SCALEOUT SOFTWARE

How to Scale the Storage and Analysis of Business Data Using a Distributed Data Grid

by Dr. William Bain, Ph.D, ScaleOut Software, Inc.

A hallmark of the Information Age is the incredible amount of business data that companies have to store and analyze. The ability to efficiently search data for important patterns can provide an essential competitive edge.

For example, an e-commerce Web site needs to be able to monitor online shopping carts to see which products are selling quickly. A financial services company needs to hone its equity trading strategy as it optimizes its response to fast-changing market conditions. Businesses that face challenges like these have turned to distributed data grids (also called distributed caches) to scale their ability to manage fast-changing data and comb through data to identify patterns and trends requiring a timely response.

Distributed data grids offer two key advantages. First, they store data in memory instead of on disk for fast access, and, second, they run seamlessly across a farm of servers to scale performance. But perhaps best of all, they provide a fast, easy to use platform for running "what if" analyses on the data they store. By breaking the sequential bottleneck, they can take performance to a level that stand-alone database servers cannot match.

Software architects and developers often say the following. "OK, I see the advantages, but how do I incorporate a distributed data grid into my data storage architecture, and how could it help me to analyze my data?"

Here are three simple steps for building a fast, scalable data storage and analysis solution using a distributed data grid.

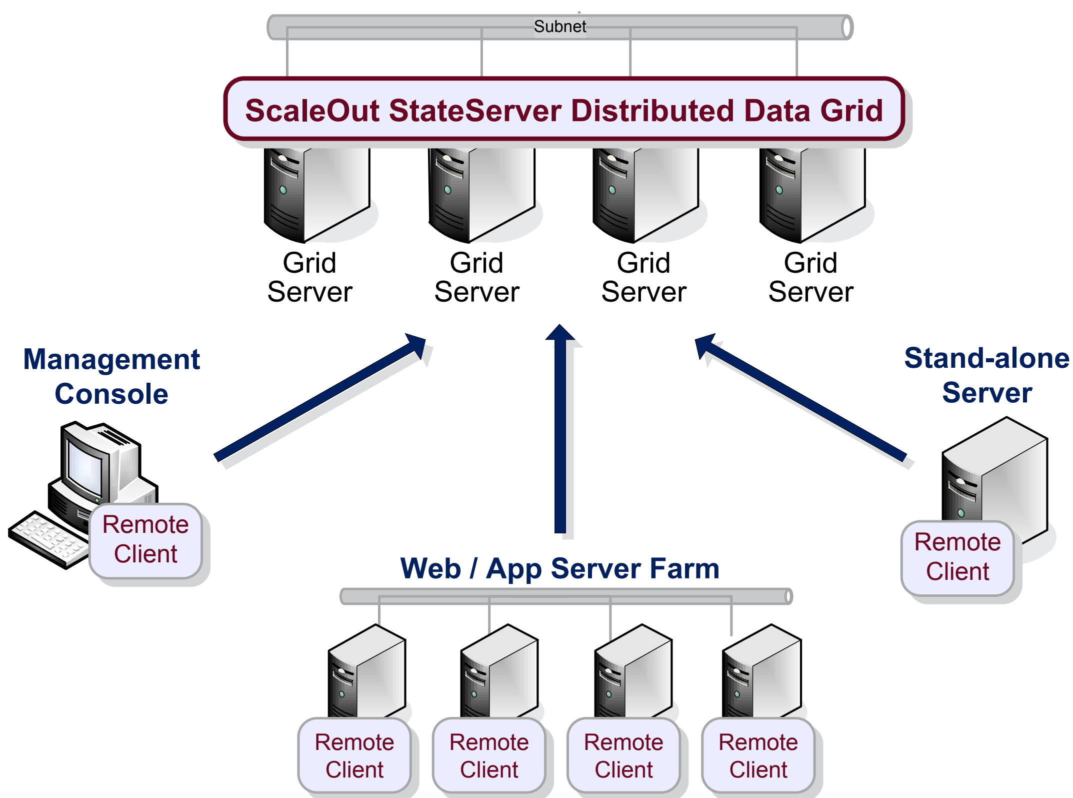
1. Store fast-changing business data directly in a distributed data grid instead of a database server.

