GridGain
IN-MEMORY COMPUTING

In-Memory Streaming

White Paper

GridGain Systems, 2013

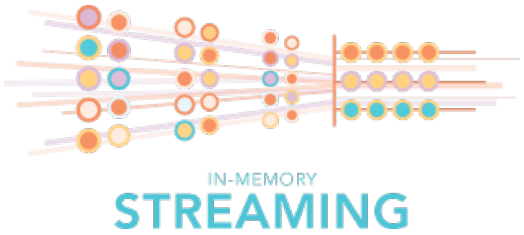


Table of Contents:

Re-Imagining Ultimate Performance

What is In-Memory Computing?

Data volumes and ever decreasing SLAs have overwhelmed existing disk-based technologies for many operational and transactional data sets, requiring the industry to alter its perception of performance and scalability. In order to address these unprecedented data volumes and performance requirements a new solution is required.

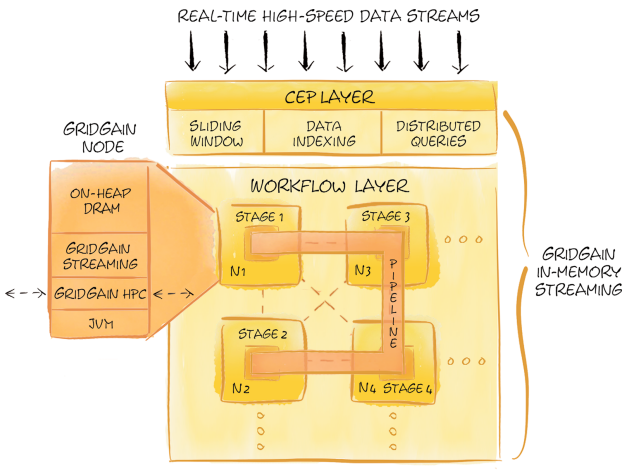
In-Memory Computing is characterized by using high-performance, integrated, distributed memory systems to manage and transact on large-scale data sets in real time, orders of magnitude faster than possible with traditional disk-based technologies.

With the cost of system memory dropping 30% every 12 months In-Memory Computing is rapidly becoming the first choice for a variety of workloads across all industries. In fact, In-Memory Computing paves the way to a lower TCO for data processing systems while providing an undisputed performance advantage.

Stream Processing At A Glance

What is In-Memory Streaming?

Stream processing fits a large family of applications for which traditional processing methods and disk-based storages, like databases or file systems, fall short. Such applications are pushing the limits of traditional data processing infrastructures. Processing of market feeds, electronic trading by many financial companies on Wall Street, security and fraud detection, military data analysis - all these applications produce large amounts of data at very fast rates and require appropriate infrastructure capable of processing data in real-time without bottlenecking.



One of the most common use cases for stream processing is the ability to control and properly pipeline distributed events workflow. As events are coming into system with high rates, the processing of events is split into multiple stages and each stage has to be properly routed within a cluster for processing. As stages are executing they produce more events which are routed further until the final result is returned. Such systems are usually coupled with a very effective and fast communication protocol as messaging cannot become a bottleneck in event workflow.

Routing and locality of events becomes especially important when stream data must be integrated with stored data for processing. Accessing data from disk is never an option as any type of disk I/O would introduce a significant bottleneck. Hence, data is usually kept in some in-memory data grid and is partitioned across multiple cluster nodes. In such cases it is extremely important to make sure that events are routed exactly to the nodes on which the required data resides. In the absence of affinity routing, data would have to be delivered across the network for every event, effectively bringing the system to a full halt.

> Affinity collocation of event processing with nodes on which data resides is essential for achieving high-throughput and scalability in real-time streaming.

Another important use case for streaming is the support for Complex Event Processing (CEP). One of the key features of many CEP systems is the ability to control the scope of operations on streamed data. As streaming data never ends, an application must be able to provide a size limit or a time boundary on how far back each request or each query should go.

The sliding window construct serves exactly this purpose and is usually one of the core concepts behind a CEP product. Sliding windows allow you to answer questions like “what are the top 10 users over the past hour?” or “what is the average sale price for product A over the past 24 hours?”. Therefore, CEP systems must keep track of the new events coming into the window on one side, as well as events coming out of the window on another side.

