Research Statement

Andrew Pavlo
pavlo@cs.brown.edu

Creating a large-scale, data-intensive application is easier now than it ever has been, in part due to the proliferation of open-source distributed system tools, cloud-computing platforms, and affordable mobile sensors. Developers are able to deploy applications in a short amount of time that have the potential to reach millions of users and collect large amounts of data from a variety of sources. Data scientists can then process and interpret this data to uncover new knowledge in commercial, medical, and scientific fields.

Many of the database management systems (DBMS) used in these applications are based on the system architectures developed in the early 1980s. But now the processing and storage needs of emerging Internet-scale, "Big Data" applications are surpassing the limitations of these legacy systems. Although there are new distributed DBMSs that can overcome some of these problems, many of them are predicated on removing features in order to obtain scalability and availability. This approach is unwieldy as it requires applications to re-implement the guarantees that are traditionally provided by a DBMS, such as atomicity and consistency. This argues for new research into solving data management problems at scales unforeseen by previous work without sacrificing the advantages of using a DBMS. Such an effort will not just be about exploiting hardware advancements, but will also require incorporating ideas from many areas of computer science and outside disciplines.

Given this outlook, the central theme of my research is on *the development of new database management systems and technologies for high-volume and data-intensive applications.* My work uses techniques from machine learning and optimization research to enable distributed DBMSs to execute workloads that are beyond what single-node systems can support. This is increasingly important as data-driven decision making becomes more prevalent in everyday society. Thus, my plan is to continue to collaborate with partners in both industry and academia to identify shortcomings in existing systems for massive-scale data sets. I will then study these problems and develop theories on how to resolve them. When necessary, I will build systems based on these ideas and validate their efficacy through a combination of scientific experimentation and real-world deployments. I believe that as the size of databases increase, new hardware is released, and the users' problems change, my research will help move the state-of-the-art forward in the database field.

I now discuss the three areas of my current DBMS research. These projects are a testament to my overall vision of using other facets of computer science to scale distributed DBMSs. I then outline future directions that I am interested in pursuing in the next stage of my research career.

Current Research

High-Volume Transaction Processing

The majority of my time as a Ph.D. student has been involved in the development of a new distributed DBMS, called **H-Store** [1], that is designed for applications with a large number of clients that need to access and store data at the same time. There are two fundamental differences in H-Store over a legacy DBMS. In older systems, the DBMS copies data from disk into memory when it is needed and uses expensive concurrency control measures to allow multiple transactions to execute at the same time to hide the latency of disk. H-Store eschews all of this overhead by storing the entire database in main memory partitions (shards) and only executes one transaction at a time at each partition [3]. This allows H-Store to process transactions several orders of magnitude faster than a disk-based system, since transactions that access data at only one partition do not require concurrency control. For example, in one benchmark, H-Store executes 200,000 transactions per second on a single machine, compared to just 5,000 transactions per second using a well-known, open-source DBMS. This significant performance improvement allows a wide variety applications, ranging from high-frequency trading systems to e-commerce Web sites, to support many simultaneous users using only commodity hardware at a fraction of the cost and energy usage.

In order to support databases that are larger than the amount of memory available on a single machine, H-Store splits the data across a cluster of nodes and processes transactions on them in parallel. But this brings about several problems that are inherent in a distributed DBMS that prevents the system from scaling. Thus, my research has focused on removing these impediments.

The greatest of these problems is that H-Store stalls whenever a transaction needs to access data that is stored at two or more partitions. The DBMS locks the partitions that a transaction needs to prevent any other transaction from accessing them at the same time. This means that these partitions are kept idle while the DBMS waits for the transaction to send a request over the network to execute queries. Whether or not a transaction needs to access multiple partitions is largely determined by how the database is divided up in the cluster. But it is an NP-Complete problem to devise the optimal configuration that allows most transactions to execute using only a single partition and avoid these network delays. Thus, I developed the **Horticulture** tool that uses search techniques from optimization research to automatically create the best physical layout of a database that minimizes the number of multi-partition transactions while also reducing the effects of temporal skew [4]. The results that I published in SIGMOD show that Horticulture increases H-Store's overall performance up to $4\times$ over another state-of-the-art tool and up to a factor $16\times$ against a baseline approach.

