GridGain

The GridGain In-Memory Data Grid

A GridGain Systems In-Memory Computing White Paper

January 2017

Contents

Re-Imagining Ultimate Performance with In-Memory Computing

With the cost of system memory dropping 30% every 12 months, in-memory computing has become the first choice for a variety of workloads across all industries. In-memory computing can provide a lower TCO for data processing systems while providing an unparalleled performance advantage. In-memory computing technologies take many forms, ranging from in-memory data caches on a single server to in-memory databases (IMDBs), in-memory data grids (IMDGs), and comprehensive in-memory computing platforms (IMCPs). High performance in-memory computing technologies can even allow real-time analytics to run on operational datasets, enabling the implementation of hybrid transactional/analytical processing (HTAP) systems that can provide significant cost and complexity savings.

This white paper provides an overview of in-memory computing technology with a focus on in-memory data grids. It discusses the advantages and uses of in-memory data grids and introduces the GridGain In-Memory Data Fabric. Finally, it presents a deep dive on the capabilities of the GridGain solution.

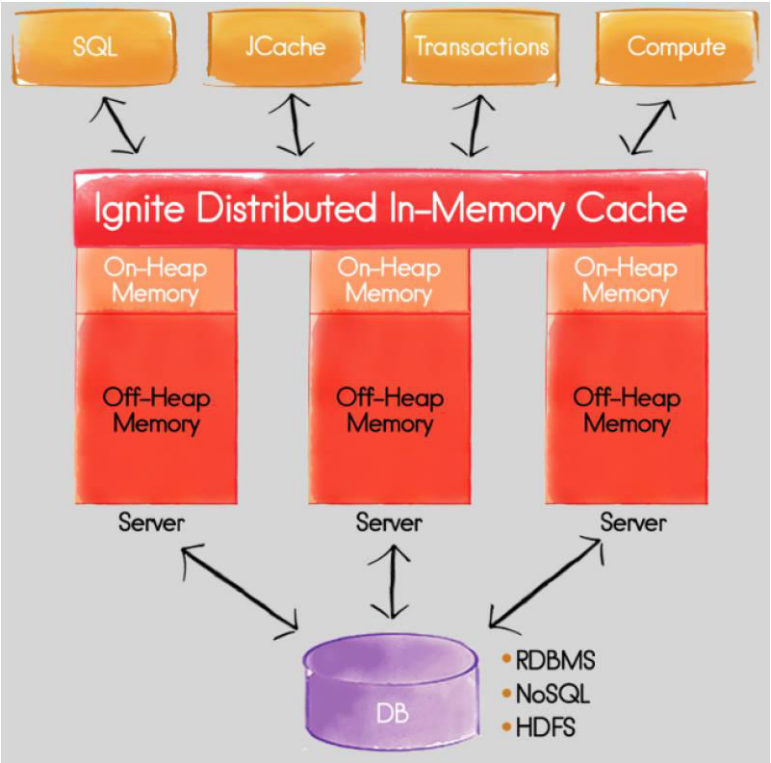
In-Memory Data Grids: Bringing High Performance to Big Data

Traditional approaches to application architecture are based on spinning disk technologies, which struggle to keep up with the expanding data volumes and velocities inherent in today's enterprise applications. To meet the need for a faster, scalable alternative, organizations are increasingly considering in-memory data grids (IMDGs) as the cornerstone of their next-generation architectures. This section will discuss how IMDGs are revolutionizing data processing – turning Big Data into Fast Data – and how their advantages are increased with added features that turn them into comprehensive in-memory computing platforms.

Why In-Memory Data Grids Are Faster and More Scalable Than Disk-Based Storage

An in-memory data grid stores all of its data in-memory, as opposed to traditional Database Management Systems that use disks as their primary storage mechanism. By making use of system memory rather than spinning disks, IMDGs are typically between a thousand and a million times faster than traditional DBMS systems.

Keeping data in memory is not the only reason IMDGs perform significantly faster than disk-based databases. Architectural differences are the main reason for the performance improvement. IMDGs use a memory-first and disk-second approach. This approach uses memory as primary storage and disk as secondary storage for backup and persistence. Since memory is a more limited resource than disk, IMDGs are built to scale horizontally: you can add nodes on demand in real-time. IMDGs linearly scale to hundreds of nodes. To reduce redundant data movement, they provide strong semantics for data locality and affinity data routing.



In contrast, disk-based databases have disk-first and memory-second architectures that are primarily optimized for disk-based, block device access. Although disk-based databases employ caching, the caching is generally a thin layer holding a small portion of the overall data in memory. Most of the data is still stored on disk. When the data-access pattern goes beyond what little can fit in cache, disk-based systems suffer from heavy paging overhead, due to data being brought in and out of memory. This overhead occurs both with traditional SQL RDBMSs and with the vast majority of new NoSQL systems that are predominately disk-based.

