Data drives knowledge which engenders innovation. Be it personalizing search results, recommending movies or friends, determining which advertisements to display or which coupon to deliver, data is central in improving customer satisfaction and providing a competitive edge. Data, therefore, generates wealth and many modern enterprises are collecting data at the most detailed level possible, resulting in massive and ever-growing data repositories. Such massive scale of data pose a number of research challenges, called *big data* challenges, which form the basis for my research. My research philosophy is to build database management systems (DBMSs) designed for large scale operations that expose abstractions to simplify application design while providing tools to ease system deployment and management. Using this philosophy as the cornerstone, my research has spanned the broad area of big data management encompassing both *transaction processing* and *analytical processing* systems exercising a synergy of both theoretical and practical system-oriented research.

1 Dissertation Research: Scalable, Elastic, and Autonomic OLTP Databases

DBMSs serving mission critical user facing web-applications must be scalable, fault-tolerant, and highly available to serve the growing number of users and handle the increasing amounts of data. Classical relational DBMSs (RDBMSs) support generic transactions but are expensive to scale-out to large clusters. Key-value stores can scale-out but provide transactional access to only single key-value pairs, thereby considerably increasing the complexity of application design. It is therefore important to design systems that balance functionality with scale-out—a hard problem. Furthermore, when such online transaction processing (OLTP) systems are deployed on pay-per-use infrastructures, the ability to autonomically load balance and scale up and down the number of nodes in the system (*elasticity*) is important to minimize operating cost, making the problem even more challenging.

My dissertation demonstrates that *with careful choice of design and features, it is possible to architect scalable DBMSs that efficiently execute transactions at scale to ease application design while being elastic and autonomic to simplify deployment and management.* My dissertation advances the state-of-the-art by improving two critical facets of OLTP DBMSs. First, we propose architectures and abstractions to support efficient and scalable transaction processing in database systems spanning clusters of commodity servers. Second, we propose techniques to migrate databases for elastic load balancing in an operational system and the design of a system controller that can autonomically orchestrate the system's resources and automate system management.

1.1 Scalable Transaction Processing in Large Database Systems

Classical RDBMSs allow transactions to access the entire database, thus making it hard to efficiently scale-out to large clusters. However, as the size of data grows, applications seldom access all the data within a single transaction. This has resulted in access driven database partitioning where data accessed together frequently form a database partition. Distributing the partitions on a cluster of servers and supporting transactional access to only the database partitions allow efficient transaction processing while scaling-out. Using this design rationale, my dissertation shows how to manage such large partitioned DBMSs where partitions can be statically or dynamically defined.

Many enterprise applications, such as classical banking or retail applications, have static data access patterns. For such applications, rich functionality can be supported even when limiting most transactions to a single partition or server. The challenge in designing such systems lies in making them scalable, fault-tolerant, highly available, and self managing. In other words, the major research questions are: how to concurrently execute transactions on thousands of partitions, automatically detect and recover from node failures, dynamically maintain partition-to-server mappings in the presence of partition re-assignments, and guarantee correctness? We proposed **ElasTraS** [2, 3], an elastically scalable transaction processing system. The unique aspect of ElasTraS is that it uses a *logically* separate cluster of nodes to store the persistent database image, thus exposing a scalable network-addressable storage abstraction. The transaction processing layer, executing on a cluster of nodes *logically* separate from the storage layer, focusses on efficient caching, transaction execution, and recovery of the partitions. A self-managing system controller monitors nodes, detects failures, and automatically recovers from these failures.

In some applications, such as online games or collaboration-based applications, data items frequently accessed within a transaction change dynamically with time. For instance, in an online game, transactions frequently access player profiles participating in the same game instance which changes with time. In a statically partitioned database, profiles of players in a game may belong to different partitions. Transactions will, therefore, be distributed across

multiple nodes resulting in inefficiencies and scalability bottlenecks. We proposed the **Key Group** abstraction [4] that allows applications to dynamically specify the groups of data items (or keys) on which it will require transactional access. To allow efficient execution of transactions on a group, we proposed the **Key Grouping protocol**, which co-locates read-write access of keys in a Key Group at a single node. The Key Group abstraction leverages the application semantics and is equivalent to a dynamically formed database partition. The group creation and deletion cost is paid for by more efficient transaction execution during the lifetime of the group. The challenges to implement transactions in such a system are: how to ensure safe formation and deletion of groups in the presence of failures, how to ensure efficient and durable transaction execution, and how to guarantee safe maintenance of system metadata corresponding to the dynamically formed groups? We proposed rigorous protocols for group formation, transaction execution, and recovery to ensure efficiency and correctness and address the above challenges.

