# Conference Reports

**In this issue:**



Every two years, 75–100 systems, database, and application developers and researchers gather at Asilomar for the Workshop on High Performance Transaction Processing Systems (HPTS). The name, something of a misnomer, stems from its origin in 1985, when Jim Gray, Dieter Gawlick, Andreas Reuter, and other luminaries invited practitioners and academics to discuss the challenges and successes in the area of large, scalable, high-throughput systems. Today, I think of HPTS as the place where people with large-scale problems come to talk with people who like to solve large-scale problems. The crowd is a seamless blend of researchers and practitioners and infrastructure suppliers and consumers.

Each HPTS seems to have a dinstinctive flavor. A few years back, it seemed that all the large online service providers were talking about how they used Lucene to solve large-scale problems. This year, there was a lot of talk about integrating NoSQL solutions into large-scale services. HPTS feels a lot like HotOS, but with more emphasis on data and less emphasis on operating systems. This year Rik Farrow went to HPTS to soak in the ambience, learn a bit about the community, and coordinate student scribes so that we could bring a taste of HPTS to the USENIX community. Unlike many USENIX workshops, HPTS is not based on paper submissions; the written record mostly consists of personal blog postings, a collection of presentations, and some less-than-one-page submissions. These reports are the closest thing you'll find to an HPTS proceedings, although Web surfing will reveal several personal blog reports.

*—Margo Seltzer, USENIX Acting Executive Director*

## 14th International Workshop on High Performance Transaction Systems (HPTS)

Pacific Grove, CA
October 23–26, 2011

### Datacenter Trends 101

*Summarized by Eugenia Gabrielova (eugenia.g@uci.edu)*

#### *Internet-Scale Datacenter Economics: Where the Costs & Opportunities Lie*

James Hamilton, Amazon

James kicked off HPTS by saying it is his favorite conference, primarily because of the people in the room. He then claimed there has been more innovation in the past five years than in the previous fifteen, primarily due to advances in cloud computing and the accessibility it provides to application developers. Datacenters are expensive and don't really help innovation—when you are spending millions or billions of dollars, you do things the way you know it will work.

At Amazon, there are always multiple datacenters under construction. In the past four years, AWS has evolved into a phenomenal business generating tons of revenue and passing on savings to customers. Amazon was approximately a $2.7 billion annual revenue enterprise in 2000. Now, every day Amazon Web Services adds enough capacity to have support ed all of Amazon.com's infrastructure in the company's first five years. There is a competitive advantage in having better infrastructure.

The talk shifted to everything below the OS, because that is generally where the money goes. Charts often show people costs, but at a really large scale these costs are very minor relative to the costs of servers and power distribution. As a rule of thumb, "If you want to show people your infrastructure, you're probably spending too much." In the monthly costs of a datacenter, servers (not power distribution) dominate. However, server costs are decreasing, while networking costs are creeping up. Networking is a problem precisely because it is "trending up," so it is broken—this is a huge opportunity for innovation.

Another area with great potential for innovation is cooling systems, which have remained the same for about 30 years. Fans moving air is expensive, and moving water is also fairly expensive. Datacenters of the future could be designed beautifully with eco-cooling, no AC required. In the meantime, modular and pre-fabricated datacenters are regaining popularity, because of how quickly they can be deployed. Making datacenters better isn't just a technical advantage, it is an enormous business advantage.

Bruce Lindsay (Independent, ex-IBM) commented on the declining cost of network ports. Someone asked about OpenFlow, and James said that Google supports Quagga for routing, and OpenFlow comes from Stanford. Both open up the infrastructure by allowing the control plane to run centrally, with cheap hardware for running the data plane.

Someone noted that standard practice in the computer industry is to "prepare for the worst." James replied that there are test sites running with high-voltage direct current and many high-profile datacenters have very robust strategies for ensuring uptime (such as fully dedicated power generators). However, due to high demand, it can be hard to know which workloads will be running in a datacenter at a given time.

Slides from this talk can be found at http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_HPTS2011.pdf. James can be reached at James@amazon.com.

### *The Rise of Dark Silicon*

Nikos Hardavellas, Northwestern University

Dr. Hardavellas was unable to make it to HPTS this year but has made the slides for his talk available at www.hpts.ws/sessions/Hardavellas.pdf.

### *The Hitchhiker's Guide to Precision Time Synchronization*

Krishna Sankar, Egnyte

Before he become Lead Architect at Egnyte, Krishna was a Distinguished Engineer at Cisco Systems. In his free time he enjoys working as a technical judge for FIRST LEGO League Robotics. He began his talk by emphasizing that time synchronization is different from time distribution. There is incredible value in offering time precision in an application. Ocean observatory networks, industrial automation, cloud computing, and many other fields would benefit. Time synchronization is also slowly finding its way into routers and blade server fabrics.

Krishna gave an overview of IEEE 1588 v2 PTP (Precision Time Protocol), which concerns the sub-microsecond synchronization of real-time clocks in components of a network distributed measure and control system. This capability is intended for relatively localized systems, like those often found in finance, automation, and measurement. The purpose of IEEE 1588 is simple installation, support for heterogeneous clock systems, and minimal resource requirements on networks and host components.

PTP uses a master/slave model to synchronize clocks through packets over unicast and/or multicast transport. It follows a simple protocol: master and slave devices enabled with PTP send messages through logical ports to synchronize their time. Of the five basic PTP devices, four are clocks. Each clock determines the best master clock in its domain, including itself. It is very difficult to achieve high precision, so some hardware-assisted time stamping can be used to help accuracy (which is more complex than it sounds). A few key lessons in working with PTP are that shallow, separate networks are preferable; anything too hierarchical will prove difficult to manage and synchronize. Accuracy depends largely on hardware and software abilities and interaction. Additionally, GPS satellite visibility is needed for the GMC (Grand Master Clocks, the most accurate).

Krishna closed by encouraging audience members to submit to ISPCS 2012, which will take place in San Francisco. Learn more at http://www.ispcs.org. A central theme of the Q&A was whether these time precision techniques are accessible to the average application developer. How can an average application, subject to layers of virtualization and delays, take advantage of precision timing? The main takeaway was that, with some planning, developers can certainly take advantage of advances in time synchronization. The slides for this talk can be found at http://www.hpts.ws/sessions/Synchronization.pdf.

## Not Your Traditional Data Management Session

*Summarized by Andy Pavlo (pavlo@cs.brown.edu)*

### *Enterprise Supercomputing*

Ike Nassi, SAP

Ike began with a harsh denunciation and lamentation about current enterprise computing hardware, which supports only a single TB of DRAM on a single motherboard. That limitation makes it difficult for servers to be used for enterprise computing systems, because they often have a much greater working set size. Ike strongly believes it is time to re-examine our current predilection for shared-nothing architectures and that the database research community should take advantage of developments in high-performance computing research from the past 25 years, which has favored a shared-everything architecture. Large memory systems on the scale required by SAP are simply not being built; thus Ike sought to create one himself.

Ike presented a new DBMS server architecture, currently under development at SAP, which uses a virtual shared-

everything paradigm built on a single rack cluster. In SAP's new system, the database executes on a single instance of Linux, while underneath the hood the ScaleMP hypervisor routes operations and data access requests over networking links (i.e., no shared buses) to multiple, shared-nothing machines. By masking the location of resources through a coherent shared-memory model, Ike argues that they are able to minimize the amount of custom work individual application developers have to do in order their database platforms.

The early morning audience was languid, but several skeptics, such as Margo Seltzer, were concerned that the data links between machines would not match the speed of DRAM. Ike assured these doubters that high-performance communication links such as InfiniBand would be sufficient for this system. He also remarked that the system currently does not support distributed transactions; thus there is no message passing needed between nodes. Roger Bamford (Oracle) asked, why divide the system into so many cores, and Ike replied that they need the RAM. Adrian Cockcroft asked how common failures are. Ike said that this is a lab test so far, and in thirty days there were no failures. Margo Seltzer said she loved this project, which reminded her of late '80s shared memory multiprocessor systems such as Encore. Ike said that unlike the early systems, which used busses, their system is using fast serial connections, and he suggested that people not be blinded by what happened in the past. Both Margo and James Hamilton wondered about the problem of having a NUMA architecture, especially when the ratio of "near" memory to "far" memory reaches 10 to 1. Ike said that he lied, that all memory is used as if it were L4 cache. Roger pointed out the cost of going to the cache coordinator, and Ike replied that identifying the location of memory has a constant cost.

### *Forget Locality*

Randal Burns, John Hopkins University

Randal Burns, a systems research professor at Johns Hopkins, raised the issue that the canonical optimizations used in DBMS systems were insufficient to achieve high-performance data processing (i.e., > 1 million IOPS) on large and complex graph data sets. This is because any algorithm that must perform a scan of the entire data set or a random walk in the graph cannot take advantage of locality in the data. Thus, optimizations such as partitioning, caching, and stream processing are rendered impotent.