Since disk is a virtually unlimited resource when compared to memory, disk-based systems often hold all or most of the data set on disk. They are rarely designed to scale horizontally because there is no need to do so from a data storage standpoint. Such an architecture makes disk-based systems poorly suited to parallel processing, as compared to IMDGs, and disk-based architectures are prone to database overloading and thrashing.

Because IMDGs are linearly scalable and use in-memory data partitioning, they can localize processing logic to the nodes where the data resides. This capability makes IMDGs quite well suited to massively parallel processing (MPP), because they can use multiple CPUs across the cluster to process computations that work on multiple data sets in parallel.

In-memory data grids can scale out to handle large data and leverage the grid-computing power with MPP. This capability takes users beyond Big Data, into the real-time realm of Fast Data.

IMDGs and Fast Data: Realizing Real-Time Insights from Big Data

Data is “Fast” rather than just “Big” if it generates actionable analysis of information in a useful time frame. According to Gartner, the core of a successful Fast Data strategy is the ability to easily handle the following “three V’s”:

- **Velocity**: the ingestion and analysis of streaming and/or transactional data
- **Volume**: the ever-growing amount of data
- **Variety**: the expanding sources and types of data in the enterprise

In-memory computing technologies – particularly IMDGs and the in-memory computing platforms built around them – can handle the three V's because their speed and scalability lets them process data with the type of latencies that produce actionable insights and machine-driven decision-making.

For example, the GridGain in-memory computing solution provides crucial real-time analysis in the following Fast Data situations:

- **Real-time fraud detection.** The ever-increasing speed of commerce and the growing sophistication and organization of people with a criminal intent create an extremely difficult problem for financial institutions. Transaction processing gateways have to deal with a high volume of transactions even as they must be able to determine, in milliseconds, the legitimacy of any given transaction. This task is further complicated by the fact that transactions cannot be considered in isolation. Transactions must be considered in the context of multiple other transactions as well as the transaction history for a given person – and the evaluation must happen in real time. Otherwise, it is effectively useless. By the time an alert is generated, the fraudsters will have discontinued use of the credit card and have moved on to the next target. With a Fast Data solution, transactions can be analyzed for suspicious patterns compared against the historical purchase history to detect fraud in real-time and deliver a decision on the legitimacy of a transaction.

* **Biometrics and border security.** Border security is a growing concern in today's global climate. It is critical for front-line border security personnel to have the most up-to-date information and risk models at their disposal. With a Fast Data solution, information such as passport scans and biometrics for all incoming passengers at a port of entry can be quickly processed. Comparing the data against vast data sets and running it through complex risk models allows security personnel to identify potential security risks before granting entry.

* **Financial risk analytics.** A firm's competitiveness – and, ultimately, its profitability – is critically dependent on how quickly market data can be ingested and analyzed against open positions in a given portfolio. Market data becomes less useful for driving trading decisions as it ages, leaving slower firms at a disadvantage. Consequently, data must be ingested and risk calculations must be performed as quickly as possible. With a Fast Data solution, the streaming data can be consumed as quickly as it becomes available from the information provider, and risk calculations can be performed in parallel in near real-time. Furthermore, more complex modeling can be done simply by adding additional nodes, with no drop in latencies.

The use cases described above have some important features in common. They all share a low-latency service-level agreement (SLA) and require that complex analytics be performed across a high volume of data. To perform optimally in these use cases, an in-memory computing solution needs both an in-memory data grid *and* an in-memory compute grid. Adding a compute grid and other features turns an IMDG into an IMCP: an in-memory computing platform.

Beyond Data Grids: The Advantages of In-Memory Platforms

In-memory data grids have been around for years but the recent availability of comparatively inexpensive memory triggered the development of a variety of new in-memory data-management platforms. Combining IMDGs with compute grids, streaming support, service grids, and other features, IMCPs offer a strategic approach to in-memory computing. They deliver performance, scale, and comprehensive capabilities far above and beyond those

traditionally offered by in-memory databases (IMDBs), in-memory data grids (IMDGs), and other in-memory-based point solutions.

In-memory computing platforms such as the GridGain In-Memory Data Fabric can be inserted between existing application and data layers to provide up to 1,000,000x performance gains, along with scaling to manage petabytes of data for existing applications. An in-memory computing platform such as GridGain also provides ACID-compliant transactions and ANSI SQL-99 support while working with any RDBMS, NoSQL, or Hadoop database. The GridGain Unified API makes it simple to insert an in-memory computing platform with little or no code changes to existing applications. The result is an integrated, distributed, memory-based system that can manage and transact on large-scale data sets in real time, orders of magnitude faster than possible with traditional disk-based technologies.

But not all IMCPs have the same set of comprehensive features. Let's take a closer look at the platform that defines the category, the GridGain In-Memory Data Fabric.

