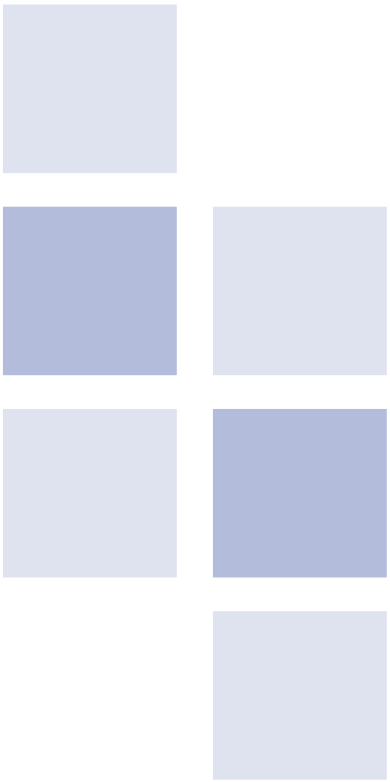


SCALEOUT SOFTWARE

Using an In-Memory Data Grid for Near Real-Time Data Analysis

by Dr. William Bain, ScaleOut Software, Inc.

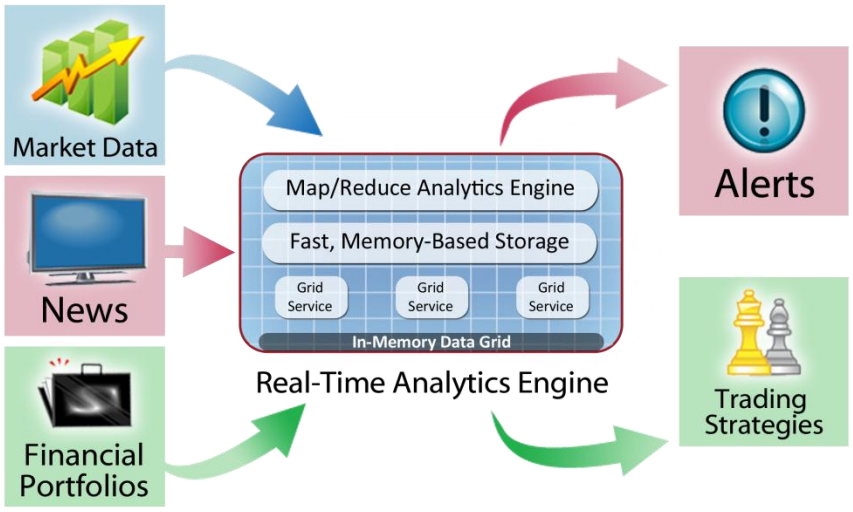
IN today's competitive world, businesses need to make fast decisions to respond to changing market conditions and to maintain a competitive edge. The explosion of data that must be analyzed to find trends or hidden insights intensifies this challenge. Both the private and public sectors are turning to parallel computing techniques, such as "map/reduce" to quickly sift through large data volumes.

In some cases, it is practical to analyze huge sets of historical, disk-based data over the course of minutes or hours using batch processing platforms such as Hadoop. For example, risk modeling to optimize the handling of insurance claims potentially needs to analyze billions of records and tens of terabytes of data. However, many applications need to continuously analyze relatively small but fast-changing data sets measured in the hundreds of gigabytes and reaching into terabytes. Examples include clickstream data to optimize online promotions, stock trading data to implement trading strategies, machine log data to tune manufacturing processes, smart grid data, and many more.

Over the last several years, in-memory data grids (IMDGs) have proven their value in storing fast-changing application data and scaling application performance. More recently, IMDGs have integrated map/reduce analytics into the grid to achieve powerful, easy-to-use analysis and enable near real-time decision making. For example, the following diagram illustrates an IMDG used to store and analyze incoming streams of market and news data to help generate alerts and strategies for optimizing financial operations. This article explains how using an IMDG with integrated map/reduce capabilities can simplify data analysis and provide important competitive advantages.

What is an In-Memory Data Grid?