Randal then discussed ongoing work at Hopkins that seeks to understand the main bottlenecks that prevent modern systems from scaling to larger I/O operation thresholds. His work shows that low-level optimizations to remove lock contention and interference can improve throughput by 40% over file access through the operating system.

Margo Seltzer asked whether making certain assumptions about the physical layout of the graphs could be exploited. That is, could performance be improved if the system stored the data in a way that optimized for a particular processing algorithm? Randal responded that such techniques would be unlikely to work for attribute-rich graphs, since there is no optimal ordering. Roger suggested that he put the answer in their database and be done with it, eliciting laughter. Mike Ubell asked whether the cache was throttling IOPS, and Randal said yes, that there is lots of bookkeeping and page structures to manage. James Hamilton asked why not have the database use memory directly, and Randal said that is where they are going. They want to get away from local and global data structures. James pointed out that databases had already done this. Mohan asked about latches, and Randal replied that they want only locks that matter, such as a read lock on dentry and on mapping.

Someone suggested proper indexing, declaration of graph processing, having the database make decisions in advance. Randal replied that that is ground that has been trod before. Someone else pointed out that it seemed they were looking for storage memory that had DRAM-like characteristics. Randal agreed, saying that without a memory hierarchy his talk would be a no-op.

### *Flexible Hardware for Flexible Data Intensive Software*

Arun Jagatheesan, Samsung

Arun Jagatheesan from Samsung shared his perspective on new hardware trends and configurations for big-data systems and supercomputing platforms. He was specifically focused on the flexibility of both the hardware systems (i.e., allowing administrators to configure the hardware) and the software platforms that they support (i.e., allowing users to execute variegated workloads). Arun began with an overview of the flash-based Gordon system that he helped to develop while at the San Diego Supercomputer Center in 2009. Arun said that the three main lessons that he learned from this project were (1) not all the configuration options that one needs are available in hardware, (2) there is a nebulous tradeoff between flexibility and performance, and (3) manufacturers, applications, users, and administrators are unprepared for new hardware.

From this, Arun then introduced his more recent work on Mem-ASI at Samsung. Mem-ASI is a memory-based storage platform for multi-tenant systems that is designed to learn the access patterns and priorities of applications and react to them accordingly in order to improve throughput. Such priorities could be either service-level hints from applications, service-level requests from the computing platform's infrastructure, or simply how the individual application accesses data. This additional information could be used by the

system for more intelligent scheduling and resource management. Arun believes that such a model could both improve performance and possibly reduce energy consumption.

James Hamilton said he could understand the power savings, but not the factor of 4 for performance gains. Arun said that the idea is that you can change something on the memory controller to change what is happening at the transport layer. James asked if this had to do with the number of lanes coming off the core, and Arun replied that it is not about lanes but about what you can do behind those lanes.

## Mapping the NoSQL Space

*Summarized by Aaron Elmore (aelmore@cs.ucsb.edu)*

### *The NoSQL Ecosystem*

Adam Marcus, MIT

Adam Marcus provided a brief history of the origins of NoSQL, beginning in the late 1990s with Web applications developed using open source database systems. Applications that saw increased load began wrapping stand-alone DBMSes to allow for sharding to achieve scale. Additionally, relational operations were removed and joins were moved to the application layer to reduce costly database operations. These modifications led to the creation of databases that went beyond traditional SQL stores and came to be referred to as Not Only SQL (NoSQL). With a plethora of recent NoSQL options, Adam lightheartedly introduced Marcus's Law, which tells us that the number of persistence options doubles every 1.5 years.

The majority of NoSQL stores rely on eventual consistency and are built using a key-based data model, sloppy schemas, single-key transactions, and application-based joins. However, exceptions to these properties were highlighted, including alternatives to data models, query languages, transactional models, and consistency. For example, while many NoSQL databases utilize eventual consistency, many alternatives exist, such as PNUTS's timeline consistency or Dynamo's configurable consistency based on quorum size. With a basic understanding of NoSQL properties, real-world usage scenarios were outlined.

Recently, Netflix has undergone a transition from Oracle to Cassandra, to store customer profiles, movie watching logs, and detailed customer usage statistics. Key advantages that motivated the migration include asynchronous datacenter replication, online schema changes, and hooks for live backups. More information about this migration is detailed in Adrian Cockcroft's paper at http://www.slideshare.net/adrianco/migrating-netflix-from-oracle-to-global-cassandra. Contrasting Cassandra, Facebook chose HBase for the new FB Messages storage tier, primarily due to difficulties in programming against eventual consistency. HBase also provides a simple consistency model, flexible data models, and simplified distributed data node management. MongoDB usage for Craigslist archival and Foursquare check-ins were briefly highlighted.

After detailing NoSQL databases and use cases, Adam presented takeaways for the database community. First, and most contentiously, is developer accessibility. Adam said that the ability of a programmer to set up and start using a NoSQL db really mattered. Bruce Lindsay (ex-IBM) strongly objected to the question on "whether first impressions made within five minutes of database setup and use matter." Margo Seltzer (Harvard) countered that a new generation of developers, who use frameworks such as Ruby on Rails, do make decisions on accessibility and that these developers should matter. Adam furthered the argument by claiming accessibility will matter beyond minutes in schema evolution, scaling pains, and topology modifications. Database development should also examine the ecosystem of reuse found in some NoSQL projects. This is exemplified in Zookeeper, LevelDB, and Riak core becoming reusable components for systems beyond their initial development. Lastly, the NoSQL movement espouses the idea of *polyglot persistence*, where a specific tool is selected for a task. Selecting various data solutions can create painful data consistency issues, as an enterprise's data becomes spread among disjoint systems.

In closing, Adam presented several open questions. These focused on data consistency, datacenter operational tradeoffs, assistance for scaling up, the ability to compare NoSQL data stores, and next-generation databases. A question by C. Mohan (IBM) about the need for standardization of a query language drew mixed reactions.

### *The Present and Future of Apache Cassandra*

Jonathan Ellis, DataStax

Jonathan Ellis, of DataStax and a major contributor to the Apache Cassandra project, outlined developments in the recent version 1.0 release and goals for future Cassandra releases. Inspired by Google's BigTable and Amazon's Dynamo, Cassandra began as a project at Facebook before becoming an Apache incubator project. Cassandra's popularity is partly due to the ability for multi-master (and thus multi-datacenter) operation, linear scalability, tunable consistency, and performance for large data sets. Cassandra's user base today includes large companies such as Netflix, Rackspace, Twitter, and Gamefly.

For release 1.0 of Cassandra, leveled compaction was introduced to improve the reconciliation of multiversion data files. Advantages over the previous size-tiered compaction include improved performance due to lower space overhead and fewer average files required for read operations. Addition-

ally, individual nodes can construct local secondary indexes on columns; however, denormalization and materialized views are necessary to avoid join operations. Improvements mentioned but not discussed were compression, expiring columns, and bounded worst-case reads.

An interesting application that was developed for Cassandra was the ability for eventually consistent counters. Every node in the system maintains a list of counter values associated with each node. Any local modification for the counter performed only modifies the replica's value of the counter. To ascertain the value of the counter, all replica values are summed. This allows for concurrent modifications to the counter without needing synchronization between nodes.

Heavy optimizations were undertaken to improve read and write performance for Cassandra, including JVM tuning and garbage collection. Future advances in Cassandra will involve easing administration and use, improving the query language, support for range queries, and introducing entity groups. Pat Helland (ex-Microsoft) asked how to improve the performance of random reads for large data sets. Jonathan said a reliance on SSD would be needed to make significant gains. Someone wondered why Facebook had moved from Cassandra to HBase; Jonathan answered that it was mostly a personnel issue within Facebook. Mehul Shah (Nou Data) asked about the advantages of developing in Java. Jonathan said they included core consistency, memory management, immutable collections, and a rich ecosystem. The last question was about the largest install of Cassandra. Jonathan thought that it was around 400 nodes and 300 TB of data.

### *Oracle's NoSQL Database*

Charles Lamb, Oracle

Charles Lamb began the presentation on Oracle's latest data store with what NoSQL means to Oracle. A NoSQL database encompasses large data, distributed components, separation of OLTP from "business intelligence," and simplified data models, such as key-value, document stores, and column families. Lamb said that Berkeley DB alone does not meet all of these requirements and that the focus of the Oracle NoSQL DB is a key-value OLTP engine. Requirements for the database include support for TB to PB scale data sets, up to one million operations per second, no single point of failure, predictably fast queries, flexible ACID transactions, support for unstructured or semi-structured data, and the ability to have a single point of support for the entire stack, from hardware up to the application.

