Scalable Transaction Management in Cloud Data Systems

NSF Award 1319333 (Principal Investigator: Anand Tripathi)

Graduate Students: Vinit Padhye, Tara Sasank Sunkara, Jeremy Tucker, Gowtham Rajappan, B. Thirunavukarasu, Varun Pandye, Rahul Sharma, Manu Khandelwal, Tanmay Mehta

REU Participants: Indrajith Premanath, Kris Lange, Alexander Cina, Wei-Wei Wu, Henry Hoang

http://ajanta.cs.umn.edu
Department of Computer Science & Engineering
University of Minnesota, Minneapolis MN 55455

Research Goals
- Provide scalable multi-row serializable transactions support for key-value based NoSQL systems.
- How to utilize Snapshot isolation model for scalable transactions.
- Provide transaction support for geographically replicated data.
- Develop scalable transaction management model for partially replicated data in large-scale systems.
- Develop models and techniques for supporting different consistency models for an application.
- Utilize transaction model for data analytics applications on clusters.

Transactions in NoSQL Systems
- Transaction support is either not present (e.g., Amazon Dynamo, Yahoo PNUTs) or available with certain restrictions (Google Megastore).
- Strong consistency requires synchronous replication which incurs higher latencies.
- Due to this system dynamic and PNUTs adopted eventual consistency model.
- eventual consistency models may not be adequate for many applications requiring strong consistency guarantees.

Research Approach
- Transaction management techniques investigated here are based on the Snapshot Isolation (SI) model.
- Because the SI model does not guarantee serializable execution, this project has developed techniques to ensure serializability under the Snapshot Isolation model on NoSQL systems.
- Building upon the causal consistency model and Snapshot Isolation, this work has developed a model to simultaneously support multiple consistency models, ranging from serializability to eventual consistency.
- Replication management models are based on asynchronous update propagation.

Publications
- Incremental Parallel Computing using Transactional Model in Large-scale Dynamic Graph Structures, Anand Tripathi, Rahul R. Sharma, Manu Khandelwal, Tammy Mathis, Varun Pandey, in the International Workshop on Big Graph Processing (BiGP) 2017, in conjunction with ICDCS-2017.

Multi-Row ACID Transactions on HBASE
- Goal is to develop scalable techniques for serializable multi-row transactions on Hadoop/HBase.
- Transactions are performed entirely outside storage system without additional storage scalability of the storage system.
- Cooperative recovery model: Any application process can execute recovery actions on behalf of a failed transaction process to complete its commit/abort.
- Both the transaction management data and the application data are stored in the global key-value based storage service.

System Architecture
- The scalability limitation of the original design motivated us to develop a key-value based ACID transactions support for key-value systems.
- The scalability limitation of the original design motivated us to develop a key-value based ACID transactions support for key-value systems.
- The scalability limitation of the original design motivated us to develop a key-value based ACID transactions support for key-value systems.
- The scalability limitation of the original design motivated us to develop a key-value based ACID transactions support for key-value systems.
- The scalability limitation of the original design motivated us to develop a key-value based ACID transactions support for key-value systems.

Partitioned Causal Snapshot isolation (PCSI)
- Goal of this work is to provide a scalable model for transaction management in systems with partially replicated data.
- A key-value based data store is shared into partitions, and each partition is replicated on a subset of the sites.
- A partition is not required to be replicated at all sites.
- A site is responsible for managing the consistency of all partitions that it manages.
- Goal is to provide a weaker yet useful model based on causal consistency and snapshot isolation.

PCSI Model
- PCSI model is based on Snapshot Isolation with causal consistency guarantees.
- Properties of the PCSI Model:
  - Transaction Update Ordering:
    - Causal ordering of transaction updates
    - Global ordering of partition updates
    - Replica-based ordering of per partition updates
  - Globally Consistent Snapshot:
    - A transaction can read data from any set of partitions (local or remote) using a snapshot which is globally consistent.
    - A snapshot is causally consistent.
    - Atomically of transaction updates (all or none of the updates of a transaction are visible in the snapshot).
  - Updates are propagated asynchronously to remote sites storing the updated partitions.
- Remote sites apply updates in causal order.

Performance Scaling of the PCSI Protocol
- Scalability evaluations were conducted for both in-memory data storage and as well storing data in HBase at each site.

Design Revision
- Approach: A transaction updating a non-local partition obtains the update sequence number from a remote replicas of the partition
- Challenges:
  - Stale reads at local transactions at the site issuing sequence number to a remote transaction, because all update transactions on a partition replica are applied in their sequence number ordering.
  - Potential of deadlocks when sequence numbers are obtained from multiple remote partition replicas because of conflicting ordering implied by less or more such numbers.

Scalability Evaluations
- Used three workloads for evaluations
- Each site Si maintains for each partition p:
  - A sequence for assigning monotonically increasing sequence numbers to the local update transactions.
  - A vector clock called partition view (V), with one entry for each replica site.
  - A set of vector clocks called partition dependency view (D).
- Scalability of a transaction is a set of vector clocks, with one vector clock value for each of the partition to be accessed.
- In the original design, for updating a non-local partition by a transaction a "ghost replica" was created at the execution site.
- A ghost replica does not store any data but contains a local sequence and vector clock based dependency information.
- Scalability of the initial design was limited to about 30 nodes.
- Overheads imposed by vector clock operations increased with the creation of "ghost replicas", which were used for assigning update sequence number for a non-local partition update.
Graph structure of a graph data analytics problem may change dynamically and evolve over a period. Periodic re-executions of an analytics program on a large-scale graph structure can become expensive.

Incremental computing models can be utilized for supporting continuous queries or evolving graph structures.

**Model for Incremental Computation**
- An update to the graph after the execution of an analytics program for some problem may change certain properties of the result state.
- Updating the graph data would require some corrective tasks to be executed to restore such required properties.
- A transactional task performing a graph update operation creates a set of cascaded tasks for these incremental computations to be performed by the corrective tasks.
- Incremental computations can progress concurrently with a continuous stream of graph update transactions. When a stream of updates terminates and incremental computations are completed, one can query for results in a stable state.

**Experimental Evaluations**
- We conducted experimental evaluations of this approach for several graph problems: Single Source Shortest Paths, Maximal Cliques, Graph Coloring, Connected Components, K-Nearset Neighbors.
- Experiments with 1% of edges or nodes added/deleted.

The benefits of performing incremental computations depend on:
- Cost and complexity of the initial computation.
- Cost and complexity of performing the incremental computations for a specific type of update operation.
- Complexity depends on the problem and structure of the input graph. For small number of updates, incremental computing approach has clear benefits in most of the cases.

**Acknowledgements**
This work was conducted with support from NSF grant 1319333 and experiments were conducted on the resources provided by the Minnesota Supercomputing Institute.