however is that "managing data wastes scientists' time and money," and off-the-shelf solutions are much preferred.

To overcome the re-invention problem, collaboration between key people from different domains is essential. Building such collaborations with appropriate funding may be more difficult than the technical challenge, though.

3.9 Large-scale process flow

Industrial users have built systems to perform ad hoc analytics even at extremely large scales, but these systems are generally too expensive for academics. Scientists instead have traditionally built systems that are optimized for throughput, using careful planning and coordination to handle the most intensive and bulky analyses with limited hardware and simple software.

In HEP, the undisputed leader in analytics scale, scientists are organized into strict groups. Each group vets all publications from its members and blesses data for publication only after rigorous checking of provenance, calibration, and removal of all systematic effects. A member's good idea must be individually tested and then officially re-run by the group, commonly yielding turnaround times of weeks to months. Some experiments (e.g., Alice) process with a "data train," or continuous data scan, with batch-like scheduling.

Non-HEP analyses are currently at smaller scales, though they are often more complex. They frequently require computations across related data items, whereas HEP analyses are based on statistics of (essentially) uncorrelated events.

3.10 80/20 rule

Most data-intensive users, including industry (e.g., Nokia, Facebook, eBay) and science, observed a common set of "80/20" characteristics although the exact numbers may vary:

a) less than 20% of data is accessed 80% of the time,

b) 20% of data changes all the time,

c) 20% of users consume 80% of the available resources.

Some industrial cases are more extreme. In one example, 70% of the users consumed only 2% of the available resources, so adding even a large number of this type of "simple reporting" user is still negligible. In all communities the hardest analytics were run by a very small number of experts; i.e., it is not unusual to see 2% of the users using 50% of the available resources. This disparity appears to be because the barriers to accessing and performing useful analyses on these large data sets are high.
3.11 Importance of raw data

A common theme among industrial and scientific users is the hunger for raw data. Industrial users have discovered the huge potential in mining a wide variety of logs (including website logs), which typically have been discarded. Sciences dealing with images expressed a strong interest in running complex analyses on the pixel data, not just derived data, and they would like to keep and analyze the pixels and their associated metadata in a single system.

Unfortunately, almost everybody is forced to throw away raw data. Sometimes this is justified. In HEP, only rare events are interesting – most events are well understood – but some fraction of those must still be discarded. Radio astronomy has plans to save only a small fraction of the incoming data. The biology community discards older raw data, since there is no desire to fund the archiving of the 5-6 PB generated each year, even though this makes it impossible to redo full analyses on previous samples. Industrial users discard some data (such as SMS messages) for legal reasons but still must discard other data due to cost.

3.12 Data ownership and usage

Virtually all scientists want to “own” and “control” their data. They do not want their data to live on shared remote servers. But as their data sizes increase, their working sets may no longer fit on their desktops or laptops. The customary practice is to extract or derive subsets of the data and perform deeper or more specialized analysis on a local machine. Final analysis often involves specialized tools such as MATLAB, R, or their own code. In the past this local (offline) analysis mode was also often justified by poor or intermittent network connectivity, including the difficulty of accessing remote data while traveling. Yet as scientists attempt to use data more broadly and deeply, they are coming to accept the inevitable, frequent reliance on data servers; after all, finding the answers is more important than performance.

Pushing analyses into those servers can only succeed if the servers provide better support for scientists' familiar and currently local-only tools. Merely providing interfaces to download data does not address this issue. Also, the performance of the centralized systems should approach the performance of local analysis. Finally, ownership in a shared environment requires authentication and authorization controls to protect not only data but also code, letting users work privately without worrying about being “scooped” or letting others “steal” a Nobel prize. Centralized analysis requires a paradigm shift, but nobody doubts that it will be a huge win in the end.

3.13 Funding

Funding problems were discussed extensively at XLDB1 and XLDB2. At XLDB3, the geoscience representatives pointed out a disturbing trend: while data volumes have increased, the proportion of funding devoted to data management (DM) has decreased. Geoscience currently sees less than 10% of project budgets allocated for DM, whereas best estimates are that 30% could be needed to do an adequate job. One main reason is the idea that adding funds for DM “gets in the way of doing science” by reducing more direct science funding. While some centralized data managers such as the SDSS and Space Telescope Science Institute have demonstrated their value in terms of publications, in many other areas DM has struggled to prove its return on investment. The result is a “chicken-and-egg” problem – good DM-accelerated science results are difficult to produce without DM funding, but DM funding depends on the existence of those science results.
The user communities agreed that funding is somewhat misallocated, usually emphasizing the collection and production of data sets and neglecting their usage, analysis, hosting, or storage. Furthermore, the typically distributed nature of funding leads to the ownership challenges discussed in the previous section and makes it difficult to adopt potentially more-efficient centralized DM practices.

### 3.14 Inertia

The scientific and industrial communities differ greatly in terms of inertia, or resistance to change. Industry seems much more flexible, adjusting data models and tool sets rapidly, although perhaps still slower than desired. One participant noted: “Whatever we do, however agile we are, management asks for more and more and more.” Scientists have a lot more inertia, preferring to avoid risk where the benefit (from DM) is difficult to measure, hence the prevalence of legacy software in science. Also, scientific environments are often centered on large collaborations whose use of tightly integrated tools and applications hamper change.

The ROOT system used by HEP is a good example of legacy, integrated software. It handles almost everything: data model and data storage, workflows, compression, schema evolution, histograms, statistics, and visualization. It contains much legacy code and is almost irreplaceable, for better or worse. Any new approach, such as map/reduce processing, must be carefully integrated with the ROOT format and framework.

### 3.15 Other notes

#### 3.15.1 Imbalanced systems

The XLDB3 participants worried that most systems funded and built for the science communities are poorly balanced for data-intensive scientific computing (DISC), instead being more suited for traditional high-performance computing (HPC). HPC systems are designed to process data, not move large quantities to and from storage, which may be perfect for running simulations, but bad for running large scale analytics.

When hardware is configured to provide the required I/O performance for DISC, software may become the bottleneck. When both CPU cores and disk bandwidth are plentiful, data management systems must use the available memory bandwidth well. Current systems often do not pay attention to this problem, as they are typically used in disk-bandwidth-limited configurations.

#### 3.15.2 Cloud computing

Neither industrial nor scientific users currently rely on public clouds to store or analyze extremely large data sets. Science does not even use private clouds, using grids instead, which are much less elastic. The key obstacle may be the pricing model – for huge data sizes, private storage is cheaper than paying for bandwidth to and from the cloud and incurring monthly storage fees there. Yet there are signs of clouds in the future for analytics: the University of Washington's SciFlex system is an example, and there have been frequent inquiries whether SciDB will run on a cloud as a service. In the meantime, some are considering using a private cloud for bulk data processing and offloading to a public cloud during peak load.
3.15.3 Self-management

Everybody facing petabytes of data realizes the importance of auto-tuning. The sheer number of disks needed to manage petabytes and frequently changing hot spots means that manual administration efforts (e.g., for load balancing) require too many people and are thus too expensive in terms of labor. As one participant mused: "It's about having one or fewer DBAs [database administrators], who may be amateurs."

3.15.4 Append-only

All agreed that petascale systems are (almost) all append-only – written once and never updated. There are many ways to take advantage of this feature to greatly optimize ingest throughput and improve concurrency.

3.15.5 Green computing

It is clear that the power costs of petascale computing are enormous. When many participants declared that electricity costs would soon exceed the purchase price of the computing hardware, others pointed out that this has already come true in some places. Hence petascale system design must consider power efficiency.

4 RDBMS VS. MAP/REDUCE

Many petascale users, scientific or industrial, have declined to adopt the RDBMS model for data management. The map/reduce model, on the other hand, has wide adoption within petascale industrial users and is undergoing preliminary testing (but not yet serious usage) among petascale scientific users. This section discusses how these models differ and how these two paths appear to be converging.

4.1 Key differences

4.1.1 Procedural steps vs. monolithic query

The number one difference from an application programmer's perspective is the way data is accessed. In the map/reduce (MR) world, data is accessed by a pair of functions, one that "maps" all inputs independently and one that "reduces" the results from the parallel invocations of the first. Problems can be broken down into a sequence of MR stages whose parallel components are explicit. In contrast, a DBMS forces programmers into less-natural, declarative thinking, giving them very little knowledge of or control over the flow of query execution. They must trust the query optimizer's prowess in "magically" transforming the query into a query plan. Compounding the difficulty is the optimizer's unpredictability: even one small change to a query can make its execution plan either efficient or painfully slow.

Some XLDB3 attendees reasoned that scientists, along with the engineers at Google/Facebook/Yahoo! types of companies, usually have PhDs and tend to think they can do better than an optimizer. They just "want the data and will deal with it." Databases "get in the way." Mathematicians and statisticians, who may have the most complex analytics, develop analytics procedurally and, unsurprisingly, favor the non-declarative map/reduce approach.
4.1.2 Checkpointing vs. performance

Map/Reduce systems perform frequent checkpointing: the outputs of each map step and each reduce step are saved, and users must express their tasks as sequences of MR stages. This checkpointing limits performance, but becomes critical in handling failures. In contrast, attendees pointed out that databases are built with the optimistic assumption that failures are rare: they generally checkpoint only when necessary due to resource limitations, e.g., when a sort/merge grouping algorithm is executed on a data set that won't fit into memory. Avoiding checkpointing leads to superior performance when there are no failures but much longer recovery time when failures occur. This has been shown through various studies\(^{11}\). At small scales, the assumptions made by databases are perfectly valid. In the petascale regime, thousands of disks often participate in answering a single query, making sophisticated fault-tolerance, perhaps even as far as re-observing lost data, increasingly necessary.

4.1.3 Flexibility and unstructured data support

The map/reduce paradigm treats a data set as a set of key-value pairs (sometimes with complicated values), much like an RDBMS has tables of tuples (relations). However, while the RDBMS model operates on sets, MR functions operate on a single pair at a time. The latter was thought to be more approachable for end users.

Data in databases are structured strictly in records according to well-defined schemata. Some participants noted that a database is “kind of like a prison,” and it can be a hassle to put data in and to take data out. MR is structure-agnostic, leaving interpretation to user code and thus handling both poorly-structured and highlycomplex data. Loose constraints on data allow users to get to data more quickly, bypassing schema modeling, complicated performance tuning, and database administrators.

Relational databases require queries to be expressed in SQL, possibly with the addition of user-defined functions that operate through non-standard and non-portable interfaces. MR processing steps can be programmed in any language. Indeed, MR encourages users to employ their languages of choice, leveraging whatever libraries are available.

4.1.4 Cost

Finally, the last key difference is cost. High-end database system software is very expensive, and low-end alternatives require a lot of custom code on top to be usable in large-scale environments. There are no frelicense, scalable (shared-nothing massively parallel) database systems on the market today. The maintenance cost of a large database setup is also non-negligible, and when database administrators are required, they can contribute significant overhead. At the same time, the MR world has Hadoop, a free open-source system. Hadoop's simplicity tends to lower the administration cost.

Unfortunately, MR jobs often lack optimizations employed by databases. Large tasks that can be executed on a small number of database nodes may require thousands of MR nodes to achieve comparable performance. For example, eBay employs a 96-node database cluster that routinely handles 70,000 queries per day over 6.5 petabytes of production data. The eBay team was told that a comparably performing MR cluster would require tens of thousands of nodes. Thus, in practice, MR has significant physical costs in hardware and operations (power and cooling).
4.2 Convergence

Despite their differences, the database and map/reduce communities are learning from each other and seem to be converging.

The map/reduce community has recognized that its system lacks built-in operators. Although nearly anything can be implemented in successive MR stages, there may be more efficient methods, and those methods do not need to be reinvented constantly. MR developers have also explored the addition of indexes, schemas, and other database-ish features. Some are building a complete relational database system on top of MR.

Database systems are becoming more aligned with the map/reduce processing style in two ways:

1) Every parallel shared-nothing DBMS can use the map/reduce execution style for internal processing – while often including more-efficient execution plans for certain types of queries. Though systems such as Teradata or IBM’s DB2 Parallel Edition have long supported this, a number of other vendors are building new shared-nothing-type systems. It is worth noting that these databases typically use MR-style execution for aggregation queries.

2) Databases such as Greenplum and Aster Data (soon, Teradata) have begun to explicitly support the map/reduce programming model with user-defined functions. DBMS experts have noted that supplying the MR programming model on top of an existing parallel flow engine is easy, but developing an efficient parallel flow engine is very hard. Hence it is easier for the DBMS community to build map/reduce than for the map/reduce community to add full DBMS functionality.

5 SOLUTION PROVIDERS

XLDB3 included unstructured talks from solution providers on various topics. This section describes what they presented.

All presenting solution providers were interested in working with the scientific community.

5.1 MonetDB

The MonetDB team has demonstrated significant interest in scientific data applications. Their successful port of the SDSS multi-terabyte database to MonetDB was remarkable in light of several other unsuccessful attempts to port SDSS data to other systems. They noted that the process required some changes to their SQL dialect for compatibility with SDSS's dialect such as adding a “top-k” function, removing 32-bit limits, adding custom spatial indexing, and adding support for UDFs. They changed the SDSS-provided queries by generalizing SQL Server-specific portions, in particular rewriting some spatial index C# code in pure SQL. Bulk data migration was facilitated by a special SQL Server-MonetDB connector.

