What can Zing® do for my project?

AZUL SYSTEMS®

To many Java developers, the JVM you use is a given – you build your application on the Java environment that works with your app server or framework and OS. If the app runs into trouble, you look within your organization for a JVM specialist, re-tune the JVM, and hope for the best. That's how Java developers have been (mostly) successful for 20 years.

Nobody disputes the power and flexibility of Java; it is the enterprise standard worldwide for a reason. However, Java's memory management is arguably both the best and the worst thing about the language. On the plus side, Java developers can ignore a huge class of programming errors that keep users of other languages up nights, or that can show up months or even years later in supposedly clean production code. But there's a downside to the garbage collection (GC) technology that makes Java developers so productive: a lack of consistency and predictability in application performance, a problem that only gets worse as an application's memory space grows.

Taming the Memory Management Monster

Like most GC implementations, Java's memory management waits in the background as applications create new objects.

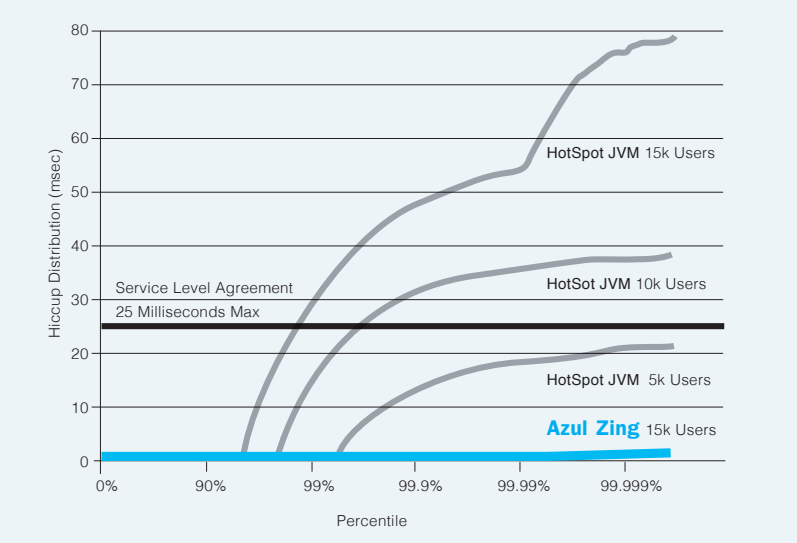
Eventually a threshold is reached and Java's GC leaps into action, identifying objects that are still in use and reclaiming space held by unused "dead" objects. Having cleared up lots of space, the GC waits in the background until it is needed again.

The performance problems come from the monolithic nature of some of Java's GC operations and the need to freeze application execution until those operations complete. The bigger the memory and the more objects that need to be scanned, the longer the application freezes. Even a moderate sized Java heap of a few gigabytes of memory can produce measurable delays due to garbage collection, on top of the application's normal processing time.

Application freezes due to GC are a fact of life for Java applications. Building time critical applications is that much harder when you have to worry about unpredictable delays, whether you're working with ecommerce transactions measured in small numbers of seconds or low latency processing in handfuls of microseconds. The best most Java developers can do is either trade off frequent small freezes against less regular but longer ones or try to push the freezes far enough into the future that they become somebody else's problem.

WHAT DO ZING INNOVATIONS MEAN FOR DEVELOPERS?

- Smooth, predictable, pauseless operation
- Fit all your critical data in memory at once
- No need for specialized frameworks to ensure predictable application behavior
- Faster launch times with reduced need for JVM tuning
- Minimize the need for 3rd-party APIs and off-heap data stores
- Zing is a JVM that ships as part of a complete JDK
- Zing is transparent to your application– you can even build using any JVM, then deploy on Zing
- Zing ReadyNow! technology solves Java warm-up problems for good, giving developers more control over Java compilation and allowing optimizations to be saved and reused across runs



Zing vs. Oracle's HotSpot: *Independent test results of Apache Lucene search transaction times shown by percentile at 200 queries/sec against an in-memory Wikipedia English-language index. Both JVMs were configured with 78 GB of index data in 140 GB Java heap.*

Zing: The Kinder, Gentler Memory Manager

Azul's Zing doesn't merely hide application freezes or reduce their occurrence; it eliminates them completely. Zing offers the only true concurrent memory manager, one that permits the GC and the application to run at the same time, not just most of the time but 100% of the time. Zing removes GC freezes and provides smooth, consistent application performance as Java's memory footprint grows from a few gigabytes of memory to terabytes. Transactional applications can count on predictable response. Low latency processing in Java becomes a practical reality. And applications can benefit from large in-memory data, without having to deal with the negative effects of managing all that extra memory.

Zing supports X86-64 systems all major Linux distributions, including Red Hat, SLES, Ubuntu, Debian, Amazon Linux, Oracle Linux and CentOS, including Docker and Linux containers and both VMware and KVM. For more on Zing: *www.azul.com/zing*

Find Out More

Solve Java Warm-Up problems with ReadyNow!

www.azul.com/readynow

Download Zing:

www.azul.com/zingtrial

Open Source Developer?

Request a free development copy by emailing us:

zing_oss@azul.com

AZUL SYSTEMS®

Monotype™