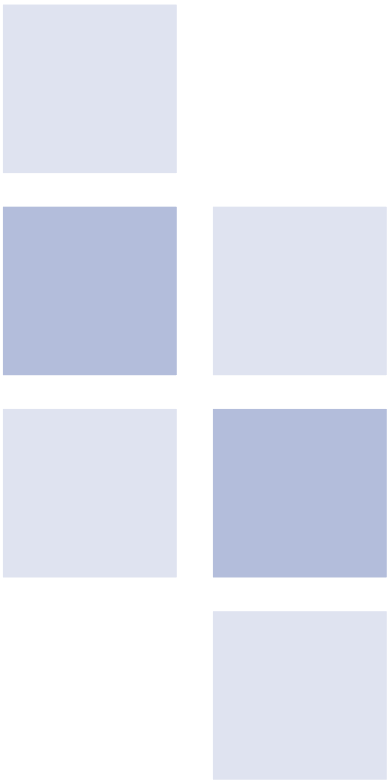


SCALEOUT SOFTWARE

How In-Memory Data Grids Can Analyze Fast-Changing Data in Real Time

by Dr. William Bain and Dr. Mikhail Sobolev, ScaleOut Software, Inc.

Twenty-first century computing infrastructures, especially cloud computing, are handling larger workloads than ever and generating huge, fast-growing data sets. "Big data" analysis platforms have enabled these data sets to be mined for important patterns and trends. With big data analysis, e-commerce vendors can target customers more precisely, financial analysts can quickly spot changing market conditions, manufacturers can tune logistics planning, and the list goes on.

Parallel computing techniques such as "map/reduce" have opened the door to dramatically reducing analysis times and are now proliferating in platforms such as open source Hadoop. However, conventional approaches have not addressed the need to analyze fast-changing data in real time to meet the needs of operational systems. For example, financial trading applications need to rapidly respond to fluctuating market conditions as market data flows through trading systems. Smart grid monitoring systems need to analyze a stream of telemetry from many sources to anticipate and respond to unexpected changes in a power grid. In both of these examples, the data sets hold live, fast-changing data in active, real-time operations.

The ability to continuously analyze operational data using big data techniques unlocks the potential for organizations to extract important patterns and trends that otherwise cannot be seen as the data rapidly changes. Unfortunately, popular big data systems such as Hadoop, which employ file-based storage and batch processing techniques, are not well suited for this challenge. However, the technology of in-memory data grids (IMDGs) offers important breakthroughs that enable real-time analysis of operational data. For example, recent measurements have demonstrated that an IMDG can deliver a complete map/reduce analysis every four seconds across a terabyte data set which is continuously being updated at the rate of one gigabyte per second.

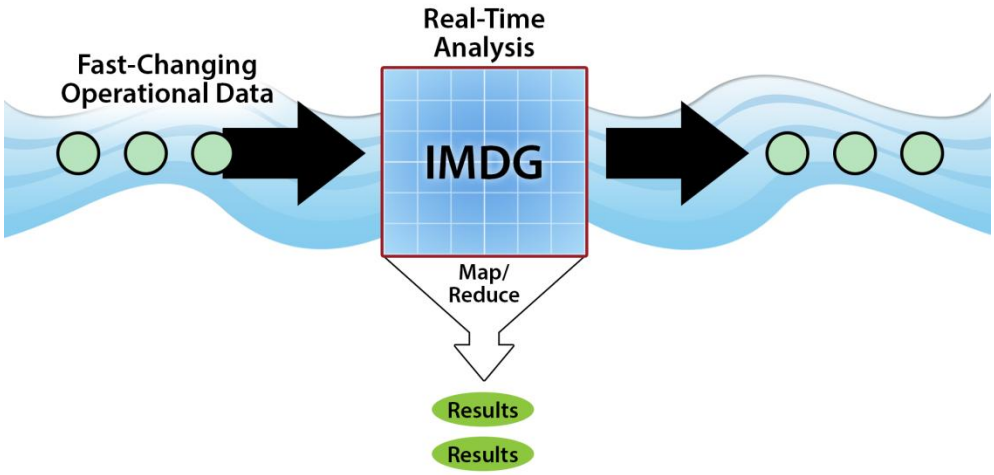
This article explains how IMDGs differ from other big data systems and deliver this important and exciting new capability to analyze fast-changing, operational data.