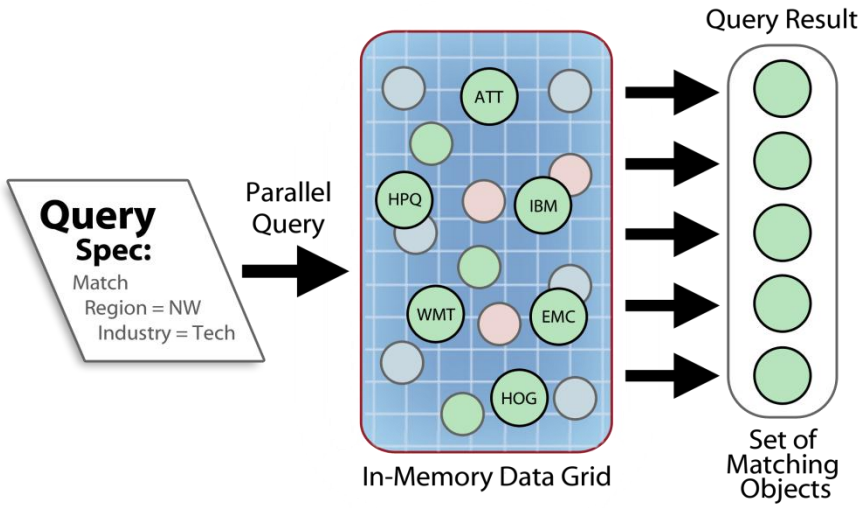
By storing fast-changing data within a middleware software tier, IMDGs enable applications to seamlessly scale performance by adding servers that access and update a shared, memory-based data set. To maximize scalability, IMDGs automatically load-balance data across servers on which the grid is hosted. They also redundantly store data on multiple servers to ensure high availability in case a server or network link fails.



Using an IMDG to Analyze Fast-Changing Data

Additional capabilities, including eventing and distributed locking, make IMDGs a powerful data storage platform.

IMDGs typically integrate their data storage model with object-oriented programming languages, such as Java and C#. They store data as a collection of objects which are accessible either by specifying an identifying key or by querying object properties. The IMDG's built-in parallel query mechanism can quickly scan a large data set for objects whose properties match a query specification. This provides an important tool for identifying data to be reviewed or analyzed. The following diagram illustrates the use of parallel query for selecting stock history data.



Parallel Query of Stock History Objects Stored in an IMDG

Using an IMDG for Analytics

Without a doubt, the field of data analytics has gained a powerful new tool with the "map/reduce" analysis model, which has recently surged in popularity as open source solutions such as Hadoop have raised awareness. In fact, the roots of the map/reduce pattern date back to pioneering work in the 1980s which originally demonstrated the power of data-parallel computing.

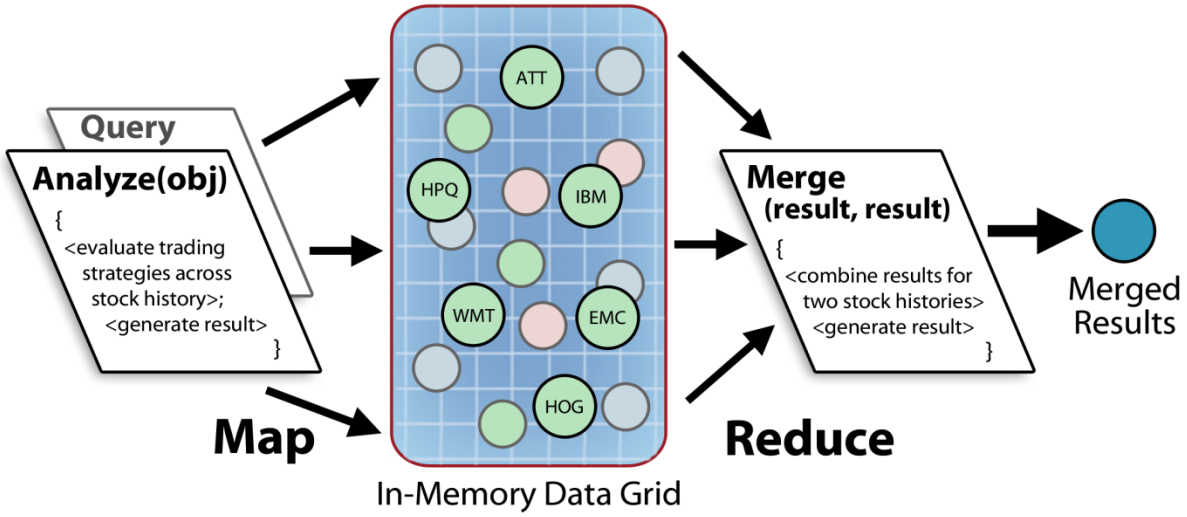
Map/reduce implementations take many forms and are offered as components in several competing frameworks. Nearly all of these solutions are aimed at accelerating data analysis for disk-based data. With some data sets reaching petabytes in size, the benefits are often measured in reducing batch job processing times from hours to minutes for these “big data” analyses.

However, the overhead (and complexity) of disk-based map/reduce platforms is too high for applications which must quickly analyze fast-changing data sets measured in hundreds of gigabytes or terabytes. (Estimates by some analysts indicate that as much as sixty percent of data sets are smaller than ten terabytes.) In many situations, an answer in hours or minutes is not acceptable. For example, an e-commerce Web site may need to monitor online shopping carts to see which products are selling. A financial services company might need to hone its equity trading strategy as it optimizes its response to fast-changing market conditions.