Other notable aspects of MonetDB include its application of compiler optimization techniques to SQL, such as instruction parameter matching (common subexpression elimination) and instruction subsumption (strength reduction).
5.2 Cloudera

Cloudera provides enterprise-level support for Hadoop, the popular open source implementation of map/reduce. They employ 15 of the 200+ Hadoop development team and are extending Hadoop to support columnar storage. Cloudera cited their experience working with scientific communities to ease adoption of the map/reduce model and expressed interest in continuing such activities.

5.3 Teradata

Teradata reported work in several areas that specifically address petascale needs. First, their automatic tuning and management features mean that no additional DBAs are needed as data scales up. They noted that poor query optimizations should be treated as bugs and not a DBA tuning task – no optimizer hints should be needed. Second, their storage virtualization layer for heterogeneous systems provides automatic migration of data to faster or slower storage based on “temperature” or frequency of access – hot data moves to fast storage and cold to slower/cheaper. Some participants noted that this technique is purely reactive and that human intervention can be proactive to help optimize the system for known future workloads.

5.4 Greenplum

Greenplum claimed to have a customer with the largest database installation – 6.5 petabytes on 96 nodes. They report one of the customer's biggest problems was the infrastructure proliferation problem: the maintenance of dozens of data marts and frequent new data mart births at a single site. Ideally, the system would support an “elastic infrastructure” in which a large pool of servers could be remapped to different warehouses dynamically and centrally. Greenplum expects another release later this year to include columnar storage, external table support, additional indexing support, and parallel network ingest/export.

5.5 Astro-WISE

The Astronomical Wide-field Imaging System for Europe, or Astro-WISE, was built specifically to manage scientific data sets. Though astronomy-focused initially, it is now used for some non-astronomy projects. Astro-WISE stores pixel data in file servers and metadata in a database, mediating all I/O through the database. Instead of periodic data releases, Astro-WISE reprocesses data dynamically, which trades off access latency in favor of flexibility. Process-provenance tracking is facilitated by restricting processing to uploaded and registered executables, making this a comprehensive, integrated system somewhat like ROOT.

5.6 SciDB

The SciDB team built and demonstrated a from-scratch prototype system. Their system was notable for its ndimensional array (with nesting) data model.

Their prototype stores data in compressed chunks distributed across multiple nodes. Queries were executed in parallel on heterogeneous nodes, using a scatter/gather technique for data redistribution. Their UDF model provides array-level access to cells. The demonstration illustrated scientific tasks such as object detection and regridding on raw pixel data and also showed a more relational filtering operation, confirming that tables map easily onto arrays. They pointed out the
natural fit of gridded raw data to an array model, in contrast to a relational model where the overhead of storing dimensions would be prohibitive.

Future SciDB development will produce an alpha version more suitable for early-adopter experimentation and open development. The release will include source, binaries, and documentation and is scheduled for the end of March 2010.

## 6 SCIENCE BENCHMARK

### 6.1 Concept

XLDB2 introduced the concept of a science benchmark as a means to validate and compare solutions for large scale scientific analytics. The benchmark specification was to be derived from existing practice and would not be created just to demonstrate software capabilities. It was noted that the existing TPC benchmarks were overly complex and micro-optimized by vendors and therefore were no longer good examples.

Michael Stonebraker drafted the first specification. It covered raw data processing (pixel or gridded data processing), derived data analytics in an array model context, and was intended to simulate astronomy and geoscience usage. Specifically, the benchmark included “cooking” observations from raw images, matching observations between images, finding near neighbors, and time-series analysis.

### 6.2 Discussion

XLDB3 participants believed that the benchmark should be comprehensive, representing important large-data problems from a variety of science domains, but admitted this was very ambitious and would be best achieved in multiple, focused stages. They fully agreed that the array data model coverage was a very good approach as a first stage.

The current draft of the benchmark was mostly composed of descriptive text. Participants agreed that a releasable benchmark would need extra scaffolding, such as more precise mathematical requirements, rules for query execution, lists of acceptable and unacceptable configurations (e.g., permitted degree of replication), correct answers to queries, failure modes, and details on data loading. The benchmark also needed a review from a broader community of those facing array-data-related problems.

Participants suggested the formation of a dedicated working group for future benchmark development and identified a few candidates for that group. Many suggested that a “challenge paper” describing various scientific data analytics needs would be more appropriate as a starting point, given that the tasks themselves are not commonly understood in the petascale community. Such a paper would not need the detail and rigor that may be expected in a benchmark.

## 7 NEXT STEPS

### 7.1 XLDB’s focus

Some participants suggested the scope of XLDB should be extended beyond “databases,” covering broader topics related to data management, such as file management, workflows, and pipelines. Others remarked that XLDB should
“scale down to include people who are turned away or afraid of petabytes.” There was some disagreement over whether XLDB should be distinct from or tightly integrated with the SciDB project it spawned. The organizers will consider these debates and poll the community through the XLDB mailing list. The current thinking seems to be that it is better to stay tightly focused than to try to be more inclusive and risk slowing progress.

7.2 Reaching out

Each XLDB iteration has targeted different communities, yet there are still some that have never attended. These communities include oil and gas research, chemistry, medical informatics, banking and consumer credit, as well as the military. Geographically, Asia and the Southern Hemisphere have not yet participated significantly. Some communities (e.g., chemistry) may not have extremely large data sets, although they may experience complex issues in medium-scale data analytics.

7.3 Collecting use cases

The participants unanimously agreed that use cases from diverse science domains would best illuminate the needs of different communities. Industrial use cases are also useful although many come with secrecy constraints that limit their value. Representatives from biology, geoscience, and HEP agreed to provide their use cases within three months following this workshop. Collecting and publicizing use cases has been one of XLDB’s most valuable services, and summarizing them into challenges/benchmarks would be “even better.”

7.4 Funding opportunities

XLDB3 participants discussed new funding opportunities from the European Commission. The EU has recognized the emergence of “big-data science” and has allocated new funds for “e-Science” and “e- Infrastructures.” The XLDB community was advised to consider writing a ~€200k “FP7-INTEGRATION” proposal encompassing activities such as finding common roadmaps, forging international partnerships, and organizing workshops and meetings. The proposal could provide seed money for a bigger (multi-million Euro) “FP7-INFRASTRUCTURES” proposal. FP7 proposals are intended for international software development for science, which would include development of a large-scale science analytics platform. Proposal acceptance requires participation from at least three large European scientific laboratories and strongly favors US collaboration. The European Commission works with US funding agencies in the event of such international efforts. This year’s submission deadline is 24 November.

Participants noted that the preparation of such a proposal would be complex and non-trivial. Some believed that the presence of funding would change the nature of the workshop and its participants, citing cases where financial tensions have destroyed well-functioning communities. For these reasons, the XLDB3 organizers and others interested will perform appropriate research and exercise extra care before arranging what would be a complex, multinational funding structure.

7.5 Publicity

XLDB3 participants discussed how to publicize the petascale data problem. While means such as wikis, blogs, Twitter, or other social media may prove effective, they require dedicated evangelists. None of the participants could perform this role due to other responsibilities. Since the XLDB series has always operated on donated resources, dedicated funding may help. The XLDB wiki, set up shortly after XLDB1, has not been active. Even some appropriate
documents, such as collected use cases, have been posted elsewhere due to low interest in the wiki. Some suggested presenting XLDB at SC16 although this would be difficult given XLDB's present as-needed donation funding situation.

7.6 XLDB4

It was decided that XLDB4 would be held in the United States about one year after XLDB3. Although XLDB2's recommendation was to reach out to Asia after Europe, participants agreed to maintain American momentum with a US venue. In particular, a Silicon Valley location would help attract industrial users for whom overseas travel is difficult. The vast majority of XLDB3 participants came for XLDB and not for the adjoining VLDB'09 conference. Returning attendees found that XLDB "gets better each year" and rated this instance the most interactive of all with the most productive, lively discussions.

Participants strongly believed that industry attendance was insufficient and should be increased. Vendor presentations were highly appreciated, and participants looked forward to hearing about other vendors' plans at future XLDB events.

ACKNOWLEDGMENTS

- XLDB3 was sponsored by Microsoft, Teradata, and eBay.
- The organizers are grateful for the assistance of the local VLDB organizers, in particular Mohand-Said Hacid, for help with arranging the logistics in Lyon.
- The XLDB3 was organized by a committee consisting of Jacek Becla (SLAC, chair), Kian-Tat Lim (SLAC), Martin Kersten (CWI), Dirk Duellmann (CERN), and Maria Girone (CERN).

Footnotes

1 Website: http://www-conf.slac.stanford.edu/xldb09

2 Such as LHC in HEP, LSST in optical astronomy, and SKA in radio astronomy.

3 It started shortly after the workshop, before this report was written.

4 http://opendap.org

5 http://aws.amazon.com/s3/
for a list of SDSS UDFs, see functions and procedures at http://cas.sdss.org/dr7/en/help/browser/browser.asp

They often had to spend resources to obtain the raw data.

University of Nebraska-Lincoln and Caltech store simulated LHC/CMS data in Hadoop Distributed File System; anecdotal experimental usage by geoscientists and biologists; research usage by computer scientists.

See e.g. A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, A Comparison of Approaches to Large-Scale Data Analysis, Proceedings of the 2009 ACM SIGMOD

An example of that is Hive, http://hadoop.apache.org/hive/

HadoopDB http://db.cs.yale.edu/hadoopdb/hadoopdb.html

ParAccel, Vertica, Aster Data, Greenplum, DATA Allegro (now part of Microsoft), Dataupia, Exasol, SciDB, ...

The entire XLDB effort so far has been executed through in-kind resources, plus help from sponsors whose funds were used towards funding catering at the workshops and associated social events.

International Conference for High Performance Computing, Networking, Storage and Analysis
ABSTRACT

The complexity and sophistication of large scale analytics in science and industry have advanced dramatically in recent years. Analysts are struggling to use complex techniques such as time series analysis and classification algorithms because their familiar, powerful tools are not scalable and cannot effectively use scalable database systems. The 2nd Extremely Large Databases (XLDB) workshop was organized to understand these issues, examine their implications, and brainstorm possible solutions. The design of a new open source science database, SciDB that emerged from the first workshop in this series was also debated. This paper is the final report of the discussions and activities at this workshop.

Keywords: Analytics, Database, Petascale, Exascale, Very large databases, Extremely large databases

1 EXECUTIVE SUMMARY

The 2nd Extremely Large Databases (XLDB) workshop focused on complex analytics at extreme scale. Participants represented database-intensive scientific and industrial applications, database researchers and DBMS vendors. Complex analytics. Many examples of complex analytical tasks were described. Industrial applications were often in the area of finding and understanding patterns in customer behavior. These industrial analyses frequently use techniques similar to analyses run by scientists for discovering patterns and outliers, such as time series analysis and classification.
Dataset sizes are growing dramatically, and the growth rate is increasing. The largest projects are now adding tens of petabytes of new data per year. Analysis tools such as R, MATLAB and Excel are not keeping pace, forcing analysts to generate memory-sized summaries or samples instead of using all the data. Both the structure of these immense datasets and the techniques applied to them are becoming more complex as well, so XLDB systems must remain flexible in their data representation, processing, and even hardware. An exciting possibility to maximize flexibility while reducing cost, although one requiring some cultural change, is to deliver analytics as a service, using a central XLDB to support distributed and diverse analyst communities. Administrative costs must be kept from scaling with the rapidly growing data sizes, so self-adjusting systems that can keep running normally in the face of hardware faults are required.

SQL’s set orientation and low-level ODBC/JDBC interfaces increase the barriers for analysts to use databases. An array-based data model that more intuitively matches the types of data found in science and even in many industries can help break down these barriers. Integration with analytical tools and with familiar procedural languages such as C++ and IDL will also assist. The invention of new languages that more directly capture the analyst’s intent is also a possibility, although these face adoption hurdles. The procedural-oriented “MapReduce” camp and the declarative-oriented “database” camps are converging as each grows to understand the benefits of the other.

As analytics become more complex and involve ever larger data sets, the reproducibility of an analytical workflow and its results becomes very important. While provenance and reproducibility have typically been associated with science, industry is now increasingly seeing the need for this feature, which is most easily handled within the database. On the other hand, perfect reproducibility can be unduly expensive or even impossible, so the ability to optionally relax consistency guarantees is also important.

SciDB. The initial XLDB activities resulted in an effort to build a new open source science database, called SciDB. To date, the SciDB founders have identified initial partners, assembled a database research brain trust, collected detailed use cases, completed initial design, organized funding, founded a non-profit corporation and started recruiting technical talent. The SciDB design is based on a hierarchical, multi-dimensional array data model with associated array operators, including equivalents to traditional relational operations. Queries will be expressed through a parse-tree representation with expected bindings to MATLAB, C++, Python, IDL and other tools and languages. SciDB will run on incrementally scalable clusters or clouds of commodity hardware. Optionally, it will operate on “in situ” data without a formal database loading process. It will support uncertainty, provenance, named versions and other features requested by science users.

SciDB is managed through a non-profit foundation. Design is being done by the brain trust led by Mike Stonebraker and David DeWitt. Broad science involvement and the presence of some high-end commercial users have allowed the team to capture detailed requirements and use cases to help validate the initial design. The effort is supported by large industrial partners such as eBay and Microsoft. The first version of the SciDB system is expected to be available in late 2010.