Analyzing Rivers of Data

To enable rapid updates to a fast-changing data set, IMDGs store data in memory across an elastic pool of servers. This enables the grid to scale its capacity seamlessly by adding servers to store and retrieve fast-changing data and handle growing workloads. Typically organized as a middleware software tier, 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. Additional capabilities, including eventing and distributed locking, make IMDGs a powerful data storage platform for operational data.

Data sets stored within an IMDG are organized as collections of logically related objects which can rapidly be created, updated, read, and removed. Having proven their value in storing fast-changing application data and scaling application performance, some IMDGs have integrated map/reduce analytics into the grid to perform continuous, real-time map/reduce analysis of stored data sets. IMDGs incorporate straightforward, well understood programming techniques to lower the learning curve, shorten the development cycle, and reduce analysis times. Importantly, advanced IMDGs integrate a parallel execution engine with memory-based storage to minimize data motion and enable real-time analysis.

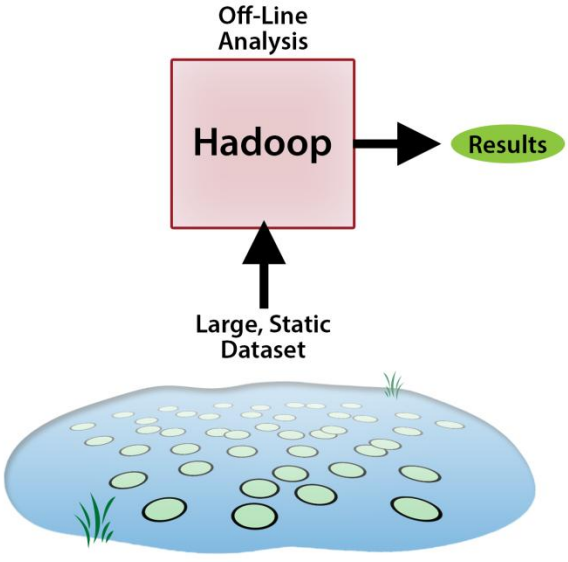
The flow of data through an IMDG can be visualized as a "river" of updates to a stored data set:



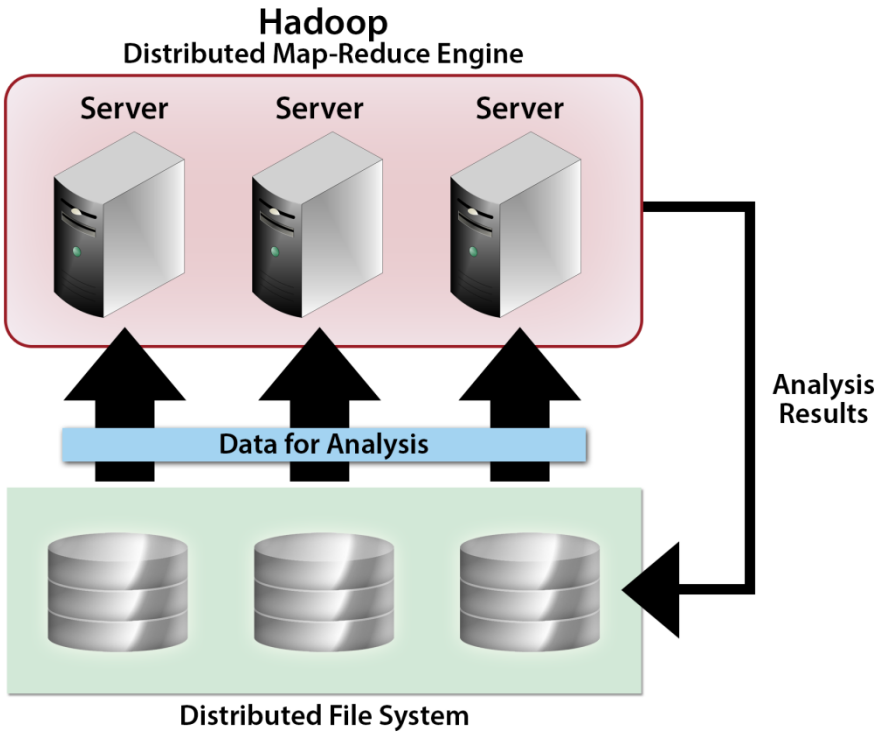
Applications typically use IMDGs to store operational data as part of their ongoing processing. For example, IMDGs store shopping carts for e-commerce systems, flight reservations for airline reservations systems, or market data for financial trading systems. In almost all cases, IMDGs are integrated as middleware into an operational system's scalable, parallel architecture to store fast-changing data with fast, predictable response times.