Introducing the GridGain In-Memory Data Fabric

The GridGain In-Memory Data Fabric is an in-memory computing platform that delivers unprecedented speed and unlimited scale to modern data processing. Built on the Apache® Ignite™ open source project, GridGain enables high-performance transactions, real-time streaming, and fast analytics in a single, comprehensive data access and processing layer. The GridGain In-Memory Data Fabric easily powers both existing and new applications in a distributed, massively parallel architecture on affordable, industry-standard hardware. GridGain can run on premise, in a hybrid environment, or on a cloud platform such as AWS, Microsoft Azure, or the Google Cloud Platform.

GridGain provides a comprehensive set of APIs for developers. The Unified GridGain API supports SQL, C++, .NET, Java/Scala/Groovy, Node.js, REST, and more access for the application layer. The unified API connects cloud-scale applications with multiple data stores containing structured, semi-structured, and unstructured data (SQL, NoSQL, and Hadoop). It offers a high-performance data environment that allows companies to process full ACID transactions and generate valuable insights from real-time, interactive and batch queries

Designed for the most demanding use cases, including sub-millisecond SLAs, the GridGain in-memory computing solution lets you programmatically fine-tune large and super-large topologies with hundreds to thousands of nodes.

How GridGain Differs from In-Memory Databases

Unlike in-memory databases, GridGain works on top of existing databases , with no need to rip-and-replace or change an existing RDBMS, NoSQL or Hadoop database. Users can keep their existing database in place and deploy GridGain as a layer above it. GridGain can even automatically integrate with RDBMS products such as Oracle, MySQL, Postgres, DB2, Microsoft SQL, and others. This automatic integration feature generates the application domain model based on the schema definition of the underlying database and then loads the data.

Moreover, in-memory databases typically provide only a SQL interface, while GridGain provides a much wider ecosystem of supported access and processing paradigms. In addition to ANSI SQL-99, GridGain supports key/value stores, MapReduce, HPC/MPP processing, streaming/CEP processing, and Hadoop acceleration, all in one well-integrated, in-memory computing platform.

As part of an integrated solution, the GridGain in-memory data grid is built on top of existing capabilities such as pluggable auto-discovery, communication, marshaling, on-demand class loading, and distributed computations.

The next section provides an overview of the key capabilities that make GridGain a full-featured IMDF platform.

Overview of GridGain Features

An in-memory data grid is just one of the capabilities that GridGain provides. The GridGain In-Memory Data Fabric is made up of many components, including the following:

- **In-memory data grid**. The GridGain In-Memory Data Grid feature is horizontally scalable with the ability to add nodes on demand in real-time. It can linearly scale to hundreds of nodes with strong semantics for data locality and affinity data routing to reduce redundant data noise. The GridGain In-Memory Data Grid feature supports local, replicated, and partitioned data sets. The GridGain In-Memory Data Fabric offers an extremely rich set of data grid capabilities, including off-heap memory support, load-balancing, fault tolerance, remote connectivity, support for full ACID transactions and advanced security. The data grid's capabilities are discussed in greater detail in the next section, "GridGain Strategies for Data Distribution and Availability" below.

- **In-memory SQL grid**. The GridGain In-Memory SQL Grid provides in-memory distributed database capabilities to the GridGain platform. The In-Memory SQL Grid is horizontally scalable, fault tolerant and ANSI SQL-99 compliant. It fully supports all SQL and DML commands. The In-Memory SQL Grid allows you to interact with the GridGain platform using standard SQL language through the GridGain JDBC or ODBC APIs without custom coding. This provides true cross-platform connectivity even from languages such as PHP and Ruby which are not natively supported by GridGain.

- **In-memory compute grid**. Beyond simply distributing data, GridGain provides the ability to work with the data in place. Its compute grid lets it send computations, queries, and entry processing to where the data resides. Distributing and parallelizing compute tasks across clusters of computers helps GridGain compute and transact on large-scale data sets in real time – thus providing a highly scalable, distributed framework for parallel High Performance Computing (HPC) and massively parallel processing (MPP).

- **In-memory streaming/CEP**. The GridGain real-time streaming feature uses programmatic coding with rich data-indexing support to provide complex-event-processing (CEP) querying capabilities over streaming data. GridGain also provides comprehensive support for customizable event workflows and windowing.

- **In-memory file system**. The GridGain acceleration feature for Hadoop is a high-performance implementation of a job tracker that replaces the standard Hadoop MapReduce and HDFS with the compliant Ignite In-Memory File System (IGFS). IGFS works in dual mode, either in tandem with HDFS or as a standalone primary

file system in the Hadoop cluster. In the latter case, it serves as an intelligent caching layer with HDFS configured as the primary file system.

- **In-memory service grid**. The GridGain service grid provides complete control over services being deployed on the cluster. It lets you control how many instances of your service should be deployed on each cluster node, ensuring proper deployment and fault tolerance.