1.2 Elasticity and Autonomic Resource Orchestration in OLTP Database Systems

A DBMS deployed on a pay-per-use infrastructure must be elastic to optimize the system's operating cost. The ability to migrate database partitions in a live system executing transactions, with low overhead and minimal impact on currently executing transactions, is critical to elastic load balancing—a problem referred to as *live database migration*. The major challenges for live database migration are how to minimize the migration overhead and how to ensure correctness in the presence of failures that might occur during migration? We proposed two live database migration techniques for shared nothing and shared storage database architectures.

In a shared nothing database architecture, the persistent database image is stored on disks locally attached to the nodes. An added challenge for this architecture is the live migration of the persistent database image (which can be tens of megabytes to gigabytes) with no unavailability. We proposed Zephyr [11] that divides migration into multiple phases. In the first phase, Zephyr freezes all the indices, i.e. prevents structural changes, and copies a wireframe of the database to the destination node. This wireframe consists of the minimal information needed for the destination to start executing transactions but does not include the actual application data stored in the partition. Once the destination initializes the wireframe, migration enters the second phase where both the source and the destination nodes concurrently execute transactions on the partition. The source completes execution of the transactions that were active at the start of migration, while the destination executes new transactions. Database pages are used as granules for migration; pages are pulled from the source by the destination as transactions at the destination access them. Once transactions at the source complete, migration enters the final phase where the remaining pages are pushed to the destination. Minimal synchronization and handshaking between the source and the destination ensures correctness.

In the shared storage architecture, the persistent database image is stored in a network-addressable storage abstraction accessible from all the database nodes. Therefore, the partition's persistent image need not be migrated. An added challenge for this architecture is the live migration of the database cache and state of transactions active on the partition. We proposed Albatross [7] that migrates a partition with no aborted transactions and minimal performance impact. In Albatross, the source takes a quick snapshot of a partition's cache and the destination warms up its cache starting with this snapshot. While the destination initializes its cache, the source continues executing transactions; the destination therefore lags the source. Changes made to the source node's cache are iteratively copied to the destination. Once the destination has sufficiently caught up, transactions are blocked at the source and migration is completed.

Elastic scaling and load balancing require the system to determine *which* partition to migrate, *when* to migrate, and *where* to migrate. Autonomic administration of large DBMSs minimizes the need for human intervention for resource orchestration. The responsibilities of such an autonomic controller include monitoring the behavior and performance of the system, elastic scaling and load balancing based on dynamic usage patterns, modeling behavior to forecast workload spikes and take pro-active measures to handle such spikes. The goal is to optimize the system's operating cost while ensuring good performance. We are currently working on using a combination of machine learning techniques to characterize behavior, which is then used by placement algorithms to propose good co-location and consolidation plans. Once an initial placement is determined, the controller continues monitoring the systems to detect dynamic changes in load and resource usage to determine the opportune moment for elastic load balancing.

2 Research in the Broader Area of Data Management

In addition to transaction processing systems, my research has spanned other areas of scalable database management: deep statistical analysis on big data, supporting multi-dimensional data analysis for large scale location based services, data stream analysis on multicore architectures, novel architectures for transaction processing over flash storage, and social network anonymization to facilitate analysis.

In classical enterprise settings, the ability to apply sophisticated statistical analysis techniques to large volumes of data is essential for marketplace competitiveness. This need poses a significant challenge to existing statistical software and DBMSs. Statistical software provides rich functionality but can only handle limited amounts of data. DBMSs can scale to petabytes of data, but provide limited statistical analysis functionality. During my internship at IBM Almaden, we built Ricardo [10], a scalable platform for deep analytics, that decomposes data-analysis algorithms into parts executed by a statistical analysis software (e.g. R) and parts handled by a DBMS (e.g. Hadoop); this decomposition minimizes the transfer of data across system boundaries. This differs from previous approaches which try to get along with only one type of system and leverages best of both worlds.

