Zing®

The Java Supercharger

Java for the Real Time Business

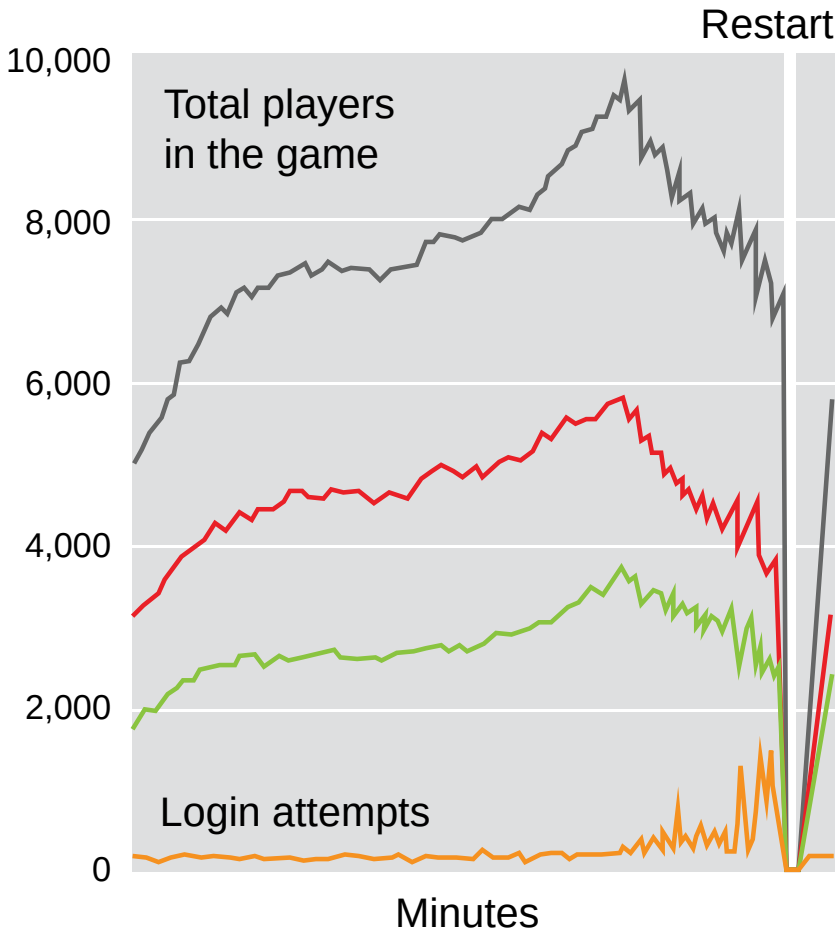
AZUL SYSTEMS®

From
the award-winning
leader in Java
runtime technology

Table of Contents

1

Java Performance Barriers



Java performance is great – until it isn't. This graph is all too familiar to Java shops. As usage grows, then suddenly processing stops or the JVM crashes. Sometimes this is due to out-of-memory errors or the JVM pausing to perform garbage collection.

Graph courtesy of Smart Bomb Interactive

Zing is a better JVM that delivers:

Consistent response times, even under load

Java SE compliance

Efficient hardware and human capital utilization

Faster time to market for new features and applications

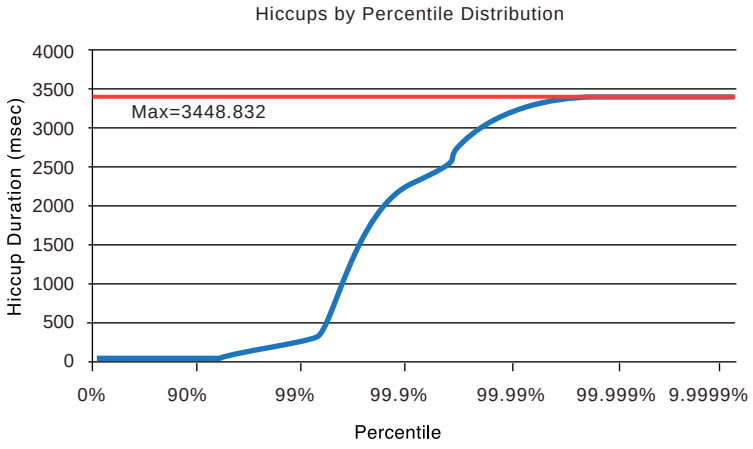
New avenues for business innovation and additional revenue

ZING INCORPORATES AZUL'S UNIQUE C4 "PAUSELESS" GARBAGE COLLECTION ALGORITHM

Measuring Java Performance

An average response time of 0.8 msec is good, right?

