
# Consistent Cassandra Performance with Zing®

## Executive Summary

Apache Cassandra® is a NoSQL distributed database written in Java and designed for reliability and high scalability. Data nodes are spread across lots of commodity servers and even between datacenters. Information is kept in-memory for fast read and write response times, then “flushed” to disk when memory fills.

Because Cassandra is built in Java, it is subject to the limitations and performance hiccups associated with traditional Java Virtual Machines (JVMs). Common issues include response time outliers, frequent Java garbage collections, out of memory errors and system hangs. Companies deploying Cassandra spend extensive time tuning and tweaking the JVM to avoid these issues, then need to re-tune each time load (or the application accessing Cassandra) changes.

The Azul Zing® runtime allows Cassandra to run smoothly, free of Java-related performance glitches and hangs. Zing is an innovative JVM that removes garbage collection as an issue. Now your deployment can have consistent response times all the time, without GC pauses, node hangs or out of memory errors, regardless of load or object allocation rates.

Zing also provides several runtime optimizations in addition to the pauseless garbage collector. These runtime optimizations enable the application to continue to perform where traditional JVMs would not. For example, Zing has some unique features that enable code to warm up sooner and remain consistent longer.

## A Simplified Overview of Apache Cassandra

Cassandra is a NoSQL distributed database. It provides a highly scalable data store that is very reliable and has no single point of failure. Every Cassandra node is a peer to the others rather than a master/slave model. Deployments can consist of any number of nodes across many commodity servers and across datacenters, and new nodes can be added easily on-the-fly.

Cassandra is row oriented. Each row has a unique key and can have different columns than the rows around it. New data is written to a log, then to a Memtable in-memory. Once the Memtable is full the contents are flushed to disk. Each row is kept on multiple nodes, so if a node is no longer reachable the data is not lost.

Cassandra is a good choice for enormous amounts of data where high write and read throughput is required. Users include Netflix for movie streaming, Constant Contact for email marketing and RockYou to track clicks in their online games.

## Introducing Azul Zing

Zing is an innovative JVM that is fully compatible with the Java SE specification. Zing is optimized for Linux server deployments and designed for enterprise applications and workloads that require any combination of large memory, high transaction rates, low latency, consistent response times or high sustained throughput. It is also the only JVM that implements Azul's C4 garbage collector, a continuously concurrent collector that removes garbage collection as a consideration in Java deployments.

Now Java apps like Cassandra can scale to high levels of performance with unprecedented response time consistency and meet the most demanding SLAs – without application changes or even recompilation. Zing allows Cassandra to run smoothly while handling far more data and delivering more throughput than traditional JVMs. It is a game-changing innovation that opens up new business capabilities and opportunities not practical with other Java software platforms.

## How Zing Improves Cassandra Performance

Because Cassandra is written in Java, certain elements are very sensitive to garbage collection issues. If you're not familiar with garbage collection, the basic idea is that when memory allocated to the JVM fills up, garbage collection is triggered to free up space for more objects. (Read more in this white paper.) A "minor collection" usually takes very little time. However, in a traditional JVM, the system will eventually need to "stop-the-world", or pause the operation of the Cassandra node to perform a "full collection" that compacts the memory heap to free space.

Cassandra's internal systems misreport performance, which might cause you to miss issues that impact your application's response times. For example, you can put a process to sleep in Cassandra for a full 30 seconds without changing the results on the performance report. When testing Cassandra performance, we recommend using jHiccup or other tools to capture response time outliers that the internal measurement methods miss.

Azul Zing's innovative garbage collection algorithm solves many common Cassandra performance issues:

**Insensitive to Memtable thresholds and sizes**. Memtables store updates in memory until they can be pushed to disk. Data kept in a Memtable usually ends up in the "old generation" in the Java heap. Lots of other information is stored in the heap as well as the Memtable, so the Memtable size should be set low enough (the default is 1/3 of the overall heap size) so that it doesn't squeeze out row caches and other important data. Companies often spend a lot of time tuning Memtable sizes to avoid full gc.

Zing allows you to avoid tuning and get your deployment launched faster. Zing also provides flexible memory allocation to the JVM, so if the heap is filling fast more memory is made available to avoid an out of memory error or a slowdown in processing.