With the proliferation of location aware devices, location based services (LBS) are growing in popularity. To handle this growth, DBMSs serving LBSs must cope with high insert rates for location updates from millions of devices, while supporting efficient real-time analysis on latest location. Even though RDBMSs can efficiently handle spatio-temporal data, popular open-source RDBMSs are overwhelmed by the high insertion rates, real-time querying requirements, and terabytes of data. Key-value stores can effectively support such high insertion rates, but do not natively support multi-attribute accesses needed to support the LBSs. We proposed MD-HBase [12], a scalable data management system for LBSs that bridges this gap between scale and functionality. Our approach leverages a multi-dimensional index structure layered over a key-value store. The underlying key-value store allows the system to sustain high insert throughput and large data volumes, while ensuring fault-tolerance, and high availability. The index layer allows efficient multi-dimensional query processing; our prototype builds the K-d tree and the Quad tree over a range partitioned key-value store.

Unlike classical databases, data stream analysis applications—such as network monitoring, click stream analysis—have long-standing, continuously-executing queries. When such queries execute on multicore processors, it is important to effectively utilize the parallelism. In locking based designs of data structures shared by multiple cores, contention for hot objects becomes a bottleneck. We proposed *cooperation* based thread level parallelism to allow for more concurrency in shared main-memory structures. We show that this paradigm is practical by implementing parallelized operators for two common stream queries—frequent elements and top-k—over a single stream of data [5] and over multiple streams of data [6]. The proposed cooperation based paradigm removes waits associated with synchronization, and allows replacing locks by cheaper atomic synchronization primitives.

The overall hardware infrastructure in data centers is also evolving rapidly. Novel advances include low latency and high throughput networks, abundant I/O operations in new storage-class memories (such as flash memory), in addition to powerful multicore processors. These advances allow us to revisit database architectures and evaluate novel designs that can leverage the evolving hardware. Hyder is a research project at Microsoft that presents a radically different database architecture [1]. Hyder supports reads and writes on indexed records within classical multi-step transactions. It is designed to run on a cluster of servers that have shared access to a large pool of network-addressable raw flash chips. During my internship at Microsoft Research, I drove the performance evaluation of the design and designed several important enhancements to the design.

The increasing popularity of social networks has initiated a fertile research area in information extraction and data mining. Anonymizing these social graphs is important to publish these data sets for analysis. We consider the problem of edge weight anonymization to enable analyzing social networks as a weighted network. We proposed Anónimos [8, 9], a Linear Programming based technique for anonymization of edge weights that preserves linear properties of graphs. Such properties form the foundation of many important graph-theoretic algorithms such as shortest paths problem and maximizing information spread. As a proof of concept, we apply Anónimos to the shortest paths problem and its extensions, analyze its complexity, and evaluate it using real social network data sets.

3 Future Research Directions

The continued growth of data sizes, advent of novel applications, and evolution of hardware and infrastructure ensures that the area of *big data* management has many interesting research challenges. Such challenges can be classified into three areas: the evolving landscape of OLTP systems, deeper insights through data analysis, and leveraging new hardware and infrastructure paradigms.

For OLTP database systems, one major challenge is to develop techniques for dynamic database re-partitioning. Access driven database partitioning techniques rely on the application's access patterns to partition data to avoid distributed transactions. However, as the application's access patterns change, it might need re-partitioning. The challenge is to re-partition the database in a live system to minimize service interruption resulting from re-partitioning.

My research plan is a two-pronged approach: develop techniques to incrementally determine the partitions based on changes in access patterns and dynamically re-organize data in a live system without any downtime. From the application developer's perspective, a fundamental challenge is to define transaction and data consistency abstractions that simplify application design while allowing the systems to scale out to thousands of servers spread across continents. Current abstractions only focus on single data items and single-step operations. I plan to extend these abstractions to encompass multiple data items and multi-step transactions.