Not if the maximum response time violates your SLA, frustrates your customers or causes you to miss revenue opportunities.

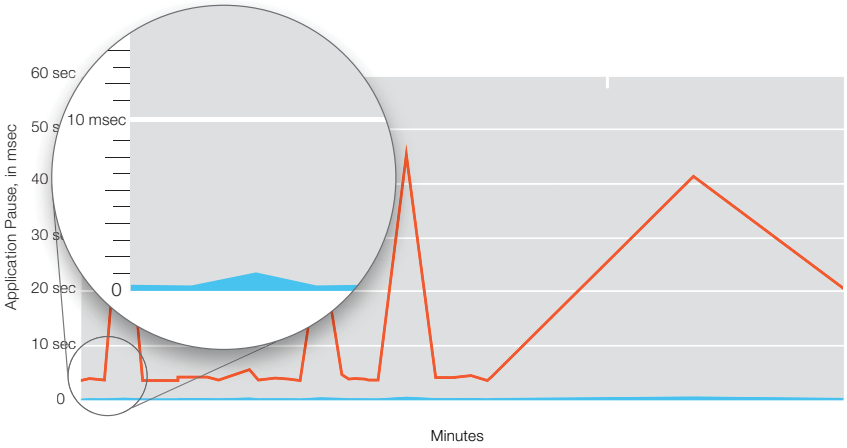


How you measure and report the performance of your systems is critical. Response time is usually NOT a normal distribution. You need to understand both the average and maximums. Azul's jHiccup tool was designed to provide an accurate view of a production application.

Read more about how NOT to measure latency here.

4

Real World Examples



Example 1:

Supercharging Search

This example is from NetDocuments, one of the first SaaS companies. Their application is built around search. In testing on Apache Solr™, their current JVM experienced pauses of almost 50 seconds.

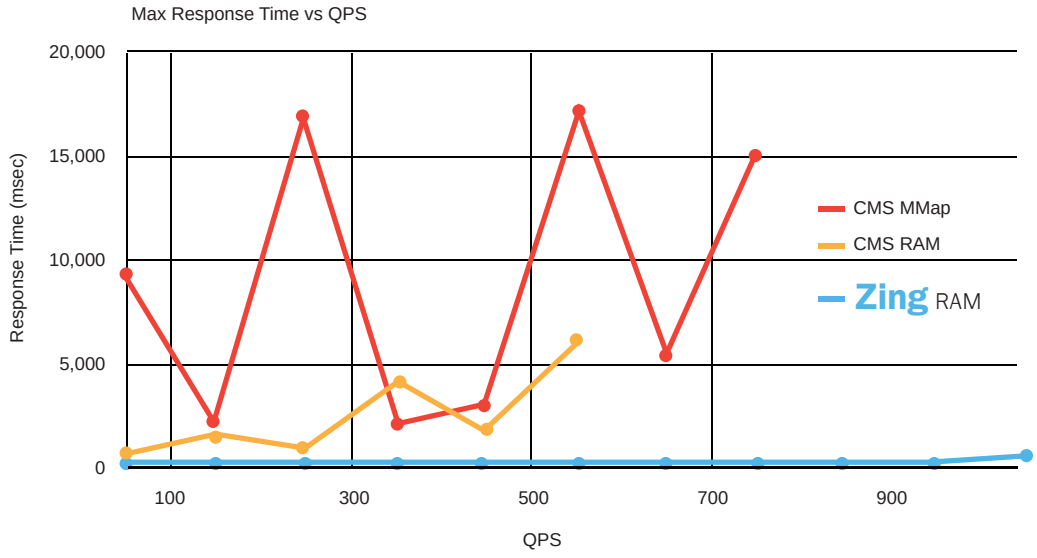
Running on Zing, the max pause was under 10 milliseconds.