Incorporating continuous analytics into this workflow enables these applications to maintain a global view of fast-changing data sets and quickly respond to significant patterns and trends. For example, an e-commerce system can track shopping behavior during a sale and modify its offers in real-time to boost sales. Likewise, a financial trading application can evaluate and optimize its trading algorithms over the course of a trading session. As depicted in the above diagram, analysis results continuously flow out of the IMDG as map/reduce operations are repeatedly performed every few seconds or minutes.

Compare this scenario to a conventional big data analytics system such as Hadoop. These systems analyze very large data sets stored in a file system, such as the Hadoop Distributed File System (HDFS), or in a database. These data sets typically have been extracted from another storage repository and are not updated during analysis. In contrast with the IMDG's processing of a fast-changing river of data, Hadoop can be visualized as analyzing a large, relatively static "lake" of off-line data:



While conventional big data analytics systems have the ability to analyze very large data sets reaching into the petabytes, they are not well suited for analyzing fast-changing, operational data. Their storage systems usually stage data sets from other sources and are not designed to manage fast-changing, operational data, which require frequent updates and rapid access. Also, they must move data into memory for analysis, as illustrated in the following diagram:



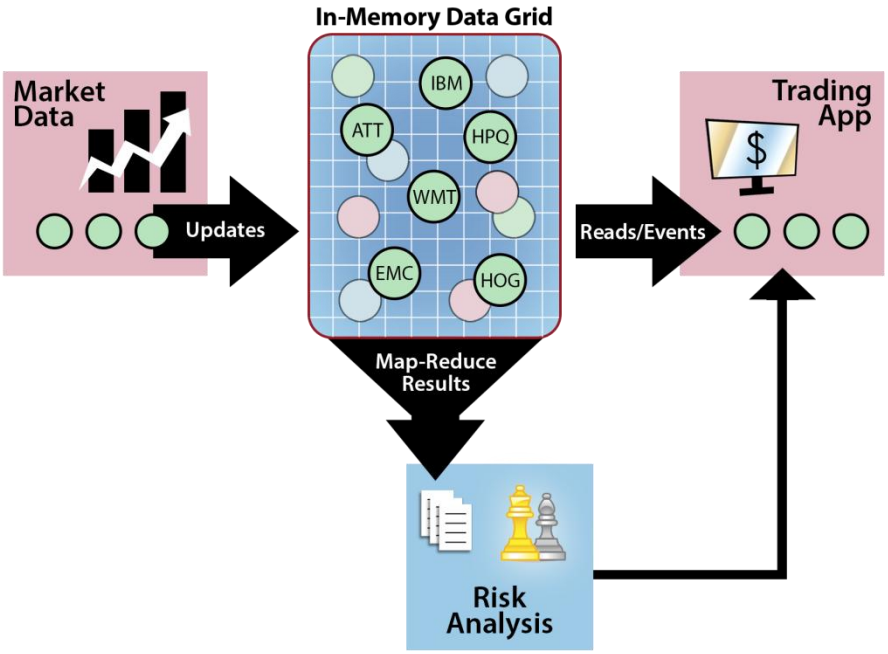
Data motion to and from a distributed file system increases both access latency and I/O overhead, significantly lengthening the execution time for analysis. In contrast, IMDGs perform analytics in place on memory-based data, avoiding data motion and driving down the time required to complete a map/reduce analysis. This enables IMDGs to analyze data significantly faster than Hadoop or other file-based analytics platforms, thereby delivering results with minimum latency. Because IMDGs typically are integrated into operational systems processing live data, they can immediately access this data for analysis and provide real-time feedback for optimizing operations or identifying exceptional conditions.

