From Databases to Big Data

Sam Madden • *Massachusetts Institute of Technology*

My primary research community is focused on *data management* — the buttoned-up world of business data, relational databases, carefully designed schemas, SQL, and strict consistency ("ACID semantics"). As the token database researcher at MIT, I'm often asked questions like, "I heard databases can store a lot of data. Does that mean they solve the *big data* problem?" The answer to this question is "no," but before telling you why, let me first try to define what I think the big data problem is.

What Is Big Data?

Among all the definitions offered for "big data," my favorite is that it means data that's *too big, too fast,* or *too hard* for existing tools to process. Here, "too big" means that organizations increasingly must deal with petabyte-scale collections of data that come from click streams, transaction histories, sensors, and elsewhere. "Too fast" means that not only is data big, but it must be processed quickly — for example, to perform fraud detection at a point of sale or determine which ad to show to a user on a webpage. "Too hard" is a catchall for data that doesn't fit neatly into an existing processing tool or that needs some kind of analysis that existing tools can't readily provide. A similar breakdown is being promulgated by Gartner (which is probably a sign that I'm oversimplifying things), citing the "three Vs" — volume, velocity, and variety (a catchall similar to "too hard").

On Databases and MapReduce

So, where do database management systems (DBMSs) fall short on these metrics? With respect to data size, commercial relational systems actually do pretty well: most analytics vendors (such as Greenplum, Netezza, Teradata, or Vertica) report being able to handle multi-petabyte databases. Although this might not be big enough for a few massive Internet companies, it probably is for almost everyone else. Unfortunately, open source systems such as MySQL and Postgres lag far behind commercial systems in terms of scalability. It's on the "too fast" and "too hard" fronts where database systems don't fare well.

First, databases must slowly import data into a native representation before they can be queried, limiting their ability to handle streaming data. The database community has widely studied streaming technologies, which don't integrate well into the relational engines themselves. Second, although engines provide some support for in-database statistics and modeling, these efforts haven't been widely adopted and, as a general rule, don't parallelize effectively to massive quantities of data (except in a few cases, as I note later).

What about platforms such as MapReduce or Hadoop? Like DBMSs, they do scale to large amounts of data (although Hadoop deployments generally need many more physical machines to process the same amount of data as a comparable relational system, because they lack many of DBMSs' advanced query-processing strategies). However, they're limited in many of the same ways as relational systems. First, they provide a low-level infrastructure designed to process data, not manage it. This means they simply provide access to a collection of files; users must ensure that those files are consistent, maintain them over time, and ensure that programs written over the data continue to work even as the data evolves. Of course, developers can build data management support on top of these platforms; unfortunately, a lot of what's being built (Hive or HBase, for instance) basically seems to be recreating DBMSs, rather than solving the new problems that (in my mind) are at the

Published by the IEEE Computer Society 1089-7801/12/$31.00 © 2012 IEEE

crux of big data. Additionally, these systems provide rather poor support for the "too fast" problem because they're oriented toward processing large blocks of replicated, disk-based data at a time, which makes it difficult to obtain low-latency responses.

The State of the Art

So, where does this leave us? Existing tools don't lend themselves to sophisticated data analysis at the scale many users would like. Tools such as SAS, R, and Matlab support relatively sophisticated analysis, but they aren't designed to scale to datasets that exceed even a single machine's memory. Tools that are designed to scale, such as relational DBMSs and Hadoop, don't support these methods out of the box.

Additionally, neither DBMSs nor MapReduce are good at handling data arriving at high rates, providing little out-of-the-box support for techniques such as approximation, single-pass/sublinear algorithms, or sampling that might help users ingest massive data volumes.

Several research projects are trying to bridge the gap between large-scale data processing platforms such as DBMSs and MapReduce, and analysis packages such as SAS, R, and Matlab. These typically take one of three approaches: extend the relational model, extend the MapReduce/Hadoop model, or build something entirely different.