Next steps. It was agreed that SciDB should remain an activity independent from XLDB. A science challenge will be created to enable various XLDB systems, including SciDB, to measure their capabilities against a common standard. The next XLDB workshop is projected to occur at CERN to take advantage of proximity of the VLDB conference in Lyon, France, in August, 2009 (http://vldb2009.org/). Its main goals will include connecting with non-US XLDB efforts and with more science disciplines.
2 ABOUT THE WORKSHOP

The 2nd Extremely Large Databases Workshop continued a series providing a forum for discussions related to extremely large databases. It was held at SLAC in Menlo Park, CA, from September 29 to 30, 2008. The main goals were to:

- continue to understand major roadblocks related to extremely large databases with an emphasis on complex analytics,
- continue bridging the gaps within the XLDB community including science, industry, database researchers and vendors,
- build the open source SciDB community.

The workshop’s website can be found at: http://www-conf.slac.stanford.edu/xldb08.

The agenda is reproduced in the Appendix.

The workshop organizing committee was composed of Jacek Becla / SLAC (chair), Kian-Tat Lim / SLAC, Celeste Matarazzo / LLNL, Mayank Bawa / Aster Data, Oliver Ratzesberger / eBay and Aparajeeta Das / LG CNS.

2.1 Participation

Participation in the workshop was by invitation only in order to keep the attendance small enough to enable interactive discussions without microphones and to ensure an appropriate balance between participants from each community.

The workshop was attended by 64 people from industry (database users and vendors), the sciences (database users) and academia (database research). Compared to the 1st workshop, the 2nd workshop had significantly higher attendance from the academic and database research communities.

The names and affiliations of all the attendees can be found on the workshop’s website.

2.2 Structure

The bulk of the time was spent in highly interactive discussions focusing on complex analytics and administrative features needed by extremely large database setups. A significant fraction of the workshop was spent on SciDB, a new database project that was initiated as a result of the 1st XLDB workshop and its follow-up activities. A small fraction of the workshop was devoted to “war stories” from CERN/LHC, Pan-STARRS, and eBay in which lessons learned from these users currently struggling with extremely large datasets were presented.

2.3 About this report

The structure of this report does not map directly to the agenda, as we attempted to capture overall themes and the main threads of discussion.

Sections 3 through 5 cover complex analytics, with emphases on examples, effects on data representation and effects on processing, respectively. Section 6 discusses SciDB. Section 7 documents the consensus on the next steps to be taken and the future of the XLDB workshops.
We have intentionally de-emphasized the names of specific projects and participants in order to draw out the commonalities and differences within and between the scientific and industrial communities.

3 COMPLEX ANALYTICS — INTRODUCTION

The focus of this workshop was intended to be on complex analytics using extremely large databases. It was pointed out that even relatively simple computations become complex when applied to peta-scale datasets. The goals were to explore beyond these ordinary statistics and aggregations to determine what the needs of sophisticated scientists and business analysts are and how these needs are affecting the structure and usage of XLDBs.

Many examples of complex analytical tasks in industry were in the area of understanding and finding patterns in customer behavior. Such patterns, or in some cases the exceptions to the patterns, can be used for many business purposes. These tasks may include targeting advertising and promotions, predicting churn, detecting spam, finding fraud, and analyzing social networks. In combination with controlled experiments, complex analytics may be used to improve products by determining the effects of a change on both behavior and revenue.

In science, similar tasks include analyzing astronomical spectra and positions; crunching high-throughput genomics and proteomics data; pulling together climate data from networks of hundreds of thousands of sensors; comparing computational simulations of the ocean, earthquakes or combustion dynamics; and sifting through the results of fusion and collider experiments. Not all of these sciences are using databases extensively today for storage of raw and derived data, but they all have large datasets in the terabyte or larger range with some collections of datasets reaching the petabyte scale. A wide variety of techniques is applied to the data, ranging from coordinate transformations on the raw data to advanced machine learning algorithms applied to derived attributes.

Scientific and industrial analytical methods overlap substantially. Both groups use statistical techniques, classification algorithms and time series analysis. Both are interested in finding outliers that do not fit patterns, while also using the data to determine those patterns.

The next two sections describe common issues across science and industry with regard to complex analytics on XLDBs. Section 4 discusses issues related to the analyst’s view of the database in terms of data representation and the query interface. Section 5 continues with topics related to the processing of data internal to the database.

4 COMPLEX ANALYTICS — DATA REPRESENTATION

4.1 Scale and cost

Not only are dataset sizes growing dramatically, but even the rate of growth seems to be increasing. Industrial dataset sizes are reaching tens of petabytes per year, with raw data in the tens of terabytes per day. Scientific dataset sizes are in the same range, with CERN planning to store 15 petabytes per year at similar daily data rates. While these are the largest databases, virtually all participants are working with existing and planned systems in the 0.1 to 10 petabyte range.

There are several approaches to scalability of data warehouse systems available from various providers, but the scalability of analysis tools may not be keeping pace. Many of the analyses that are desired are the kind that can be done by statistical tools like R or SAS, or even Excel, which is perhaps the most popular, but these can only be applied if

http://semanticommunity.info/Data_Science/Big_Data_Science_for_CODATA/Workshops_on_Extremely_Large_Databases
Updated: Sat, 19 Sep 2015 00:38:45 GMT
Powered by mindtouch
the massive data can be reduced to a set of derived statistics that fit in memory. The cost of doing analysis may actually be increasing per unit of data as algorithms get more complex and vendor licenses fail to take these scales into account.

At these sizes, maintaining complex normalized relational schemas can be difficult and expensive. Many projects have compromised by storing data in files or unstructured strings with the metadata and sometimes derived data being the only components stored in a traditional RDBMS.

It was pointed out that even free software is not free. All software entails maintenance and operations costs, in the form of contracts or personnel that can be substantial.

4.2 Complexity

Just as scale is increasing, the complexity of analyses is also increasing.

First, the structure of the data is becoming more complex, engendering similarly complex processing. Observations are acquiring a host of attributes to capture conditions that cannot be reconstructed; storing transient search results for Internet queries or postage-stamp images for astronomical detections are examples of this. Time series, for which the order and spacing of events are significant, are becoming prominent. Much scientific data and increasing amounts of industrial data have spatial attributes, so multi-point correlations in space and time are needed. Uncertain data adds uncertainty ranges and interval calculations to queries, further increasing complexity.

Second, the analyses themselves are becoming more complex. Transformations between coordinate systems (re-gridding) may be required. In many cases, well-understood derived data products such as key performance metrics are being produced by highly-optimized processes to allow simple tools to do basic analysis, but this is insufficient for today's analysts, who need to be able to explore the data in more detail, integrating raw data with the derived data in an ad hoc manner, in order to discover new patterns and new metrics.

4.3 Flexibility

Complete, accurate requirements are rarely known ahead of time for any usage of complex analytics. Scientists often do not know exactly what they are looking for. Industrial needs can rapidly change. Analytical systems must accordingly be built very flexibly in order to handle unknown requirements. In many ways, the new, unprecedented analysis is precisely what XLDBs are designed to enable.

As the data being stored become more complex and more structured, their variability also increases. Accordingly, schema flexibility, in particular the ability to add new data attributes easily and cheaply, is a critical aspect of XLDB operations. These new attributes may stem from the raw data, from new derived metrics, or from end-user annotations.

Capabilities of systems at the leading edge of technology are further enhanced as that technology advances. New developments such as multi-core computing, access to large amounts of RAM, flash storage, fast networks, or vastly increased disk I/O bandwidth can enable radically different analytical techniques. The systems must be able to cope with the introduction of these new methods over time.

4.4 Data models

A variety of data models could be used to represent information in XLDBs. Possibilities include relational tables, objects, streams, arrays, graphs, meshes, strings (e.g. of DNA or amino acid sequence), unstructured text, and XML. None of
these is perfect for any given dataset, and it is very hard for a single system to support all of these well. The relational model has succeeded for decades as a reasonable compromise between, on one hand, flexibility and representational power and, on the other, limiting the set of operators to permit an optimizer to work well. Object databases and MapReduce systems move further to one side of the spectrum, permitting great representational capability and infinitely flexible operations at the cost of little optimization.

Many participants felt that representing data in the form of arrays could be a useful step forward beyond the relational model. First-class arrays can perform much better than arrays simulated on relational tables. Arrays are a natural, intuitive data model for many sciences, including astronomy, fusion physics, remote sensing, oceanography and seismology. Typically, a small number of (physical) dimensions is needed, often no more than four (x/y/z/time or right ascension/declination/spectral frequency/time). It will be important to support irregular or ragged arrays that have varying numbers of elements in each dimension.

Arrays inherently have ordering properties for their elements. This is of some interest to industry, which can use the ordering and spacing of array elements to represent event sequences.

Some sciences require more specialized data structures. Biology works with sequences; chemistry uses graph and network structures. Both of these could be simulated using tables or arrays, but likely at some cost.

4.5 Interfaces

The final aspect of data representation is the interface by which the analyst gains access to the data. As mentioned above, end-user statistical tools such as R, MATLAB and Excel, which can only be used with small, summary datasets today, need to be integrated with databases to make use of their exploratory capabilities while fully exploiting the scalability of the XLDB back-end.

Beyond a tools interface, science and industry both use procedural languages such as C++, IDL, and even FORTRAN in order to write advanced analytics. Interfaces to these languages that are more natural than low-level ODBC or JDBC would substantially enhance analyst productivity. Microsoft’s LINQ was mentioned as an interesting example. An alternative path is to build a new language, such as Sawzall or Pig Latin, that provides a procedural structure, but concern was expressed over whether such a language could achieve widespread adoption.

Participants recognized the need to define basic operators for manipulating arrays, especially if they are to be the basis of a new data model. They foresee that the right set of powerful primitives for tasks such as time series analysis could enable a few simple lines to replace pages of unwieldy SQL code. The specialized graph processing operations available for AT&T’s Daytona system are an example.

Furthermore, an interface must be defined internal to the database to allow new operators to be defined. This interface, which might take the form of an existing language, must allow exploitation of the parallelism of the XLDB system by moving computation to the location of the data. Techniques such as the translation of Daytona’s Cymbal language to C code for compilation and execution may be useful in this area.

5 COMPLEX ANALYTICS - PROCESSING
5.1 Architecture

For the largest-scale datasets, there is no debate that computation must be moved close to where the data resides, rather than moving the data to the computation. On a micro level, this suggests a shared-nothing style architecture, which is common to many existing XLDBs, e.g. those implemented using Teradata or the Pan-STARRS GrayWulf system.

On a macro level, there are sometimes difficulties, because, as one project put it, data must move to where the funding is. Prevailing cultural attitudes in both industry and science desire ownership of and control over data, sometimes for competitive reasons. Nevertheless, having a central shared analysis platform can provide great benefits. It avoids a proliferation of data marts, each of which requires valuable administrative resources. Providing analytics on the platform as a service seems to be an exciting way forward. It allows dynamic adjustment of usage on demand as workload changes driven by external factors such as conferences or quarterly reports. It can help maximize utilization of expensive resources while allowing the full capabilities of those resources to be devoted to important problems, thereby reducing the time to discovery. Further centralization does not preclude having rigorous access controls.

There continue to be debates between a “brute force” camp that emphasizes the MapReduce framework and full table (or column) scans and a “database” camp that emphasizes the capabilities of an optimizer. MapReduce has advantages in fault tolerance, progressive results that allow mistakes to be caught rapidly and simplicity of the programming model, but it requires programming, is more suited for batch processing than interactive queries, and can be inefficient in its use of resources. Both sides agreed that there appears to be a convergence between them, with aspects of each model being adopted by the other.

5.2 Reproducibility

As analytics become more complex, it is becoming even more important to be able to reproduce an analytical procedure and its results. This involves more than just tracking what happened or maintaining the lineage of metadata; the lineage of the data itself and the ability to use older versions of them is critical, as running an old process on updated data can generate different output. This capability also allows erroneous derived data to be tracked to its source or, vice versa, erroneous raw data to be tracked to all values derived from them. While provenance and reproducibility are typically associated with science, industry is increasingly seeing the need for this feature, in some cases due to new legal requirements.

It is easiest to capture provenance for operations occurring within the database. It is rare for all operations of interest to be handled by the database, however. In many cases, external systems are used for processing raw data sources or as “black box” computational packages. In these cases, provenance information from outside and inside the database must be joined. Loading external provenance into the database’s internal structures is one possible approach that offers a unified query capability; another approach would be to export provenance information from the database into a standard format usable by external provenance tools.