With sliding windows the ability to query and aggregate data passing through such windows must be very efficient. As events continue streaming in with a high rate, such queries must be extremely fast, and therefore CEP systems generally have strong data indexing capabilities with extensive support for data aggregation routines.

In-Memory Streaming vs. Other Solutions

What makes In-Memory Streaming a unique solution?

In the past few years several technologies have emerged specifically to address the challenges of processing high-volume, real-time streaming data. Generally such technologies evolve around some specific streaming use case. For example, a product like Storm is mainly focused on providing event workflow and routing functionality without focusing on sliding windows or data querying capabilities. Products in the CEP family, on the other hand, are mostly focused on providing extensive capabilities for querying and aggregating streaming events, while generally neglecting event workflow functionality.

Customers looking for a real-time streaming solution usually require both, rich event workflow combined with CEP data querying, and as a result are left with the difficult task of integrating different streaming technologies together. Such integration is rarely effective or simple as knowledge of data locality across different products is minimal, and proper affinity data routing

is essential in achieving any kind of scalability. GridGain In-Memory Streaming is squarely focused on providing both event workflow and CEP capabilities, in an integrated product.

GridGain In-Memory Streaming combines both event workflow and CEP capabilities fully integrated in one product.

SQL vs. Programmatic Querying

In streaming CEP applications some querying mechanism must be used to find events of interest as well as to aggregate, analyze, and process them. There are two main approaches to querying stream data, using standard SQL or using programmatic coding directly, both having their advantages and disadvantages.

The ability to query event data using semi-standard SQL seems desirable at first as it provides a familiar view on streaming data using a high level query language. SQL in this case allows for a concise and pre-defined way of expressing various streaming data queries. However, traditional standard SQL does not provide sliding window capabilities and, hence, new SQL dialect needs to be created to support stream processing. Since there is no standard for such SQL dialect, each CEP product provides its own syntax which is usually quite extensive and may entail a lengthy learning curve. Additionally, these semi-standard SQL dialects are not extensible and users are limited to the set of features supported by a certain CEP vendor.

Using programmatic coding such as Java or Scala to query stream data may require additional work, but is generally more flexible as users are not limited by any proprietary syntax or pre-defined routines. Moreover, using Java or Scala directly usually renders dramatically better performance as there is no overhead associated with SQL parsing or processing, and users have much more fine-grained control over data indexing and data aggregation.

GridGain In-Memory Streaming uses programmatic coding utilizing Java or Scala with rich data indexing support to provide CEP querying capabilities over streaming data.

In-Memory Streaming Features And Key Concepts

What are the key features and concepts of GridGain In-Memory Streaming?

When events are streaming into a system at a very high rate, there are only a few ways to process them while sustaining the load without bottlenecking. All the features of GridGain In-Memory Streaming, from event workflow to indexing and data querying, are focused on minimizing network and disk access as well as providing instantaneous response times to queries.

Customizable Event Workflow
Even though majority of streaming events are often processed in one step there are many use cases when processing must be split into different stages and routed to different nodes. GridGain provides comprehensive support for customizable event workflow. As events come into the system they can go through different execution chains, supporting branching and joining of execution paths, with every stage possibly producing new types of events.

By allowing every stage to declare the next one or multiple stages for execution, GridGain In-Memory Streaming can support multiple execution paths for the same events executing in parallel on one or more nodes, supporting loops and recursive branching. The execution workflow ends when all branches finish.

At-Least-Once Guarantee
The at-least-once execution semantic provides a guarantee that as long as there is one node standing, the event workflow chain will run its course. GridGain fails over the execution chain as a whole. This means that if at any point a node responsible for some execution branch fails, the whole execution will be cancelled and restarted from the root.

Failing over and restarting workflow from the root is important as different execution stages may store partial results on individual nodes, and the aggregated result as a whole becomes invalid whenever some partial results are lost. Restarting from scratch guarantees that the final result will always be complete and consistent.