In the relational camp are traditional vendors such as Oracle, with products like its Data Mining extensions, as well as upstarts such as Greenplum with its MadSkills project (see http://db.cs.berkeley.edu/papers/vldb09-madskills.pdf). These efforts seek to exploit relational engines' extensibility features to implement various data mining, machine learning, and statistical algorithms inside the DBMS. This approach enables operations to happen inside the DBMS, near the data, and to sometimes run in parallel. However, many users would rather not be SQL programmers, and some iterative algorithms aren't easily expressible as parallel operations in SQL.

On the MapReduce front, numerous efforts are under way. Probably the best-known is Apache Mahout, which provides a framework for executing many machine learning algorithms (mostly) on top of MapReduce. Other MapReduce-like systems include UC Berkeley's Spark, the University of Washington's HaLoop, Indiana University's Twister, and Microsoft's Project Daytona. These systems

IC Welcomes New Board Member

Anirban Mahanti is a principal researcher at National ICT Australia (NICTA), Australia's center of excellence in ICT research. His research interests are in performance evaluation of distributed systems and computer networks, with current research activities focused on scalable content distribution, network measurements, and ICT for development. Mahanti has a BE in computer science and engineering from the Birla Institute of Technology, India, and an MSc and PhD in computer science from the University of Saskatchewan, Canada. He's served on the program committee of several top-tier conferences including World Wide Web, IFIP Performance, and ACM SIGMETRICS. He's a member of the ACM and the membership chair for the IEEE STC on Sustainable Computing.

provide better support for certain types of iterative statistical algorithms inside a MapReduce-like programming model, but still lack database systems' data management features.

Finally, on the new systems front, packages such as GraphLab from Carlos Guestrin's group at Carnegie Mellon University or the SciDB project (with which I'm involved) aim to provide a scalable platform for addressing some or all of these concerns. Although these systems might eventually solve the problem, they still have a long way to go. GraphLab provides a scalable framework for solving some graph-based iterative machine learning algorithms, but it isn't a data management platform, and it requires that data sets fit into memory. SciDB is a grand vision that will support integration with high-level imperative languages, a variety of algorithms, massive scale, disk resident data, and so on, but it's still in its infancy.

What's Left?

These systems all represent a great step in the right direction. Much more is needed, however. First, part of the big data craze is that CIOs and their ilk are demanding "insight" from data; actually generating that insight can be very tricky. Machine learning algorithms are powerful but often require considerable user sophistication, especially with regard to selecting features for training and choosing model structure (for instance, for regression or in graphical models). Many of our students learn the mathematics behind these models but don't develop the skills required to use them effectively in practice on large datasets.

Second, I believe these tools all fall short on the usability front. DBMSs do a great job of helping a user maintain and curate a dataset, adding new data over time, maintaining derived views of the data, evolving its schema, and supporting backup, high availability, and consistency in various ways. However, forcing algorithms into a declarative, relational framework is unnatural, and users greatly prefer more conventional, imperative ways of thinking about their algorithms. Additionally, many databases provide a painful out-of-the-box experience, requiring a slow "import" phase before users can do any data exploration. Tools based on MapReduce provide a more conventional programming model, an ability to get going quickly on analysis without a slow import phase, and a better separation between the storage and execution engines. However, they lack many of a database's data management niceties. Furthermore, none of these platforms provide necessary exploratory tools for visualizing data and models, or understanding where results came from or why models aren't working.

In summary, although databases don't solve all aspects of the big data problem, several tools — some based on databases — get part-way there. What's missing is twofold: First, we must improve statistics and machine learning algorithms to be more robust and easier for unsophisticated users to apply, while simultaneously training students in their intricacies. Second, we need to develop a data management ecosystem around these algorithms so that users can manage and evolve their data, enforce consistency properties over it, and browse, visualize, and understand their algorithms' results.

Sam Madden is an associate professor of electrical engineering and computer science in the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology, where he works on projects including the C-Store column-oriented database system and the CarTel mobile sensor network system. His research interests include databases, sensor networks, and mobile computing. Madden has a PhD in computer science from the University of California, Berkeley. He was named one of *Technology Review's* Top 35 Under 35 in 2005, and is the recipient of several awards, including a US National Science Foundation CAREER award and a Sloan Foundation Fellowship. Contact him at madden@csail.mit.edu.