The system has multiple storage nodes, potentially residing in multiple datacenters, and is accessed by a jar deployed within the application. This jar, or driver, maintains information about the state of each storage node. Data is accessed using major and minor keys, and all records with the same major key are clustered on the same replication group of storage nodes. Operations are simple CRUD (create, read, update, and delete), read-modify-write (or compare-and-set style), and iteration. CRUD may operate on one or more records with the same major key. ACID transactions are provided but may not span multiple API calls. Iteration is unordered across major keys and ordered within major keys. Management and monitoring of the system are available through a command-line interface and Web-based console. Oracle's NoSQL database is built upon the battle-tested, high-throughput, large-capacity, and easy-to-administer Berkeley DB Java Edition/High Availability. Since Berkeley DB JE/HA was built for a single replication group, features such as data distribution, sharding, load balancing, multi-node backups, and predictable latency (which was highlighted as a difficult goal) were required to achieve better scaling.

Hashing a major key, modulo the number of partitions, identifies the group of nodes responsible for storing replicas of a data record; this group provides high availability and read scalability. The Rep[lication] Node State Table (RNST) identifies the best node to interface within a replication group. The RNST is stored at the driver and is updated by responses sent to the client. From the RNST the driver can determine a group's master, staleness of replicas, last update time, number of outstanding requests, and average trailing time for a given request. Replication is single-master, multi-replica, with dynamic group membership provided by election via the Paxos protocol. Durability can be configured at the driver or request level, and there are options for disk sync on both the master and replicas and replica acknowledgment policies. Consistency can be specified on a per-operation basis as well, with options to read from (1) the master, (2) any replica that lags no more than a specified time-delta from the master, (3) any replica that is at least as up-to-date as a specific version, or (4) any replica (i.e., with no consistency guarantees). The presentation concluded with an evaluation of the database's performance and scale-out capabilities.

Mohan asked about multi-node backup. They can do that, but it will not be consistent. As with Cassandra, they can take a snapshot for a consistent backup. Roger asked how they are supporting read-modify-write. Charlie said that the application does a get, does operations, then a put-if-version, and, conditionally, updates. Mohan wondered if reads are guaranteed to see the final versions, and Charlie answered that he would cover that later, but there are no guarantees.

There was vigorous discussion after Charlie finished. Mohan asked if the data and operations log were stored on the same disk. Margo Seltzer, who is also involved with Oracle NoSQL, said that they use a log-structured data store and that data and log are stored the same way. James Hamilton wondered if they could migrate off a node if it gets hot, and Charlie

replied, Not in this version. They do use hashing for even data distribution. Shel Finkelstein (SAP) asked about time-based consistency. Charlie explained that data is tagged with a Java-based timestamp. Mehul Shah wondered if they can continue operations after a partition, and Charlie said they could do reads but not writes without access to the master for that major key. Mehul then wondered if they can move partitions around and Charlie replied, Not in this release.

Someone asked if the drivers knew about all partitions. They get initialized on the first request and can connect to any replication nodes. Roger asked Charlie to describe their access control model. Charlie said that the assumption is that the system is in a DC, producing an "OMG" response.

## Big Data Experiences & Scars

### *Netflix Goes Global*

Adrian Cockcroft, Netflix

*Summarized by Hatem Mahmoud (hatem@cs.ucsb.edu)*

Adrian Cockcroft described the process of migrating Netflix to a public cloud in order to provide highly available and globally distributed data with high performance. The migration focuses on the control plane (e.g., users' profiles, logs), not the actual movie streaming, which is done using CDN. Amazon AWS was chosen as the public cloud to host Netflix's services because it is big enough to allocate thousands of instances per hour as needed. Adrian mentioned a remarkable idea in his presentation: the notion of design anti-patterns, that is, that design is better defined by undesirable properties than by desirable ones.

The Netflix migration involved a bi-directional replication phase in which data was replicated between Oracle and Simple DB, while backups remained in the datacenter via Oracle. Later on, replication of new account information to the datacenter was eliminated. Each data item is replicated to three different zones (i.e., different buildings with different power supplies within the same datacenter). This keeps all the copies close for fast synchronization. There is a tradeoff between recoverability and latency; to achieve the lowest latency a write operation must acknowledge once it is done on at least one replica, while to achieve the highest recoverability a write operation has to wait for all three replicas to be updated before acknowledging the user. The middle path is to use a quorum of two replicas. Overall, Netflix's data are distributed across four Amazon regions, plus a backup region. Remote replication can be also achieved through log shipping.

Backup is done by: (1) taking a snapshot (full backup) periodically by compressing SSTable and storing it to S3; (2) doing incremental compressed copying to S3 triggered by SSTable writes; or (3) scraping the commit log and writing it to EBS every 30 seconds. Also, there are multiple restore modes, multiple ways to do analytics, and multiple methods for archiving. Backups are PGP encrypted and compressed, with the lawyers keeping the keys for encryption. If S3 gets broken, they also make an additional copy to another cloud vendor.

Adrian pointed out that they find cloud-based testing to be frictionless. As an example, he asked a Netflix engineer to spin up enough Amazon instances to perform one million client writes per second. It took a couple of experiments to come up with the correct number of nodes, 288, to do this, and a total of two hours and about $500 of Amazon charges.

Margo Seltzer asked the size of their biggest database. It is currently 266 GB. Adam Marcus (MIT) asked if engineers had their own machines, to which Adrian replied that they used Jenkins for build testing,and had a special Eclipse plugin for working with EC2.

### *Towards Improved MySQL Scalability and Reliability*

Ryan Huddleston, RightNow

*Summarized by Rik Farrow (rik@usenix.org)*

Ryan described RightNow as a company that provides MySQL as a service. Located in Bozeman, Montana, the one-thousand-person company provides database services, on the company's servers, for over two thousand customers worldwide. The US military is one of their larger customers. RightNow uses the Percona Server MySQL port and has paid companies like Percona to add features to MySQL. In 2001 they paid to have the Innodb file-per-table feature added. They found they needed to switch from ext3, the default Linux filesystem, to XFS, because file deletion time was scaling with file size. Someone asked if this is still an issue with ext4, and Ryan said it was. James Hamilton asked if `create table` was an issue, and Ryan said it never had been an issue.

Ryan discussed their technique for migrating customers between shared servers when a customer's load becomes too great. James Hamilton wondered how they prevent a single customer from dominating a server. Ryan said they had a system that keeps track of load and can migrate a customer to another node. It keeps track of queries and can queue queries that will take a long time and move the queries from real-time to batch. Ryan said that their goal is to remain an open source company and that they plan to push all patches up to MariaDB (a branch of MySQL).

Bruce Lindsay asked whether adding a column requires them to delete a table. Ryan said it does. They add tables/columns on a slave server, move data in batches, cut over columns and tables, then drop tables and columns. Then they snap the customer to the slave and alter the master while doing updates. The entire process appears to occur with no delay for queries.

Someone exclaimed, "This was all fixed 25 years ago!" Ryan calmly replied that if they were doing this on Oracle, it would cost them $25 million a year. Instead it costs them $100,000 for support of MySQL.

### *Storage Infrastructure Behind Facebook Messages*

Kannan Muthukkaruppan, Facebook

*Summarized by Hatem Mahmoud (hatem@cs.ucsb.edu)*

Kannan Muthukkaruppan explained why Facebook has moved from Cassandra to HBase as a storage system for Facebook messages, the architecture used, and the lessons learned from that experience.

HBase is used to store small messages, message metadata (thread/message indices), and the search index, while large messages and message attachments are stored in Haystack. HBase was chosen for its high write throughput, good random read performance, horizontal scalability, automatic failover, and strong consistency. Besides, by running HBase on top of HDFS the system takes advantage of the fault tolerance and scalability of HDFS, as well as the ability to use MapReduce to do analytics.

Each of the datacenters that host Facebook's data is considered a cell that is managed by a single HBase instance. A cell contains multiple clusters, and a cluster spans multiple racks. Each user is assigned initially to a random datacenter, although the user may later be migrated to another datacenter via a directory service. Typically, a datacenter consists of several buildings. Thus each data item stored in HBase is replicated three times, in three different buildings.

The migration to HBase took more than a year. Shadow testing was used before and after rollout. To account for potential bugs, Scribe was used to write offline backups to HDFS, both locally and at remote datacenters. The developers had to introduce several modifications to HDFS to improve reliability, including sync support for durability, multi-column-family atomicity, several bug fixes in log recovery, and a new block replacement policy to reduce the probability of data loss. Also, to improve availability, the developers introduced rolling upgrades to account for software upgrades, online `alter table` to account for schema evolution, and interruptible HFile compaction to account for cluster restarts and load balancing. The developers also added several modifications to improve performance and solicit fine-grained metrics.

Someone asked whether they have an additional sharding layer on top of HBase. Kanna said yes, but that HBase only works within a single DC. Margo Seltzer asked if users are mapped to cells randomly. Kanna said yes and that they can migrate users later. Overall, they average 75+ billion read-write IOPS per day, with a peak of 1.5 million operations/second. Their load is 55% read and 45% write, over 6 PB of data (2 PB with three replicates), all compressed using LZH. Margo asked if they lose all the users within a DC if it goes down, and Kanna replied that they do offline backups to other DCs. Cris Pedregal-Martin asked if they had non-peak hours. Kanna answered that Monday between 12 and 2PM is their peak, so in a sense, yes. Adam asked if they planned on upstreaming their patches to HBase. Kanna said that they do, as most of what he talked about is open source.

