



Distributed Data Grids Deliver Scalable Performance

Introduction

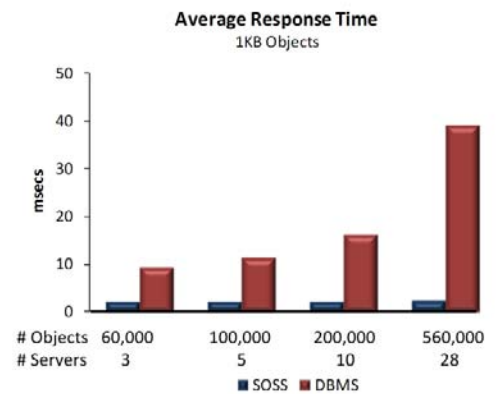
Mission-critical applications, running on server farms and compute grids, often use a DBMS to hold mission-critical but relatively short-lived data, such as Web session state, ecommerce shopping carts, and cached application data. This creates significant additional traffic to the DBMS (with a much larger proportion of writes for which the DBMS was not optimized), and it delays responses to client applications. To eliminate this performance bottleneck, ScaleOut StateServer® (SOSS) provides a memory-based distributed data grid that is hosted directly on a server farm, on an HPC compute grid, or on a cluster of dedicated data-grid servers networked to servers running client applications. SOSS's fast, in-memory storage dramatically reduces response times for repeatedly accessing grid-based data in comparison to database access, and its access throughput linearly scales as servers are added to the farm. These performance characteristics enable SOSS to maintain very fast and predictable response times even under very high access loads, while simultaneously offloading the database server.

Scalable Throughput

To maintain very fast response times under heavy workloads, ScaleOut StateServer automatically scales its throughput to handle additional client load as servers are added to the user's server farm or HPC compute grid. This keeps response times from growing due to access bottlenecks as the workload grows. In contrast, storage solutions with fixed throughput, such as database servers, experience rapidly increasing response times as their maximum throughput is reached.

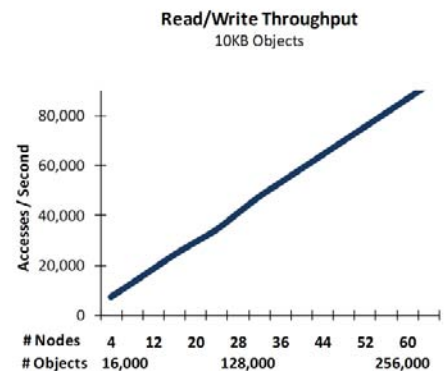
To illustrate the advantages of ScaleOut StateServer's scalable throughput, two scalability tests were performed. In the first test, the access throughput of SOSS's distributed data grid was compared to that of a relational database server. To maximize stress on the distributed data grid, each application server sequentially performed continuous read/update access requests to 20,000 1K byte objects. The average time to complete a read and update was recorded while servers were added to the farm until 28 servers with a total of 560K objects were in use. For comparison purposes, response times were also measured for the same read/update operations on a database server in which the objects likewise were stored.

The chart on the top right shows that SOSS's average response time (shown as the blue bars) remained flat as application servers were added to the farm to scale the storage and access workload. This demonstrates the benefit of the distributed data grid's scalable throughput, which increased linearly as the server farm grew from 3 to 28 servers. In contrast, the database server exhibited a non-linear increase in response times (the red bars) and was almost 20X slower than SOSS's distributed data grid under the maximum load of 28 servers and 560K objects.



The second test confirmed ScaleOut StateServer's linear scalability under very high load on a large computational grid. This grid consisted of 64 four-way multiprocessor blades and a high speed 20Gbits/second Infiniband network. Each server in the grid hosted 4,000 objects of 10KB each, and performed continuous read/update access tests as servers were added to the grid and the load was proportionately increased, reaching approximately 2.5GB on 64 servers. The aggregate access throughput was measured to confirm linear scalability and the absence of sequential bottlenecks.

As illustrated in the line graph to the right, this test demonstrated that SOSS's access throughput grows linearly as servers and access load are increased. This shows that the distributed data grid delivers maximum scalability in both its throughput and storage capacity, which enables applications to enjoy the fastest possible access times for very large, access-intensive computations.