But these database configurations only instruct H-Store on where to put the application's data in the cluster. They do not assist the DBMS in making decisions on what partitions each incoming transaction will need to access at run time. Without knowing this information, the DBMS will either lock more partitions than a transaction needs, or lock only one partition and then abort the transaction when it tries to access other partitions. To overcome this problem, I developed the **Houdini** framework that uses machine learning to predict what each transaction will do before it starts executing [5]. Houdini uses probabilistic models to learn what operations the transactions will execute and then helps the DBMS schedule them in an optimal way. One challenging aspect of this problem is that our approach needs to be fast: we cannot spend 100 ms generating predictions for transactions that only run for 5 ms. The results that I published in VLDB show that integrating this framework into H-Store improves its throughput by up to $1.3\times$ with an overhead of just 5% of transactions' total execution time [5].

In this last year, I extended this work to create the **Hermes** lock-free concurrency control scheme that allows the DBMS to execute transactions any time a node is blocked waiting for network communication. Hermes employs a technique, known as speculative execution, to run single-partition transactions whenever a multi-partition transaction is stalled. Using the same probabilistic models from our previous project, the DBMS predicts whether a single-partition transaction can safely be interleaved at an idle partition without violating consistency guarantees. Our initial results show that Hermes allows the DBMS to execute $\sim$30% more transactions without needing to use locks or maintain multiple copies of data to ensure serializability.

Although building a DBMS from the ground up was a laborious task, it gave me an intimate understanding of the system that allowed me to pursue avenues that would otherwise be difficult had we used an existing system. Most notable is that a commercial version, called VoltDB, is also being developed based on H-Store's design and is currently used in hundreds of commercial and government sites. Another derivative project is a new benchmarking framework that is designed for Web-based, transactional databases [2]. As the lead graduate student for the H-Store project, I co-advised four master's and three undergraduate student projects based on the system, including two master's theses. I also taught a semester-long seminar course on NewSQL systems where all of the students' projects were based on H-Store.

Big Data Analysis Platforms

Another project that I was involved with was on evaluating different programming models for executing analytical workloads on massive-scale data sets. This was a collaboration between me and researchers at MIT, Yale, the University of Wisconsin, and Microsoft. When we began this project, nascent MapReduce frameworks, namely Hadoop, were touted as being superior for certain analytical tasks that had been traditionally executed on parallel DBMSs. To determine whether this was true, we investigated the similarities and differences between Hadoop and two commercial DBMSs. As part of this comparison, we developed a benchmark that allowed us to measure the systems' performance for common data processing tasks. This benchmark was comprised of tasks that were either ideal for the MapReduce framework or the parallel DBMSs, as well as tasks that both types of systems were expected to execute well. The results that we published in SIGMOD showed that while Hadoop was easier to setup and configure, in most cases the relational DBMSs out performed it [6]. Our main finding was that MapReduce systems are better suited for tasks on unstructured data or when working with *in situ* data sets, whereas DBMSs provide better performance for repetitive analytical operations on structured data.

In a follow-up paper that was published together with a companion paper by the original creators of the MapReduce model at Google, we predicted that these two types of systems would eventually incorporate the key features of the other and allow for better inter-operation between them [7]. Since then, it has been interesting to see many of these predictions come true in both academic systems and commercial products.

Distributed Document-Oriented Systems

Lastly, I have also explored methods for improving the performance of data-intensive, Web-based applications on distributed document-oriented DBMSs. Rather than store data as tuples in relational tables, these systems use unstructured or semi-structured objects, called *documents*, that are represented in a hierarchical format, such as JSON or XML. This work is in collaboration with 10gen, Inc., the company behind the MongoDB NoSQL system. My research is in direct response to the fallout from a major outage of one of 10gen's high-profile customers in 2010 due to an errant database configuration. Many MongoDB users face the same issues that caused this failure, but they are often from smaller organizations that do not have the resources or expertise to ensure that it does not happen to them. Thus, I worked with engineers at 10gen to identify the issues that affect their users the most and then led a team of students at Brown to develop an open-source tool that automatically selects the best physical layout of a database on a distributed document-oriented DBMS like MongoDB. This research effort has been challenging because many of the existing techniques for relational databases, including our own work on the Horticulture project, are not directly applicable to these systems. Hence, we had to develop new algorithms to generate configurations that accentuate the properties that are important in a document DBMS. We are currently in the process of beta testing this tool with MongoDB users in the New York City area.

Future Research

As databases grow in both size and complexity, optimizing a DBMS to meet the needs of modern applications will surpass the abilities of humans. Therefore, my future research will continue to improve performance and scalability of complex systems by applying machine learning techniques to better inform the underlying DBMS architecture on the best way to execute a workload. I will leverage my industry contacts to find relevant and interesting research problems, and use these partners as a forum for my work in order to have an impact on real-world applications.