Someone asked about network speed. Kanna said they use 1G at hosts and 10G at the top of racks. Mike Caruso asked what type of changes they made to the schema. Kanna said that making threads longer meant writing metadata back to HBase, so they fixed that as an example. Then he said there is lots more work to be done, such as fixing the problem of a single HDFS Name Node and having fast hot backups. Mohan asked if all users are mapped to US DCs, and Kanna said yes. Cris asked if they ever lose messages, and Kanna said that they don't know, but they do sample, and sampling looks good.

## Big Analytics

*Summarized by Michael Armbrust (marmbrus@cs.berkeley.edu)*

### *Big Data at eBay*

Tom Fastner, eBay

There are a number of important use cases for analytics over big data at eBay, spanning daily decisions such as A/B testing for experiences or treatments on ebay.com all the way to supporting long-term and multi-step programs such as the buyer protection plan. Tom Fastner described the architecture of their system and some of the challenges they have experienced operating at such a large scale (50+ TB/day of new data and 100+ PB/day processed by 7,500+ users and analysts).

Analytics at eBay is supported by three separate platforms, each with its own strengths but with some common capabilities. At the high end they run EDW (Enterprise Data Warehouse) systems based on Teradata for all transactional data, sharing it with a wide user base and supporting >500 concurrent requests per minute. For the application logs and other structured or semi-structured data, they use a low-end enterprise-class Teradata system. The world's largest Teradata installation (256 nodes, 36 PB of spinning disks able to hold 84 PB of raw data with compression) is supporting use cases on very large, but still structured, data. This platform is called Singularity. The dominating data use today is user behavior information. It also serves as a DR for the EDW data, as most of that data is required to be joined to the user behavior data for analytics.

The ability to easily work with semi-structured data is important for several reasons. First, the use of semi-struc-

tured data greatly simplifies the process of modeling the data and results in a system that is less vulnerable to changes. Additionally, the resulting de-normalization of the data can result in improved performance, as the data is already joined. Singularity enables processing over this semi-structured data by providing developers with SQL functions that extract individual items and sequences from the key/value pairs stored in a given row.

The final platform of their data analysis system is a Hadoop cluster running on 500 commodity nodes, used primarily for structuring unstructured data and for finding patterns that are difficult to express in SQL.

There is no silver bullet to cover all forms of analytics on a single platform. Integration across the three platforms is key. eBay deployed a self-service data shipping tool and is working on a transparent bridge between Teradata and Hadoop.

Mike Caruso asked if they had ever compared performance. It is not worth the effort; Hadoop is cheaper but less efficient than Teradata or EDW. Stephen Revilak (University of Massachusetts) asked how big their DBA team was. Tom said they had four DBAs and an offshore support contract.

### *Cosmos: Big Data and Big Challenges*

Ed Harris, Microsoft

Ed Harris presented Cosmos, a multi-petabyte storage and query execution system. Used in Microsoft's Online Service Division, Cosmos is designed for large-scale back-end computation, such as parsing data from Web crawls, processing search logs, and analyzing clickstream data. Cosmos is run as a service within Microsoft; users provide the data and queries to be run, without having to worry about the underlying infrastructure. At a high level, Cosmos is broken into three major layers: storage, execution, and the SCOPE language.

The storage layer is organized around the concept of *extents*. An extent is an immutable block of data, up to 2 GB in size. The storage layer automatically handles compression and ensures that each extent is replicated to three different extent nodes for fault tolerance. Multiple extents are concatenated to form a *stream*, and the storage layer is also responsible for maintaining the namespaces of available streams.

On top of the storage layer, the execution engine is responsible for taking a parallel execution plan and finding computers to perform the work. For better performance, the system ensures that computation is co-located with data when possible. The execution model is based on Dryad, which is similar to MapReduce but more flexible, since it allows the expression of arbitrary DAGs. The execution engine, by managing failures and restarting computation as needed, also shields the developer from some of the flakiness inherent in running jobs on large clusters of commodity machines.

Finally, at the top of the stack is the SCOPE language. Influenced heavily by SQL and relational algebra, SCOPE provides developers with a declarative language for manipulating data using a SQL-like language extended with C# expressions. The SCOPE optimizer, which is based on the optimizer found in Microsoft SQL Server, decides the best way to parallelize the computation while minimizing data movement.

Since Cosmos is a hosted service, it's important to allocate resources fairly among the system's many users. This is accomplished by defining the notion of a virtual cluster (VC). Each VC has a guaranteed capacity, but can also take advantage of idle capacity in other VCs. Within any given VC the cost model is captured in a queue of work (with priority).

Harumi Kuno from HP Labs asked Ed to elaborate on how they divide cluster resources. Ed responded that each VC is provided with tokens that represent some amount of processing cores, I/O bandwidth, and memory. Mike Caruso asked if they do any migration of data, and Ed said that they have, because they bring up or shut down clusters.

### *Scal(a)ing up Big Graph Analytics*

Tyson Condie, Yahoo! Research

Tyson Condie presented ScalOps, an embedded domain-specific language in the Scala programming language designed for running machine-learning algorithms over big data. ScalOps expands on current systems such as MapReduce and Spark by providing a higher-level language based on relational algebra that natively supports successive iteration over the same data.

A motivating example for their system is performing spam classification for Yahoo! Mail. Their first prototype used Pig to extract labels and generate a training set which was then used to train a model using sequential code. This code was executed repeatedly until a satisfactory model was found. Unfortunately, this process was suboptimal, for several reasons. First, since their tools did not natively support the iteration, they needed to construct a fractured workflow using Oozie. Second, the sub-sampling required to fit the data on a single machine hurt the accuracy of the classifier. Finally, copying data to a single machine can be very slow.

An obvious improvement is to parallelize the training algorithm. This, however, does not fit nicely into the MapReduce model. Thus, real-world implementations often involve using fake mappers that cross-communicate, eliminating many of the fault-tolerance benefits of the MapReduce model. While systems like Spark provide an improvement over such practices, by allowing users to explicitly cache data for subsequent iterations, explicit caching is a point-solution that limits opportunities for optimization. In contrast, ScalOps is a Scala DSL that is capable of capturing the entire analytic pipeline. It supports Pig Latin and has a looping construct

that can efficiently capture iteration. It runs on top of the HyracksML + Algebricks runtime, which provides the system with a relational optimizer and a data-parallel runtime.

Ed Harris (Microsoft) asked how they knew when iteration for a given algorithm was complete. Tyson replied that for global models the UDF would specify completion, and for local models computation terminates when all messages stop. Mike Stonebraker asked why they didn't use R, given its popularity with analysts. Nothing in their system precludes the use of R UDFs. Mike Caruso asked how opaque UDFs are and if this is a problem for optimization. There is no visibility into the UDFs, but the looping construct can look at the underlying AST and perform algebraic optimizations.

## Consistency Revisited

*Summarized by Aaron Elmore (aelmore@cs.ucsb.edu)*

### *Eventually Consistent Is Eventually Not Enough*

Mehul Shah, HP

In an analysis of eventual consistency, Mehul Shah shared experiences and insights with building a large distributed key-value store at HP. Some applications need scalable solutions to support high availability and globally distributed data. Traditional DBMSes have limited scalability, due to their consistency requirements, resulting in the creation of NoSQL databases that dropped ACID and traditional models to achieve scale. This equates to traditional databases providing CP and NoSQL databases providing AP. Eventual consistency then becomes the standard tool to enable availability in a distributed environment. There are two myths about NoSQL today: eventual consistency is enough, and adding stronger consistency later is easy.

Shah described a database built at HP as a geo-distributed, highly available, large object store that supports a large user base and versioned keys by use of timestamps. Conceptually the database is similar to S3, with unique accounts owning multiple buckets, and each bucket having a unique name and containing many objects. Buckets have unique ownership and can be shared with other users via bucket permissions. With this context, two partitioned users attempting to concurrently create the same bucket is a conflict simplified by strong consistency, whereas eventual consistency for metadata operations, such as bucket creation and deletion, could result in a user viewing unowned objects.

To facilitate strongly consistent operations and prevent undesirable situations, the Atomic Conditional Update (ACU) was introduced as a primitive for achieving consistency. Multi-key and atomic get/test/put are operations that ACU needed to satisfy with a single RPC call. Pat Helland asked whether this was effectively concurrency control, or a strongly consistent tool that can be used for optimistic concurrency control. Shah answered that larger transactions could be built out of ACU primitives if needed.

Not all operations need the strong consistency of ACU, so applications can mix strong and weak consistency operations for the same data store. This introduces subtle interactions, such as weakly consistent operations that are not serialized against strongly consistent operations. Developers are not used to thinking about these interactions, and that typically results in workarounds in higher layers. Armando Fox (UC Berkeley) asked if the operations are not serializable, due to core operations checking different targets. Shah answered that they are serializable, because a read-write dependency exists between the operations. If they were strongly consistent operations, they would be serialized by the system.