- **Topic-based messaging**. This feature allows for topic-based, cluster-wide communication among all nodes. Messages with a specified topic can be distributed either to all nodes or only to the sub-group of nodes that have subscribed to that topic.

- **Apache Spark integration.** GridGain provides an implementation of the Spark RDD abstraction that lets you easily share state in-memory across Spark jobs. GridGain also provides ANSI SQL-99 compliance which allows to use GridGain SQL queries directly from Spark getting performance boost because GridGain uses indexes while querying the data while Spark must run a full scan of the data for every query.

- **Advanced security**. The GridGain Enterprise Edition comes with a set of security features that enable privileged access for remote clients and other cluster members. This capability is achieved by configuring pluggable authentication and authorization mechanisms. In addition, full auditing capabilities allow recreating of any event in the system so it can be traced back to the user responsible for the event.
- **Advanced clustering**. GridGain provides one of the most sophisticated clustering technologies on Java Virtual Machines (JVMs). The ability to connect and manage a heterogeneous set of computing devices is at the core of GridGain's distributed processing capabilities. Clustering capabilities are fully exposed to the end user, and developers have full control with many advanced features. Some features are pluggable, including cluster-topology management, automatic discovery (on LAN and WAN as well as AWS, and other cloud platforms), cluster-wide message exchange (unicast, broadcast, and actor-based), "split-brain" cluster-segmentation resolution, and event storage. Other important clustering capabilities include on-demand and direct deployment, cluster-aware versioning, support for complex leader-election algorithms, and support for virtual clusters and node groupings.

These comprehensive capabilities make GridGain suitable for a much broader set of use cases than a typical IMDG. The upcoming sections provide a closer look at three important aspects of GridGain: how it handles data availability and consistency issues, how it supports both transactional and analytical data processing, and how it supports user and developer access with a wide array of tools.

GridGain Strategies for Data Distribution and Availability

An in-memory data grid provides extremely high data availability by keeping the data in memory and highly distributed (that is, parallelized). With this strategy come challenges. How do you make sure all data remains available and efficiently distributed as nodes or data centers join or leave the grid? How do you use tiered memory effectively to minimize latency in data processing? And how do you integrate the in-memory data grid with persistent data storage to keep the two in sync?

The sections that follow describe how GridGain meets these challenges through strategic data replication, rebalancing, off-heap allocation, and integration with persistent data storage.

How GridGain Replicates Cache Data

One of the distinguishing features for most IMDGs is the way they support data replication. There are several common approaches, ranging from no replication at all to full replication, in which every node sees an identical data set. The most versatile approach is partitioning in which each node receives and is responsible for a chunk of data. Systems that support this approach are usually the most scalable, as the addition of nodes to the cluster allows more data to be stored in memory. What makes an IMDG truly powerful is the ability to use different replication modes on different data sets within the same application — and the ability to easily cross-query between these data sets in real time.

The GridGain In-Memory Data Fabric offers both of these capabilities: multiple, simultaneous replication modes and the ability to cross-query among them. To support the full range of replication approaches, it provides three different modes of cache operation: Local, Replicated, and Partitioned.

Local mode. Local mode is the most lightweight mode of cache operation since no data is distributed to other cache nodes. It is ideal for scenarios in which data is read-only or can be periodically refreshed at a defined expiration frequency. It also works very well with read-through behavior where data is loaded from persistent storage on misses. While local caches are not themselves distributed, they still have all the features of a distributed cache, including automatic data eviction, expiration, disk swapping, data querying, and transactions.

Replicated mode. In replicated mode, all data is replicated to every node in the grid. This mode provides the highest availability because all data is available on every node. However, in this mode, the need to propagate every data update to all other nodes can affect performance and scalability. Also, because the same data is stored on all grid nodes, the size of a replicated cache is limited by the memory available on the node that has the smallest amount of RAM. In other words, no matter how many nodes are in your data grid, the maximum amount of data you can cache is fixed. This mode is best for scenarios in which cache reads are much more frequent than cache writes, and in which the most important criteria for your use case are data availability and locality.

Partitioned mode. Partitioned mode is the most scalable mode of distributed caching. The overall data set is divided equally into partitions, and all partitions are split equally among participating nodes – essentially creating one massive, distributed memory for caching data. This approach lets you store as much data as can fit in the total memory available across all nodes. The more nodes you have available, the more data you can cache. Unlike in replicated mode, where updates are expensive because every node in the grid needs to be updated, updates in partitioned mode are cheap. The reason is that only one primary node (and, optionally, one or more backup nodes) must be updated for each key. However, reads become somewhat more expensive, because only certain nodes have the data cached. In order to avoid extra data movement, it is important to always access the data on the node that has that data cached. This approach is called affinity colocation and is strongly recommended when working with partitioned caches.