**Reduce out of memory errors**. The default JVM setting in Cassandra triggers a major collection when the heap is around 92 – 93% full. Because this threshold is so high, sometimes the heap fills before garbage collection can be triggered, causing an out of memory error.

Zing eliminates this issue by continuously and concurrently garbage collecting (that is, without stopping the node) and by providing 'elastic memory', so more memory is automatically made available to the JVM in real time.

**Solve memory fragmentation**. Attempts to tune away long garbage collection pauses on Cassandra nodes can lead to insufficient available space for new objects. As you tune away full collection to avoid response time issues, node memory can become too fragmented - the spaces between existing objects may be too small to accommodate the next object that needs to be stored, causing performance degradation over time and eventual out of memory errors. A full garbage collection with heap compaction, which in a traditional JVM stops the processing of the node, is required to free up this space.

Zing is the only JVM that is able to continuously and concurrently compact. It constantly reclaims memory fragments to make larger spaces available for new objects and never falls back to "stop-the-world" to compact the heap.

**Prevent long GC on a node from bringing down the cluster**. In Cassandra every row is stored on multiple nodes and every write is sent to all nodes that contain the row. If one node fails, it's ok; all read and write requests can still be handled. However, every node has a coordinator queue that will wait for all requests to finish – even if this is after the response has been given to the client. What this means is that a very slow node (for example, one experiencing a full GC) can cause the queue to fill up and make that node unable to respond to any requests. This can lead to the entire cluster becoming unresponsive.

Zing eliminates GC pauses on the nodes, preventing GC-caused cluster failures.

**Prevent GC thrashing.** If the JVM heap size is set too small or if residency in the heap is over 50%, Cassandra can spend most of its time doing garbage collection. Of course, the time needed to do

garbage collection is directly proportional to the size of the heap. If you're tuning to try to make full garbage collection pause times low, you can end up causing the system to garbage collect all the time.

Zing allows you to use a large heap without garbage collection pauses. It removes the tradeoff between holding lots of information in memory and keeping response times low.

**Support row caches**. The row cache in Cassandra allows information that has been moved to disk to be cached in memory for faster read response times. The memory consumption of a row cache is high, and with a traditional JVM an efficient row cache will trigger lots of full garbage collections that pause operation of the node.

Zing is designed for systems that benefit from large amounts of in-memory data. Zing allows the row cache to use lots of memory without any garbage collection penalties.

**Mitigating compaction pauses due to tombstones**. When data is deleted from disk, Cassandra creates a 'tombstone' to indicate its new status. The column continues to exist for a short period of time before compaction permanently deletes it. Cassandra compaction of these files can trigger a full Java garbage collection.

Zing never falls back to a "stop-the-world" full garbage collection. It constantly compacts the heap while the application is running and does not cause a stall due to the removal of tombstones in Cassandra.

## Cassandra Performance with Zing – Online Advertising

Online advertising platforms are very sensitive to response time issues. Ads are placed in real time bidding. When an ad request is received, the system has just a few milliseconds to respond with a targeted ad or another ad service may get the placement.

One online advertising company using Cassandra contacted Azul about improving response times and throughput on their cluster. In testing:

| | **Other JVM** | **Zing** |
|---|---|---|
| Throughput: requests/sec | 1,000 | 1,300 |
| Request response time | Baseline | 24 – 30% faster |

Zing also removed response time outliers and allowed the company to make better use of available server CPU and memory.

## Zing and DataStax

DataStax powers the apps that transform business for more than 200 customers, including startups and 15 of the Fortune 100. DataStax delivers a massively scalable, flexible and continuously available big data platform built on Apache Cassandra.

DataStax is an Azul Systems partner and recommends the Zing runtime to allow Cassandra to run smoothly, without the issues that can be caused by Java garbage collection.

## Find Out More

To learn more about Zing or to request a free trial copy to try with your Cassandra deployment, contact Azul Systems today.

Learn more about Zing: http://www.azulsystems.com/products/zing/whatisit
Request a free trial copy of Zing here: http://www.azulsystems.com/products/zing/trial

Azul Systems, Inc.
1173 Borregas Ave.
Sunnyvale, CA 94089
+1 650 230 6500
www.azul.com
info@azulsystems.com