With the assumption that partitions will occur, CAP presents a choice between consistency and availability. However, the terms in CAP are not crystallized. Consistency could include notions of recency, isolation, or integrity. Availability could encompass uptime, latency, or performance. Partition toleration could be supporting a single node or a minority partition. Shahed claim that CAP is not a theorem to be applied, but more of a principle. With the many semantics that exists for consistency and availability, an ideal single system should support various consistency options that span a spectrum from consistency (transactions) to availability (eventual consistency). This was likened to isolation levels, which are easy to understand, configurable, and compatible.

Several options exist when adding consistency to a weakly consistent system. Layering services, coordinated components, and an integrated approach are techniques to provide a consistency primitive, such as ACU, to a weakly consistent database. The integrated approach still requires insight into mixed consistency operations, but these complexities are abstracted from application developers. Shah said that if you are starting over, your system design would benefit from relaxing a strongly consistent core rather than strengthening a weakly consistent core. Eventual consistency is required and, at the same time, is not enough, and now is the time to rigorously examine our understanding of consistency.

### *Flexible OLTP Data Models in the Future*

Jags Ramnarayan, VMware

Jags Ramnarayan presented his view of the future of OLTP databases. He noted the high demand that exists currently for databases that can support low latency, predictable performance, graceful handling of large spikes in load, big data support, and in-memory operations on commodity hardware. Data input is increasingly trending toward streaming and bi-temporal behavior. Additionally, rapid application development requires a more flexible schema to support frequent changes. Most significantly, while a single database

instance is ACID, rarely does ACID hold for enterprise-wide operations. This results in data silos and duplication across databases, so enterprises must live with cleaning and de-duplication of data. Jags concluded that people actually do not want ACID but, rather, deterministic outcomes.

Having outlined trends, Jags provided a brief overview of VMware's GemFire and the similar SQL-interfaced SQLFire. GemFire is a highly concurrent, low-latency, in-memory and distributed key-value data store. Keys and indexes are stored in memory, with persistence of the data handled by compressed rolling logs. Tables can be partitioned or replicated, with replicas acting as active-active for reads, but using serialized writes by a single master for any given key. Distributed transactions are supported, but effort is undertaken to prune queries to a single partition for co-located transactions. Jags mentioned that GemFire supports asynchronous WAN replication and a framework for read-through, write-through, and write-behind operations; however, no details were given.

While hash partitioning typically provides uniform load balancing, databases should exploit OLTP characteristics to go beyond key-based partitioning. Jags made a key observation: the number of entities typically grows, not the size of each entity. Additionally, access is typically restricted to only a few entities. If related entities can be grouped, they can be co-located and thus minimize the number of distributed transactions. To build entity groups, compounded primary keys should be constructed, using foreign keys to capture relationships between entities. Grouping will largely prune operations to single-entity groups that are co-located, allowing for scalable cluster sizes, transactional write sets on entity groups, serializabilty for entity groups, and joins within a group. This does not eliminate distributed transactions across entity groups, since access pattern complexity invariably goes beyond grouping semantics. Despite the promise of hashed keys and grouping, hotspots and complex queries create difficult scenarios for "partition aware" designs. Smart replication of reference data, which is frequently joined with partitioned data, can help with this.

Looking forward and beyond the traditional SQL models, Jags described a *polyglot* OLTP database. This multi-purpose data store should support (1) continuously changing, complex object graphs, (2) structured transactional data, and (3) flexible data models, such as JSON.

### *Inconsistency and Outconsistency*

Shel Finkelstein, SAP, and Pat Helland

Shel Finkelstein's presentation focused on approaches to handle views based on inconsistent data sources. He began with a quote by F. Scott Fitzgerald, "The test of a first-rate intelligence is the ability to hold two opposed ideas in the mind at the same time, and still retain the ability to function." Similarly, the goal for database systems should be to maintain functionality despite having potentially inconsistent data sources. To understand whether data is inconsistent, a complete view of the data's context may be required. A set of weather measurements without location or time may seem like inconsistent data but is simply lacking event details and provenance. After challenging notions of consistency, Shel provided Jim Gray's definition of consistency: "A transaction is a correct transformation of the state. The actions taken as a group do not violate any of the integrity constraints associated with the state."

Data inconsistencies can occur for a variety of reasons. First, inconsistencies can derive from integrity constraint violations, such as impossible address information, corrupted-entity foreign relations, violated business rules, or domain constraints. Second, logical impossibilities can occur. This can include unanticipated data unknowingly being transformed, incomplete data, or real-world data contradictions, such as a location changing that clearly could not relocate. Third, replication issues include asynchronous data feed corruption and relaxed consistency models for replication protocols. Fourth, many databases run with read committed as the default isolation level, which does not guarantee serializability and can result in data inconsistencies. Normann and Ostby's *A Theoretical Study of Snapshot Isolation* in EDBT 2010 was given as a reference for inconsistencies that can arise even when using snapshot isolation.

In an ideal world inconsistencies in data would not exist, but databases reside in the real world and need to handle inconsistent data sources. Every application operates with assumptions about the consistency of data, and disjoint applications can make different assumptions about the same inconsistent data source. Shel introduced the concept of *outconsistency*, which involves providing an "outwardly consistent view of the data" and guidelines for how applications should operate on inconsistent sources. This provides a regimen that enables different applications to operate on the same data source with some understanding of methods for dealing with inconsistencies.

Approaches for addressing outconsistency were defined as the following techniques, which are likely to be used in combination. *Preventing inconsistent data* provides an identical view to the data, relying on approaches such as strong integrity constraints, checking business rules, and utilizing "transactional intent" (CIDR 2011) to prevent inconsistencies at the source. *Tolerating inconsistent data* utilizes expected inconsistencies to transform data for the outward view. *Ignoring inconsistent data* filters the outward view to include only data consistent for the purposes of the application. *Fixing inconsistent data* requires the application to take an active role in correcting the inconsistencies in the source

data. For each approach a set of challenges were discussed. A final claim was that all data, and subsequently transaction processing, consists only of events, reports, and decisions. Transactions and consistency should be discussed in this context. The work presented is just the beginning of examining the fluid relationship between applications and data.

Graefe Goetz wondered whether "kicking the can down the road" really means "eventually consistent"? Shel countered that "kicking the can down the road" means that something else deals with it. It can be data cleansing applications, services with alerts, interpolation and extrapolation, or renewal processes, such as SAP's APO. Roger Bamford (Oracle) asked if Shel would consider compensating transactions, and Shel said that this fits into this category. Shah said that he had experience with fixing these problems, and that it comes down to cost, earlier versus later. Can you comment a little on costs? Shel replied that the tradeoff has to do with coping with inconsistencies or fixing them. Mike Caruso mused that you could have an outconsistency in one system that would be an inconsistency in a second. Shel concluded by asking the audience to consider whether his factoring is correct, and if it is whether we should write applications based on it.

## It May Be Fast, But Is It Right?

*Summarized by Yingyi Bu (yingyib@ics.uci.edu)*

### *Debugging Designs*

Chris Newcombe, Amazon

Chris Newcombe presented a model checking–like approach for finding bugs at the design stage. He used Stoica et al.'s 2001 Chord paper as an example. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications" is one of the most cited computer science papers (8,966 cites as of Nov. 11, 2011). However, Pamela Zave from AT&T Research recently found eight major defects in the Chord ring membership protocol, using exhaustively testable pseudo-code (written using Alloy). The testable pseudo-code is remarkably simple. The example reveals that even top work done by the best people and reviewed by very smart peers can still have bugs! In particular, those systems bundling concurrency control, recoverability, failure handling, and business logic together are very hard to debug. However, test tools can help.

Chris proposed that pseudo-code should be written in a support tool rather than only in design documents, and people should use the tool to do exhaustive testing on the pseudo-code so that bugs could be found at the design phase. He proposed that TLA+ and PlusCal should be the pseudo-code language, and he used Michael J. Cahill's SIGMOD 2008 paper "Serializable Isolation for Snapshot Databases" as a running example to show how to get testable pseudo-code from the pseudo-code in the paper. Finally, Chris showed that the TLA tool can automatically check Cahill's algorithm. Chris also suggested that the audience read Lamport's book *Specifying Systems* as well as *TLA+ Hyperbook* (http://research.microsoft.com/en-us/um/people/lamport/tla/hyperbook.html).

Margo Seltzer said that the testable pseudo-code actually is a specification, and TLA+ could be thought of as a specification language. Chris replied that the word "formal" is like death for many people, and he steers away from using that word. He claimed to be an escaped video game programmer who has never proven anything in his life. Mike Caruso asked if Lamport's TLA+ can generate code to the state machine level, and Chris replied that Lamport designed his tool to be very expressive declaratively, but it cannot be used to compute. Adrian Cockcroft wondered why he couldn't find Lamport's book at Amazon, and Ernie Cohen replied that the PDF is available for free.