How GridGain Maintains Availability When Nodes Fail

A common misconception about IMDGs is that you lose all data in the event of a node failure. This is not the case if you designed your grid topology correctly. In a partitioned scenario, you can configure as many backups as you wish. In the event of a primary node failure, the grid will automatically promote the backup node to primary and data access will continue without interruption. Furthermore, you can write your data through to underlying persistent storage in a variety of ways, as discussed in "How GridGain Integrates its IMDG with Pluggable Persistent Data Stores" below. By writing through to persistent storage, you can ensure that data is not lost in the event the whole cluster is lost.

For systems involving multiple data centers, GridGain provides a data-center replication option. This option ensures that if one data center goes down, another data center is fully capable of picking up its load and data. When data center replication is turned on, each data center consistently backs up its data to one or more other data center, as specified by your configuration. When a crashed data center comes back online, it will receive all the updates it has missed from a backup data center. The data center replication mode options are described below.

Active-active replication mode. In this mode, both data centers are fully operational online and each acts as a backup copy for the other.

Active-passive replication mode. In this mode, only one data center is active, and the other data center serves as its backup.

Asynchronous, non-transactional replication mode. GridGain currently supports this mode of replication between data centers. In this mode, a transaction will usually complete before the replication is completed. Typically, the data is concurrently buffered on one data center and then gets flushed to another data center either when the buffer fills up or when a certain time period elapses. Asynchronous non-transactional mode is generally a lot faster than the other modes but it also introduces a lag between updates on one data center and replication to another.

How GridGain Rebalances Data When Nodes Join or Leave

Whenever a node joins or leaves the topology, the remaining nodes will make an attempt to repartition and rebalance data so that it is spread uniformly across the cluster. The data in the grid will therefore remain consistent and equally balanced. In cases where data is replicated to all nodes, the new node will try to load the full set of data from the existing nodes. When data is partitioned across the grid, the current node will try to load only the entries for which the current node is either primary or backup.

GridGain's rebalancing options are described below.

Synchronous rebalancing. In synchronous mode, distributed caches will not start until all necessary data is loaded from other available grid nodes. Essentially, existing data grid nodes will keep operating as usual, but the new nodes will not be allowed to start until they have all the necessary data.

Asynchronous rebalancing. If asynchronous mode is turned on, which is the default behavior, distributed caches will start immediately and will load all necessary data from other available nodes in the background. This is the most efficient rebalancing mode and should be used unless the application logic absolutely requires synchronous mode.

Delayed rebalancing. There are some cases in which it is inefficient to start preloading immediately after a new node starts or an existing node leaves. For example, if you start multiple nodes one by one, you may want to wait until all nodes are started, in order to avoid multiple preloading processes that take effect sequentially as each individual node starts. Similarly, if you plan to restart existing nodes, you will find it most efficient to start rebalancing only after the nodes are restarted, rather than right after they are stopped. Using GridGain's option for delayed rebalancing in such cases can significantly reduce network overhead and improve system performance. Note that delaying rebalancing has no effect on data consistency. The data in the grid always remains consistent, regardless of when rebalancing starts or completes.

How GridGain Uses Off-Heap Memory to Speed Processing

GridGain provides a tiered approach to memory that allows data to migrate between on-heap, off-heap, and swap storages. Off-heap overflow allows the grid cache to store data outside of the JVM heap (that is, in off-heap memory). When data is allocated off-heap, Java Virtual Machine Garbage Collection (JVM GC) does not know about it and therefore does not slow down or affect your application with long garbage collection pauses. Long garbage collection pauses typically occur with large data sets using on-heap memory configurations. Depending on the specificity of a Java application, you can start with a relatively small heap (e.g., less than 512 MB) and allow GridGain to use hundreds of gigabytes of memory as off-heap data cache. One of the distinguishing characteristics of GridGain off-heap memory is that the on-heap memory footprint is constant and does not grow as the size of off-heap data grows.

Traditionally, JVM GC pauses have been mitigated by starting multiple JVMs on a single server. However, this approach is less than ideal for applications that need to co-locate large amounts of data in a single JVM in order to meet low-latency processing requirements.

How GridGain Integrates its IMDG with Pluggable Persistent Data Stores

In-memory data grids are often used in conjunction with an external persistent data store, such as a disk-based database or a file system. When a persistent store is configured in GridGain, the system automatically handles loading data and updating data into the data store. Moreover, the persistent store plugs into data grid transactions, so updating the underlying persistent store is part of the same transaction as the data grid update and if one fails, the other fails as well. GridGain allows read-through, write-through, and write-behind integration strategies, discussed below.

Read-through and write-through. Properly integrating with persistent stores is important whenever you want read-through or write-through behavior. *Read-through* means that data will be automatically read from the persistent store whenever it is not available in cache. *Write-through* means that data will be automatically persisted whenever it is updated in cache. With GridGain, all read-through and write-through operations participate in overall cache transactions and are committed or rolled back as a whole to ensure data consistency.

