JAVA PROBLEMS IN THE CLOUD: THE NEED FOR MULTITENANT JAVA

White Paper

ID: 12062000
June 2012

COPYRIGHT

June 2012

RESTRICTED RIGHTS LEGEND

Information in this document is subject to change without notice and does not represent a commitment on the part of Waratek Limited.

TRADEMARKS

Waratek and Replicode are registered trademarks of Waratek Limited.

Java, JVM, Hotspot, JRocket are registered trademarks of Oracle

All other company and product names may be the subject of intellectual property rights reserved by third parties.

TABLE OF CONTENTS

LIST OF FIGURES

JAVA PROBLEMS IN THE CLOUD: THE NEED FOR MULTITENANT JAVA

INTRODUCTION: CLOUD COMPUTING REQUIRES TECHNOLOGY DESIGNED FOR THE CLOUD

In a world of 24/7 on-demand online access there is growing pressure on business managers to ensure that their business applications are always-on and always-available at acceptable levels of responsiveness. Meeting this demand comes with a price: if the service performs poorly or proves unreliable, every business application or IT service poses a risk to business, brand and reputation.

In response to these challenges, two high-growth technologies have emerged: virtualization and cloud computing. Virtualization, as a percentage of x86 architecture workloads running in virtual machines, has risen dramatically from 2% in 2005 to 56% in 2012, and is expected to reach 86% by 2018.

Cloud computing is the sharing of resources to deliver computing and storage capacity as a service. Cloud computing offers dramatic breakthroughs in the cost and elasticity of computing operations through features such as high application density, multitenancy, accurate and real-time metering of usage, and fine-grained elasticity with instant scalability.

Cloud computing allows businesses to get their applications up and running faster, while at the same time offering improved manageability and reduced maintenance. It also enables IT to rapidly adjust resources to meet fluctuating business demand, thus reducing or eliminating the challenge of predicting demand. Cloud computing is changing the entire IT industry as we know it today and we need to be able to respond to this change.

Developed in the 1990s, Java is the single most important enterprise and e-Commerce language used in corporate computing. Java is used by 80% of top firms worldwide. If Java were a region, it would have the same GDP as the State of New York, bigger larger GDP than countries such as The Netherlands, Switzerland or Saudi Arabia. Many of the latest innovations (Spring Source, JRuby, Scala and Groovy) are derived from Java.

According to a Forrester report, it may take a decade for a competitor to Java to emerge. (Source: The Future Of Java – A Social Computing Report: Java Will Be Constrained By The Bounds Of Oracle's Business Model, by John R Rymer and Jeffrey S Hammond, 2010.)

Java is already under threat from the rise of cloud computing. Java was not designed for applications to run in the cloud, which may encourage the emergence of new languages. Ultimately, this could involve the need to rewrite $US 2 trillion worth of legacy Java computer applications.

It is estimated that nearly half of all applications deployed on virtualized infrastructure today are Java applications. With Java applications comprising such a significant percentage of the installed base of applications running on virtualized infrastructure, the inefficiencies of hosting Java applications in this manner are becoming increasingly obvious. Any comprehensive solution for hosting Java applications on cloud computing infrastructure currently available,

involves learning new languages, adopting proprietary frameworks, or operating under restrictions and limitations that will inevitably result in the inefficient use of resources.

All this has driven the demand for a virtualization solution that lowers the complexities and inefficiencies of developing and hosting Java and Java-derived applications on cloud computing infrastructure.

In describing their Reference Architecture for Multitenancy, Gartner (June, 2012) points out that computing in the cloud will ultimately require specialist technologies and architectures that are designed natively for cloud platform environments. An efficient Java cloud platform needs to offer effective multitenancy, use tracking, massive scalability (including horizontal scaling), accessible productivity tools and self-service management. All this will have to be delivered natively, i.e. supported by a particular system with minimal computational overhead and additional components. Currently, Java is only able to provide limited support for these advanced cloud features.