Maintaining the provenance of data at this level comes with a price. It was pointed out that XLDB systems typically have large numbers of disks in order to have enough spindles to maintain sufficient I/O bandwidth. In typical usage, only 10% or so of those disks are actually useful for “hot” data. The remainder of the disk space may be used for storing provenance and versioning information.

http://semanticommunity.info/Data_Science/Big_Data_Science_for_CODATA/Workshops_on_Extremely_Large_Databases
Updated: Sat, 19 Sep 2015 00:38:45 GMT
Powered by mindtouch
At the same time, a strong argument was made that perfect reproducibility is a mirage that is not necessary all the time, particularly during exploratory analysis. First, the computational environment outside the database such as the hardware, the operating system, compiler libraries and so forth may affect results. While this environment can be recorded, reproducing a given configuration may be unduly expensive or even impossible. Second, there may be great cost savings in relaxing the accuracy guarantees. Given the probability of failures in hardware in these large-scale systems, the ability to execute analyses on incomplete data, dropping a few elements in an unbiased fashion, may be highly desirable. Similarly, it may be possible to provide greater responsiveness (see below) and performance by relaxing the standard ACID (atomicity, consistency, isolation and durability) criteria used with transactional relational databases. In such cases where incomplete or potentially inconsistent data are used, the system must give an indication of the uncertainty in the result.

5.3 Workflows

There are several occasions in the course of a complex analysis when workflow management may be necessary. Initially, as data is loaded into a system, processing may occur to transform the data from their raw form into a suitable long-term persistent format. A “cooking” process may be used to turn raw data into more-easily-analyzed derived data. Finally the analysis itself may require multiple steps.

Tracking these workflows is essential for provenance and reproducibility, as in the previous section. Allocating resources to them in an appropriate fashion is also important. Industrial systems often have strict requirements in order to meet service level agreements, but even scientific systems must ensure fair sharing of the available resources. Managing workflows inside the database is the most powerful paradigm in terms of maintaining provenance and allowing for potential optimizations such as bringing the computation to the data. In many cases, however, bringing all of the processing inside the database will be impossible, and so the database must be able to integrate with external workflow management systems.

5.4 Responsiveness

Despite the immense scale of XLDBs, many end users need rapid response capabilities. Businesses need to react to events happening in real time, and scientists need to respond to interesting transients that may have limited observation windows. In many cases, processing raw data as they are collected or loaded into an XLDB is used to trigger these responses, but it would be desirable to bring the full power of the database system to bear on fresh data.

Another aspect of responsiveness is the ability of a system to deliver partial and progressively more accurate results. Having such a capability, rare if not nonexistent in traditional SQL RDBMSes, would allow erroneous queries to be recognized faster and correct queries to be terminated early when their accuracy reaches desired levels.

5.5 Administration

At extremely large scales, administrative costs that increase with the size of the data are unsustainable. Systems must be self-sustaining, self-healing, self-load-balancing and self-adjusting to avoid requiring a large number of database administrators. In the experience of the participants, detailed monitoring of large systems at every level is essential. Applying the power of the database to the task of analyzing its own logs is highly rewarding. The feedback loop between database performance metrics and database configuration helps to automate index management and reduce data skew,
but the loop should not totally exclude human oversight, as transient load anomalies could otherwise cause long-term
problems.

Resources used by a database need to be managed carefully. Because accurate cost estimation is often difficult due to
the presence of many correlated factors and data dependencies, it is desirable for a system to be able to catch
unproductive queries rapidly during execution and also to pause and resume queries that are temporarily exceeding
their resource quotas. This naturally leads to a fine-grained, operating-system-like priority scheme, as implemented in
several systems of workshop participants.

Fault tolerance is essential when large numbers of computers are involved in a single system. A particular concern was
failures of complex, long-running user-defined functions. While a database may handle fault tolerance in its own
operations, providing fault tolerance to user code, perhaps by including checkpointing capabilities in the programmer
interface, may be more difficult.

6 SCIDB

The 1st XLDB workshop clearly exposed the lack of shared infrastructure needed by large-scale database users. Each
data-intensive community, with very few exceptions, “rolls its own” software on top of the bare operating system. This
results in building software with almost no applicability to other projects. While data-intensive industrial users with
extensive financial resources can afford building custom solutions, such an approach is not sustainable inside science.
To address this issue, the recommendations from the 1st workshop included:

a) improving collaboration between science and database research and
b) defining common database requirements shared by different science domains.

A follow-up Science-Database mini-workshop organized at Asilomar, CA, in March of 2008 resulted in a decision to build
a new open source science database, called SciDB. Since then, the SciDB founders have identified initial partners,
assembled a database research “brain trust,” collected detailed use cases, completed an initial design, organized
funding, founded a non-profit corporation and started recruiting technical talent. This chapter highlights the initial SciDB
design and some of the aforementioned activities.

6.1 Science needs

The 1st XLDB workshop, the Science-Database mini-workshop, and the use cases supplied by the initial partners and
lighthouse customers all highlighted the fact that science database users are almost universally unhappy with relational
DBMSes. The following main reasons include many described above:

Wrong data model: science data very rarely naturally fits into a relational table-based model. The ideal model varies
somewhat by science, but in each case, it is far from pure tables, and simulating it on top of tables is extremely
inefficient.

Wrong operators: the most frequently performed operations, for example regridding or Fourier transforms are nearly
hopeless in relational DBMSes. Similarly, frequently performed complex analytics such as time series analysis are
impossible to express in SQL.
No provenance: all scientists want a DBMS to support provenance and reproducibility.

No time travel: science users must be able to reproduce published results, hence overwriting previous data as is done in current DBMSes is not an option.

Insufficient scalability: some scientific projects have already reached the petabyte level, and others are quickly approaching it. None of the existing DBMSes offers multi-petabyte level scalability. The situation worsens when the costs of licensing and, in some cases, specialized hardware are taken into account.

6.2 Design

Building a system supporting all of the features requested by science requires substantial development at the lowest levels of the system, such as the storage manager. This requirement makes it difficult to build on top of existing open source systems (DBMS or MapReduce). While it is expected that some relevant pieces of existing appropriately-licensed software can be reused, the overall design of SciDB has to be from the ground up. This section highlights the key decisions made, the technological challenges to be overcome, and future plans in the area of SciDB design.

6.3 Data model

It is clear there is no single universal data model that would make every science happy. After evaluating which models are applicable to the largest number of users, which models are most realistic to implement and which models can be easily implemented on top of other models, the array model has been selected. This model is a very good fit for most sciences, including astronomy and many branches of geoscience including oceanography, remote sensing and atmospheric sciences.

SciDB will support nested multi-dimensional arrays. Two types of arrays will be supported:

- basic arrays (MATLAB style), with integer dimensions starting with 0. Dimensions can be bounded or unbounded.
- enhanced arrays, where a user-defined shape function defines the outline of the array. Enhanced arrays can be irregular in any dimension.

Each element of an array will contain a tuple of attributes, similar to a row in a relational database. Any of the attribute values can be a nested array.

It is expected array-based data will compress well. Techniques such as storing deltas or run-length encoding will be used to provide maximum possible loss-less compression.

6.4 Query language and operators

A parse-tree representation will be used to define the SciDB query language. Bindings to commonly used tools, such as MATLAB, C++, Python, IDL and others are expected to be built. The C++ binding will likely be the first one available because of its usefulness for internal development. It is expected that the community will help with some of the additional bindings.
SciDB will support standard relational operators, such as filter or join, plus many other commonly-used array operators. The complete list of such operators has not been determined yet and will require polling different scientific communities. Frequently mentioned examples include regridding and Fourier transforms. In addition to natively supported operators, users will be able to define their own operators, PostgreSQL-style.

Some of the research topics in this area include how to represent array operations in languages like C++ or Python. On one hand it would be useful if the syntax would look similar in all languages, but on the other hand some tool languages already have well defined array-based operators.

### 6.5 Infrastructure

SciDB will run on incrementally scalable clusters or clouds of industry standard hardware, ranging from a single laptop through small clusters managed by individual projects and laboratories to very large commercial clouds. It will have built in high availability and failover and disaster recovery. Data will be partitioned across the available hardware to maximize throughput.

**In situ processing**

SciDB will optionally operate on data “in situ”, i.e. on external data not loaded into SciDB. Such data will not have some of the SciDB services such as replication or crash recovery. SciDB will provide a means for describing the contents of this type of data. In addition, adapters for the most popular array formats such as HDF-5 or NetCDF will be implemented.

**Uncertainty**

Based on input received from scientists, virtually all sciences make some use of uncertainty. Beyond that, the uncertainty-related requirements vary significantly. For that reason, SciDB will initially support the specification of uncertainty and their use in simple computations such as comparisons. Even this level of support is non-trivial to implement. More sophisticated error models may be implemented later; once more commonalities across different sciences are identified.

**Provenance**

To capture provenance, the SciDB engine will record every update it executes. In addition, it will be possible to load external provenance through special interfaces, and export of provenance will be supported. In combination with versioning features for data, it will be possible to reproduce the conditions of any operation and trace its inputs and effects. It is expected that special facilities for querying provenance will be developed.

### 6.6 Project organization and current status

The SciDB organization is based on a partnership between three groups:

Science and high-end commercial users. Their tasks include providing use cases and reviewing the design.

A database research brain trust. The tasks of this group include designing the system, performing necessary research and overseeing software development.
A non-profit foundation. The tasks of the foundation include managing the open source project and providing long-term support for the resulting system.

The initial science partners include Large Synoptic Survey Telescope/Stanford Linear Accelerator Center (now SLAC National Accelerator Laboratory) (LSST/SLAC), Pacific Northwest National Laboratory (PNNL), Lawrence Livermore National Laboratory (LLNL), and the University of California at Santa Barbara (UCSB). The first lighthouse customers are LSST and eBay. Some of these teams have already provided an initial set of use cases. A science advisory board has been created to coordinate input from science, translate use cases from scientific terminology into database terminology and prioritize the requested features.

Industrial partners include eBay, Vertica, and Microsoft.

The brain trust team members include Mike Stonebraker (MIT), David DeWitt (University of Wisconsin → Microsoft), Jignesh Patel (University of Wisconsin), Jennifer Widom (Stanford), Dave Maier (Portland State University), Stan Zdonik (Brown Institute), Sam Madden (MIT), Ugur Cetintemel (Brown University), Magda Balazinska (University of Washington) and Mike Carey (UC Irvine). The brain trust has already produced an initial design and is currently in the process of refining and improving it.

6.7 Timeline

It is expected that a demo version of SciDB system will be available in late 2009, and the first production version (“V1”) in late 2010.

6.8 Summary

The SciDB project sparked a lot of interest among workshop attendees. It was unanimously agreed that it is an ambitious project, and therefore in order for it to succeed it must initially stay focused on a minimum set of well-defined core features. It is to be built from the ground up, so users should not expect “Teradata-like bells and whistles” in one or two years — it may take a long time to get to truly good performance.

7 NEXT STEPS

The last section of the workshop was devoted to discussions about the future. The two main topics discussed were (1) the future relationship between SciDB and XLDB and (2) the future of the XLDB workshops.

7.1 SciDB and XLDB

SciDB came about as a result of the 1st XLDB workshop. It is clear there is substantial overlap between XLDB and SciDB, but it was also obvious that SciDB design and development is not appropriate for a big and diverse environment like XLDB and should thus be an independent activity. The attendees agreed that it was very valuable to devote a large fraction of this workshop to SciDB to bring the XLDB community up to date. It was also agreed that it would be very useful to keep the XLDB community aware of SciDB activities on a regular basis but that this need not be as great a proportion of the workshop time in the future.
It was suggested that the SciDB team should publish its collected use cases for other vendors to study. The attendees also advised that SciDB should reach out to more sciences.

One of the hot topics related to the future of SciDB was a measure of success. The conclusion was that some metrics of success should be defined early on and that the design should be periodically validated against such measures.

### 7.2 Science challenge

It was agreed we should define a science-oriented database challenge. Because this was envisioned to be different from benchmarks such as the TPC-x series, the word challenge was preferred over benchmark. Some believed a challenge based on the Sloan Digital Sky Survey data set and queries, which are well characterized and understood, would be a comprehensive measure. In the end, however, it was believed that this task has been overly optimized for relational engines. Mike Stonebraker and David DeWitt agreed to take on the task of defining a science challenge.

### 7.3 Wiki

The XLDB Wiki was briefly discussed. The attendees felt it should be made more visible and publicized better. An active moderator with some degree of journalistic skill will likely be required to maintain an attractive site. We will attempt to recruit one.

### 7.4 Next workshop

There was unanimous agreement that we should continue with XLDB workshops. The workshops have a very well-defined, unique focus and are a “great place for interactions” between database users and database community members (researchers and vendors). It was also pointed out there is no overlap between XLDB and other well established conferences and workshops; while other groups (such as SSDBM) have tried to create similar forums, they have never fully succeeded.

Most attendees felt that the XLDB workshops should continue as annual meetings. Two days turned out to be an ideal length. We will continue with the by-invitation-only rule to foster discussion and frank exchange. XLDB3 should continue to seek to be interactive, although it was suggested that we might introduce several presentations or papers to drive the discussion.

Some felt it might be advisable to attach the next XLDB to another database venue like SIGMOD or VLDB, however in the end the conclusion was to continue with an independent workshop. It was clear to everybody that the larger database community gathered around VLDB/SIGMOD conferences would benefit from hearing about real science requirements, and therefore we should try to organize a tutorial at the next VLDB conference, but that would not substitute for a real XLDB workshop.