### *Verifying Real-World Transaction-Processing Code with Microsoft VCC*

Ernie Cohen, Microsoft

Ernie Cohen argued that testing sucks, and, instead, deductive verification should be widely used in production software development. He proposed that programmers should be able to write contracts such as pre-conditions, post-conditions, and invariants in their code. Therefore, verification could be done in a program-centric way. Ernie clarified that the cost of deductive verification should be comparable to complete functional testing.

After giving the high-level vision, Ernie briefly introduced VCC (Verified Concurrent C), which was developed by his group. VCC allows programmers to annotate their original C code with contracts. Then he did a live demo to show how VCC can find bugs in a C binary search program. Ernie added several preconditions and invariants as annotations to the code, and then bugs such as race conditions, buffer overflow, and value overflow were quickly caught. After the live demo, Ernie illustrated several useful constructs in VCC, such as data invariants, ownership, and ghost data and code. Data invariants are invariants on objects and can be defined as part of type declarations. Ownership is mostly used for specifying the contracts of concurrent reads/writes. Ghost data usually represent abstract states, while ghost code is actually executable contract and only run at verification time. Ernie also showed how to add annotations such as invariants and ghosts to make a piece of lock-free, optimistic, multiversioned transaction processing code verifiable. Finally, Ernie said that they should add prophecy support in VCC in order to verify properties such as "whether a timestamp obtained from the DB will be its final timestamp."

Armando Fox (UCB) asked if VCC can handle runtime polymorphism. Ernie pointed out the C includes runtime polymorphism, such as function pointers. Armando asked if VCC works for languages other than C. They may port it to C++.

### *Data Without Provenance Is Like a Day Without Sunshine*

Margo Seltzer, Harvard University

Margo Seltzer argued that provenance is playing an increasingly important role in computer systems. It is the "how, when, why" metadata about the data. She used Wikipedia revision history to illustrate provenance: by looking at the editors historically, one can gain someconfidence about the Wikipedia content. Provenance can come from instruments, application software, system software, or software tools. Provenance reminds people what has happened and gives people a way to understand why something happened. Margo pointed out that nowadays provenance is usually managed manually, implied, embedded, or part of a workflow system.

Margo emphasized that provenance is everywhere! Every day, people ask questions such as "Why does Facebook recommends this ad to me?" "Where does this file come from?" "What did the customer do before she hit this bug?" Margo advocates that provenance be built into every system in a layered way. The key concept there is that each layer collects provenance and each layer associates its provenance objects with both upper- and lower-layer provenance objects. The example systems Margo's group has built include a provenance-aware storage system, simple provenance in PostgreSQL, and a provenance-aware Python workflow engine.

Rusty wondered why the person who wrote an algorithm couldn't supply provenance, and Margo said she wants the algorithm to include the generation of the provenance, so that the information generated can be used to improve the algorithm. Pat Helland said that machine learning is like Mulligan stew, it's "ginormous," and Margo agreed. But Margo said she still wants everything, which is why disk vendors love her. Jim Waldo (Harvard) said that for non-disk vendors, transporting all the provenance data will not be wonderful (or cheap). Margo pointed out that this is HPTS, so with a provenance handle you can make distributed queries on replicated stores wherever you want. Margo's group has worked on how much provenance you are likely to want. Jim asked if this was a ratio of provenance to data. Margo answered that it depends. Someone asked if provenance was like using CVS, and Margo replied that CVS is "the poor man's provenance."

## Trusting the Cloud

*Summarized by Steve Revilak (srevilak@cs.umb.edu)*

### *Clouds & Condos*

Pat Helland

Pat asked, "What can condos teach us about cloud computing?" Quite a few things! Condos place constraints on living environments, but they also provide benefits: for example, most repair and maintenance work is taken care of for you. One can take advantage of these benefits, as long as there's a willingness to live within the constraints. Similar trends can be seen with other types of buildings. Retail space and office parks are built with a notion of how the space will be used but without knowledge of who will be using them. This allows building developers to support a wide variety of tenants; they build to a common set of usage patterns and impose a few constraints on what the building tenants can do.

Cloud computing can develop along similar lines. Cloud computing can provide basic services, such as stateless request processing, session management, load balancing, provisioning, and scalability. These services may not fit the needs of every conceivable cloud user, but they will fit the needs of *most* cloud users. We can design cloud computing systems according to common patterns of use, just as we do for buildings, even if we don't know who the cloud's users will be.

Laws and norms governing landlord-tenant relationships have evolved over time and work to the advantage of both parties. Pat believes we could benefit from a set of common rules that govern cloud providers and cloud users. Such rules would provide fair treatment to users and offer protection to service providers.

In summary, our relationship with buildings has changed over time. As we've done with buildings, we need to develop usage models and constraints, as well as rights and responsibilities, governing the cloud computing environment.

Mike Caruso pointed out that customers will need to tell the cloud providers what they want/need. Pat agreed, but said that we already know some patterns. Someone pointed out that it took many years for landlord-tenant law to evolve. Armando Fox said that he already uses Heroku, and it provides many of the things he wants.

### *Go Fast and Don't Break Things: Ensuring Quality in the Cloud*

Scott Hansma, Salesforce.com

Salesforce began life as a CRM application. It has evolved into a full-blown development platform that conducts 575 million transactions per day. All Salesforce customers run in a hosted environment, and all customers use the same

version of Salesforce's software. This scenario makes quality control extremely important; upgrades must work for all users, and upgrades cannot break functionality users have come to depend upon.

Salesforce is intensely focused on software quality, and this commitment to quality manifests itself in several ways. Salesforce uses a continuous integration (CI) system to test changes as they are committed to their source code repository. This CI system runs 150,000+ tests in parallel across many machines, and it will do binary searches across revision history to pinpoint the precise check-in that caused a test to fail. Developers do not get off easy—once the CI system has identified an offending check-in, it will open a bug for the developer to address the problem.

Salesforce allows customers to customize their applications with a programming language called APEX. As a best practice, Salesforce requires customers to test their APEX code prior to deployment. These customer-written tests provide an excellent way to regress new releases; Salesforce can run customer-written tests against new releases to identify problems prior to deployment. (Salesforce developers are given access to information about failing customer-written tests, but they are not given access to the underlying customer data.)

Finally, Salesforce maintains a Web site (http://trust.salesforce.com) where they publish availability metrics and service announcements. The company believes that this transparency—publishing their uptime metrics—helps to promote user confidence in the platform and keep the company focused on quality.

### *A Non-Proprietary Social Internet*

Monica Lam, Stanford University

Cloud computing offers a long list of benefits, but that list does not always include privacy. Take Facebook as an example: Facebook provides a great user experience, and a great platform for application development. But Facebook is also a social intranet—all interactions pass through Facebook's servers, and Facebook controls access to user data. Monica believes that social networking should be more like email. Two people can exchange email without having to use the same email provider, so, why should two users need to use the same social network provider in order to have social interactions?

Monica presented an application called Musubi, to demonstrate how open social networking could work. Musubi is a mobile application that runs on the Android platform and permits peer-to-peer social networking. Social networks are created through the users' address book and do not require a central service provider. This platform preserves privacy by eliminating the need for a central provider (all of your data lives on your mobile phone) and by encrypting communications. During the talk, Monica set up a social networking group for HPTS attendees; several people joined and began exchanging messages.

Monica also demonstrated how smartphone applications could be turned into collaborative social applications. She presented an application called We Paint, which is a collaborative drawing application.

Stanford has conducted several usability studies with Musubi. The reactions have varied by age group. Some adults believe that this is the future of social networking. College students were indifferent; they preferred to use Facebook and found nothing new and attractive in Musubi. Elementary school students were the most receptive; they thought Musubi was "awesome."

An attendee who worked at Facebook was very upset with Monica for suggesting that Facebook might sell user data. Monica said that people should be free to use Facebook if they want to, but she also believes that users should have the freedom to use different social networking platforms if they choose to do so.

## Debate Panel: Scale Up vs. Scale Out

*Summarized by Pinar Tozun (pinar.tozun@epfl.ch)*

Panelists: Michael Stonebraker, MIT; Mark Callaghan, Facebook; Michael Cahill, Wired Tiger; Andy Gross, Basho

The debate panel of this year's HPTS was about whether to focus on scaling up, utilizing a single node in the system with useful work as much as possible, or scaling out, increasing the performance of the system by adding more machines. A node was initially defined as a single processor by the panel but during the panel it was sometimes also referred to as a single multiprocessor machine. The panel chairs, Margo Seltzer (Harvard) and Natassa Ailamaki (EPFL), had a slider where 0% indicated total focus on scaling up and 100% indicated focusing only on scaling out. The chairs asked the panelists where on this slider they stood while building their systems.