To address this challenge, leading-edge IMDGs have incorporated map/reduce analytics engines, transforming them from just scalable, memory-based data stores into parallel computing platforms for analyzing data and providing fast, near real-time results. IMDGs leverage the grid's automatic load-balancing to minimize data motion and speed up analysis. Instead of migrating data into memory from disk, an IMDG analyzes data in place. Results also are stored and combined in memory, minimizing file I/O to calculate the final results. By eliminating these overheads, IMDGs dramatically reduce network usage and thereby shorten analysis time.

Moreover, by simplifying the programming model, IMDGs offer another advantage over popular, disk-based map/reduce platforms. Instead of requiring the application developer

to create a key space for identifying objects to be analyzed, they make use of object-oriented query specifications to select objects. Also, both the analysis ("map") and merge ("reduce") codes can be structured as straightforward, object-oriented methods written as if to be executed on a single workstation. These capabilities shorten design time and enable analysis applications to be quickly developed and revised.



Map/Reduce Analysis of Stock History Objects in an IMDG

The preceding diagram illustrates a map/reduce analysis of stock trading strategies across a set of stock histories held in the IMDG. A parallel query selects stocks for analysis, and the IMDG analyzes the stocks and merges the results using the supplied methods.

Running Map/Reduce on an IMDG

ScaleOut Analytics Server™ from ScaleOut Software is an example of an IMDG with an integrated data analytics engine. Using it as an example, the following steps demonstrate how an IMDG performs a map/reduce data analysis:

- The data set to be analyzed in the IMDG originates from one of two sources. In many cases, especially those with tight latency requirements, the application continuously updates the grid as data flows through for processing. Alternatively, the application may stage the data set in the grid from persistent storage via a bulk loading operation. In either case, the IMDG holds the data, creates replicas for high availability, and load-balances it across servers to avoid hot spots.

- ScaleOut Analytics Server allows a query specification to be written either in Java using filter methods or in C# using the Microsoft language integrated query (LINQ) mechanism. This query specification selects the data to be analyzed, for example, ticker symbols, sales data, machine data, etc.

- In ScaleOut Analytics Server, the analysis and merge methods can be written either in Java or C#. Since the IMDG holds the objects to be analyzed or merged in memory, these methods are written without the need to use grid APIs. The analysis method specifies the analysis logic for a single data object selected by the query specification. For example, it might calculate stock trading profits for one company's recent history of stock prices. The merge method combines the results of analyzing multiple objects and is repeatedly executed as necessary to merge all results. In the above example, it might calculate the average return for stock trades spanning many companies.

- Using a special API in ScaleOut Analytics Server called "invoke" and supplying the query specification and both the analysis and merge methods, the application starts a map/reduce computation called a "parallel method invocation" (PMI). The IMDG automatically performs the query, analysis, and merge steps in parallel across all grid servers using a multi-threaded computation engine and then returns the final, merged result back to the application. PMI operations can be performed repeatedly to provide a continuous stream of results. Because ScaleOut Analytics Server avoids batch scheduling and keeps the overhead for starting and running the analysis low, it returns results with minimum latency for near real-time performance.

When using an IMDG, all computations are performed “in-place,” reducing data motion which is the enemy of high performance for map/reduce. Also, the IMDG leverages its features for maximizing scalability and high availability, such as partitioning, peer-to-peer architecture, and load-balancing. In addition, ScaleOut Analytics Server implements special features for ensuring the high availability of map/reduce computations.