In the data analysis front, designing a scalable analysis platform that allows sophisticated statistical analysis, modeling, and learning is paramount to gaining deep insights from data. My future research agenda is to explore multiple facets in this front. The first challenge is to design systems that enable statistical analysis over location data to support advanced on-line location aware recommendation algorithms. My planned approach is to combine a statistical analysis platform, such as Ricardo, with a scalable multi-dimensional DBMS, such as MD-HBase. Another interesting future direction is to leverage crowd sourcing (knowledge of the masses, such as Amazon Mechanical Turk) for better data analysis. Putting a human in the data analysis loop can help solve many complex problems such as entity resolution, data cleaning, and data integration, which are fundamental to analysis systems that combine data from multiple sources. The major research challenges here are in designing the right models and abstractions to integrate humans into the loop and to associate a level of confidence to the crowd-sourced results to provide results with high confidence.

On the infrastructure side, challenges lie at the microscopic as well as macroscopic scale. At the micro-scale, the advent of storage class memory, nanosecond latency network interconnects, and specialized hardware such as FPGAs pose a fundamental question: how can we leverage such advances to build faster and bigger DBMSs? Can we exploit radically different architectures to break barriers that limit the current generation of systems? My future research agenda is to leverage such novel hardware in both transaction and analytical processing systems exploring shared storage database architectures and using custom hardware as query co-processors for efficient query processing. At the macroscopic scale, administering large scale database systems is expensive and is labor intensive. The design of an autonomic system controller for such large scale systems is important to ease administration of these systems. The goal is to ensure that the performance guarantees are met while ensuring effective resource utilization. As the scale of such systems increases, there is a super-linear growth in complexity of the problem, so the challenge is to make this problem tractable while ensuring competitive bounds. I plan to explore leveraging machine learning techniques for the design of autonomic control.

References

[1] P. A. Bernstein, C. W. Reid, and S. Das. Hyder A Transactional Record Manager for Shared Flash. In *CIDR*, pages 9–20, 2011. **[Awarded Best Paper]**.

[2] S. Das, S. Agarwal, D. Agrawal, and A. El Abbadi. ElasTraS: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud. Technical Report 2010-04, CS, UCSB, 2010.

[3] S. Das, D. Agrawal, and A. El Abbadi. ElasTraS: An Elastic Transactional Data Store in the Cloud. In *USENIX HotCloud*, 2009.

[4] S. Das, D. Agrawal, and A. El Abbadi. G-Store: A Scalable Data Store for Transactional Multi key Access in the Cloud. In *SoCC*, pages 163–174, 2010.

[5] S. Das, S. Antony, D. Agrawal, and A. El Abbadi. CoTS: A Scalable Framework for Parallelizing Frequency Counting over Data Streams. In *ICDE*, pages 1323–1326, 2009.

[6] S. Das, S. Antony, D. Agrawal, and A. El Abbadi. Thread cooperation in multicore architectures for frequency counting over multiple data streams. *PVLDB*, 2(1):217–228, 2009.

[7] S. Das, S. Nishimura, D. Agrawal, and A. El Abbadi. Albatross: Lightweight Elasticity in Shared Storage Databases for the Cloud using Live Data Migration. *PVLDB*, 4(8):494–505, May 2011.

[8] S. Das, Ömer Eğecioğlu, and A. El Abbadi. Anonymizing Weighted Social Network Graphs. In *ICDE*, pages 904–907, 2010.

[9] S. Das, Ömer Eğecioğlu, and A. El Abbadi. Annimos: An LP Based Approach for Anonymizing Weighted Social Network Graphs. *IEEE Trans. on Data and Know. Engg. (to appear)*, 2011.

[10] S. Das, Y. Sismanis, K. Beyer, R. Gemulla, P. Haas, and J. McPherson. Ricardo: Integrating R and Hadoop. In *SIGMOD*, pages 987–998, 2010.

[11] A. J. Elmore, S. Das, D. Agrawal, and A. El Abbadi. Zephyr: Live Migration in Shared Nothing Databases for Elastic Cloud Platforms. In *SIGMOD*, pages 301–312, 2011.

[12] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi. MD-HBase: A Scalable Multi-dimensional Data Infrastructure for Location Aware Services. In *MDM*, pages 7–16, 2011. **[Awarded Best Runner-up Paper]**.