An Example in Financial Services

Consider a stock trading application that receives a market feed of stock price changes occurring during the trading day. This application employs various trading strategies to place new trades based on tracking the history of price changes for individual stocks and changing risk profiles. It can store a large set of stock histories in an IMDG as a collection of objects, one for each stock symbol being tracked and containing the price history of the stock. These objects are updated within the IMDG as prices frequently change.

Every few seconds, the IMDG can perform map/reduce analytics across either all or a selected set of stock symbols (such as a market sector), comparing potential returns, evaluating risk profiles, and optimizing the overall trading strategy. This ability to scan a large, fast-changing data set in real time gives the analyst an important new tool for detecting changing market conditions and optimizing the selection of trades to place.

The following diagram illustrates the flow of market data through an IMDG in this stock trading application and the continuous flow of map/reduce results used to analyze overall risk and tune the trading application in real time:



This architecture is equally suitable for many other applications, such as:

- a financial lending application reviewing incoming credit applications and apportioning funds across these requests to continuously minimize overall credit risk,
- an e-commerce application scanning shopping carts to detect popular product categories and optimize offers on the Web site in real time,
- a fraud detection system for credit card fraud transactions analyzing a flow of transactions to detect potential fraud and quickly allocate resources to highest threats,
- a logistics system or real-time control system (such as a smart grid) watching changes to assets within the system and alerting when potentially dangerous conditions are detected.