> Without Zing we would not have been able to deploy Apache Solr for our production system. Our customers could have experienced long pauses when searching for critical documents.
>
> – Mou Nandi, Search Engineer and Architect NetDocuments

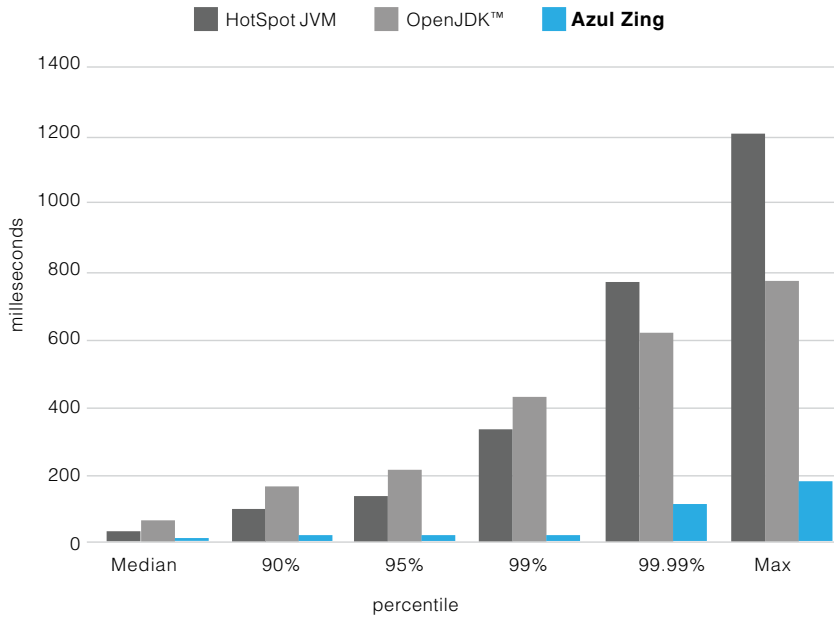
> ...I remain impressed with Zing, and I wish its C4 collector were the default for Java! Then we all would stop having any GC worries and could freely use Java with very large heaps, fundamentally changing how we build software in this age of very cheap RAM
>
> – Mike McCandless
> Apache Lucene committer
> and PMC member

Zing: The Java Supercharger

Mike McCandless, Apache Lucene committer and PMC member tested Lucene on Zing. His results clearly show the response time advantages of using Zing with Lucene deployed with a large RAMDirectory:



READ MIKE'S BLOG POSTS



This example is from a digital and web-based services provider that did a side-by-side comparison of Apache Solr on three different JVMs.

Using Zing they were able to lower response times 60% and eliminate outliers.