Write-behind caching. In a simple write-through mode, each cache put and remove operation will involve a corresponding request to the persistent data store. Therefore, the overall duration of the cache update might be relatively long. Additionally, an intensive cache-update rate can cause an extremely high load on the underlying

persistent store. For such cases, GridGain offers an option to perform an asynchronous storage update, a strategy known as *write-behind*. The key concept of this approach is to accumulate updates and then asynchronously flush them to the persistent store as a bulk operation. In addition to providing performance benefits (because cache writes become faster), the write-behind approach scales much better than the write-through approach, assuming that your application can tolerate delayed persistence updates. When the number of nodes in the topology grows and every node performs frequent updates, it can be too easy to overload the underlying persistent store. By using a write-behind approach, you can maintain a high throughput of writes in the GridGain In-Memory Data Fabric without bottlenecking at the persistence layer. Moreover, GridGain can continue operating even if your underlying persistence store crashes or otherwise becomes unavailable. In this case, the persistence queue will keep storing all the updates until the persistent data store once again becomes available.

Hybrid Transactional/Analytical Processing with GridGain

Just as important to an in-memory data platform as its data-distribution strategies are how it supports transactions and how it supports queries and analysis. Does it support ACID transactions and employ efficient strategies for lock-free concurrency? Does it support a full range of SQL queries and distributed joins? Robust support for both transactions and queries in the distributed environment is a crucial part of a successful Hybrid Transactional/Analytical Processing (HTAP) strategy, along with the speed and scale that an in-memory data platforms enable. The sections that follow describe how GridGain supports transactions and queries to meet the challenges of HTAP.

How GridGain Supports Distributed ACID Transactions

GridGain supports distributed ACID transactions that use a two-phase commit (2PC) protocol to ensure data consistency within the cluster. The GridGain 2PC protocol has been optimized to ensure minimal network chattiness and lock-free concurrency.

Distributed data grid transactions in the GridGain in-memory computing platform span data on both local and remote nodes. GridGain supports automatic enlisting into JEE/JTA transactions. It also allows users to create more lightweight transactions than they could otherwise. Such transactions are often more convenient to use.

Full ACID property support. GridGain transactions support all ACID properties that you would expect from any transaction, including support for optimistic and pessimistic concurrency levels and read-committed, repeatable-read, and serializable isolation levels. When the optimistic concurrency level is used with the serializable isolation level, you get all the benefits from GridGain Deadlock-Free Transactions. In this mode, keys can be accessed in any order inside of transactions. This accessibility occurs because transaction locks are acquired in parallel with an additional check, allowing GridGain to avoid deadlocks. If pessimistic transactions are used because optimistic and serializable modes do not work for a particular use case, you can enable the GridGain deadlock-detection capability which provide easy debugging and fixes for possible deadlocks. Finally, if a persistent data store is configured, the transactions will also automatically span the data store.

Multi-version concurrency control (MVCC). GridGain uses an advanced and optimized implementation of MVCC. This implementation provides practically lock-free concurrency management by maintaining multiple versions of

data instead of using locks with a wide scope. Thus, the GridGain MVCC provides a backbone for high performance and overall system throughput for systems under load.

GridGain In-Memory SQL Grid

Most in-memory data grids provide access to the data by means of a primary key. Few systems provide a custom, expressive query language for rich data-querying capabilities. Very few systems allow the use of standard SQL for data querying. Systems that allow for execution of distributed joins on the data are the most rare. Generally, the ability to execute standard SQL joins in a distributed environment is one of the hardest features to support. However, when implemented successfully, as it is in GridGain, this capability provides the most elegant and efficient method for accessing data.

Supported SQL grammar, functions, and joins. The GridGain In-Memory SQL Grid provides the ability to query and work with data using standard SQL. You can issue SQL commands either via API-based mechanisms or via a JDBC or ODBC interface. There are almost no restrictions as to which SQL syntax you can use. The only rule is that SQL queries must be ANSI-99 compliant. There is full support for all SQL and DML commands including SELECT, UPDATE, INSERT, MERGE and DELETE queries. All inner, outer, or full joins are supported. The ability to join different classes of objects stored in the DBMS or across different caches makes GridGain queries a very powerful tool. Typically, all indices are kept in memory, either on-heap or off-heap – resulting in very low latencies for query execution.

GridGain supports two types of distributed joins: collocated and non-collocated. Both types of joins let you execute cross-cache SQL queries to get a valid and consistent result. The collocated distributed joins require you to collocate the data of the caches that are being joined in a query. GridGain allows you to easily enable collocation by using the affinity key concept in which the data of one business entity is stored on the same node where the other business entity resides.

In cases where it's not feasible to collocate all the data, GridGain's non-collocated distributed join capabilities enable applications to execute very complex analytics and operational queries. Performance of non-collocated joins is worse than performance of affinity-collocation-based joins, though, because queries require more network round-trips and data movement between nodes. Usually, it is possible to collocate enough data such that 95% of queries will be executed in the fastest and most efficient manner with the remaining 5% using the less efficient non-collocated distributed joins. This strategy lets you execute 100% of your queries with GridGain.

