Elastic In-Memory Transaction Processing for Multi-Tenant Database Systems

Seyedeh Sharareh Mirzargar
seyedeh.mirzargar@epfl.ch

ABSTRACT

Traditional database systems try to maintain high throughput in the presence of load spikes by manually reconfiguring the database system or by overprovisioning the hardware. Recent research focuses on providing elasticity for distributed scale-out OLTP engines. For example, E-Store [1] provides an end-to-end design for shared-nothing scale-out OLTP engines. SQLVM [2] provides absolute resource assurance for relational database-as-a-service providers and mostly focus on providing performance isolation rather than fine-grained elasticity.

All current approaches achieve elasticity by adding some number of identical commodity servers to the database cluster. This fits a transaction computing model that has single-threaded execution engine per partition and a fixed number of partitions per node. However, it fails for environments in which partitions can be accessed by a variable number of threads and nodes host variable numbers of partitions.

It is possible to provide elasticity for OLTP engines by keeping the number of servers the same and increasing the size of the servers by adding additional DRAM or CPU cores. Public cloud vendors like Amazon AWS have made it easy to scale up by simply choosing a different instance type [3]. Achieving elasticity in a scale-up environment is a different problem compared to scale-out systems. Data partitioning and migration play a crucial role in the scale-out scenario but they are irrelevant to a scale-up setting in which all the data is stored in globally-accessible shared memory. Moreover, resources are allocated in a fine-granularity (e.g. core) in a scale-up environment in contrast to the scale-out environment where resources are allocated at the granularity of a whole computing node.

In order to exploit such fine-grained flexibility, an OLTP engine should be able to react to dynamic changes in the underlying processor topology. It should be able to dynamically manage its thread pool. When cores are added to the OLTP engine, it should be able to start running new threads and when we release cores from it, threads should stop accessing those cores. Data placement is crucial in a scale-up setting. Cross socket accesses have higher latency compared to accessing data that is resident in the local socket. Another interesting research direction is identifying the right amount of resources that we need to provision and also how we should change the hardware topology to get the best performance improvement.

CIDR '13 January 13–16, 2019, California, USA

DOI:

CCS Concepts

•Computer systems organization → Cloud computing;

Keywords

OLTP,Main Memory, Multi-Tenancy

1. REFERENCES

[1] R. Taft, E. Mansour, M. Serafini, J. Duggan, A. J. Elmore, A. Aboulnaga, A. Pavlo, and M. Stonebraker, "E-store: Fine-grained elastic partitioning for distributed transaction processing systems," *Proc. VLDB Endow.*, vol. 8, no. 3, pp. 245–256, Nov. 2014. [Online]. Available: http://dx.doi.org/10.14778/2735508.2735514

[2] V. Narasayya, S. Das, M. Syamala, S. Chaudhuri, F. Li, and H. Park, "Sqlvm: Performance isolation in multi-tenant relational database-as-a-service," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 1077–1080. [Online]. Available: http://doi.acm.org/10.1145/2463676.2463686

[3] "Amazon ec2 pricing," https://aws.amazon.com/ec2/dedicated-hosts/, accessed: 2010-09-30.