Michael Stonebraker chose 15% on the slider. Focus on using all the cores available in your processor as efficiently as possible first, but also think about how to scale out: unless you have both in your database today, you are not going to be successful. One size does not fit all. Different markets should optimize their systems for their needs. In OLAP (online analytical processing), for example, a column-store beats a row-store, and you need clever overhead cleanup and, in scientific databases, array-based designs. He emphasized getting rid of the shared-data structures (buffer pool, B-trees, etc.),

locking, and latching bottlenecks in database management systems in order to scale up to many cores in a node, as they do in their VoltDB-related work. He claimed that Facebook could have done what they are doing with a 4000-machine cluster with only 40 machines if they had been using VoltDB. He believes that in the next 20 years: there will be around five gigantic public cloud vendors, so we should trust the cloud and try to adapt our systems to its environment.

Mark Callaghan picked 90%. He leads the MySQL engineering team in Facebook; they run the MySQL at Web scale across many machines. He tried to answer why they were using that many machines to handle Facebook's workload. They know they can never be working on a single node with Facebook's enormous scale, so as long as the software is efficient enough, they focus on how to scale out rather than how to scale up. He believes their market requires a focus on scaling out. They are mostly I/O bound and not CPU bound, and he does not think using database designs focused on in-memory databases, such as VoltDB, will help them. To have better IOPS and provide lower latency to their customers they need to buy more machines and think about how to scale out.

Andy Gross put his choice at 70% , arguing for focusing on both but favoring scaling-out. His background is in distributed systems; he likes Dynamo-like systems. Naturally, he said, he is interested in more than one machine. He also pointed out different ways of scaling out and scaling up: not just thinking how to use one node or more machines better but also dealing with how to exploit different technologies, such as SSDs, GPUs, and FPGAs. He also said that some people who do not have the choice of using specialized solutions need to use general-purpose products. Public cloud environments such as EC2 are good spaces as general-purpose solutions, and they also try to address problems related to power and energy consumption for general-purpose systems.

Michael Cahill chose 0%. Working inside the storage engine, he argued that we need to revisit the assumptions we make in storage engines to ensure serializable isolation among transactions. We have to find non-blocking algorithms to get the best out of a single processor. People mostly focus on scaling out across multiple machines, but he wants to make contributions within a single node. People should design their software in a way that it will work well on new hardware. He said that big companies such as Facebook have the luxury to focus on scaling out, but smaller companies should focus on the storage engine first and do their best to scale up there.

Natassa Ailamaki asked for an example of a technique that is good for scaling up but not good at all for scaling out. Michael Cahill answered that optimistic concurrency control is definitely a good technique for scaling up in a single node, but since it is hard to coordinate across nodes, it is not good for scaling out. Natassa wondered whether buying more machines to have more IOPS is a waste of machines and power. Mark said if they tried to get rid of the disk and be in-memory they would need 10 times as many machines as they have now. Michael Stonebraker said that if they are I/O bound, they should use what is optimal for the data-warehouse market, have column-stores with better compression, and reduce their I/O load. Mark argued that having compression does not reduce the number of IOPS you need linearly, and it does not solve the random I/O problem. On the other hand, Margo Seltzer opted for focusing on scaling up first: if you have a system that cannot saturate the memory and CPUs you have, then you have a badly built system.

All the panelists thought open source products were great. However, Michael Cahill said that his team cannot maintain an open source product for their own needs. Open source products such as MySQL also might end up having so many versions around that it will not be clear which one should be used. However, Mark Callaghan argued for MySQL, since there is one main MySQL version and only two or three other versions to choose from.

Someone asked, If you have a 160-core machine, how is scaling up within that machine different from scaling out? Michael Cahill said it is the same, but there are more failure cases when you have more machines. Andy Gross said that such a machine will make you use ideas from distributed systems in a single machine. Another person asked what academic researchers should focus on. Andy Gross argued that the interesting papers are the applied ones and they are mostly about scaling out. Michael Stonebraker advised that academics should talk to real customers, understand their problems first, and then try to solve them in their research. What should we do if we had non-volatile main memory? Andy Gross said that SSDs can be thought of in that way.

There was discussion about the NoSQL databases and database knobs that require DBAs. Mark Callaghan argued that NoSQL systems are good, because even though they do not focus on performance they are easier to manage, and that is what matters for some users. Andy Gross also supported NoSQL systems. On the other hand, C. Mohan (IBM) argued that NoSQL systems made users optimizers of databases, and Stonebraker argued that SQL provides an abstract layer on top of a database for their customers. Stonebraker also supported the need to get rid of as many knobs as possible so as not to be dependent on the database vendor and the DBAs. However, Armando Fox (UC Berkeley) argued that as a customer he would prefer not knowing about the database design details and that people who can tune are good if they know how to do it well.

## New Age OLTP

*Summarized by Pinar Tozun (pinar.tozun@epfl.ch)*

### *All the Rules Have Changed*

Michael Stonebraker, VoltDB

Michael Stonebraker started his talk by categorizing the OLTP market: OldSQL, NoSQL, and NewSQL. OldSQL represents the major RDBMS vendors, which Stonebraker called the "elephants." They are disk-based and use ARIES (Algorithms for Recovery and Isolation Exploiting Semantics), dynamic record-level locking, and latching mainly to ensure ACID properties. On the other hand, NoSQL is supported by around 75 companies today and favors leaving SQL and ACID. Finally, NewSQL suggests not leaving SQL and ACID but changing the architecture of the traditional DBMS used by OldSQL.

He pointed to an earlier study that shows that in OldSQL-like databases only 4% of the total CPU cycles go to useful work, with the remainder almost evenly shared by latching, locking, logging, and buffer pool management. He argued we cannot benefit much from trying to optimize those architectures, so a redesign of the DBMS architecture without compromising SQL and ACID is needed to increase the percentage of the useful work. This is what NewSQL databases are trying to achieve. Some examples of the NewSQL movement are VoltDB, NuoDB, Clustrix, and Akiban.

He focused on his own work on VoltDB in this field. VoltDB is a shared-nothing database which has hash or range partitioning on the data. It gets rid of the buffer pool by deploying an in-memory database design, because most of the OLTP databases today can fit in the aggregate main memory of few commodity machines. Main memory is statically divided per core, and there is a single partition for each core in a machine. Only a single worker thread executes transactions within a partition. There are no shared data structures, hence, no need for locking and latching. K-safety measures are used by replication of the data to ensure durability. VoltDB uses stored procedures for transactions, but on-the-fly compilation of ad hoc queries is possible with minor overhead. Speculative execution can be used for distributed transactions to reduce their overhead, although VoltDB does not have such a technique yet. Overall, in terms of performance, the NewSQL database VoltDB achieves a 60x improvement compared to an OldSQL vendor, offers an 8x improvement over Cassandra, which belongs to the NoSQL movement, and has the same performance as Memcached. Stonebraker argues that VoltDB is good for both scaling up and scaling out.

Stonebraker also discussed some of the recent techniques they introduced in VoltDB and some future work. To deal with cluster-wide failures, they have continuous and asynchronous checkpointing and a log that keeps the stored procedures executed with their inputs so that they can re-execute those after a failure, starting from a checkpoint. Moreover, for the databases which are too big to fit in memory, they are investigating how to do semantic caching for the current working set of the workload. In addition, they work on WAN replication, compression, and on-the-fly reprovisioning in VoltDB.

Bruce Lindsay exclaimed that the CPU cycles breakdown on the presentation was not true for the big OldSQL vendors. Stonebraker replied that they cannot do that measurement with the products of major vendors, since the source code is not available, and Oracle does not permit benchmarking of its products. C. Mohan (IBM) asked whether they support partial rollbacks in VoltDB. Stonebraker said no. Bruce also wondered how VoltDB will do semantic caching if indexes are not subsetted.

### *HyPer-sonic: Combined Transaction AND Query Processing*

Thomas Neuman, T.U. Munich

Neuman described a main memory database system that combines the execution of OLTP and OLAP workloads, called HyPer, which they built at T.U. Munich. OLTP and OLAP workloads have different characteristics. OLTP has frequent short-running transactions that observe high data locality in accesses and require high performance with updates; OLAP has a few long-running transactions that touch a lot of database records and need to see a recent consistent state for the database. Most of today's systems are optimized to be good in either. Two separate systems are kept for each workload, and data is transferred from the OLTP side to the OLAP side with an ETL (extract, transform, load) process. However, this causes both high resource consumption and the OLAP copy to see an older version of the data.

HyPer tries to bring the query processing from OLAP workloads to an efficient OLTP system. It has an in-memory database design, wherein the data is partitioned and only a single thread operates on a partition at any time. This way they can run OLTP transactions lockless and latchless, very similar to VoltDB design. Whenever an OLAP query needs to be executed, the OLTP process is forked to create a virtual memory snapshot of the database and the query processing is done on that snapshot, which provides a fresh version of the database for query processing. Updates from the OLTP side to the database while a query is running are not reflected in the database snapshot seen by the OLAP query. This is handled efficiently because the initially forked OLAP process shares physical memory with the OLTP process. Only when there are updates from the OLTP process does a copy-on-update take place and only on the updated memory locations,

to separate the update done by the OLTP (parent) process from the OLAP (child) process.