Geospatial functionality. The GridGain in-memory computing platform provides geospatial functionality for applications and services that require gathering, storing, and processing geospatial data. For this reason, the GridGain in-memory computing platform allows for querying and indexing of geometric data types such as points, lines and polygons and the consideration of the spatial and locational relationships between them.

Text-query support. In addition to supporting SQL queries, GridGain supports text queries by using the Lucene text engine for efficient indexing of text data. You can have continuous queries that listen for cache updates and perform an action based on the type of update. You can also have scan queries that allow iteration over cache data with maximum efficiency and fault tolerance.

Tools for System Management and Security

The final piece in any in-memory platform is the user toolkit. Powerful and easy-to-use tools are essential for helping users make the most of the platform's capabilities. The sections that follow describe some important tools that GridGain provides for users, developers, and service providers.

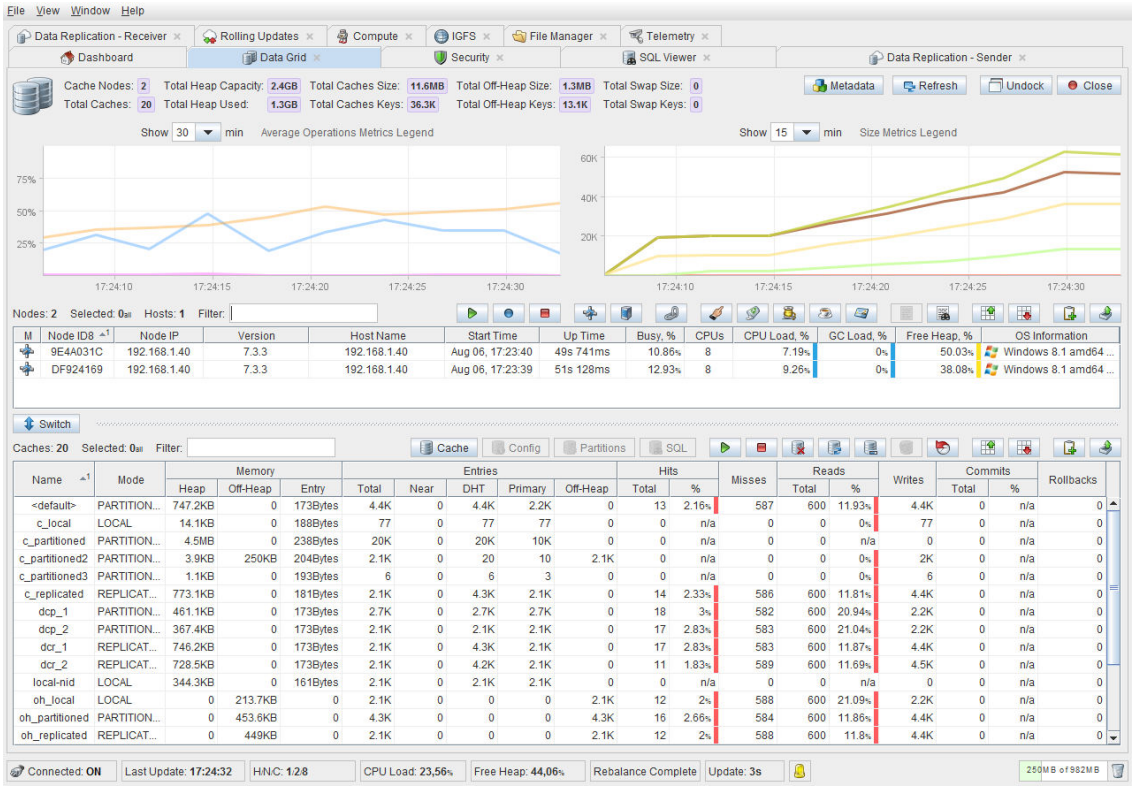
Web Console: A Free Management and Monitoring Tool

The GridGain Web Console, a free online tool for GridGain and Apache Ignite, allows you to manage and monitor GridGain or Apache Ignite clusters. It also provides an interactive configuration wizard that helps you create and download configuration files and code snippets for your GridGain or Apache Ignite projects. Additionally, the tool lets you automatically load SQL metadata from any RDBMS, run SQL queries on your in-memory cache, and view execution plans, in-memory schema, and streaming charts.

GridGain Visor: A Comprehensive System-Management Interface

The GridGain In-Memory Data Fabric comes with a comprehensive, GUI-based management and monitoring tool called GridGain Visor. Visor provides substantial capabilities for system operation, management, and monitoring, extending the feature set available with the GridGain Web Console.