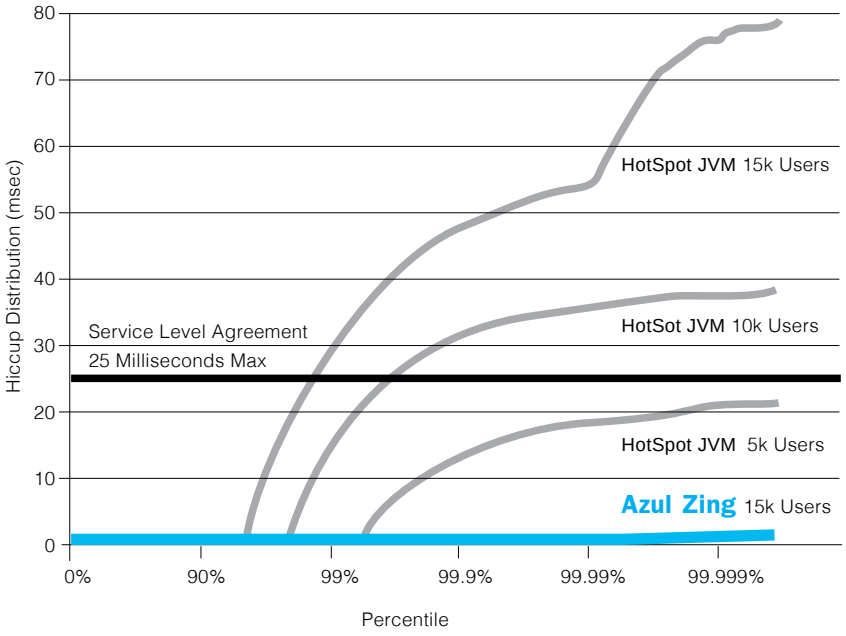
Zing Supercharges Search

In-memory indices of 60 GB and more

2-3X more throughput on the same hardware

Seamless service and consistent response times

New opportunities for innovation



Example 2

Low Latency Applications

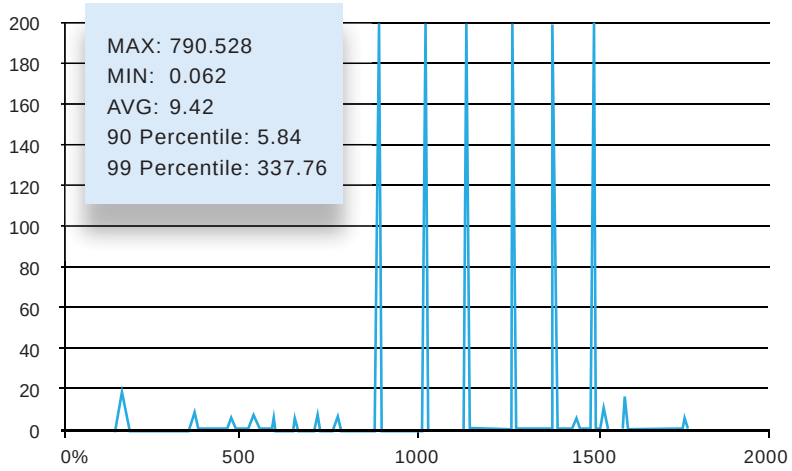
Low latency can refer to machine-level (microseconds) or user interactive levels (milliseconds). Both types of applications benefit from Zing's ability to deliver consistent response.

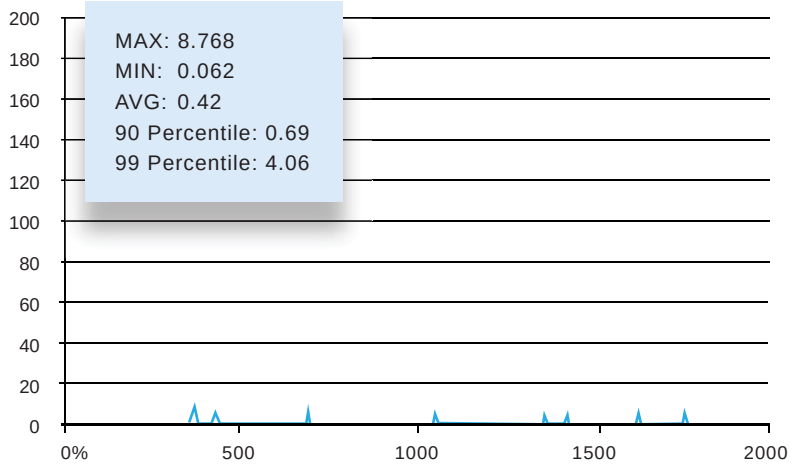
This machine-level benchmark from Push Technology highlights the advantages gained from deploying Zing.

Zing delivered 15X more throughput than Oracle's HotSpot and eliminated long response time outliers.

> ...by running Diffusion™ on Zing, application pauses in latency critical deployments can be effectively removed.
>
> – Push Technology Benchmark Report August 2013

DOWNLOAD THE BENCHMARK REPORT





This benchmark from a financial industry consulting firm compared performance of an algo trading engine running under high load stress on Oracle's HotSpot vs. Zing.