They use a datacentric query execution model rather than the iterator model. There is a pipeline of operators for a query. They have a producer and consumer interface: the former pushes the data to the current operator and the latter accepts the data and pushes it further up to the next operator. The functions are generated in assembly code at compile time using LLVM, which is fast when you want on-demand compilation and creates portable code. Moreover, it achieves better data and instruction locality than the iterator model.

To evaluate HyPer they designed the CH-benchmark. This mainly keeps TPC-C database schema and its transactions with some additional tables from TPC-H schema, and it converts TPC-H queries to be compatible with this schema. The performance numbers are good, although the memory price might be high unless the working set is fairly small.

Bruce Lindsey wondered how they could run serialized transactions. It's easy if the transactions are not touching the same data. Mohan asked how they knew this statically at runtime; Neuman said, Through partitioning. Mike Ubell asked how they could get copying without copying all the data. They used the MMU to detect when a write would occur and created a copy only at that time. Russell Sears (Yahoo!) commented on the use of LLVM to create queries, saying that code generators blow away the instruction cache.

### *Scaling Out with Meld*

Phil Bernstein, Microsoft

Phil Bernstein presented a database design where a shared binary-tree log is the database. The log supports multiversion concurrency control, and the most recently used parts of it are cached in each server's memory. In this design there is no need for cross-talk between the servers, so the design is very suitable for scaling out without dealing with the burdens of partitioning, which is the more common technique for scaling-out today.

Each server has its local (partial) copy of the log in-memory, and it has the last committed database state. Transactions executing on these servers append their intention log records to these local copies of the log. At the same time, the meld operation takes place at each server that processes the log in log order to see whether there are any transactions that conflict with each other. Depending on this process, the meld operation decides which transactions to commit or abort. If a transaction is to be committed, then the meld merges its updates (intention log records) to the database state. The meld operation basically performs optimistic concurrency control for the transactions.

Even though such a design is good for scaling out, the simple way of doing it has bottlenecks. Optimistic concurrency control can hinder performance severely if the conflict rates are high. However, this is dependent on the application. Reading from and writing to the log might be bottlenecks, but these can improve in the future with technology trends. On the other hand, the meld operation, as a single-threaded operation that reads many entries in the log, might become a huge bottleneck, because the speed of a single processor no longer improves, so the meld operation should be optimized first.

The main idea was for the meld process to have a lot fewer log records to check for conflicts for a transaction, by storing more information about the transaction in each transaction's intention. This is mainly done by keeping version numbers for each node of the binary tree.

Mohan asked if they broadcast the log. Everybody has to read it. Margo Seltzer noticed a potential conflict on a slide, and Bernstein responded that she had found a bug. Someone asked how they handle scan operations. They currently do not support scans.

## Mobility Trends & Implications

*Summarized by Yingyi Bu (yingyib@ics.uci.edu)*

### *Data Management Challenges in Location Based Services*

Srinivas Narayanan, Facebook

Srinivas Narayanan talked about the status and vision of location-based services in Facebook. Mobile users usually add location check-ins, upload photos, and create events. Srinivas emphasized that location is not only latitude and longitude, but also people, activities, and places. In the future, social events will be built on top of locations, and interesting applications such as social events/friends discovery on top of places will become very popular.

Srinivas listed several challenges. First, a good location search needs to address queries with either strong location bias or weak location bias, and considers rankings. Second, social queries need to scale up and scale out; currently Facebook uses a MySQL back-end but does not use complex queries. In the future, Facebook wants to support queries more complex than NoSQL queries. Third, everybody can have privacy policies for every data point, and to apply rules to each data point would be expensive. Fourth, social recommendations add a new dimension (location) to personal data, which advanced machine learning and data mining algorithms should leverage. Fifth, data quality will be challenging;the true real-world location for many data sources (which get tagged by user annotations), and crowd-sourcing or machine learning might be a solution.

Harumi Kuno (HP Labs) asked if Srinivas could say more about the index used for queries. Srinivas said you can use lots of predicates. Adam Marcus (MIT) pointed out that the information is both sensitive and has many compelling use cases, and he wondered about privacy. Srinivas said Facebook provides you with complete control over who sees your data. Mike Caruso wondered if they could use the location data to reveal where someone lives. Srinivas said that he didn't have a specific answer for Mike, and he pointed out that Facebook is not just an urban product.

### *Urban Data Analysis—Projects, Methods and Tools Used to Describe 21st Century Cities*

Oliver Senn, MIT SENSEable City Group

Oliver Senn described several interesting projects built by their group. The Copenhagen Wheel includes sensor devices on bikes that continuously collect data from the bikes. With gathered sensor information from all bikes, the back-end system can do real-time data analysis, considering both traffic and social information. Co2go uses accelerometer traces in smartphones to estimate CO2 emissions and enables aggregated CO2 emission analysis among users. LIVE Singapore! is an enabling platform for applications that collect, combine, distribute, analyze, and visualize either historical or real-time urban data. Applications such as realtime call locations, formula one city, and raining taxis have been built on top of LIVE Singapore! (http://senseable.mit.edu/livesingapore/).

After describing those interesting urban data analysis applications, Oliver mentioned software tools they used in the project, including Matlab, R, OpenMP, MPI, Boost Graph Library, GNU Scientific Library, C++, Java, Python, awk, sed, Oracle, MySQL, and PostgreSQL. One challenge is that the project group has only a few computer science people, and everyone is sticking to some tools; thus, to integrate different components requires a lot of work. The other challenge is to understand the data set in terms of what system produced the data, what parts of the data should be included/excluded, and how inconsistency should be resolved.

Someone asked about future problems/challenges. Oliver said that one challenge is that they don't have access to huge clusters, so for certain tasks (e.g., graph analysis) they are currently restricted. Also, certain data sets (such as the LIVE Singapore data) belong to the owners (are under NDA) and cannot be exported offshore.

### *Mobile Data Management in the CarTel System*

Sam Madden, MIT CSAIL

Sam Madden talked about problems in mobile data management and their solutions in the CarTel system. Applications of mobile data management include smart tolling insurance, urban activity monitoring, and personal medical monitoring. The volume of road sensing data is huge, and personal trajectory data is sometimes sensitive. Madden's group developed software to efficiently store and access such data while providing users control over privacy.

One subproject of CarTel is CTrack, which transforms sensed raw data to meaningful trajectories. CTrack handles cell phone signal location points with incorrect data, using probability-based estimation techniques, and pre-processes this data to visualizable road traces. The other subproject is TrajStore, a storage system for indexing and querying trajectories. TrajStore recursively divides a region into grid cells and dynamically co-locates/compresses spatially and temporally adjacent segments on disk in order to minimize disk I/Os.

Sam pointed out that there are more and more location-aware mobile devices, from a database researcher's view: cleaning, matching, filtering, visualizing, and animating mobile data at large scale is challenging.

Mehul Shah wondered why they didn't put all the data into a database. Sam replied that some of the group are machine intelligence experts trained in tools that don't look like SQL. If you push the data into SQL, they won't use it. Adam Marcus asked to what extent this is an interface problem. Sam said that some of the algorithmic issues haven't been figured out yet.

### *Scalability Under the Hood at Foursquare*

Jorge Ortiz, Foursquare

Jorge Ortiz introduced Foursquare, a location-based social networking service provider which has general social networking utilities, games, and city guides At the launch in March 2009, Foursquare used a single-node PostgreSQL database, which served 12,000 check-ins/day and 17,000 users. At the beginning of 2010, carrying a workload of 138,000 check-ins/day and 270,000 users, the system broke trying to serve all the reads/writes on a single database. From then on Foursquare used MongoDB to serve reads but still used one PostgreSQL node to serve check-in writes. In 2011, with 2.8 million check-ins/day plus 9.4 million users, Foursquare began to use MongoDB for both reads and writes. By October 2011, there were over 13 million users and more than 4 million check-ins per day. Drawbacks of PostgreSQL include connection limits, VACUUM (free space recovery), and lack of monitoring tools; advantages of MongoDB include auto-balancing, shard routing, and synchronization.

Jorge explained that MongoDB does not shard geography queries. Therefore they use Google's s2-geometry library, which turns polygons into sets of covering tiles and turns the geography index problem into a search problem.

# GO WHERE UNIQUE TALENTS CONVERGE

At EMC, innovative thinking is sustained through diverse perspectives. The EMC workforce is a world of more than 52,000 thought leaders working together to drive the future of cloud computing and information management solutions.

Through innovative products and services, EMC accelerates the journey to cloud computing, helping IT departments to store, manage, protect and analyze their most valuable asset – information – in a more agile, trusted and cost-efficient way.

To learn more about EMC, visit www.emc.com. To learn about working at EMC, visit www.emc.com/careers.

EMC²