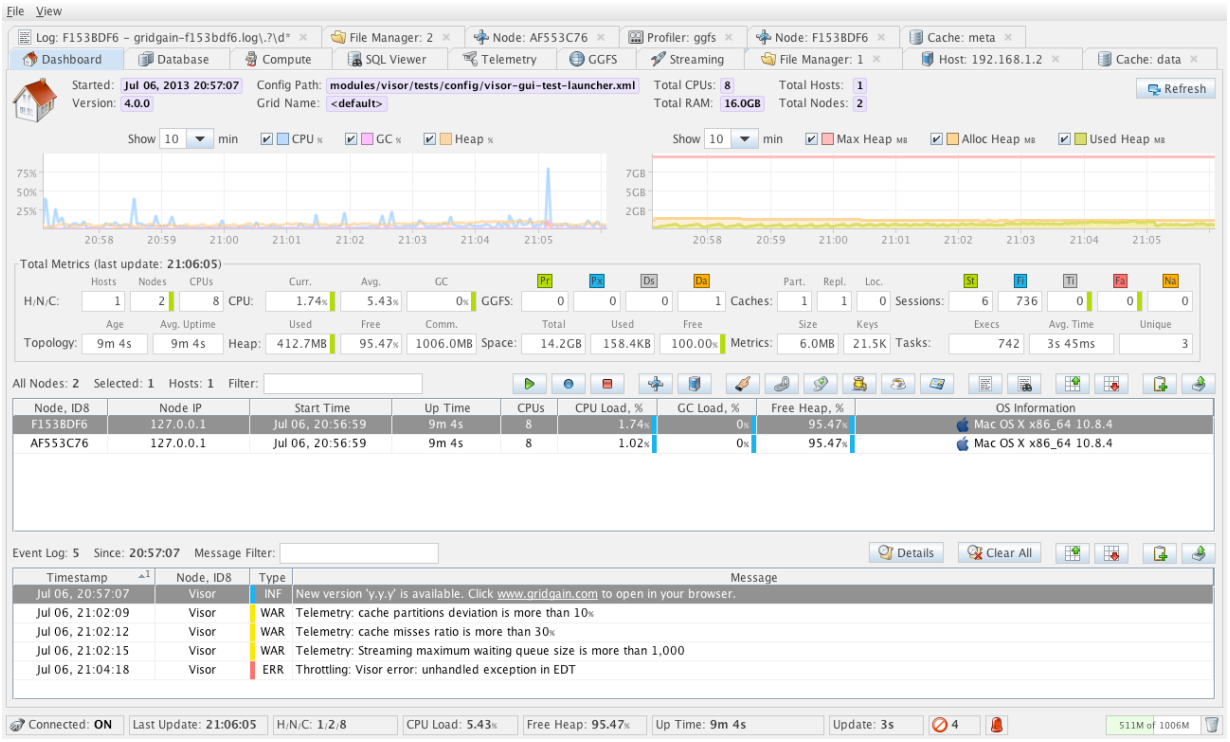
Sliding Windows
As streaming data is fairly constant, it is important to define the scope of streaming data operations by limiting the size of data being queried. Sliding window constructs supports exactly that. GridGain rich windowing functionality includes sliding windows that can be limited by size or time, windows that slide with either every individual event or in batches, windows that are unique or allow duplicates, and windows that can be sorted or snapshotted.