Participants agreed that we should try to reach out to XLDB communities outside of the USA. Large scale, database-focused activities are happening in Europe (e.g., MonetDB), and in Asia (with Japan and China specifically mentioned). For that reason, locating XLDB3 in Europe or Asia was considered. Most felt the best location would be at CERN immediately before or after the VLDB conference which will be held in August 2009 in Lyon, France, near CERN. The XLDB3 location will be finalized once CERN officially confirms it can host the workshop. Assuming these plans hold,
Europeans Maria Girone/CERN and Martin Kersten/CWI would take charge of the organizing committee; at least one of the two key organizers of the first two XLDB workshops should also be involved.

The attendees pointed out that we have not had sufficient representation from several major scientific communities. In particular biology was underrepresented; the biology representatives present at XLDB2 were all database people who had worked with biologists.

Two important goals for XLDB3 were identified:

- Connect with non-US XLDB efforts.
- Connect with more science disciplines and communities.

XLDB3 should also include a short report from the SciDB project. Another possible agenda item mentioned was “embedded sensors and RFIDs.” One possible structure for the workshop might be to spend the first day on existing engines and solutions with a focus on systems developed in Europe and then spend the second day on discussing how to move the state of the art forward.

Participation of DBMS vendors in future XLDB workshops was encouraged; keeping the vendors aware of large scale problems and needs stimulates research and ultimately will result in more DBMSes supporting useful XLDB features.

It was also agreed that it would be beneficial to make the XLDB3 agenda available well in advance, e.g. around December 2008.

8 ACKNOWLEDGMENTS

The organizers gratefully acknowledge support from the sponsors:

- eBay
- Greenplum
- Facebook
- LSST Corporation.

9 GLOSSARY

CERN – The European Organization for Nuclear Research
DBMS – Database Management Systems
HEP – High Energy Physics
LHC – Large Hadron Collider
LLNL - Lawrence Livermore National Laboratory
LSST – Large Synoptic Survey Telescope
MIT – Massachusetts Institute of Technology
Pan-STARRS – Panoramic Survey Telescope & Rapid Response System
RDBMS – Relational Database Management System
SDSS – Sloan Digital Sky Survey
SLAC – SLAC National Accelerator Laboratory, previously known as Stanford Linear Accelerator Center
2008: Report from the First Workshop on Extremely Large Databases

Source: https://www.jstage.jst.go.jp/article...becla0223/ pdf (PDF)

REPORT FROM THE FIRST WORKSHOP ON EXTREMELY LARGE DATABASES
J Becla*1 and K-T Lim 2
Stanford Linear Accelerator Center, Menlo Park, CA 94025, USA
*1 Email: becla@slac.stanford.edu
2 Email: ktl@slac.stanford.edu

ABSTRACT

Industrial and scientific datasets have been growing enormously in size and complexity in recent years. The largest transactional databases and data warehouses can no longer be hosted cost-effectively in off-the-shelf commercial database management systems. There are other forums for discussing databases and data warehouses, but they typically deal with problems occurring at smaller scales and do not always focus on practical solutions or influencing DBMS vendors. Given the relatively small (but highly influential and growing) number of users with these databases and the relatively small number of opportunities to exchange practical information related to DBMSes at extremely large scale, a workshop on extremely large databases was organized. This paper is the final report of the discussions and activities at the workshop.

Keywords: Database, XLDB

1 EXECUTIVE SUMMARY

The workshop was organized to provide a forum for discussions focused on issues pertaining to extremely large databases. Participants represented a broad range of scientific and industrial database-intensive applications, DBMS vendors, and academia.

The vast majority of discussed systems ranged from hundreds of terabytes to tens of petabytes, and yet still most of the potentially valuable data was discarded due to scalability limits and prohibitive costs. It appears that industrial data warehouses have significantly surpassed science in sheer data volume.

Substantial commonalities were observed within and between the scientific and industrial communities in the use of extremely large databases. These included requirements for pattern discovery, multidimensional aggregation, unpredictable query load, and a procedural language to express complex analyses. The main differences were the availability requirements (very high in industry), data distribution complexity (greater in science due to large
collaborations), project longevity (decades in science vs. quarter-to-quarter pace in industry) and use of compression (industry compresses and science doesn't). Both communities are moving towards parallel, shared-nothing architectures on large clusters of commodity hardware, with the map/reduce paradigm as the leading processing model. Overall, it was agreed that both industry and science are increasingly data-intensive and thus are pushing the limits of databases, with industry leading the scale and science leading the complexity of data analysis.

Some non-technical roadblocks discussed included funding problems and disconnects between vendors and users, within the science community, and between academia and science. Computing in science is seriously under-funded: the scientific community is trying to solve problems of scale and complexity similar to industrial problems, but with much smaller teams. Database research is under-funded too. Investments by RDBMS vendors in providing scalable multi-petabyte solutions have not yet produced concrete results. Science rebuilds rather than reuses software and has not yet come up with a set of common requirements. It was agreed there is great potential for the academic, industry, science, and vendor communities to work together in the field of extremely large database technology once the funding and sociological issues are at least partly overcome.

Major trends in large database systems and expectations for the future were discussed. The gap between the system sizes desired by users and those supported cost-effectively by the leading database vendors is widening. Extremely large database users are moving towards the use of solutions incorporating lightweight, flexible, specialized components with open interfaces that can be easily mixed and matched with low-cost, commodity hardware. The existing monolithic RDBMS systems face a potential redesign to move in this direction. Structured and unstructured data are coming together; the textbook approach assuming a perfect schema and clean data is inadequate. The map/reduce paradigm popular in many places lacks efficient join algorithms, and therefore it is likely not the ultimate solution. Recent hardware trends will disrupt database technology, in particular the increasing gap between CPU and I/O capability, and emerging solid-state technologies.

Next steps were discussed. It was agreed that collaborating would be very useful. There should be follow-up workshops, and perhaps smaller working groups should be set up. Defining a standard benchmark focused on data-intensive queries and sharing infrastructure ranging from testbed environments to a wiki for publishing information were also highly desired.

### 2 About the Workshop

The Extremely Large Database (XLDB) Workshop was organized to provide a forum for discussions focused specifically on issues pertaining to extremely large databases. It was held at SLAC (See the Glossary for definitions of abbreviations used in this document). on October 25, 2007. The main goals were to:

- identify trends and major roadblocks related to building extremely large databases,
- bridge the gap between users trying to build extremely large databases and database vendors,
- understand if and how open source projects such as the LSST Database can contribute to the previous two goals in the next few years.

The workshop’s website can be found at: [http://www-conf.slac.stanford.edu/xldb0](http://www-conf.slac.stanford.edu/xldb0).

The agenda is reproduced in Appendix A.
The workshop organizing committee was composed of Jacek Becla (chair), Kian-Tat Lim, Andrew Hanushevsky and Richard Mount.

2.1 Participation

Participation in the workshop was by invitation only in order to keep the attendance small enough to enable interactive discussions without microphones and to insure an appropriate balance between participants from each community. Based on feedback from participants and the workshop outcome, this strategy turned out to be very successful.

The workshop was attended by 55 people from industry (database users and vendors), the scientific community (database users) and academia (database research). During the panel discussions, the XLDB user community from industry was represented by AOL, AT&T, EBay, Google and Yahoo!. The database vendors present included Greenplum, IBM, Microsoft, MySQL, Netezza, Objectivity, Oracle, Teradata and Vertica. Academia was represented by Prof. David DeWitt from the University of Wisconsin and Prof. Michael Stonebraker from M.I.T. The scientific community had representation from CERN, the Institute for Astronomy at the University of Hawaii, IPAC, George Mason University, JHU, LLNL, LSST Corp., NCSA, ORNL, PNL, SDSS, SLAC, U.C. Davis and U.C. Santa Cruz.

The user group was selected to represent a broad range of database applications. On the industrial side, these ranged from search engines (Google, Yahoo!) and web portals (AOL) through on-line bidding systems (EBay) and telecom (AT&T). On the scientific side these ranged from high energy physics (LHC, BaBar) and astronomy (SDSS, PanSTARRS, LSST, 2MASS) through complex biological systems. The names and affiliations of all the attendees can be found in Appendix B.

2.2 Structure

The workshop was held on a single day to facilitate attendance and encourage constructive discussion. The bulk of the time was spent in highly interactive panel sessions, although there were two initial presentations from the two largest scientific projects: LHC representing today’s usage and high energy physics and LSST representing future usage and astronomy. The agenda was divided into three parts:

- user panels from the scientific and industrial communities that were meant to reveal how extremely large databases are used, how they have been implemented, and how the users would like to use them
- vendor and academic responses to the issues heard
- discussion about the future and possible next steps.

2.3 About This Report

The structure of this report does not map directly to the panel organization, as we attempted to capture overall themes and the main threads of discussion.

Section 3 shows how extremely large database systems are used in production now and discusses current technological solutions and problems. Section 4 describes issues related to collaboration among the various groups interested in XLDB. Section 5 summarizes participants’ thoughts on the future evolution of XLDB. Section 6 documents the consensus on steps that should be taken next.
We have intentionally de-emphasized the names of specific projects and participants in order to draw out the commonalities and differences within and between the scientific and industrial communities. The Facts in Appendix C give some specifics.

## 3 TODAY’S SOLUTIONS

This chapter describes the current state of extremely large database technology and practice in both science and industry, as revealed through the panels and discussions at the workshop.

### 3.1 Scale

The workshop was intended to discuss extremely large databases. Unsurprisingly, the vast majority of the systems discussed have database components over 100 terabytes in size, with 20% of the scientific systems larger than 1 petabyte. All of the industry representatives had more than 10 petabytes of data, and their largest individual systems are all at least 1 petabyte in size.

Size is measured in more than just bytes, however. Industry systems already contain single tables with more than a trillion rows. Science has tables with tens of billions of rows today; multi-trillion-row tables will be required in less than ten years.

Peak ingest rates range as high as one billion rows per hour, with billions of rows a day common. All users said that even though their databases are already growing rapidly, they would store even more data in databases if it were affordable. Estimates of the potential ranged from ten to one hundred times current usage. The participants unanimously agreed that “no vendor meets our database needs”.

### 3.2 Usage

The largest databases described are variations on the traditional data warehouse used for analytics. Common characteristics include a write-once, read-many model; no need for transactions; and the need for simultaneous load and query to provide low-latency access to valuable fresh data. These analytical systems are typically separate from the operational, often on-line transaction processing (OLTP), systems that produce the data.

Representatives from both science and industry described highly unpredictable query loads, with up to 90% of queries being new. An incisive phrase used to describe this was “design for the unknown query”. Nevertheless, there are common characteristics here, too. Most of the load involves summary or aggregative queries spanning large fractions of the database. Multidimensional queries involving value or range predicates on varying subsets of a large set of attributes are common. While science has been able to develop specialized indexes in some cases, the lack of good general index support for these queries leads to the frequent use of full table scans. To deal with the variability, it was suggested that well-designed systems would monitor their own usage and allow adaptation to the current query load.

As mentioned above, not all the data required by the various projects represented are going into databases today. Some are being discarded, but other data that are less queriable and do not require other database features such as transactions are being managed in alternative, cheaper, more scalable ways like file systems. Most scientific users and a few industrial users store aggregate data in a database while leaving the detailed information outside. There are other
uses of databases, of course. Critical transaction-oriented data management needs are almost universally handled by databases, typically using an off-the-shelf RDBMS. Operational data stores requiring low latency and high availability but with strictly defined query sets were also described. These other uses tend not to be the very largest databases (although they might be the metadata for the largest custom-designed databases). These many uses often require the use of multiple database packages, with each used in its area of expertise. For example, an off-the-shelf RDBMS might be used as the source of data for a custom map/reduce analysis system, or, inverting the flow, an off-the-shelf RDBMS might be used to hold quick-access aggregates from a different system.

3.3 Hardware and Compression

In order to achieve the scalability necessary to handle these extremely large database sizes, parallelism is essential. I/O throughput was often mentioned as being more important than raw CPU power, although both groups, especially science, have some complex CPU-intensive processing. Most systems use a shared-nothing architecture in which the data was typically partitioned horizontally by row. The total number of nodes used in parallel in a single cluster range as high as tens of thousands for industry, thousands for science.

Increasing the number of nodes exponentially increases the chance of failure. In fact, hardware failures are so common in large systems that they must be treated as a normal case. Today’s predominant solution for these clusters is to handle hardware failures through software, rather than relying on high-end hardware to provide high availability. Once software-based, transparent failure recovery is put into place, modestly greater failure probabilities (as high as 4-7% per year for disks in one report) do not affect the overall system availability. Many projects in both science and industry are thus able to use low-end commodity hardware instead of traditional “big iron” shared memory servers. This approach is also typically associated with the use of local disks, rather than network-attached storage or SAN. The number of disk drive spindles is seen as being more important than the total amount of storage, dramatically so for random-access systems. One of the advantages of map/reduce type systems is that they decrease the need for random access.

These large systems also require a large amount of power, cooling, and floor space; the database component of the system was cited as often being one of the largest contributors, as disk arrays can produce large amounts of heat. One way of conserving both disk space and I/O bandwidth is to compress the data. Everyone in industry is compressing their data in some fashion. The scientific projects, on the other hand, do not typically compress their data, as the structure and composition of the data apparently do not permit compression ratios sufficient to justify the CPU required.

3.4 SQL and the Relational Model

The relational model and the SQL query language have been very successful since the “Great Debate” many years ago. Keeping the data and processing models simple has given them great power to be used in many diverse situations. Many users felt constricted by the implementations available in off-the-shelf RDBMS packages, however.