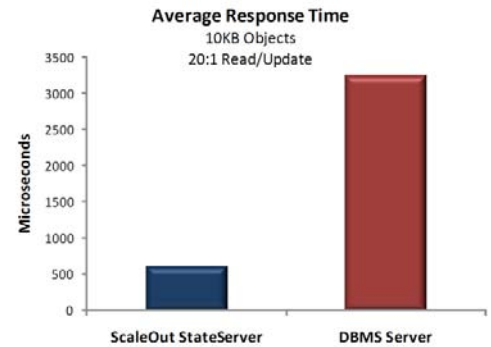
Fast Response Time

ScaleOut StateServer incorporates several internal mechanisms that automatically accelerate access requests and minimize response times. ScaleOut StateServer can deliver read access times that are very close to the performance of an "in process" store and significantly better than access times for other "out of process" stores, such as database servers. For example, the chart to the right shows SOSS's average read access time (in microseconds) for multiple reads to a 10KB grid object. This chart shows that SOSS reduces the average read access time six-fold in comparison to the use of a database server.

SOSS transparently employs multiple, internal caches that automatically accelerate accesses to cached objects. These internal caches include:

- a **client-side ("near") cache** of references to recently accessed, deserialized data objects, which is maintained by the SOSS API client libraries in each application process, and
- a **server cache** of recently accessed, serialized objects maintained by the SOSS service on each server in the data grid.

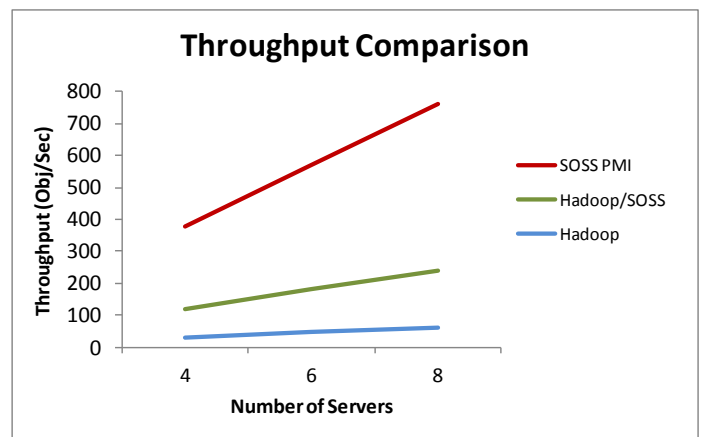
These caches are tightly integrated and kept fully coherent with SOSS's highly optimized, distributed, in-memory storage to ensure much lower access times than other storage alternatives, while maintaining global accessibility across the grid to all stored data.



Parallel Query and Map/Reduce Analysis

ScaleOut StateServer Grid Computing Edition™ (GCE) takes scalable performance to the next level by enabling fast, parallel query and analysis of data hosted in the distributed data grid. Object properties can be queried in parallel using Microsoft Language Integrated Query (LINQ) or Java query filters. Parallel data analysis can be performed via map/reduce style computation, called "parallel method invocation." These capabilities let developers easily examine and analyze large volumes of data and quickly merge results into a final report.

To illustrate the power of this analysis capability, a real-world financial analysis application was constructed to compare various stock trading strategies based on historical market data stored in ScaleOut StateServer's distributed data grid. This application made use of SOSS's parallel method invocation mechanism to automatically execute a parallel "map/reduce" analysis across all grid servers and merge the results. Performance tests evaluated throughput scaling by proportionally increasing the number of stock histories analyzed as the number of servers was increased. As the graph illustrates, SOSS delivered linearly scalable throughput on this application (shown as the red line in the graph) as the problem size (number of stocks examined) and the number of grid servers was increased. This high performance and linear scalability results from SOSS's ability to reduce start-up time, hold the dataset for the analysis in memory, automatically distribute the analysis workload among the grid servers, and avoid unnecessary data motion.



For comparison with other popular map/reduce platforms, which typically access file-based data, an alternative implementation of this application was measured using the open source Hadoop map/reduce environment. Hadoop delivered about 16X lower throughput (shown as the blue line in the graph) due to significant overhead introduced by file I/O and batch scheduling. By staging the stock history data in SOSS's in-memory data grid instead of the Hadoop file system (HDFS), Hadoop's throughput was increased by up to 6X (shown as the green line). However, SOSS's map/reduce execution model and highly efficient scheduling, as well as the elimination of file I/O between the map and reduce phases, still provided a significant performance advantage for map/reduce analysis on memory-based data.