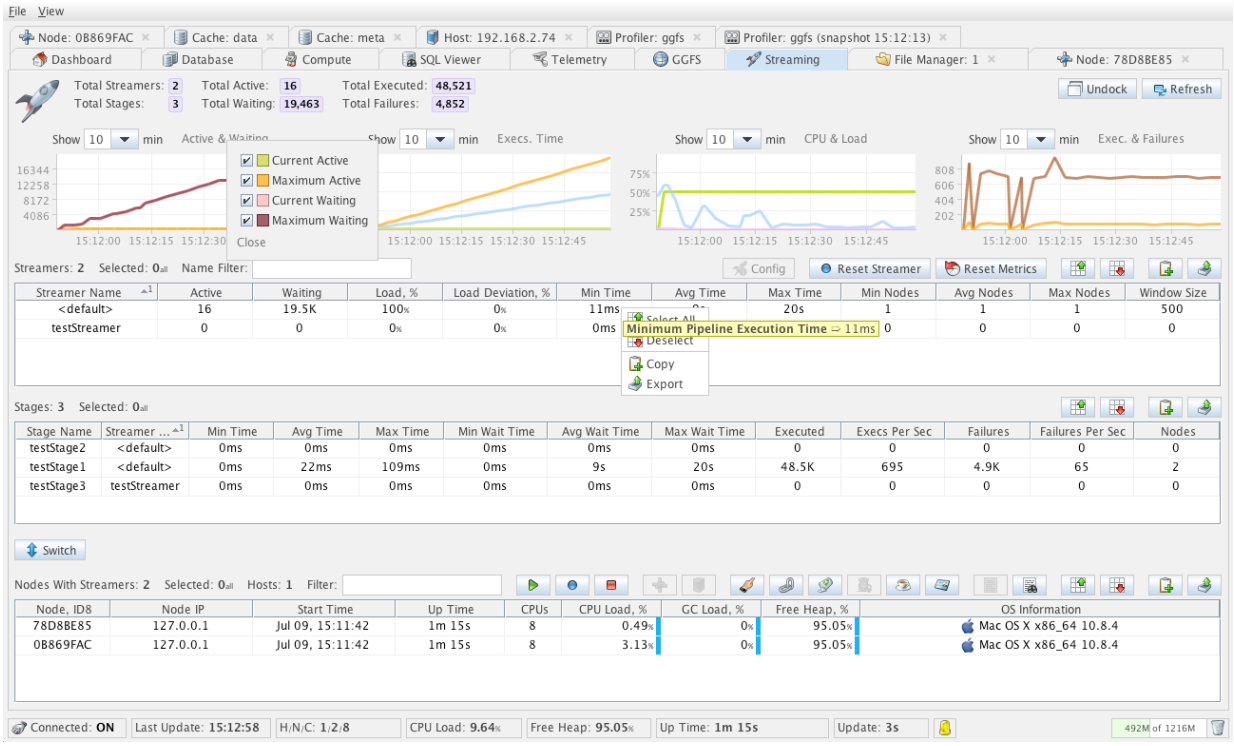
Processing of sliding windows usually involves reacting to new events or a batch of events entering the window, as well as reacting to events that leave. This essentially allows users to control any sort of time- or size-based metrics such as sliding averages, counts, sums, as well as instantaneous selection of any group of events by sorting them in any custom order. Different windows may coexist together and can optionally store and control different types or groupings of events.

By being able to query sliding windows programmatically, users are free to implement any custom algorithms on a streaming window of data.

Data Indexing
Quite often iterating through windows is not performant, especially when window sizes are large, and indexing into window data is necessary to achieve efficient query processing. GridGain provides comprehensive indexing APIs which allows you to create any type of index, as well as maintain any type of running aggregate metrics within indexes.

Every window can have as many indexes as needed and every index can be based on any arbitrary field or group of fields. Such flexibility allows users get immediate responses for a





variety of use cases and queries, from maintaining running averages to figuring out best selling products, or top co-selling product groups over a certain sliding period of time.

Distributed Streamer Queries

When processing streaming data it is important to keep any network communication or disk level access to the bare minimum. Otherwise there is always a possibility for some workflow stage to start bottlenecking and essentially bring the whole system to a full stop. To avoid such a scenario, GridGain took special care to make sure that no data or event distribution happens unless it is absolutely required.

All windows on every node accepting streaming events are local and are not copied over network for redundancy (if redundancy is required for certain events, then such events should be stored in in-memory data grid). Whenever a collective cluster-wide result needs to be computed, a streamer query is issued across all participating nodes which will gather results from all participating nodes, aggregate them, and return to user.

GridGain supports various types of streamer queries, allows for local and remote result aggregation, as well as support for visiting queries, which perform computations remotely on the nodes where the event data resides without sending anything back.

Co-location With In-Memory Data Grid

To preserve data integrity for mission-critical information and to provide a fault-tolerant highly available system, certain streaming events may need to be stored in an in-memory data grid. This way applications will avoid any disruptions in real-time processing, survive node crashes, and ensure that all event-related data will always remain intact and consistent.

However, to utilize data stored in an in-memory data grid while minimizing any data migration during stage execution, stages must be routed exactly to the nodes where the data is cached. To achieve this GridGain provides a special router which automatically co-locates stage execution with required in-memory data, based on affinity information provided from incoming events.

Moving stage execution logic directly to the data is a lot cheaper than moving data to the execution. Transferring data over network is one of the most expensive operations a distributed system can perform and should be avoided whenever possible. In fact, whenever a GridGain cluster topology is stable, there will be zero data movement during event processing. Instead, stages will be automatically routed to the appropriate nodes for execution.

Management

GridGain In-Memory Streaming, as with any other GridGain platform product, comes with a comprehensive and unified GUI-based management and monitoring tool called GridGain Visor. It provides deep operations, management and monitoring capabilities.