Distributed data grids are designed to plug directly into the business logic of today's enterprise applications and services. By storing data as collections of objects instead of relational database tables, they match the in-memory view of data already used by business logic. This makes distributed data grids exceptionally easy to integrate into existing applications using simple APIs, which are available for most modern languages, like C#, Java, and C++.

Because distributed data grids run on server farms, their storage capacity and throughput scale just by adding more grid servers. When hosted on a large server farm or in the cloud, a distributed data grid's ability to store and quickly access large volumes of data can grow well beyond that for a stand-alone database server.

2. Integrate the distributed data grid with database servers as part of an overall storage strategy.

Of course, distributed data grids are used to complement and not replace database servers, which are the authoritative repositories for transactional data and long-term storage. For example, in an ecommerce Web site, a distributed data grid would hold shopping carts to efficiently handle a large workload of online shopping traffic, while a backend database server stores completed transactions, inventory, and customer records. The key to integrating a distributed data grid into an enterprise application's overall storage strategy is to carefully separate application code used for business logic from other code used for data access.



The distributed data grid would hold shopping carts to efficiently handle a large workload of online shopping traffic, while a backend database server stores completed transactions, inventory, and customer records.

Distributed data grids naturally fit into business logic, which usually manages data as objects. This code is also where rapid access to data is needed, and that's where distributed data grids provide the greatest benefit. In contrast, the data access layer typically focuses on converting objects into a relational form (or vice versa) for storage in database servers.

Interestingly, a distributed data grid optionally can be integrated with a database server so that it can automatically access data from the database server if it's missing from the distributed data grid. This is very useful for certain types of data, such as product or customer information, which is kept in the database server and just retrieved when needed by the application. However, most types of fast-changing, business logic data can be kept solely in a distributed data grid and never written out to a database server.

3. Analyze grid-based data using simple analysis codes and the "map/reduce" programming pattern.

Once a collection of objects, such as a Web site's shopping carts or a financial company's pool of stock histories, has been hosted in a distributed data grid, it's important to be able to scan all of this data for important patterns and trends. Over the last 25 years, researchers have developed a powerful, two-step method, now popularly called "map/reduce," for analyzing large volumes of data in parallel. In the first step, each object in the collection is analyzed for an important pattern of interest by writing and running a simple algorithm that just looks at one object at a time. This algorithm is run in parallel on all objects to quickly analyze all of the data. Next, the results that were generated by running this algorithm are combined to determine an overall result, which hopefully identifies an important trend.

For example, an e-commerce developer could write a simple code which analyzes each shopping cart to rate which product categories are generating the most interest. This code could be run on all shopping carts several times during the day (or perhaps after a marketing blitz on the Web site has been launched) to identify important shopping trends.

Programmers do not need to learn parallel programming techniques or understand how the grid works.

Distributed data grids offer an ideal platform for analyzing data using this "map/reduce" programming pattern. Because they store data as memory-based objects, the analysis code is very easy to write and debug as a simple "in-memory" code. Programmers do not need to learn parallel programming techniques or understand how the grid works. Also, distributed data grids provide the infrastructure needed to automatically run this analysis code on all grid servers in parallel and then combine the results. The net result is that by using a distributed data grid, the application developer can easily and quickly harness the full scalability of the grid to rapidly discover data patterns and trends that are vital to a company's success.

As companies become ever more pressed to manage increasing data volumes and quickly respond to changing market conditions, they are turning to distributed data grids to obtain the "scalability" boost they need. As clouds become an integral part of enterprise infrastructures, distributed data grids should further prove their value in harnessing the power of scalable computing to provide an essential competitive edge.

Dr. William L. Bain is founder and CEO of ScaleOut Software, Inc. Bill has a Ph.D. in electrical engineering/parallel computing from Rice University, and he has worked at Bell Labs research, Intel, and Microsoft. Bill founded and ran three start-up companies prior to joining Microsoft. In the most recent company (Valence Research), he developed a distributed Web load-balancing software solution that was acquired by Microsoft and is now called Network Load Balancing within the Windows Server operating system. Dr. Bain holds several patents in computer architecture and distributed computing. As a member of the Seattle-based Alliance of Angels, Dr. Bain is actively involved in entrepreneurship and the angel community.