Real-time Analytics & Machine Learning

Most of my research thus far has focused on improving the performance of front-end DBMSs. The transactions in these systems are noted for being short-lived and only touching a small subset of the database at a time. But users increasingly want to execute analytical operations on these front-end DBMSs to do data-driven decision making in real-time, instead of having to wait for updates to propagate to a separate data warehouse. The system architectures of these DBMSs are not designed for such workloads: if the application invokes a complex query or deploys a continuously running machine learning algorithm that needs to access the entire database, then the DBMS must block all other work while these operations are running. This is because the DBMS needs to ensure that these operations have a consistent view of the database. In the case of H-Store, one could add back an onerous concurrency control scheme to allow these operations to interleave with the application's regular workload, but this will reduce the performance of all transactions. Further problems arise when dealing with exploratory data analysis applications that need to perform multiple scans across the entire database or if the machine learning algorithm needs to store intermediate results.

I contend that new approaches are needed for allowing users to execute real-time analytical and machine learning algorithms directly on a front-end DBMS, like H-Store or MongoDB. In particular, I will investigate techniques like dynamically relaxing consistency guarantees of the system to allow the DBMS to execute these operations with higher throughput. For example, instead of locking the cluster when an analytical query needs to scan an entire table, the DBMS could split that query into smaller tasks that are scheduled independently at partitions within a guaranteed window of time. We have already experimented with running MapReduce-style jobs on H-Store in this manner and our preliminary findings show that we are able to retrieve results without slowing down the system. The challenging part of this problem is coming up with the methods to allow the DBMS to automatically identify when it is acceptable to use a more permissive consistency policy. More research is also needed to better understand the trade-offs between accuracy and performance, how to accommodate failures, and what the best programming model is for machine learning and exploratory data analysis operations on transactional systems. Although there are recent attempts at solving this problem, the proposed methods are insufficient because they only support single-node systems with databases that are smaller than the available amount of memory.

Non-Volatile Memory

Just as changes in the number of cores per CPU and the reduction of memory prices enabled H-Store's departure from a traditional DBMS architecture, the onset of non-volatile memory (NVM) devices will require us to rethink the dichotomy between memory and durable storage. These devices promise to overcome the disparity between processor performance and DRAM storage capacity limits that encumber data-centric applications.

I believe that there are several avenues for research with these NVM devices. Foremost, I am excited about the potential of resistive NVM devices to dynamically change their storage area into executable logic gates. I plan to explore integrating machine learning components into a DBMS to automatically enable these executable configurations in NVMs. For example, if a user is sequentially scanning large segments of DNA data, then the DBMS could migrate the processing logic for identifying interesting sequences from the application down into the NVM. Once again, the research challenge lies in how to enable the DBMS to identify when it is appropriate to perform this optimization and to understand the trade-offs. I will also explore partitioning techniques that consider locality when storing data in these NVM devices, since the difference between accessing data on the same CPU socket versus a different socket in the same node will be significantly greater than it is now. My hypothesis is that NVM-based systems that account for these types of problems will greatly outperform existing DBMSs because they do not need to copy data from storage to the CPU.

As part of our collaboration with Intel for the Big Data Initiative at MIT, we plan to study NVM technologies in 2013. With this, we will begin our initial foray into understanding the performance characteristics of NVMs in the context of big data systems and build the groundwork for new DBMS architectures.

References

[1] H-Store: Next Generation OLTP DBMS Research. `http://hstore.cs.brown.edu`.

[2] C. A. Curino, D. E. Difallah, A. Pavlo, and P. Cudre-Mauroux. Benchmarking OLTP/Web Databases in the Cloud: The OLTP-bench Framework. In *CloudDB*, pages 17–20, 2012.

[3] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. J. Abadi. H-Store: A High-Performance, Distributed Main Memory Transaction Processing System. *Proc. VLDB Endow.*, 1(2):1496–1499, 2008.

[4] A. Pavlo, C. Curino, and S. Zdonik. Skew-Aware Automatic Database Partitioning in Shared-Nothing, Parallel OLTP Systems. In *SIGMOD*, pages 61–72, 2012.

[5] A. Pavlo, E. P. Jones, and S. Zdonik. On Predictive Modeling for Optimizing Transaction Execution in Parallel OLTP Systems. *Proc. VLDB Endow.*, 5:85–96, October 2011.

[6] 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. In *SIGMOD*, pages 165–178, 2009.

[7] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin. MapReduce and Parallel DBMSs: Friends or Foes? *Communications of the ACM*, 53(1):64–71, 2010.