A starting point of the Visor management console is the Dashboard tab which provides an overview on grid topology, many relevant graphs and metrics, as well as event panel displaying all relevant grid events.

To manage and monitor configured streamers there is a Streaming tab which displays various metrics and routing information for streamer events and stages.

End-to-End Stack & Total Integration

What are different GridGain editions?

GridGain provides full end-to-end stack for in-memory computing: from high performance computing, streaming, and data grid to Hadoop accelerators, GridGain delivers a complete platform for low-latency, high performance computing for each and every category of payloads and data processing requirements. Total integration is further extended with a single unified management and monitoring console.

Platform Products
GridGain's platform products are designed to provide uncompromised performance by providing developers with a comprehensive set of APIs. Developed for the most demanding use cases, including sub-millisecond SLAs, platform products allow to programmatically fine-tune large and super-large topologies with hundreds to thousands of nodes.

In-Memory HPC	Highly scalable distributed framework for parallel High Performance Computing (HPC).
In-Memory Data Grid	Natively distributed, ACID transactional, SQL and MapReduce based, in-memory object key-value store.
In-Memory Streaming	Massively distributed CEP and Stream Processing system with workflow and windowing support.

GridGain Foundation Layer

What are the common components across all GridGain editions?

GridGain foundation layer is a set of components shared across all GridGain products and editions. It provides a common set of functionality available to the end user such clustering, high performance distributed messaging, zero-deployment, security, etc. These components server as an extensive foundation layer for all products designed by GridGain.

Hyper Clustering®
GridGain provides one of the most sophisticated clustering technologies on Java Virtual Machine (JVM) based on its Hyper Clustering® technology. The ability to connect and manage a heterogenous set of computing devices is at the core GridGain's distributed processing capabilities.

Clustering capabilities are fully exposed to the end user. The developers have full control with the following advanced features:

- Pluggable cluster topology management and various consistency strategies
- Pluggable automatic discovery on LAN, WAN, and AWS
- Pluggable “split-brain” cluster segmentation resolution
- Pluggable unicast, broadcast, and Actor-based cluster-wide message exchange
- Pluggable event storage
- Cluster-aware versioning
- Support for complex leader election algorithms
- On-demand and direct deployment
- Support for virtual clusters and node groupings

Zero Deployment®
The zero deployment feature means that you don’t have to deploy anything on the grid – all code together with resources gets deployed automatically. This feature is especially useful during development as it removes 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 will be automatically be deployed on all running grid nodes. Note that you can change existing code as well, in which case old code will be undeployed and new code will be deployed while maintaining proper versioning.

Advanced Security
GridGain security component provides two levels by which security is enforced: cluster topology and client connectivity. When cluster-level security is turned on, unauthenticated nodes are not allowed to join the cluster. When client security is turned on, remote clients will not be able to connect to the grid unless they have been authenticated.

SPI Architecture And PnP Extensibility
Service Provider Interface (SPI) architecture is at the core of every GridGain product. It allows GridGain to abstract various system level implementations from their common reusable interfaces. Essentially, instead of hard coding every decision about internal implementation of the product, GridGain instead exposes a set of interfaces that define the GridGain’s internal view on its various subsystem. Users then can use either provided built-in implementations or roll out their own when they need different functionality.

GridGain provides SPIs for 14 different 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 ability to change the implementation of each of these subsystems provides tremendous flexibility to how GridGain can be used in a real-world environment. Instead of demanding that other software should accommodate GridGain, GridGain software blends naturally in almost any environment and integrates easily with practically any host eco-system.

Remote Connectivity
GridGain products come with a number of Remote Client APIs that allow users to remotely connect to the GridGain cluster. Remote Clients come for multiple programming languages including Java, C++, REST and .NET C#. Among many features the Remote Clients provide a rich set of functionality that can be used without a client runtime being part of the GridGain cluster: run computational tasks, access clustering features, perform affinity-aware routing of tasks, or access in-memory data grid.

Summary

The case for in-memory computing is actively winning converts. Analyst firm Gartner says that in 2012, 10% of large and medium-sized organizations had adopted in-memory computing in some capacity. By 2015, that figure will have more than tripled to 35%.

GridGain's new In-Memory Accelerator For Hadoop product extends the performance value chain to Hadoop distributions while also significantly cutting an organization's storage costs. It's flexibility, scalability and plug-n-play architecture allow for seamless integration and improved velocity of analytics and reporting.

###