Oracle's HotSpot
Max response time

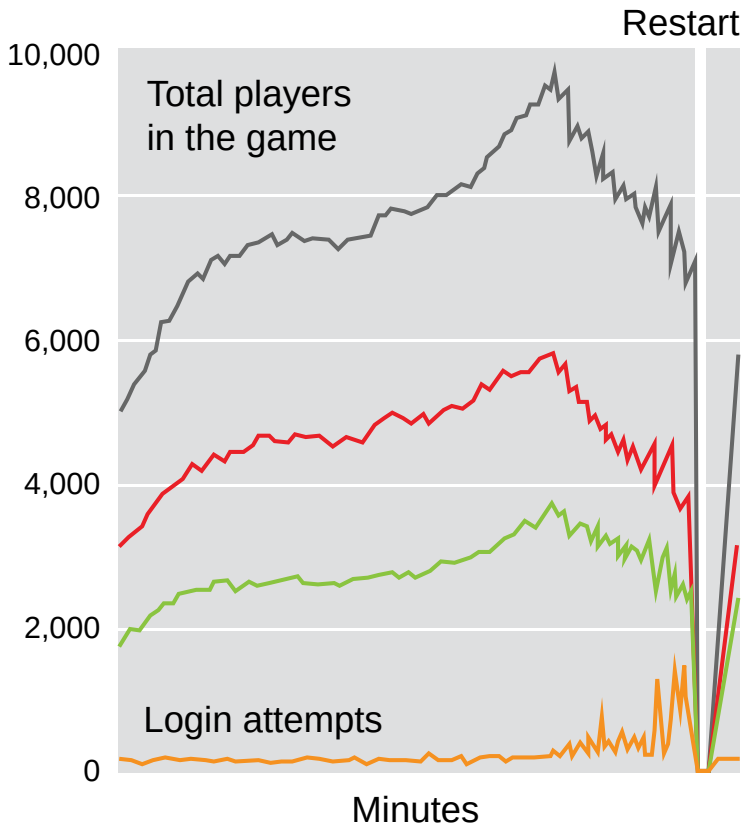
790
msec

Zing
Max response time

8.768
msec

> Under high stress, Zing shows much lower application latency in comparison to HotSpot.
>
> – Consulting Report

READ THE WHITE PAPER JAVA WITHOUT THE JITTER



Example 3

User Interactive Applications

User interactive applications also need low latency, just on a different scale. Users are demanding and will only wait a couple seconds before clicking away. Remember this graph?

It's a recipe for user frustration.

Zing eliminated pauses, stalls and restarts, and allowed Smart Bomb Interactive to grow from 8,000 to 20,000 social gaming users per server– 2.5X more capacity on the same hardware– just by implementing Zing.

> Azul Zing removed an unacceptable limit on the scalability of our online gaming experience. We'll use Zing everywhere we need to avoid garbage collection issues.
>
> – Jeff Amis, VP Product Development
> Smart Bomb Interactive

Vocalabs, which provides SaaS-based customer feedback programs, provides this example. Their report generation system continually receives new data. As the number of users grew, this led to unacceptably long GC pauses.

Performance Issues

GC Pauses up to 2 minutes long

Reports couldn't be rebuilt often enough as new data was added

Adding report filters was too slow

Developers spending too much time on performance tuning and not enough on features

Solution – Deploy on Zing

Eliminated GC Pauses

Reports can be rebuilt more often to incorporate the latest data

Adding report filters is nearly instantaneous

Developers have more time to spend adding value for customers

Scale Up Not Out!