Often industrial data originates in fully-normalized OLTP systems but then is extracted into denormalized or even semi-structured systems for analysis; it is rare for scientific data (but not metadata) to ever be stored in fully-normalized form. The poor performance of general-purpose billion-to-trillion-row join operations has led users to pre-join data for analysis or use the map/reduce paradigm\(^1\) for handling simple, well-distributed joins. With full table scans being common, as mentioned above, indexing is of lower priority, although the partitioning scheme is inherently a first-level index. Column-oriented stores are playing an increasing role in these large systems because they can significantly reduce I/O requirements for typical queries.
Procedural languages are required by both science and industry for processing and analyzing data. In industry, higher-level languages such as Sawzall2, Pig3, or the Ab Initio ETL4 tool were mentioned. In science, lower-level languages such as C++ tend to be used.

When industry does use SQL for analytics, the queries are most often generated by tools, rather than hand-coded, with one project reporting as many as 90% of queries being tool-driven. In science, hand-coded queries are common, with even lower-level programmer-level access frequent, as query-generating tools have been tried and generally found insufficient.

Object-relational adapters and object-oriented databases are not felt to be appropriate, at the current level of development, for these extremely large databases, particularly the warehouse-style ones.

Overcoming the barriers to mapping scientific data into the relational model or developing useful new abstractions for scientific data and processing models are seen as necessities to allow scaling of both hardware and people as new extremely large databases are built. Jim Gray worked hard to overcome this barrier, successfully showing in several cases that science can use relational systems.

3.5 Operations

Industrial systems typically require high availability, even during loading and backup. These systems are often integrated into essential business processes. Science, on the other hand, can often deal with availabilities as low as 98%, although not if the downtime interrupts long-running complex queries.

Smaller databases that are components of these systems may require more than just high availability: they may also need to provide real-time or near-real-time response. Examples occur in both science, where detector output must be captured, and industry, where immediate feedback can translate directly into revenue. One industrial project processes as much as 25 terabytes of streaming data per day. On the other hand, some scientific databases may be released as infrequently as once per year, preceded by reprocessing with the latest algorithms and intensive quality assurance processes.

Manageability of these systems was deemed important. Neither group can afford to have a large number of database administrators, or to scale that number with the size of the system.

Replication of data and distribution of systems across multiple sites, sometimes on a worldwide basis, is necessary for both groups to maintain availability and performance, but it does add to the manageability headaches. Science is particularly impacted here, as its collaborations often involve tens to hundreds of autonomous partners around the globe operating in radically different hardware, software, network and cultural environments with varying levels of expertise, while industry can exert firmer control over configurations.

3.6 Software

While industry may have more resources than science, both do not want to pay more than necessary for database solutions. Both groups often use free and/or open source software such as Linux, MySQL, and PostgreSQL extensively to reduce costs. Both groups also write custom software, although at different levels. Industry tends to implement custom scalable infrastructure, including map/reduce frameworks and column-oriented databases, which provides
abstraction for programmers and eventually analysts. Science tends to implement custom top-to-bottom analysis, utilizing minimal data access layers that nevertheless provide isolation from underlying storage to enable implementation on environments that may be heterogeneous across varying locations and across the lifetime of longrunning projects.

Stereotypically, industry is fast-moving, operating on a quarter-to-quarter or month-to-month basis, while science moves at a slower pace with decade-long projects. In actuality, industry needs to amortize infrastructure development over multi-year timeframes, while science needs to plan to upgrade or replace technology in midproject. Software in both cases is continually evolving with new requirements and features.

All participants expressed a need for performing substantial computation on data in databases, not just retrieving it. Extracting patterns from large amounts of data and finding anomalies in such data sets are key drivers for both groups. These analytic, discovery-oriented tasks require near-interactive responses as hypotheses must be repeatedly tested, refined, and verified. The algorithms required to generate useful attributes from scientific data today are often orders of magnitude more computationally expensive, particularly in floating point operations, than those found in industry. As a result, science tends to do more pre-computation of attributes and aggregates than industry, though this slows down the discovery process.

The map/reduce paradigm has built substantial mind-share thanks to its relatively simple processing model, easy scalability, and fault tolerance. It fits well with the aforementioned need for full table scans. It was pointed out that the join capabilities of this model are limited, with sort/merge being the primary large-scale method being used today.

3.7 Conclusions

There are substantial commonalities within and between the scientific and industrial communities in the use of extremely large databases. Science has always produced large volumes of data, but contemporary science in many different fields of study is becoming increasingly data-intensive, with greater needs for processing, searching, and analyzing these large data sets. Databases have been playing an increasing role as a result. Industrial data warehouses have surpassed science in sheer volume of data, perhaps by as much as a factor of ten.

The types of queries used by industry and science also exhibit similarities, with multidimensional aggregation and pattern discovery common to both. The overall complexity of business analyses is still catching up to that of science, however.

The relational model is still relevant for organizing these extremely large databases, although industry is stretching it and science is struggling to fit its complex data structures into it.

Both communities are moving towards parallel, shared-nothing architectures on large clusters of commodity hardware, with the map/reduce paradigm as the leading processing model.

4 COLLABORATION ISSUES

This chapter summarizes non-technological roadblocks and problems hindering the building of extremely large database systems, including sociological and funding problems, as revealed through the panels and discussions at the workshop. In some cases possible solutions were discussed.
4.1 Vendor/User Disconnects

The scale of databases required in both science and industry has been increasing faster than even the best RDBMS vendors can handle. Vendors have been working with large customers, learning how best to apply existing technologies to the extremely large scale and discovering requirements for the future, but they are still generally seen as not keeping up. The overall feeling was that they may be building on lagging indicators, instead of leading, resulting in their creating solutions for yesterday’s problems. The perception of users was that “existing RDBMSes could now easily solve all the problems we had five years ago, but not today’s problems”. One possible explanation that was mentioned is insufficient exposure of the database vendors to real-world, large scale production problems; there was even a suggestion that vendor representatives should rotate through customer sites to get a better feel for how they operate.

4.2 Internal Science Disconnects

A common perception outside of the scientific community is that that community is inefficient at software development: it rebuilds rather than reuses. One reason cited for this inefficiency was the “free labor” provided by graduate students and post-docs; treating this labor as zero-cost means that sharing code provides little value. On the other hand, significant portions of scientific software cannot be reused, even within the same science, as it performs specialized computations that are tightly coupled with experimental hardware, such as calibration and reconstruction.

The longevity of large scientific projects, typically measured in decades, forces scientists to introduce extra layers in order to isolate different components and ease often unavoidable migrations, adding to system complexity.

Unfortunately, those layers are typically used only to abstract the storage model and not the processing model. In conclusion, the scientific community needs to try harder to agree on common needs, write more efficient software and build more sharable infrastructure to the extent possible.

4.3 Academia/Science Disconnects

In the past, computer scientists working in the database area have tried to collaborate with scientists. These efforts generally failed. Technical failures included difficulties with supporting arrays, lineage, and uncertainty in databases. Social failures included mismatched expectations between the two groups, with computer science able to produce prototypes while science was anticipating production-quality systems. This resulted in a lack of adoption of new techniques and consequently a lack of feedback on the usefulness of those techniques. Jim Gray was much lauded as the exceptional person who managed to span the divide between the two communities by dint of full-time work and his ability to leverage the resources of Microsoft.

A suggestion to have computer science graduate students work in science labs was deemed infeasible. Science project timescales are too long to enable the students to have sufficient publications to provide good career prospects.

It is possible for the scientific community to reengage academia. Both sides must be willing to partner and set reasonable expectations. Science must take data management and databases seriously, not treat them as an afterthought. The primary need is for the community to develop a set of distilled requirements, including building blocks such as desired data types and operations. Projects might contribute key data sets to a data center where academics could access and experiment with them.
4.4 Funding Problems

The high-end commercial systems are very expensive to purchase and to operate. Science certainly cannot afford commercial systems, and even industry has balked at the price tags. Industry has responded by investing in building custom database systems that are much more cost-effective, even though the cost of development can only be amortized across internal customers. Science, meanwhile, has underinvested in software development. With problems of similar scale and complexity to industry, the scientific community is trying to get by with much smaller teams. While industry does need to move faster and thus has shorter development timescales, its return-on-investment timescales are similarly shortened.

Database research within computer science was felt to be underfunded as well. The result has been a lack of major changes and discoveries in the database field since the introduction of the relational model. One pithy commenter said, “twenty years of research, and here we go, we have map/reduce.”

4.5 Conclusions

There is great potential for the academic, industry, science and vendor communities to work together in the field of extremely large database technology. Past difficulties that have limited progress in this area must be overcome, and increased funding for database research and scientific infrastructure must be obtained.

5 THE FUTURE OF XLDB

This chapter describes trends in database systems and expectations for the future. These were the subject of several discussions, primarily during the vendor and academic panels.

5.1 State of the Database Market

Based on the discussions at the workshop, standard RDBMS technology is not meeting the needs of extremely large database users. The established database vendors do not seem to be reacting quickly enough to the new scale by adapting their products. It appears instead that the gap between the system sizes they support cost-effectively and those desired by users is widening.

Meanwhile, open source general purpose RDBMS software is becoming more capable and gaining in performance, at a low price point. Some of these systems include open interfaces that allow users to plug customized components such as storage engines into a standard, well-tested framework to tune the product to their needs. The open source database community has not yet solved the scalability problems either, but users willing to invest in custom development have found this software useful as components of larger scalable systems.

Simultaneously, specialized niche engines, including object-oriented, columnar, and other query-intensive, non-OLTP databases, are increasingly finding traction on the large-scale end. Order of magnitude improvements in minimizing I/O through compression and efficient clustering of data, effortless scalability, and the resulting gains in the ratio of performance to price make the deployment of these systems worthwhile, despite their unique nature and difficulties with interoperability.
The traditional RDBMS vendors are facing increased competition because of these trends. One participant provocatively suggested that as a result, they will fade away in the next ten to twenty years. Even if that proves not to be the case, they will likely have to undergo a substantial redesign to succeed in the extremely large database market.

People managing very large data sets strongly dislike monolithic systems. Such systems are inflexible, difficult to scale and debug, and tend to lock the customer into one type of hardware, frequently the high-end, expensive type. The emerging trend is to build data management systems from specialized, lightweight components, mixing and matching these components with an appropriate mixture of low-cost, commodity hardware (CPU, memory, flash, fast disks, slow disks) to achieve the ultimate balance.

Structured and unstructured data are coming together. The textbook approach assuming a perfect schema and clean data is inadequate. Today’s analyses on very large data sets must handle flexible schemas and uncertain data yielding approximate results.

The map/reduce paradigm is gaining acceptance by virtue of its simple scalability. It will likely not be the final answer as its lack of join algorithms beyond sort/merge will prove to be an increasing limitation. Finally, it was observed that academic computer scientists are not focusing on core database technology any more. Data integration has taken over as the hot topic in the field.

5.2 Impact of Hardware Trends

The number of CPU cores / processing power is increasing rapidly. For this reason alone, databases will have to go massively parallel to consume multi-core CPUs. This parallelization will need to take place at all levels of the software, including both query execution and low-level internal processing. It is very unclear if and when we might see true optical computers. Such technology, once available, would certainly be very disruptive for databases.

Disks are becoming bigger, and denser. The raw disk transfer rates have improved over the years, but the I/O transaction rate which is limited by disk head movements has stalled; disks are effectively becoming sequential devices. This severely impacts databases, which frequently access small random blocks of data.

Participants emphasized that power and cooling is often overlooked. Databases are typically the single biggest consumer of power in a complete system, primarily due to their spinning disks which generate a lot of heat. It seems inevitable that disk technology will be replaced in the not-too-distant future by a solid-state, no-moving-parts solution. One of the most promising candidates is flash memory. Large-scale deployment of flash, with its random access abilities, would completely change the way large data sets are managed, having a large effect on how data is partitioned, indexed, replicated and aggregated.

5.3 Conclusions

The next decade will be very interesting. Trends in hardware and software will enable the construction of ever-larger databases, and an increased level of research from the academic community and database vendors could significantly simplify building these systems. These trends include the movement towards massively parallel commodity boxes and solid-state storage and the simultaneous evolution of software towards both lightweight components and specialized analysis engines.
6 NEXT STEPS

There was unanimous agreement that we should not stop with this one workshop, but that this should instead mark the start of a long, useful collaboration. Some agonized over their lack of available time, but the overall sentiment was, "if you can't spend some time on collaborating, extremely large databases are not your core problem".

Specifically it was agreed that we should:
- conduct another workshop
- try to setup smaller working group(s)
- try to define a standard benchmark focused on data-intensive queries
- set up shared infrastructure, ranging from testbed environments to a wiki for publishing information, including "war stories" of experiences that could be useful to others
- raise awareness by writing a position paper, creating an entry in Wikipedia, and other actions.

6.1 Next Workshop

Participants felt that the next workshop should be similar to the first one. It should be held approximately one year after the first, giving time for progress to be made in smaller working groups and for several projects to accumulate experience, including LHC, PanSTARRS, and the Google/IBM/academic cluster.

The workshop should probably be extended to two or possibly three days to enable more detailed sharing and more extensive discussion. Now that the content and value of the session has been established, attendees will be able to justify additional time away from their offices. At more than one day long, attaching the workshop to an existing conference was thought to extend travel times too much, so it is best kept separate.