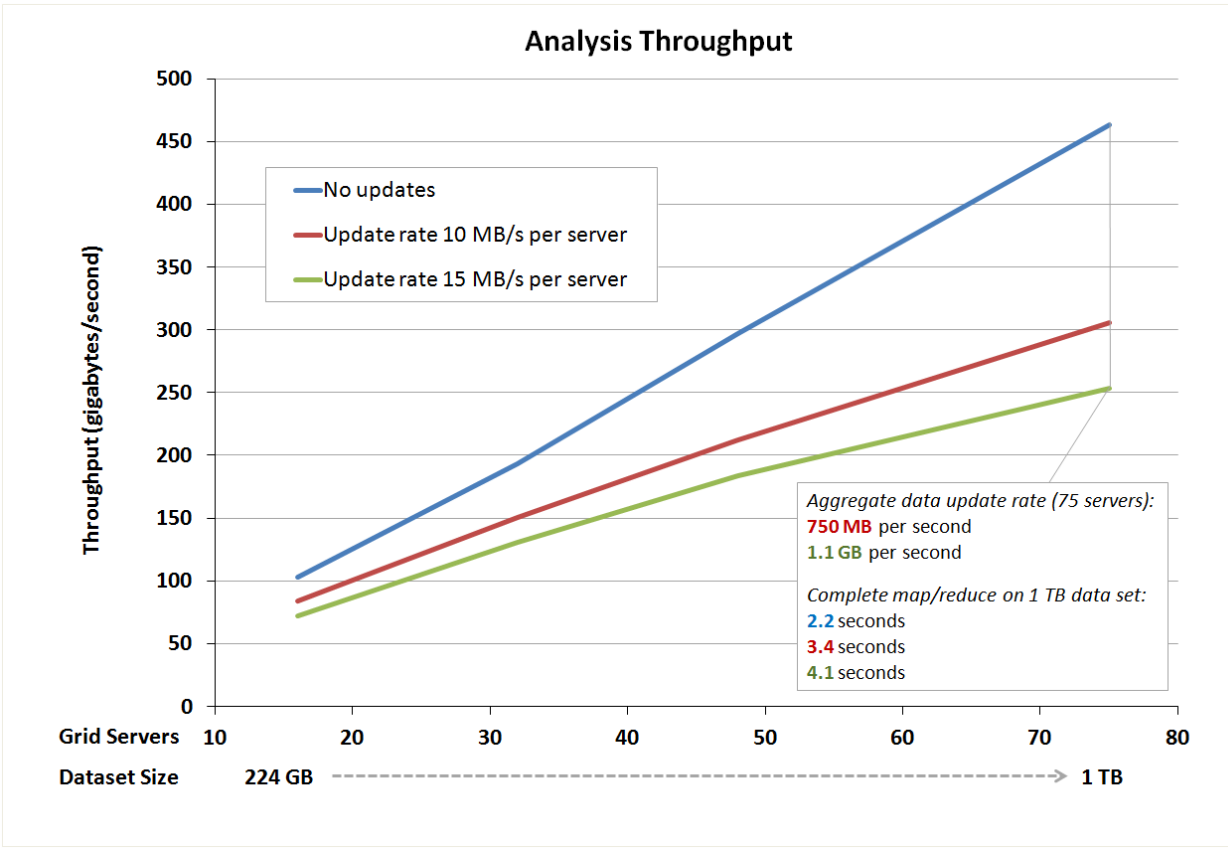
Scalable, Real-Time Performance

To demonstrate the performance capabilities of IMDGs running continuous map/reduce analytics on fast-changing data, ScaleOut Analytics Server™ was deployed on a compute cluster within the Amazon Web Services EC2 cloud environment. This in-memory data grid ran on an elastic pool of virtual servers and held partial price histories for a large pool of stock symbols. While the stock history objects were continuously updated to simulate a market feed updating stock prices, the IMDG repeatedly executed map/reduce analytics on the data set to model an ongoing analysis of stock trading strategies during a trading session.

With each stock history object holding 2 megabytes (MB) of data, a 75-server grid was able to host a terabyte (TB) of data and another terabyte of replicas to ensure high availability in case a server failed. Updates were performed at the rates of 10 MB/second and 15 MB/second per server. For 75 servers, the higher update rate resulted in an aggregate update throughput of 1.1 gigabytes (GB) per second. Note that the grid applies all updates to both the target objects and their replicas, doubling the overall update bandwidth required to sustain this rate.

Map/reduce operations were performed repeatedly on all stock objects. As shown in the following chart, the IMDG was able to complete a map-reduce operation with latencies

ranging from 4.1 seconds under the higher update load to 2.2 seconds when no updates were in progress:

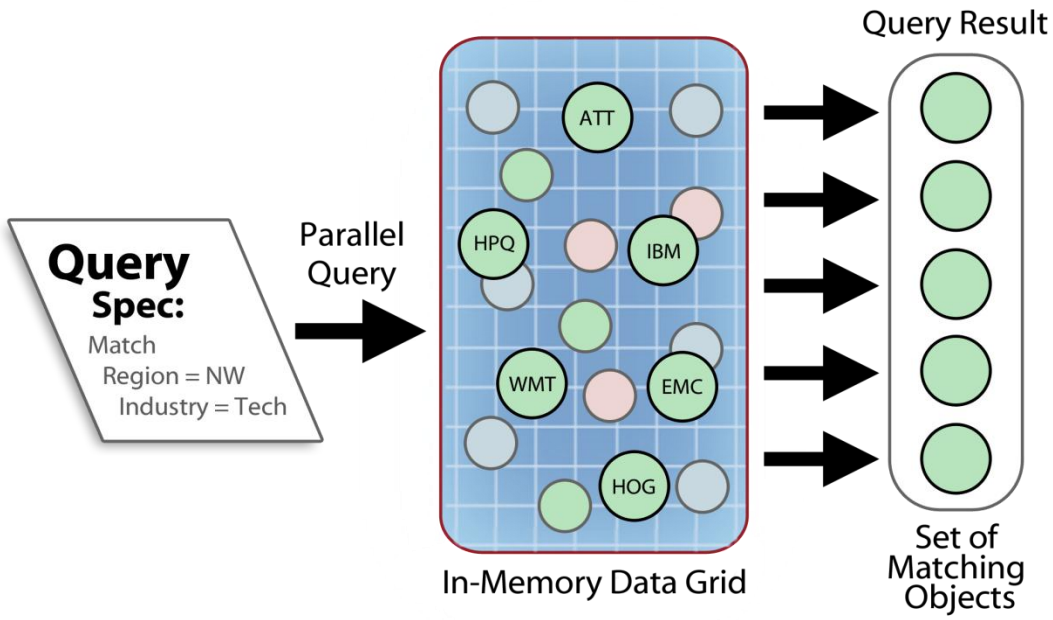


This chart shows the measured throughput of analysis operations in gigabytes per second of analyzed data for data sets ranging from 224 GB hosted on 16 servers to 1 TB hosted on 75 servers. For example, with 75 servers, the IMDG was able to analyze a 1 TB data set at the rate of 250 GB/second under a high update load. The linear increase in throughput shown in the chart enables analysis time to remain fixed as more servers are added to handle larger data sets. The IMDG's ability to store and update a terabyte of data at 1.1 GB/second while performing map/reduce analyses on the entire data set every 4.1 seconds makes it a very powerful software platform for managing fast-changing, operational data.

Simplifying the Development Model

Beyond delivering breakthrough performance for analyzing fast-changing data, IMDGs offer key advantages in simplifying the development model. Because they typically are

integrated into the application logic of operational systems, they employ object-oriented techniques which match the data schemas of application data. IMDGs store data as a collection of logically related objects which are accessible either by specifying an identifying key or by querying object properties. For example, an e-commerce Web site can store a collection of shopping cart objects identified by user IDs and queryable based on properties such as dollar value or time of last change. As seen above in the financial trading example, a large collection of stock histories can be stored as objects containing the price history of each stock and queryable on properties such as sector, market cap, or other criteria:

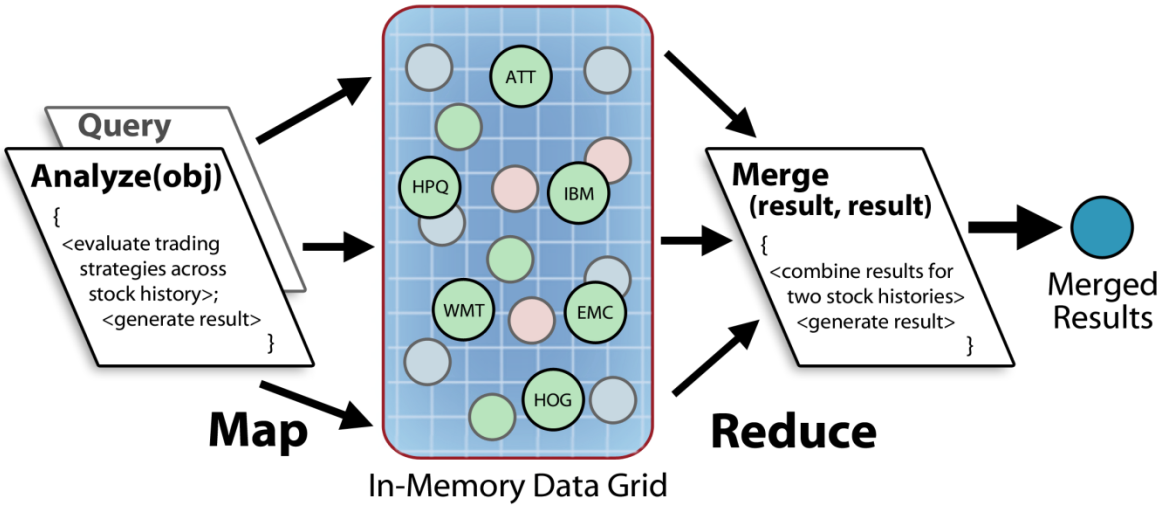


Storing data as queryable, object-based collections of operational data provides a straightforward means for identifying data to be analyzed using map/reduce techniques. In contrast, conventional big data analysis platforms, such as Hadoop, require that applications integrate with the data storage layer instead of with operational data residing in the business logic layer. As a result, applications must read and parse files to generate key/value pairs for analysis, which adds significant complexity to the application.

The IMDG's object-oriented storage and query mechanism can quickly scan a large data set in parallel to identify objects to be analyzed. Because the IMDG automatically load-balances the objects across a pool of grid servers, the data can be analyzed in place

without the need for network overhead or file I/O to input data for analysis. This avoids a key bottleneck and speeds up analysis times.

IMDGs take full advantage of the language's typing system in organizing and querying objects and in defining analysis code. This simplifies the creation of applications for parallel analysis. The developer typically just defines two methods, an "analyze" method that analyzes all queried objects of a given type and a "merge" method that combines the results generated by the analyze method. Developing these methods requires no special knowledge of parallel programming because they are written exactly as if they were to be sequentially executed on an object collection hosted on a single workstation. Some IMDGs can automatically deploy these "analyze" and "merge" methods on the grid servers for execution. The following diagram illustrates this application logic in the stock trading application described above:



A great deal of complexity in conventional big data analysis platforms can be found in the merging of analysis results using multiple key/value spaces and associated reduction algorithms used by conventional analysis platforms. While the full map/reduce programming pattern offers powerful semantics, its inherent complexity adds to development time and requires careful tuning. Many applications can sidestep this complexity and simplify application development by employing straightforward binary merging. ScaleOut Analytics Server uses this merging technique to automatically combine

results across all grid servers into a final result which is delivered back to the user in memory instead of in the file system.

In Summary

With the ongoing explosion in live, fast-changing data being managed by operational systems, such as e-commerce sites and financial trading platforms, the need for fast insights on emerging trends has become essential. Having already proven their value in storing fast-changing data, IMDGs provide an important tool for incorporating map/reduce analysis into operational systems and delivering continuous, real-time results.

IMDGs complement conventional big data analysis platforms, such as Hadoop, which target very large, static data sets typically hosted in file systems and employ batch processing techniques. In contrast to these platforms, IMDGs host fast-changing, operational data sets in memory and employ real-time analysis techniques. Today's IMDGs can hold terabytes of data and meet the needs of most operational systems. (Estimates by some analysts indicate that as much as sixty percent of data sets are smaller than ten terabytes.) Recent announcements by cloud vendors have extended the memory capacity of cloud-based servers to 240 gigabytes, which scales the storage capacity of IMDGs to the tens of terabytes and larger.

By simplifying the development model and automating execution, IMDGs also lower the learning curve in developing analysis codes and eliminate the tuning steps required by conventional analysis platforms. Because IMDGs analyze data already staged in memory and load-balanced across grid servers, they automatically deliver analysis results with minimum latency and maximum scalability. By using an IMDG, operational systems easily can start analyzing their fast-changing application data and discover data patterns and trends that are vital to optimizing the performance of these systems.

About the Authors

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.*

Dr. Mikhail Sobolev *is a software architect at ScaleOut Software, Inc. Mikhail has a Ph.D. in Applied Mathematics from the Moscow Institute of Physics and Technology.*

■■■

www.scaleoutsoftware.com