Zing® ReadyNow!™:
The Only Solution for Java "Warm-Up" and De-optimization Issues

ZING'S READYNOW! FEATURES

- Designed for latency sensitive Java-based systems
- Allows Java applications to start up fast and stay fast
- Provides operations teams with the ability to save and reuse code optimization profiles across runs
- Reduces de-optimization risk at market open and startup or when conditions change during the day
- Gives developers fine-grained control over Java compilation
- Greatly improves application performance consistency
- Included as part of Zing, Azul's innovative, "pauseless" JVM

WHAT'S THE JAVA WARM-UP PROBLEM?

Until today, Java developers and operations teams working with low-latency applications have faced a daily challenge – overcoming Java's warm-up issue. Operations teams weren't able to save compiler optimizations across runs, and developers weren't able to tell the compiler ahead of time which code to optimize. So companies developed elaborate Java warm-up exercises that if not done just right could mean that systems would stall just when you need peak performance.

Warm-up is a problem for every Java runtime except one – Zing. We've taken the folklore and guess-work out of low-latency Java, and given you control over how Java behaves when it matters most – at market open, when a retailer launches a flash sale or a new ad goes viral.

WHAT CAUSES JAVA WARM-UP ISSUES?

Java was designed to start up quickly, then improve performance over time by repeatedly compiling and optimizing frequently used code. The JVM's just-in-time (JIT) compilers use constantly-updated profile data that describes which parts of the application are called the most (the "hot" code). JIT compilation allows the JVM to optimize performance, but it takes time, and every time conditions change the JVM falls back to interpreted code ("de-optimization") until it identifies which methods (and even which code branches) are now most frequently used.

In use cases like capital markets, operations teams can't save JVM compiler optimizations from the day before. So they "warm up" their systems prior to market open, in order to ensure that systems are ready to go and fully optimized when the opening bell rings. This process works great – until it doesn't.

If conditions are different on a particular trading day – warm up attempts may not work anyway – the JVM de-optimizes and slows performance to a crawl until key methods are recompiled and re-optimized.

Today there is only one solution to manage this warm-up problem: ReadyNow! technology from Azul Systems – a key feature of the Zing runtime for Java.

INTRODUCING ZING WITH READYNOW! TECHNOLOGY

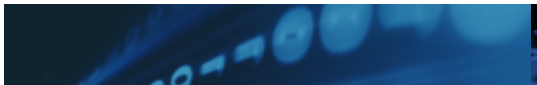
Designed for low latency systems and all workloads that must start and stay fast in spite of volatile traffic flows, ReadyNow! technology makes Java applications optimized and market-ready from the opening bell – and they stay fast.

ReadyNow! technology is built into the Zing runtime for Java. It allows systems to achieve optimum performance and consistency at the start of the trading day. ReadyNow! technology gives operations teams and DevOps the ability to save and reuse accumulated compiler optimizations across runs. For developers, ReadyNow! also provides a set of robust APIs and compiler directives that provide more control over Java compilation.

With ReadyNow! warm-up issues can be a thing of the past. Just save the compiler profile from the day before, or load one from a prior day expected to have similar conditions. Save optimizations across runs or any time you reboot to avoid having to warm up the code. Developers can use ReadyNow!'s robust APIs to direct the compiler to pre-compile code they know will be commonly used, or code that must be fast even though it's called infrequently.

Common warm-up techniques may sometimes optimize for the wrong conditions leading to de-optimization slowdowns that can affect your business results. Zing's ReadyNow! technology minimizes this risk and puts you in control. Specify how the compiler should optimize, and when the trading day is over you can save the compiled profile and use it again – the next day, or at any time you choose.

Zing with ReadyNow! technology is in use in Java-based trading, online retail, telecom and online advertising platforms worldwide.



Zing ReadyNow! Features

- Improves the effectiveness of warm-up strategies and reduces de-optimization risk at market open and at other key points throughout the trading day
- Delivers more consistent low latency performance
- Allows operations teams and DevOps to use compiled code profiles across runs
- Provides developers with APIs that control compilation policy, reduce de-optimization, and direct aggressive initialization tactics that minimize the need to warm-up the JVM
- Optimized for 64-bit Linux on x86
- Works with Zing's Azul-optimized JIT compilers

SUPPORTED PLATFORMS

Processor

- Intel: Xeon server class processors released 2009 and later
- AMD: Opteron server class processors released 2010 and later

Memory and CPU Cores Recommended

- 1 GB to 2 TB heap
- 2 cores or more

Supported Operating Systems

- 64 bit Linux (x86)
- Red Hat Enterprise Linux/CentOS 5.2 or later
- Red Hat Enterprise Linux/CentOS 6.0 or later
- Red Hat Enterprise Linux/CentOS 7.0 or later
- Red Hat Enterprise MRG Realtime 2.3 or later
- SUSE Linux Enterprise Server 11 sp1, sp2, sp3
- Oracle Linux 5.x or 6.x (kernel specific)
- Ubuntu 10.04 LTS (Lucid Lynx)
- Ubuntu 12.04 LTS (Precise Pangolin)
- Ubuntu 14.04 LTS (Trusty Tahr)
- Debian Wheezy
- Amazon Linux
- Other Linux distributions supported via Dynamic Kernel Module System

JDK Versions

- Java 8, 7 and 6

Selected Zing ReadyNow! use cases

- Low-latency financial systems
- In-memory Big Data analytics
- Online Retail
- Stream processing
- Large portals and other web-scale applications
- Real-time advertising networks
- Large-scale online and social gaming
- SLA-driven complex event processing
- Real-time messaging

Contact Azul Systems:

385 Moffett Park Drive
Suite 115
Sunnyvale, CA
94089 USA

T + 1.650.230.6500
F + 1.650.230.6600

www.azul.com/solutions/zing/readynow

Monotype