"Neutral ground" was thought to be better than a vendor or industry location. While there is undoubtedly selection bias present, the participants felt that holding the workshop in the San Francisco Bay Area minimized travel for the greatest number. We could meet at SLAC again; Asilomar (1) was also mentioned as a possibility.

The number of attendees should not be significantly expanded, as it would be hard to make progress with a much larger group. Participation should remain by invitation only.

The content of the next workshop should focus more on experience sharing to fully bring out commonalities that can be developed into community-wide requirements. If vendors are present, users wanted to be able to question them more.

6.2 Working Groups

The discussion at the workshop was at a relatively high level. It was agreed we need to dive deeper into specific problems. One example that was frequently given was that academics would like to better understand scientific needs. To tackle these problems, smaller dedicated working groups that could meet separately at more frequent intervals would be desirable. A possible agenda for a science/computer science meeting might include:
- developing a common set of requirements for scientific databases including difficult queries, a limited number of desired primitive data types, and a small set of algebraic operators
- developing a mechanism for the scientific community to give academics access to large data sets.
6.3 Benchmarks

Well defined benchmarks have been a good way to describe problems, attract the attention of both database vendors and academia, and drive progress in the field. Existing benchmarks such as TPC-H and TPC-DS are useful, but they do not directly address the usage scenarios of extremely large databases. The group will first try to understand and reach consensus on our common requirements and then define a benchmark that focuses specifically on data-intensive queries.

6.4 Shared Infrastructure

Progress in this area will happen most efficiently if groups can avoid duplication, particularly repeating mistakes. Common shared infrastructure will help here. Initially, we will build a wiki site for groups to publish lessons learned, describe problems, and discuss issues. This site will be moderated but open to all interested parties, including those who did not have a chance to participate in the workshop.

It was also noted that we should try to leverage a recently announced (2) data center, intended for academic use, that has been set up with Google hardware, IBM management software, and Yahoo!-led open source map/reduce software (3).

7 ACKNOWLEDGMENTS

The organizers gratefully acknowledge support from our sponsors: LSST Corporation and Yahoo!, Inc.

FOOTNOTES

1


2


3

http://incubator.apache.org/pig/

4

http://www.abinitio.com

(1)

http://www.visitasilomar.com

Yahoo! also announced a large cluster for use by academia after the workshop: http://biz.yahoo.com/bw/071112/20071112005373.html

8 GLOSSARY

CERN - The European Organization for Nuclear Research
DBMS – Database Management Systems
ETL - Extract, Transform and Load (data preparation)
GFS – Google File System
HEP – High Energy Physics

IPAC - Infrared Processing and Analysis Center, part of the California Institute of Technology
JHU - The Johns Hopkins University
LHC – Large Hadron Collider
LLNL - Lawrence Livermore National Laboratory
LSST – Large Synoptic Survey Telescope
NCSA - National Center for Supercomputing Applications
OLTP - On-Line Transaction Processing
ORNL - Oak Ridge National Laboratory
PanSTARRS – Panoramic Survey Telescope & Rapid Response System
PNL - Pacific Northwest National Laboratory
RDBMS – Relational Database Management System
SDSC - San Diego Supercomputer Center
SDSS – Sloan Digital Sky Survey
SLAC – Stanford Linear Accelerator Center
VLDB – Very Large Databases
XLDB – Extremely Large Databases

APPENDIX A – AGENDA

<table>
<thead>
<tr>
<th>Start Time</th>
<th>Duration (mins)</th>
<th>Speaker Moderator</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>20</td>
<td>Jacek Becla</td>
<td>Welcome</td>
</tr>
<tr>
<td>9:20</td>
<td>40</td>
<td>Dirk Duellmann (LHC) Kian-Tat Lim (LSST)</td>
<td>Examples of future large scale scientific databases LHC (20 min), LSST (20 min)</td>
</tr>
<tr>
<td>Time</td>
<td>Duration</td>
<td>Name</td>
<td>Talk Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10:00</td>
<td>45</td>
<td>Jacek Becla</td>
<td>The two talks will introduce xldb issues in the context of two scientific communities managing large data sets (High Energy Physics and Astronomy)</td>
</tr>
<tr>
<td>10:45</td>
<td>20</td>
<td></td>
<td>Coffee break</td>
</tr>
<tr>
<td>11:05</td>
<td>85</td>
<td>Kian-Tat Lim</td>
<td>Trends, road-blocks, today’s solutions, wishes Panel discussion, scientific community representatives Panel will reveal how the scientific community is using and would like to use databases.</td>
</tr>
<tr>
<td>12:30</td>
<td>60</td>
<td></td>
<td>Lunch</td>
</tr>
<tr>
<td>1:30</td>
<td>15</td>
<td>Andrew Hanushevsky</td>
<td>Summary of panel discussions</td>
</tr>
<tr>
<td>1:45</td>
<td>70</td>
<td>Andrew Hanushevsky</td>
<td>Vendor response Panel discussion, vendor representatives Directed questions from the moderator reflecting previous discussions, as well as open time. No sales talks please.</td>
</tr>
<tr>
<td>Time</td>
<td>Duration</td>
<td>Activity</td>
<td>Notes</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2:55</td>
<td>20</td>
<td>coffee break</td>
<td></td>
</tr>
<tr>
<td>3:15</td>
<td>30</td>
<td></td>
<td>Thoughts from academia Panel discussion, academic representatives Representatives from academia give their thoughts about preceding discussions, and their vision of how to improve the connection between the research community and practical peta-scale databases.</td>
</tr>
<tr>
<td>3:45</td>
<td>60</td>
<td>Richard Mount</td>
<td>Future Round table discussion, all How to organize xldb-related work most efficiently including leveraging future large scale applications to advance database technology</td>
</tr>
<tr>
<td>4:45</td>
<td>15</td>
<td>Jacek Becla</td>
<td>Conclusions</td>
</tr>
<tr>
<td>5:00</td>
<td></td>
<td>Adjourn</td>
<td></td>
</tr>
</tbody>
</table>

**APPENDIX B – PARTICIPANTS**

**Academia**
- DeWitt, David – Univ. of Wisconsin
- Stonebraker, Michael – M.I.T.
- Murthy, Raghotham – Stanford University
Industrial database users
• Baldeschwieler, Eric – Yahoo!
• Brown, Phil – AT&T Labs Research
• Callaghan, Mark – Google
• Das, Aparajeeta – eBay
• Hall, Sandra – AT&T
• McIntire, Michael – eBay
• Muthukrishnan, S – Google
• Priyadarshy, Satyam – AOL
• Ratzesberger, Oliver – eBay
• Saha, Partha – Yahoo! Strategic Data Solutions
• Schneider, Donovan – Yahoo!
• Walker, Rex – eBay

Science
• Abdulla, Ghaleb – Lawrence Livermore National Laboratory
• Becla, Jacek – SLAC
• Borne, Kirk – George Mason University
• Cabrey, David – PNNL
• Cai, Dora – NCSA, University of Illinois at Urbana-Champaign
• Critchlow, Terence – PNNL
• Dubois-Felsmann, Gregory – SLAC
• Duellmann, Dirk – CERN
• Handley, Tom – Infrared Processing and Analysis Center (IPAC)
• Hanushevsky, Andrew – Stanford University/SLAC
• Heasley, Jim – Institute for Astronomy, University of Hawaii
• Kahn, Steven – SLAC/Stanford
• Kantor, Jeffrey – LSST Corporation
• Lim, Kian-Tat – SLAC
• Luitz, Steffen – SLAC
• Matarazzo, Celeste – Lawrence Livermore National Laboratory
• Monkewitz, Serge – IPAC/Caltech
• Mount, Richard – SLAC
• Nandigam, Viswanath – San Diego Supercomputer Center
• Plante, Raymond – NCSA
• Samatova, Nagiza – ORNL / North Carolina State University
• Schalk, T L – U.C. Santa Cruz
Database vendors

- Aker, Brian – MySQL
- Bawa, Mayank – Aster Data Systems
- Brobst, Stephen – Teradata
- Ganesh, Amit – Oracle
- Guzenda, Leon – Objectivity
- Hamilton, James – Microsoft
- Held, Jerry – Vertica Systems & Business Objects
- Hu, Wei – Oracle
- Hwang, JT – Netezza
- Paries, Lee – Teradata
- Tan, CK – Greenplum
- Tate, Stewart – IBM
- Jakobsson, Hakan – Oracle
- Lohman, Guy – IBM Almaden Research Center
- Lonergan, Luke – Greenplum

APPENDIX C – FACTS

Here are some numbers / technologies mentioned at the workshop. Please keep in mind that this is not a comprehensive overview - there was no time to delve into details.

Sizes

Currently in production

- Google: tens of petabytes
- Yahoo: tens of petabytes, 100s TB in Oracle, 25TB/day ingest rates
- AOL: few petabytes, 200 TB in PostgreSQL
- eBay: over 1 PB, will have 2-4 in the next 12 months
- AT&T: 1.2 PB. 3.2 trillion rows in single largest table
- BaBar: 2 PB in files (structured data), few TB in database (metadata)
- SDSS: 30 TB
Planned
- CERN (starts 2008): stream of data from detector: 1 PB/sec, most data discarded. Kept: 20 PB/year
- PanSTARRS (starts 2008): few PB
- LSST (starts end of 2014): 55 PB in files (images), 15 PB in database

Database & Database-Like Technologies Used

Currently in production
- Google has large installation of MySQL for ads, combined with reporting, map/reduce plus BigTable plus GFS for searches.
- Yahoo is using Oracle, proprietary column based engine, and open source map/reduce (Hadoop)
- AOL is using mixture of home-grown solutions, Sybase, Oracle, PostgreSQL and MySQL
- eBay is using Oracle where transactions are needed and Teradata for analytics
- AT&T is using home grown RDBMS called Daytona
- BaBar used to rely on Objectivity/DB (object oriented database), now uses hybrid solution: structured data in files plus metadata in database (MySQL and Oracle)
- SDSS is using SQL Server

Planned
- CERN will rely on hybrid solution structured data in files plus metadata in database. <2% of data in database (~300 TB, in Oracle).
- PanSTARRS: pixel data in files, everything else in SQL Server. Over 50% of all data in database.
- LSST: pixel data in files, everything else in database-like system, possibly MySQL + map/reduce.

2008: Report from the SciDB Workshop

Source: https://www.jstage.jst.go.jp/article.../0/7_7-88/ pdf (PDF)

REPORT FROM THE SciDB WORKSHOP
J Becla*1 and K-T Lim 2
Stanford Linear Accelerator Center, Menlo Park, CA 94025, USA *1 Email: becla@slac.stanford.edu
2 Email: ktl@slac.stanford.edu

ABSTRACT

A mini-workshop with representatives from the data-driven science and database research communities was organized in response to suggestions at the first XLDB Workshop. The goal was to develop common requirements and primitives for a next-generation database management system that scientists would use, including those from high-energy physics, astronomy, geoscience and fusion, in order to stimulate research and advance technology. These requirements
were thought by the database researchers to be novel and unlikely to be fully met by current commercial vendors. The two groups accordingly decided to explore building a new open source DBMS. This paper is the final report of the discussions and activities at the workshop.

Keywords: Database, XLDB

1 ABOUT THE WORKSHOP

The workshop was held March 30 through April 1, 2008, in Asilomar, CA. The workshop organizing committee was composed of Jacek Becla, Mike Stonebraker, David DeWitt, and Kian-Tat Lim. Participation was by invitation only in order to keep the discussion focused.

Scientific community representatives, in alphabetical order:

- Astronomy: Kirk Borne (GMU), Robert Lupton (Princeton), and Alex Szalay (JHU)
- Biology: Gordon Anderson (PNL)
- Fusion: Tim Frazier (LLNL)
- Geoscience: James Frew (UCSB)
- HEP: Gregory Dubois-Felsmann (SLAC) and Dirk Duellmann (CERN)

Database research community representatives:

- Academia: David DeWitt (Univ. of Wisconsin), Jignesh Patel (Univ. of Michigan), and Mike Stonebraker (MIT)
- Industry: Tom Barclay (Microsoft), Mike Carey (BEA), Guy Lohman (IBM), and Chris Olston (Yahoo!)

“Unclassified” participants:

- Jacek Becla (SLAC), Kian-Tat Lim (SLAC), and Oliver Ratzesberger (EBay)

2 SCIENTIFIC REQUIREMENTS

This report captures the key results of the workshop, summarizes the discussions that led to them, and documents the resulting action items. It does not attempt to reproduce the exact flow of the highly interactive, ad hoc conversations that occurred during the workshop.

One major result of the workshop was a set of requirements that a database management system should meet in order to support the storage and analysis needs of several fields of data-intensive science over the next decade. These requirements were distilled from wish lists presented by each field and from more in-depth analysis of specific desired features. This future science-oriented database management system will be referred to as “SciDBMS.”

Summary of Requirements for SciDBMS

Adoption

- Open source with a stable developer community.

Scalability and Performance
• Scalability up to hundreds of petabytes using parallelized single queries on commodity hardware.

• Fault tolerance with intra-query failover.

• The performance, extensibility, and compression of a column store, but also with efficient “SELECT *” (or “SELECT many columns”).