This example is from a Global telecommunications company:

Long pauses limited their existing JVM to 4 GB heaps

This wireless operator replaced 16 JVM instances with a single 98 GB Zing instance

Better SLA compliance (response times consistently under 2 msec)

Able to consolidate their online store processing onto a single server from 16, freeing up resources for other initiatives

This example is from SuccessFactors, a major SaaS provider:

Could use only a few GB of heap/instance before pauses became intolerable

Under load the environment became unstable and repeatedly crashed due to out-of-memory errors

The company overprovisioned the systems with memory, but this led to low utilization rates outside of peak periods

Increased throughput 4X and user capacity/instance by 2.4X

Simplified the infrastructure 3X improved reliability

> With Zing, we can scale up with software.
>
> – CIO, SuccessFactors

> With Zing, we get 2 ½ times the number of users on the same hardware–without crashing.
>
> – CIO, Social Gaming Company

Tuning? Nah

For most JVMs, tuning parameters look like this

Developers and IT staff can spend hours, days, and weeks tuning and retuning their JVM flags each time the application changes. This is time that would be better spent adding new capabilities for your business. (And your developers will be more productive, too!)

```
Java -Xmx12g -XX:MaxPermSize=64M -XX:PermSize=32M-XX:MaxNewSize=2g
     -XX:NewSize=1g -XX:SurvivorRatio=128 -XX:+UseParNewGC
     -XX:+UseConcMarkSweepGC -XX:MaxTenuringThreshold=0
     -XX:CMSInitiatingOccupancyFraction=60 -XX:+CMSParallelRemarkEnabled
     -XX:+UseCMSInitatingOccupancyOnly -XX:ParallelGCThreads=12
     -XX:LargePageSizeInBytes=256m ...
```

```
Java -Xms8g -Xmx8g -Xmn2g -XX:PermSize=64M -XX:MaxPermSize=256M
     -XX:-OmitStackTraceInFastThrow -XX:SurvivorRatio=2 -XX:-UseAdaptiveSizePolicy
     -XX:+UseConcMarkSweepGC -XX:+CMSConcurrentMTEnabled
     -XX:+CMSParallelRemarkEnabled -XX:+CMSParallelSurvivorRemarkEnabled
     -XX:+CMSMaxAbortablePrecleanTime=10000 -XX:+UseCMSInitiatingOccupancyOnly
     -XX:CMSInitiatingOccupancyFraction=63 -XX:+UseParNewGC -Xnoclassgc ...
```

For Zing, tuning parameters look like this

That's it. Just set the memory high and go.

Now your developers can spend their time doing what they like to do – adding features and building new capabilities.

And with Zing, there's no need to re-tune your JVM when your app changes.

```
Java -Xmx40g
```

6 Dramatically improve your ROI

Eliminate most JVM tuning

- ✓ Days, even weeks of lost developer hours each time the application is modified

Capture lost revenue opportunities

- ✓ Retail/eCommerce: increase shopping cart success rates 3% or more
- ✓ Insurance: increase success rates for online quoting
- ✓ Real time advertising: more successful bids for increased click-through and revenue
- ✓ HFT, algo trading, Forex: never miss a trade due to platform stalls

Increase infrastructure efficiency

- ✓ Handle 2 - 3X more users or transactions on existing hardware
- ✓ Get 40% more utilization from your current servers
- ✓ Delay or cancel new hardware purchases

Pursue new opportunities

- ✓ Launch Cloud or SaaS services not practical with other JVM technology
- ✓ Deploy new algos faster to create competitive advantage
- ✓ Add memory-intensive eCommerce features that increase conversion rates and order size
- ✓ Manage risk of new initiatives better using more comprehensive real time information

