Thesis Proposal: Automatic Adaptive Partitioning in Distributed Transaction based Data Storage models

Vivek Shah(bonii@kompiler.org)

May 3, 2012

1 Thesis Research Question

How can one design an adaptive auto-partitioning mechanism in an Online Transaction Processing(OLTP) [7] system using a distributed data-store which guarantees strong ACID semantics and transaction support. How much of an advantage does such a mechanism have on the performance of the system.

2 Overview of the Proposal

Online Transaction Processing Systems(OLTP) are gaining increased importance owing to the new age of Internet service demands. The service demands range from email systems, blogging platforms to web collaboration services. Any interactive online service which utilizes the web and is large enough to need a distributed data-store requires some form of an online transaction processing service. What characterizes these services are the two demands of scalability and availability. Though these demands can be conflicting at times, the design of the data-store and its properties go a long way in satisfying both these constraints. A data-store can be a noSQL [6] non sharing database system, in essence a distributed key value store which guarantees high scalability but the lack of strong ACID semantics make application development on top of it complex. At the other end, a data-store can be a relational database system [9] which guarantees strong ACID semantics thus aiding application development but can make it difficult to scale.

In both these database systems, a key requirement is to partition the data sets so that accesses(read and write) can be made faster and complex locking, contention and agreement necessities can be avoided. An online service would not want to waste a lot of machine cycles in busy-waiting and deadlock resolution for its transactions. Partitioning [8] data-stores thus assumes utmost importance. A lot of times partitioning turns out to be the key factor in the performance of the data-store as the agreement and contention protocols depend on the partitioning layout. There are multiple approaches that are taken to solve this bottleneck.

The most obvious approach is for human experts to statically partition the data-store based on history of transactions and the data-store performance. This

[illegible]

is the most widely used approach currently. Another approach is to predict the future transactions that might crop up based on a history of transactions as outlined in [5]. It provides an interesting innovative model for predictive modeling of transactions. On the basis of future predicted transactions and its subsequent scheduling, the inter partition access costs can be minimized. Another approach is to come up with a mechanism to partition the data-store automatically using graph partitioning algorithms as outlined in [3]. This mechanism has been proposed for a disk based shared-nothing database but it can be easily extended to main memory databases like H-Store [4]. [1] proposes a similar idea but for a data-store which provides a middle ground between a noSQL and relational database system while maintaining availability and scalability. However, in this system the partitioning is done based on entity groups which depend on the structure of data in the application and need to be specified. Transactions involving multiple entity groups use expensive 2 phase commits but try to minimize the cost by using the asynchronous messaging system. The authors argue that the division of the entity groups minimizes transactions across entity groups. However, in this system the partitioning(division of entity groups) is dependent on application data and is specified manually by the application developers.

The idea of an auto-partitioning system seems a promising research idea but where it can go a step further is to research for the viability of an adaptive transactional auto-partitioning system for noSQL, relational and a system which is an overlap of the two database systems. The system would, based on the current partitioning layout, its cost model and the transaction history determine whether a new partitioning is necessary and then partition it. This idea forms the core of this thesis and the motive of the thesis would be to answer if such a research goal in terms of system development is feasible and how it can be achieved. This research can be done by abstracting the data-store as a memory model which can be partitioned. Each partition is responsible for the maintenance of the memory addresses residing in that partition and a transaction accessing addresses lying in multiple partitions would require some form of distributed concurrency control(a distributed lock manager) [2, pages 494-508] and distributed deadlock resolution mechanism [2, pages 490-494] if the transactions are conflicting for deciding the schedule. If the transitions are not conflicting then they can easily be serialized. Ideally, the partitioning should be such that the number of such distributed inter partition conflicting transactions should be minimum. Also, based on the number of inter-partition access by multiple transactions, the system should be able to recognize that the current partitioning layout is not satisfactory and another auto-partitioning needs to be done.

The thesis aims to design such an adaptive system, research possible partitioning algorithms, architect meaningful experiments to test the system and provide conclusion from the experiments about the viability of the approach. In order to meet the time constraints for the thesis, a memory model to map the data store can be used and the design of cost model can be borrowed from existing systems like [3]. In order to resolve distributed conflicting transactions a simple timeout based distributed lock manager can be used. This would enable greater focus on the partitioning and adaptive engines and also keep the thesis within a reasonable time duration. The thesis would abstract the data space as a main memory engine(a memory address space) and would provide a simple log based durability mechanism.