Interfaces

• SQL (with appropriate extensions, if needed).

• An object-relational mapping layer for external applications.

• User-defined functions and stored procedures expressed in familiar procedural languages operating on row cursors or post-ORM object cursors that have ordering and grouping properties.

• Partial results, query progress indicator, and query pause/restart/abort.

• Pre-execution query cost estimate, preferably in wall-clock time.

Features

• No transactions needed for the largest tables, but atomic multiple-table batch appends are needed. Metadata tables may need transactions.

• Support for spatial and temporal operations.

• Support for arrays as a first-class column type.

• Lightweight support for “error-bar” uncertainty of data elements by associating error columns with data columns and adding an “approximately equal” operator.

• Cheap one-to-one and one-to-many joins when both tables are partitioned on the join key.

• Versioning of tables and code, including the ability to tag or label sets that go together.

• Support for provenance of data elements, including querying and import of provenance.

• A resource management system including CPU and disk quotas.

• Support for tiered storage: migration from faster media to slower media (e.g. flash, fast disk, slow disk).

Less Important Features

• Column grouping: asserting for efficiency that certain columns will always be used together.

• Access controls and perhaps namespaces for sets of columns in wide tables.

• Support for probabilistic uncertainty for nominal values.

• XQuery-like engine for querying hierarchical data.

Explanation of Requirements

Adoption

SciDBMS must be open source if it is to achieve adoption in the scientific community. Many projects have been burned by over-dependence on a single commercial vendor. Source escrow is insufficient, as it still leaves ongoing support of the software in a questionable state. Open source alone is also insufficient, however. A robust community of developers must be present to ensure that the software will not languish in obscurity.
Scalability and Performance

SciDBMS must be able to scale to databases of hundreds of petabytes, with individual tables measured in trillions of rows. These sizes are already part of the requirements for the Large Synoptic Survey Telescope (LSST) database, which will store trillions of observations of tens of billions of astronomical objects. It must be able to parallelize single queries, rather than simply allowing multiple queries to execute in parallel, as many sciences will have queries that touch large fractions of the data but must still return results on interactive timescales. Achieving this performance must be possible on commodity hardware; proprietary hardware leads to cost and support constraints that are undesirable for large scientific projects. Since it is using commodity hardware, the system must be fault tolerant. Since some queries will still have long execution times, the ability to continue a running query if a node fails, rather than restarting it from the beginning, is necessary.

Column stores are extremely interesting as a model for SciDBMS for several reasons. They provide performance advantages for queries that operate on a small subset of columns; such queries are expected to make up a significant fraction of scientific workloads. Column stores provide easy extensibility of tables, with low-cost addition and deletion of columns. These operations are frequent in research-oriented databases where schemas are not fully understood in advance. Column stores also promise much greater compression of data on disk, providing advantages both in cost and in performance due to reduced I/O. SciDBMS will require all of these advantages. On the other hand, scientists may need to extract many columns from a wide table to perform complex analyses. Accordingly, SciDBMS must also allow efficient “SELECT *” or at least “SELECT many columns” queries.

Fault tolerance will likely require storage of two copies of the data. Fortuitously, the second copy might be just what is needed to support a separate “SELECT *” view. Requiring more than two copies is less desirable.

Interfaces

SciDBMS must provide a standard SQL interface, with appropriate extensions, if needed. The astronomy community has become familiar with SQL and is able to express complex analytical operations in the language. Many fields are using relational databases and SQL for querying metadata, such as experimental conditions, sometimes within more user-friendly tools. Overall, it provides a common baseline for declarative relational queries.

Beyond SQL, SciDBMS should provide an object-relational mapping layer that external applications can use to process data. This layer will simplify access to the database and allow scientists to think in terms of scientifically-relevant objects instead of normalized rows. Full object support, e.g. with pointers, is likely not needed; mapping to hierarchical structures should be adequate. The mapping layer should have interfaces to at least C++, Java, and Python.

SciDBMS should allow users to define functions and stored procedures in familiar procedural languages such as those mentioned above. These procedures should be able to operate on row cursors or post-ORM object cursors that have ordering and grouping properties. The goal here is to allow scientists to write procedures that compute values dependent on multiple rows, such as looking for significant changes in time series, in a familiar fashion, without having to learn new syntax such as windowing functions. Users should be allowed to specify that the function or procedure is distributive and can be executed in parallel. Since the function or procedure is user-written code, it may have errors, so exceptions must be handled reasonably, and the user must be able to debug the code easily. The functions and
procedures must be able to share scans of large data tables with other simultaneously-running queries, particularly when interacting with tiered storage (see below).

The above functions and procedures will essentially provide a workflow capability within the database engine. One of the dangers of this embedding is that using such features ties projects closely to the DBMS chosen. Most large scientific projects span decades, often requiring migrations from one system to another as technology and products change. Such migrations become more difficult as the amount of non-portable code increases. On the other hand, giving the database engine control of the workflow not only permits movement of computation to data that is essential for peta-scale datasets, but also allows the engine to track provenance more simply and accurately (see below).

Many queries to SciDBMS are expected to run for substantial periods of time. It should be possible to pause such long-running queries, restart them, and abort them. These queries should generate partial results so that the user can immediately determine if they are computing useful values and abort them if not. Also, the user needs to get feedback from the system about query progress.

A similar requirement that helps to enable efficient exploration of the data is that SciDBMS should provide a reasonably accurate estimate of the cost of a query before it is executed. This estimate would be most useful if it is in terms of wall-clock time.

Features

The large tables in scientific databases are, like commercial data warehouses, typically read-only after they have been loaded. Instrumental observations should never be updated; derived data is typically not updated in place to ensure reproducibility. As a result, transaction capabilities should not be needed for the largest tables. Instead, what is needed for loading these tables are batch appends that are atomic across multiple tables. Concurrent access to these large tables by a single writer and multiple readers is required; dirty reads are acceptable. Other tables, particularly DBMS-internal metadata tables, may need transactions in order to maintain consistency.

SciDBMS must provide efficient support for spatial and temporal operations. Astronomy and earth sciences operate on two- or three-dimensional spatial grids, often using a plethora of spherical coordinate systems. Transformations between those coordinate systems need not be embedded in the database, but operations relating records using spatial information, such as finding near-neighbors, are essential. Nearly all sciences need to deal with time series of data; it is frequently necessary to understand relationships between consecutive elements in time or analyze entire sequences of observations.

All sciences need to work with non-scalar values like vectors and arrays. SciDBMS must provide support for arrays as a first-class column type.

All sciences must deal with observations and derived data that have inherent uncertainties. The simplest form of this associates each data element with an “error bar,” typically given as a standard deviation. SciDBMS must at a minimum allow an error column to be similarly associated with each data column. Using such a column is easiest if an “approximately equal” operator is provided. This operator would be used in WHERE and JOIN clauses. Such an operator will need to be able to take a third parameter to indicate the width of the error bar in terms of the number of standard deviations specified by the error column, i.e. the “3” in “± 3 sigma.” More complex uncertainty operations...
require detailed understanding of correlations between errors and so were thought to be best left to application code, rather than embedded in the database engine.

Each column should be able to be associated with an appropriate unit of measure. SciDBMS should prevent operations on incommensurate units. Ideally, support would be provided for converting conformable units, like that provided by the Unix units utility.

Scientists often need to annotate data, adding information based on their own analyses. In some cases these annotations can be columns in a shared table, but in other cases they are most easily thought of as columns in a separate table joined to a primary one. It should be cheap to do one-to-one and one-to-many joins between such annotation tables and the main table when both tables are partitioned on the join key. Enabling the storage of these annotation columns in separate tables could be used to give the equivalent of per-user “namespaces” for columns, can allow differing access control parameters for different sets of columns, and enables such common scientific tasks as classifying an observation in multiple simultaneous ways, with probabilistic weightings of and different attributes attached to each classification.

SciDBMS should allow versioning of both tables and code elements like the procedures described above. The ability to tag or label sets of tables and code that go together is also desirable. The versioning model of the Subversion version control system is one that scientists felt familiar with and believed would be adequate. These capabilities will allow scientific applications to generate reproducible results. They will also make simpler the management of the process of applying new algorithms to old data, a frequent occurrence in all sciences.

Along with versioning, SciDBMS must provide support for recording the provenance of data elements. Provenance information allows users to determine the answers to questions such as:

- What operations led to the creation of this element?
- What operations used this element?
- What data elements were used as input to this operation?
- What data elements were created as output from this operation?

In order to answer these queries, the database will likely need to log all operations that modify data, including creation, deletion, and updates of data elements. Since operations on data will also occur outside the database, it must be possible to import provenance from external systems when loading data. The attributes of an operation that need to be recorded for provenance purposes include not only what was done but also the state of the system at the time. This system state information may be as detailed as operating system patch levels, compiler versions, etc.

Large data-intensive science projects often have diverse user bases, in some cases extending to the general public. To support such environments, it is essential that SciDBMS incorporate a complete resource management system that controls the CPU, disk, and other resources available to the database, including the ability to set quotas.

It will not always be possible or desirable to store all data on the fastest media. Many existing scientific systems provide support for tiered storage, in which data may be migrated from fast disk to slow disk to tape, and maybe also in the other direction to flash memory in the future. SciDBMS must also support this capability.
Less Important Features

There are several features that were considered to be less important or more speculative, but still potentially interesting for science.

In several cases, scientists know that certain columns will always be used together. For example, spatially-oriented fields like astronomy and geoscience will typically have two coordinates, a latitude and longitude or right ascension and declination that are always used as a pair. For efficiency, it may be desirable to indicate this column grouping to the database.

If annotation columns are added directly to a main table rather than placed in separate joined tables as described above, the ability to have access controls and perhaps separate namespaces for sets of columns within a single wide table would be desirable.

Classification probabilities, or probabilistic uncertainties associated with nominal values, are a common element stored in scientific databases, but the use of such probabilities is more complex. In some cases, thresholds will be applied; in others, the probabilities may be used as weights for aggregations. There does not yet appear to be a consensus on the operations and uncertainty model required for using these probabilities.

High-energy physics in particular has traditionally stored its largest datasets in a hierarchical form, rather similar to an XML document, that corresponds closely to the way that physicists think about the data. Other sciences, while amenable to pure-SQL relational queries over their data, may also have more hierarchical views of it, particularly for analyses coded in programming languages as described above. An XQuery-like engine for querying hierarchical data, which could still be stored in a "shredded" relational form, might be interesting as a way of providing a familiar data model while still reducing the amount of code that needs to be written to perform an analysis.

SCIENTIFIC DATABASE CUSTOMERS

Database researchers present at the workshop believed that scientific database requirements are on the bleeding edge of what database management systems can provide but that industrial users will soon catch up in terms of the complexity of their analytics and may need similar features from a DBMS. It was also believed that science itself, particularly as it becomes more data-intensive, is a sufficiently large and growing database market to support a science-oriented DBMS.

During the discussions it became clear that the HEP community has significant experience with handling large-scale data but that it is unlikely to seriously drive the requirements for a near-term SciDBMS both because of the timing of their system deployments and because of the additional complexities of satisfying their community’s historical preference for code-driven rather than query-driven operations on complex, hierarchical data.

On the other hand, other sciences are poised to adopt large scientific databases in the near future. Astronomy, in particular the LSST project, perhaps presents the most significant immediate opportunity. While other fields are rapidly approaching similar scales, the requirements and needs of LSST appear to be a superset of those of other scientific communities.
DEVELOPMENT PLANS

The database researchers present agreed that the requirements presented by the scientific representatives are reasonable and that methods for addressing them are sufficiently well-understood so as not to pose insurmountable research problems. People familiar with large commercial DBMS vendors did not feel that those companies would take on the task, however. As a result, building a new system optimized for data-intensive science is necessary.

A startup company was proposed as the best vehicle for building this system. This company would lead the open source development of SciDBMS, building a working prototype in about a year and productizing it the next. Its primary source of revenue, as for others based on open source, would be from support contracts. The company would likely be located in Silicon Valley to take advantage of the developer talent in the area.

In order to obtain startup funding from the venture capital community, market interest from the scientific community would need to be demonstrated. Sufficient “buy-in” could be shown if there are 2–3 “lighthouse” customers seriously interested in using the product and if science is willing to commit 1–2 full-time-equivalent developers to the project. The personnel portion of this requirement could be assisted by grants from funding agencies such as the Department of Energy’s Advanced Scientific Computing Research program.

There was some discussion as to which code base to use as a starting point. Systems mentioned included PostgreSQL and its derivatives such as Greenplum and EnterpriseDB, Vertica, BerkeleyDB (SleepyCat), Hadoop, MonetDB and MySQL. Some believed it would be easiest to start from PostgreSQL, but this remains to be evaluated.

ACTION ITEMS

- Produce a 10-15 page document sketching a proposed design for SciDBMS [lead = Mike Stonebraker; due end of April].
- Solidify science participation. This includes identifying at least two projects (“lighthouse” customers) interested in using the SciDBMS and willing to work closely with the development team, as well as assessing the level of interest and participation by other potential users [lead: Jacek Becla; due April 20].
- Organize fund raising [lead = Mike Stonebraker; due May 10, after the previous item].
- Discuss the possibility of collaboration with industry (eBay, Google and others) [lead: Jacek Becla; due mid-May, after the design sketch is developed].