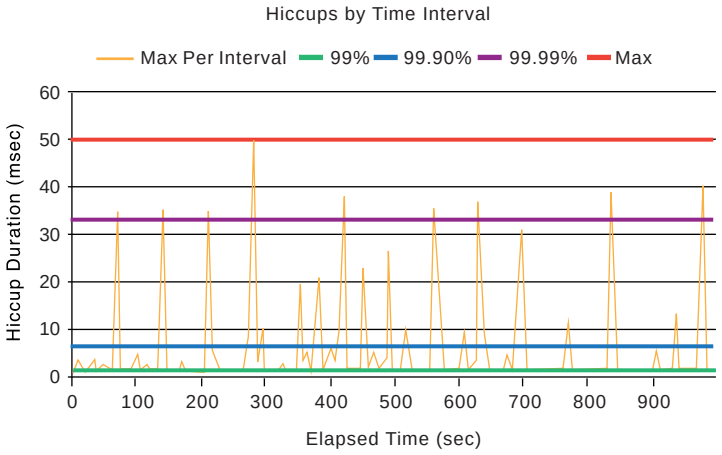
DOWNLOAD THE ROI WHITE PAPER

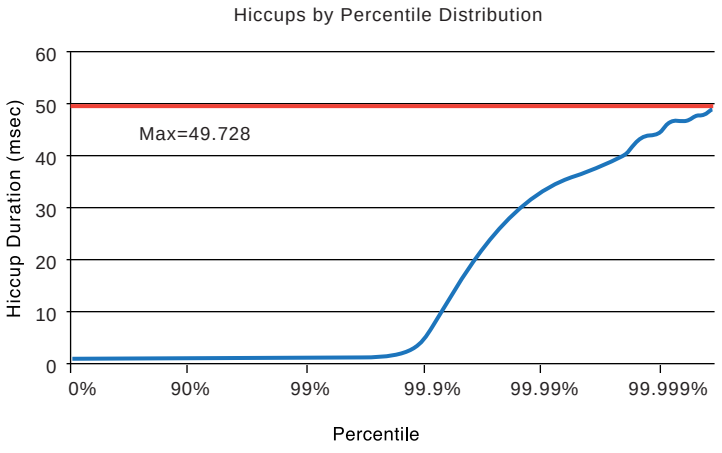
Getting Started

1. First, see what your application performance looks like now.

2. Download and run jHiccup, Azul's open source Java performance measurement tool. It will create graphs like these:

Read Gil Tene's Blog Post: How Java Got the Hiccups





DOWNLOAD JHICCUP (FREE)
AND READ MORE ON HOW TO USE IT HERE

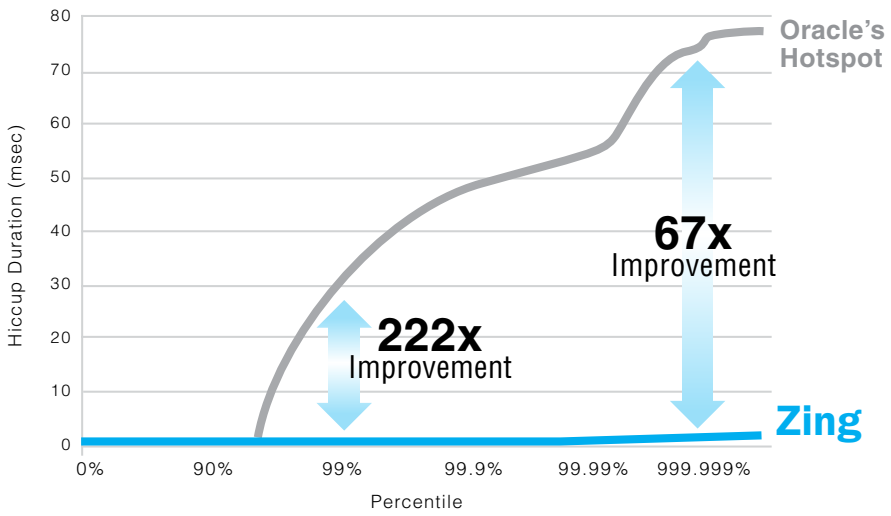
3 Next, get Zing.

Download and run Azul Inspector to gather information on your current Java deployment.

Request a free trial copy of Zing.

Download and install your copy (takes about 5 minutes).

Work with your assigned Azul Engineer to optimize and tune your app and environment to achieve the best results.



Summary

8

Zing – The Java Supercharger

Zing makes all your Java applications run better – with consistent and reliable performance – to better support the needs of your business

Eliminates pauses, stalls and jitter

Supports heaps over 300 GB with no performance penalty

Reduces tuning to just a few parameters

Opens up new avenues for business innovation

Frees Developers to spend more time adding value for customers

Learn More

www.azulsystems.com/zing

Transform all your applications for real time business.

CONTACT US