3 Thesis plan

As part of the thesis the following work is planned to be accomplished

1. Survey existing literature for related work in the field of partitioning data-stores preserving ACID semantics for transactions and graph partitioning algorithms. [1 week]
2. Analyze existing systems and their trade-offs like scalability, availability, ease of application building, performance in terms of latencies, CPU work-load and identify how the trade-offs are linked to the system design and also to each other. [1 week]
3. Survey and analyze existing cost models to depict the cost of accessing various partitions. For the purposes of this thesis, the cost model outlined in [3] may be used so that the focus can be laid on the partitioning and adaptive engine. [1 week]
4. Design a memory model to implement a data-store which is flexible enough to model both noSQL and relational data-stores and supports the necessary partitioning mechanism. The model needs to be flexible to support scalability since distributed data-stores need to be supported. The data store will be modeled as a main memory engine and would not be disk based. The main memory engine would model the entire address space and parts of the address space would be segregated into partitions and transactions would access the address space.[1.5 weeks]
5. Design partitioning algorithms and adaptive algorithms to determine whether the current partitioning is not useful any more and a re-partitioning is needed. [2.5 weeks]
6. Build the system as per the design, re-design the partitioning and adaptive engine(as necessary) and iterate until the system is ready.[10 weeks]
7. Design experiments to test the system for various rigorous workloads(preferably OLTP workbench) to identify the best, worst and average case performance of the system. The results need to be quantified and presented in meaningful forms which can be extrapolated to system behavior properties. [4 weeks]

Consolidation of the entire research work in the form of a thesis is planned to overlap with the above phases. It is to be noted that the implementation phase and evaluation phase would go hand in hand as feedback from each of the phases would influence the other.

4 Learning Goals

The learning goals for the thesis entail the following

1. Review existing literature for existing data storage architectures and partitioning schemes, cost models of partitions and existing costs. Compare various trade-offs and characteristics of the designs and draw necessary conclusion for the target system design.

2. Identify nature of cost models and how they are linked with the partitioning.

3. Analyze and design algorithms for partitioning and for adaptive feedback.

4. Design the overall system to satisfy the key features of the research question.

5. Design the experimental setup to evaluate the key aspects of the designed system and its performance.

6. Present the system design, experimental data and conclusions as a scientific work in the form of a thesis and presentation.

References

[1] J. Baker, C. Bond, J. C. Corbett, F. JJ, A. Khorlin, J. Larson, J.-M. Leon, L. Yawei, A. Lloyd, V. G. Yushprakh, and Inc. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. In *In Biennial Conference on Innovative Data Systems Research (CIDR 2011)*, pages 223–234, January 2011.

[2] G. Couloris, J. Dollimore, and T. Kindberg. *Distributed Systems Concepts and Design.* Addison-Wesley, third edition, 2001.

[3] C. Curino, E. Jones, Y. Zhang, and S. Madden. Schism:a Workload-Driven Approach to Database Replication and Partitioning. In *In The 36th International Conference on Very Large Data Bases(VLDB 2011)*, pages 48–57, September 2010.

[4] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. J. Abadi. H-Store: a high-performance, distributed main memory transaction processing system. *Proc. VLDB Endow.*, 1(2):1496–1499, 2008.

[5] A. Pavlo, E. P. Jones, and S. Zdonik. On Predictive Modeling for Optimizing Transaction Execution in Parallel OLTP Systems. In *In The 38th International Conference on Very Large Data Bases(VLDB 2011)*, pages 85–96, August 2012.

[6] Wikipedia. NoSQL. `http://en.wikipedia.org/wiki/NoSQL`. Online; Accessed 25-April-2012.

[7] Wikipedia. Online transaction processing. `http://en.wikipedia.org/wiki/Online_transaction_processing`. Online; Accessed 25-April-2012.

[8] Wikipedia. Partition (database). `http://en.wikipedia.org/wiki/Partition_(database)`. Online; Accessed 25-April-2012.

[9] Wikipedia. Relational database. `http://en.wikipedia.org/wiki/Relational_database`. Online; Accessed 25-April-2012.