Amr El Abbadi submission to
NSF PI Meeting: The Science of Cloud Computing

Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 93106-5110
amr@cs.ucsb.edu

Topics of Interest:

1. Cloud Architectures and Systems
2. Data Portability, Consistency, and Management

Summary of Current Cloud Research

DBMSs powering cloud application platforms must serve large numbers of applications with unpredictable load patterns while minimizing the operating cost leveraging the underlying pay-per-use infrastructure. We have designed **ElasTraS**, an **Elas***tic* **Tra***n***S***actional relational database* for cloud platforms. ElasTraS is a confluence of two major design philosophies: traditional relational database systems (RDBMS) that allow the efficient execution of OLTP workloads and provide ACID guarantees for small databases and the Key-value stores that are elastic, scalable, and highly available. Effective resource sharing and the consolidation of multiple tenants on a single server allows ElasTraS to efficiently manage tenants with small data and resource requirements, while advanced database partitioning and live migration allows it to serve tenants that grow, both in terms of data as well as load. ElasTraS achieves low overhead on-demand elasticity using *live migration* of tenant databases allowing expansion of the cluster size during high load and consolidation during usage troughs.

Our second project in the cloud aims at providing consistent access to data for applications that frequently access groups of data items. This is an important and significant challenge and examples of such collaboration driven applications include online gaming, social networks, and collaborative editing. We proposed the **Key Group** abstraction that defines a relationship between groups of data items through their keys and which represents the granule for on-demand transactional access. The **Key Grouping protocol** leverages the Key Group abstraction to *collocate* control for the data items in the group to allow efficient access. Using the Key Grouping protocol, we have designed and implemented **G-Store** that uses a key-value store as an underlying substrate to provide efficient, scalable, and transactional access to groups of data items.

Both **ElasTraS** and **G-Store** require efficient elastic scaling and load balancing; which necessitates live and on-demand migration of a tenant's database with low performance impact and minimal service interruption. We designed live migration techniques for the two most common database architectures. On one hand, **Iterative Copy**—a technique for shared storage architectures—migrates the database cache and the state of active transactions ensuring minimal impact on transaction execution while allowing transactions active during migration to continue execution. Iterative Copy guarantees serializability for transactions active during migration while ensuring correctness during failures. On the other hand, **Zephyr**—a technique for shared nothing architectures—uses phases of on-demand pull and asynchronous push of data, requires minimal synchronization, and results in minimal service interruption with few or no aborted transactions, while minimizing the data transferred, and providing ACID guarantees during migration.

Future Research Problems: Moving towards Multiple Clouds

We plan to explore the emerging transformations of cloud computing architectures that will further leverage computing, communication, and storage resources in the network as well as enable a new class of distributed applications. The current model of the cloud comprises a powerful *core* which is typically a large data center hosting the computation and storage of majority of Web applications. Looking forward to the future, we envision new trends in cloud infrastructures which deviate from this model of computing *cores*: namely *dynamic* and *edge* clouds. These trends originate from the observation that a tremendous amount of computation and storage also exist outside the *cores*–for instance surplus capacity from enterprises with diurnal trends of usage peaks followed by almost zero usage (case for *dynamic clouds*), or small computational and storage capacity often used by edge providers like content delivery networks (case for *edge clouds*)–which can be potentially integrated with the capacity at the *core*. These new cloud infrastructure paradigms are collectively referred to as *multiple clouds*.

In the absence of an integrative layer over multiple clouds, the management of data and resources over multiple clouds is the responsibility of the application. Ideally, we would like to create a cloud computing application processing and data management framework that seamlessly integrates and manages both data and resources that are present in multiple cloud settings. We start by focusing our attention to the naming service in Google's Cloud infrastructure which is based on the Paxos distributed consensus protocol and is implemented as the Chubby lock service by Google and as Zookeeper in its open-source counterpart. We will refer to this component as Paxos. In the multi-cloud setting, we envision a global naming service as a collection of naming services local to each core. Each local Paxos layer will manage its own physical and logical resources, and hence the issue of conflicting assignment does not arise. However, in its simplest form, the application layer needs to be aware of the existence of all the clouds, and if a particular data element is not found at the local layer, it will need to do an exhaustive search over all local Paxos layers at all the clouds. We can obviate the need for an exhaustive lookup by periodically merging the state of all local Paxos instances. A straightforward approach of merging all the views can give rise to inconsistency especially in an asynchronous environment where the communication sub-system can reorder the messages. Our proposal is to create a global view over multiple local Paxos views by drawing upon the classical notion of causality that has been studied extensively in distributed computing. In particular, we propose to maintain the local states as a distributed dictionary. This will ensure that applications running in a multi-cloud environment observe the state of the system which is causally consistent. When we combine the fact that no conflicting and competing decisions will be made by the local Paxos layers in different clouds with causal consistency of the Paxos global view, it can be asserted that the system state will remain consistent. We underscore the relative elegance of our design since the notion of causal consistency is a natural notion of correctness in distributed systems as we transition from a synchronous environment to an asynchronous environment. Furthermore, numerous implementations such as ISIS have demonstrated that causality implementation has relatively low implementation overhead.

In the above discussion, we considered a multiple cloud setting in which index and data resources remained confined within a single data center. We would like to extend the proposed causality model for dynamic management and migration of partitions across data centers by explicitly introducing a transfer event, which asynchronously propagates across clouds. The correctness of application level operations can be easily established since transfer events managed by Paxos instances across multiple clouds are causally consistent. Furthermore, no concurrency violations could occur in the system since the ownership of a partition at all times is with at most one cloud. Hence, it is impossible that two conflicting transfer events for the partition could be concurrently issued in the system.

Finally, the key tenet of distributed file system design such as GFS and HDFS is the separation of data storage from data ownership. In particular, the ownership of data is controlled via the index partitions whereas all data is stored using a distributed storage. This separation enables light-weight dynamic migration of data ownership (i.e., control) among different server nodes while the data storage remains fixed unless it is absolutely necessary, e.g., when the server storing the data fails. We propose to leverage from our work on database migration to provide explicit data transfer operations. Propagation of these events should also be causally consistent to make sure that the data does not become inconsistent. As before, during this data transfer, the copies of objects in a source cloud become read-only copies until the data transfer operation is complete.