# GridGain and Azul Systems: The Industry's Highest Performance In-Memory Computing Platform for Real-Time Data Processing

The GridGain® in-memory computing platform accelerates and scales out data-intensive applications across a distributed, JVM-based computing architecture. GridGain solves the performance needs of companies launching digital transformation, omnichannel customer experience, Internet of Things, and similar data-intensive initiatives. GridGain is built on Apache Ignite®, which was originally contributed to the Apache Software Foundation (ASF) by GridGain Systems. Ignite is a top five ASF project and has been downloaded over four million times since the project launched in 2014.

GridGain can massively scale out to thousands of nodes and power millisecond performance for petabytes of in-memory and on-disk data. It supports multiple access APIs, including ANSI-99 SQL and key-value, and supports ACID transactions. GridGain is used for real-time transactions, hybrid transactional/analytical processing (HTAP), and high performance data integration hub use cases for companies in the financial services, telecommunications, software/SaaS, healthcare, transportation and logistics, and many additional industries.

The high performance GridGain platform is ideal for digital transformation use cases. However, some low latency/high transaction volume scenarios, such as in the financial services and telecommunications industries, strain the capabilities of standard Java Virtual Machines (JVMs). GridGain keeps all the in-memory data it caches in unbound, off-heap regions of memory. Java on-heap memory is utilized extensively by temporary objects generated by applications in runtime, though. Traditional JVMs can struggle to deliver the low latency required for certain transaction-intensive applications. In those specific use cases, GridGain users may need to scale out the number of nodes in their cluster in order to parallelize the workload.

The alternative to adding more parallelization to the cluster is Azul Zing®, a 100% Java-compatible JVM based on Oracle HotSpot. Unlike traditional JVMs, Zing decouples application performance from the amount of data kept in-memory in the Java heap. Zing is unique in its ability to provide high performance and low latency for memory-intensive applications. Zing is able to grow and shrink the memory heap elastically based on real-time application demands. The Azul C4 garbage collection algorithm is also able to limit JVM-related pauses to less than 30 milliseconds.

Running GridGain on Zing allows enterprises to increase the on-heap memory allocated on each GridGain node. High transaction applications generate significant Java garbage but it is cleared efficiently by Zing. As a result, GridGain users can avoid adding extra nodes to overcome the typical JVM garbage collection challenges while maintaining consistently low latency by avoiding JVM garbage collection pauses. This allows GridGain users with applications with high read/write requirements that need low latency with low jitter to achieve their SLAs with minimal infrastructure expenditures.

**Running GridGain on Zing allows enterprises to increase the on-heap memory allocated on each GridGain node.**

## Real-Time Business with GridGain and Azul Zing

In-memory computing (IMC) platforms can power very low latency, massively scalable applications for large datasets. However, the IMC platform must be able to hold and process large amounts of data in-memory with highly controlled and limited JVM-related garbage collection pauses. The GridGain platform running with Azul Zing delivers the necessary performance for very low latency transactional, HTAP and high-performance data integration hub applications.

## No Consistency or Durability Sacrifices

Low latency use cases can typically benefit from in-memory computing but often cannot accept data loss or inconsistency. In some cases, it may be impractical or unaffordable to keep an application's entire data set in RAM. Unlike other in-memory computing platforms, the memory-centric GridGain architecture can store the entire data set to disk while maintaining only a subset of the data in memory. The platform can process the data no matter where it resides, preferentially processing against the data in memory. This active disk-based storage tier approach allows users to trade off performance versus infrastructure costs. Combined with distributed ACID transactions, using GridGain with Azul is a highly-optimized computing platform that combines the speed of memory and the durability guarantees of disk-based systems with the strong consistency required by many high-value use cases.