Lowering the Complexity Barrier

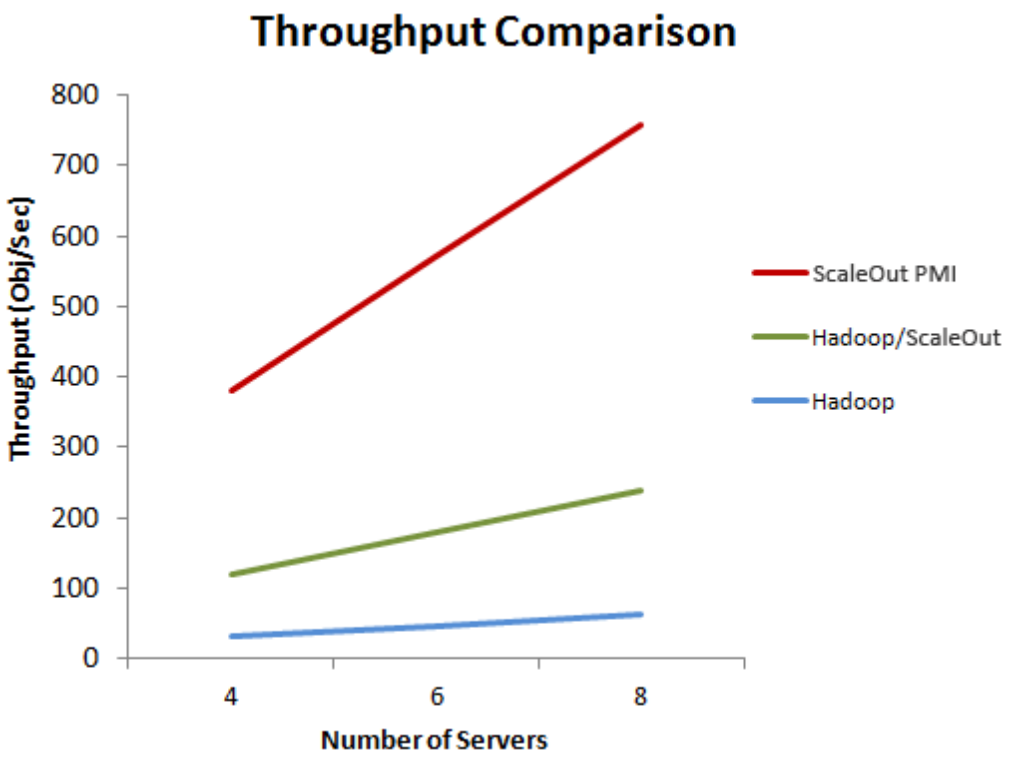
The map/reduce programming model has generated widespread interest in large part due to the popularity of the Hadoop open source software stack. However, Hadoop introduces

a complex programming model and deployment architecture which must be thoroughly understood for Hadoop to be used effectively. For example, applications need to be written to fit Hadoop's specific parallel execution model, incorporating several specialized elements such as record readers, mappers, combiners, and reducers. The number and interaction of these elements impact performance and require tuning. Beyond this, Hadoop's execution environment, including the HDFS file system, job tracker (that is, the batch scheduler), and task trackers on each execution node must be deployed and managed. It may take a seasoned Java developer with knowledge of parallel programming weeks to become proficient with Hadoop. These complexities create a steep learning curve which impedes rapid adoption.

In contrast, the IMDG-based approach to map/reduce data analysis eliminates much of Hadoop's complexity. Its object-oriented approach offers a simpler parallel execution model that reduces development time and eliminates the need for tuning. The user invests much less time in learning the model and focuses more on the *analytical* challenges of the business problem. Learning curves are flattened, and productivity is increased.

Delivering High Performance

To see the performance benefits of using an IMDG with integrated map/reduce, consider a real-world financial analysis application that compares various stock trading strategies based on historical market data stored in the IMDG. This application makes use of the IMDG's analytics engine to perform a map/reduce analysis across all grid servers and merge the results. Each stock history is stored as a separate object within the IMDG, and specific stock histories are selected for analysis using a parallel query. The analysis method evaluates a set of trading strategies across a single stock history, and the merge method combines the results across two stocks. The analytics engine repeatedly executes these methods to analyze all selected stocks and merge the results.



Throughput Comparison of ScaleOut's IMDG and Hadoop

Performance measurements were made for this application using ScaleOut Analytics Server's IMDG to evaluate throughput scaling as the number of stock histories and grid servers was proportionally increased. As the graph above illustrates, the IMDG delivers linearly scalable throughput (shown as the red line in the graph). An alternative implementation of this application was measured using Hadoop's map/reduce environment. Hadoop provided linear scaling with 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 the IMDG instead of the Hadoop file system (HDFS), Hadoop's throughput was increased by about 6X (shown as the green line), although it was still significantly below the IMDG's throughput due to file I/O between the map and reduce phases.

In Summary

With the ever-increasing explosion in data for analysis and the need for fast insights on emerging trends, IMDGs offer a highly attractive platform for hosting map/reduce analysis. By simplifying the development model, IMDGs shorten the learning curve in developing analysis codes and eliminate the tuning steps required by more complex platforms. Because IMDGs run the analysis on data already staged in memory and load-balanced across grid servers, file I/O is eliminated and data motion is minimized. IMDGs also provide the infrastructure needed to automatically run analysis code on all grid servers in parallel and then combine the results with minimum latency. The net result is that by using an IMDG, application developers can easily analyze fast-changing, memory-based data and discover data patterns and trends that are vital to a company's success.

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.*

■ ■ ■

www.scaleoutsoftware.com