The starting point for the GridGain Visor management console is the Dashboard tab, which provides an overview of grid topology and many graphs and metrics, as well as an event panel displaying all relevant grid events. The Data Grid Tab displays detailed information about the data grid and lets you manage and monitor the data nodes. You can also manage and monitor individual caches within the data grid.



Coding Tools and the Zero Deployment Feature

The GridGain zero deployment feature means that you don't have to deploy any components individually on the grid – all code and related resources are deployed automatically. This feature is extremely useful during both development and production stages, as it removes the need for lengthy Ant or Maven rebuild routines or copying of ZIP/JAR files. The philosophy is very simple: write your code, hit a run button in the IDE or text editor of your choice, and the code is automatically deployed on all running grid nodes. Note that you can change existing code as well, in which case the old code is automatically undeployed and new code deployed with proper versioning maintained.

Advanced Security and Auditing Tools

When the GridGain pluggable authentication and authorization mechanisms are turned on, cluster members cannot connect to the grid unless they have been authenticated, and they cannot perform any actions they are not authorized to perform.

In the event of an undesirable occurrence, it's important to be able to recreate what happened and trace it back to the authenticated subject responsible for the event. GridGain has the following audit mechanisms for ensuring that every event occurring in the system is traceable. You can trace the responsible party's username, IP address, and cluster-member address, as well as information about affected data (field names and values), indirect access, and parent events (in the case when one event is caused by another). You can also use the pluggable Event Storage SPI to log events in any desired format to any underlying storage system (file system or a database).

Interfaces for Service Providers (SPIs) and PNP Extensibility

The Service Provider Interface (SPI) architecture is at the core of the GridGain In-Memory Data Fabric. It allows GridGain products to abstract various system-level implementations from their common reusable interfaces. Instead of hard-coding every decision about internal implementation of the product, GridGain products expose a set of interfaces that define their internal view on their various subsystems. Users can then choose either to use the built-in implementations or to create their own when they need different functionality.

GridGain products provide SPIs for the following 14 subsystems, all of which can be freely customized:

- Cluster discovery
- Cluster communication
- Deployment
- Failover
- Load balancing
- Authentication
- Task checkpoints
- Task topology resolution
- Resource collision resolution
- Event storage
- Metrics collection
- Secure session
- Swap space
- Indexing

Having the ability to change the implementation of each of these subsystems allows GridGain products to be used very flexibly in a real-world environment. GridGain software blends naturally into almost any environment and integrates easily with practically any host ecosystem.

Remote Connectivity Tools

GridGain includes a number of Remote Client APIs that allow users to remotely connect to the GridGain cluster. Remote Clients are available for multiple programming languages including Java, C++, and .NET C#. Additionally, you can write your client for your favorite programming language by relying on the GridGain REST protocol. The Remote Clients provide a rich set of capabilities that can be used without a client runtime being part of the GridGain cluster. For example, they enable you to run computational tasks, access clustering features, perform affinity-aware routing of tasks, and access the data grid.

In-Memory Infrastructure for a Fast Data Future

As disk-based technologies fail to keep up with the demands of Big Data, the superior speed and scalability of in-memory computing technologies make them the clear choice for data-intensive applications. Platforms built around in-memory data grids can inexpensively harness the memory and computing power of a large array of nodes built on commodity hardware. With the right combination of additional features, they can provide the essential tools for ingesting and analyzing very high volumes of streaming and transactional data with ease – turning Big Data into Fast Data.

The GridGain In-Memory Data Fabric is a comprehensive in-memory data platform built on the Apache® Ignite™ open source project. It combines a powerful data grid with other key features – compute grid, service grid, streaming support, Hadoop acceleration, and more – to deliver large-scale transaction processing and actionable analysis in real time. With data replication strategies that ensure consistency and availability, GridGain supports ACID transactions and ANSI SQL-99 compliant queries (including joins) in a distributed data environment. These capabilities combine with an extensive set of tools – for system management, security, ease of deployment, and more – to create an ideal platform for hybrid transactional/analytical processing in a Fast Data world.

Contact GridGain

To learn more about how GridGain In-Memory Data Fabric can help your business, please email our sales team at sales@gridgain.com, call us at +1 (650) 241-2281 (US) or +44 (0) 7775 835 770 (Europe), or complete our contact form to have us contact you.

About GridGain Systems

GridGain Systems is revolutionizing real-time data access and processing by offering enterprise-grade in-memory computing solutions built on Apache® Ignite™. GridGain solutions are used by global enterprises in financial,

software, ecommerce, retail, online business services, healthcare, telecom and other major sectors. GridGain solutions connect data stores (SQL, NoSQL, and Apache™ Hadoop®) with cloud-scale applications and enable massive data throughput and ultra-low latencies across a scalable, distributed cluster of commodity servers. GridGain is the most comprehensive, enterprise-grade in-memory computing platform for high volume ACID transactions, real-time analytics, and hybrid transactional/analytical processing. For more information, visit gridgain.com.