### The Zing Approach

Zing uses the continuous concurrent compacting collector (C4). Unlike the other collectors, C4 is truly pauseless. Application threads run concurrently (and quite safely) with the object marking and object relocation necessary for garbage collection. In addition, Zing replaces the decades-old C2 JIT compiler with a modern and modular JIT called Falcon. Based on the open-source LLVM project, Zing can deliver even more optimized code that uses the latest processor features like AVX512 instructions.

### Benchmark Results

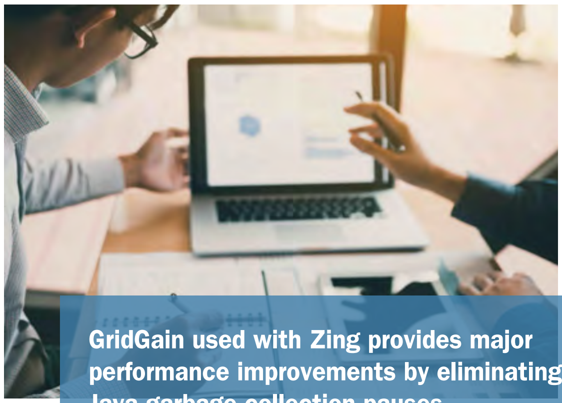
To benchmark GridGain running with Azul, a three-node GridGain cluster was connected to a Java application which performed reads and writes to the cluster. Three AWS i3en.6xlarge (3.1 GHz Intel Xeon Scalable Skylake processors) servers were used which had a total of 72 cores, 576 GB RAM, and 45 TB of disk
A number of scenarios were run to compare the performance of the standard Java G1 collector to C4 in Zing. The benchmark results were based on use of the transactional persistence capability in GridGain.

### The following credit card processing performance requirements are typical for banks:

- Each transaction accesses 20 records
- Distributed Transactional Reads
  - Target throughput – **1,000 reads/sec**
  - Target latency – **15ms for 99.99th percentile**
- Distributed Transactional Updates
  - Target throughput – **2,000 updates/sec**
  - Target latency – **50ms for 99.99th percentile**
  - RAM and disk must be updated for primary and backup copies

Tests were run for two hours each.

JVM latency was measured using jHiccup, a tool developed by Gil Tene, CTO of Azul Systems. jHiccup adds an extra thread to the JVM being monitored but the thread does not need to interact with the application code and spends most of its time asleep. The extra thread repeatedly sleeps for 1ms and records any difference between when it expected to wake up and when it actually wakes up. The nanosecond resolution of the system timer allows highly accurate readings of the observed difference between expected and actual wake up time.



GridGain used with Zing provides major performance improvements by eliminating Java garbage collection pauses.

For transactional reads, G1 consistently provides about 200ms of latency at the 99.99th percentile. Using Zing and C4, the system provides 6ms latency at the 99.99th percentile, with a maximum latency of 15.31ms.

For transactional writes, the latency target is 50ms at the 99.99th percentile in this case. G1 latency was slightly worse than with reads, consistently around 250ms (5x the goal) but with some spikes up to 479ms. With Zing and C4, the maximum at the 99.99th percentile was 25.1ms (half the maximum target) and the maximum latency was 33.87ms, never exceeding the target latency.

### There are several conclusions we can draw from this data for this use case:

- Zing solves the problem of GC pauses
- Zing combined with GridGain delivers the performance needed to support typical credit card processing performance parameters consistently and predictably
- To meet the goals of this use case with G1 is possible using GridGain but would require significantly more nodes in the cluster, increasing infrastructure costs versus using Zing

### Summary

For high transaction read/write applications with low latency requirements, GridGain used with Zing provides major performance improvements by eliminating Java garbage collection pauses. For GridGain users, this can significantly reduce potential infrastructure costs for high transaction volume/low latency use cases by reducing the number of required GridGain nodes to achieve adequate performance for demanding use cases.

Azul Systems, Inc.
385 Moffett Park Drive, Suite 115
Sunnyvale, CA 94089 USA
+1.650.230.6500
www.azul.com
info@azul.com