THE CENTRAL ROLE OF THE JAVA VIRTUAL MACHINES

The most critical element in the operation of the Java platform is the Java Virtual Machine (JVM). A JVM is a specifically designed virtual machine capable of executing Java bytecode, and is the code execution component of the Java software platform. The major challenge facing Java and existing JVMs is that Java does not support multitenant virtualized and cloud computing configurations natively. Consequently, existing JVMs are not able to provide support for developing cloud-hosted and cluster-hosted applications. To date, Java middleware platform software has relegated the JVM to merely a hardware abstraction layer role. Yet ultimately it is the JVM that determines the

runtime application performance, and it is the JVM that also affects the application's behavior in real time.

As application requirements become more numerous and/or more complex, efficient operation of cloud-hosted Java applications poses ever more complex challenges. Traditionally, Java applications have been operated like any other application-type when deployed on virtual server infrastructure. Until now, virtualized hosting models for Java applications have provisioned an entire x86 virtual machine to host a single JVM instance, which in turn has hosted the single desired Java application.

There are several factors behind Java's inefficiency when hosted in this traditional manner:

- Because the Java language was originally designed for one application per JVM, multiple mutually-distrusting independent applications cannot be safely or securely hosted within a single JVM. For example, various Java language resources in java.lang.System and java.lang.Runtime classes are singleton facilities intended for exclusive use by a single application.

- A JVM is a large application that seeks to consume most of the memory and computational capacity available in a hosting machine - which is why the adjective "greedy" is sometimes used to describe this phenomenon. For example, most JVMs automatically scale their garbage-collector task facilities to occupy the number of processors available on the underlying hardware. As a result, when starting multiple simultaneous JVMs on the same operating system, each JVM competes with other JVMs for the available computational resources of the underlying server.

- Many applications today are service-oriented. This is very different from the traditional monolithic applications in earlier years. These service applications are ideally suited for provisioning in smaller compute and memory increments than those provided for by existing x86 virtualization solutions. This means that deploying multiple service applications within separate JVM instances operating within separate x86 virtual machines can rapidly lead to underutilized server capacity, while presenting problems of demand and resource management. A similar problem provided the impetus for server virtualization and consolidation initiatives in the late 1990s.

THE NEED FOR MULTITENANT JAVA

Multitenancy is fundamental to Platform-as-a-Service (PaaS), a platform hosted by a provider and delivered as a service to customers. Multitenancy is defined as facilitating deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers. (Source: Multitenancy and Model Driven Engineering, necessary assets of a Platform-as-a-Service, Johan den Haan, 2010.)

Multitenancy is a sharing of common application computing resources among multiple tenants, and is a distinct characteristic of cloud computing. As companies seek to achieve multitenancy in today's cloud platform environments, some providers are looking for expedient shortcuts. Other providers are looking to invest in discontinuous change, a concept introduced by Charles Handy to describe change that arrives in sudden, unpredictable bursts that threaten existing or traditional structures and systems.

In the short term (3-5 years), existing models of technology and architecture will help the IT industry make the transition to cloud computing, although some of these models may not endure. In the long term, cloud computing will require dedicated technologies and architectures.

To deliver a comprehensive multitenant Java solution for cloud computing, the Java platform, and critically the JVM, must be further enhanced to deliver on the characteristics (some listed below by way of example) that Gartner (June 2012) identify as essential requirements for genuine multitenant Java:

- Isolation of tenant data and tenant work space (memory)
- Isolation of tenant execution characteristics (performance, availability)
- Tenant-aware security, management, reporting
- Isolation of tenant customizations and extensions to business logic
- Fine-tuned tracking and recording of resource use
- Tenant-aware application version control and error-tracking/recovery
- Dynamic allocation of resources to tenants as required
- Horizontal scalability to support real-time addition of new tenants
- Redundancy to support transparent hot migration in the event of resource failure

While multitenancy through x86 server virtualization has become a feature of mainstream cloud and enterprise computing environments, Java applications have not enjoyed the same availability of mainstream virtualization and multitenancy solutions. This is because existing Java application servers are simply unable to deliver multitenancy support in their enabling platform technologies to cloud-based applications.

The emergence of the major architectural approaches to supporting multiple tenants in today's evolving cloud computing environments is characterized by:

(a) Shared-nothing approach: focuses on providing isolated tenancy requirements in which application multitenancy is custom-coded into the business application;

(b) Shared-hardware approach: focuses on providing tenants with their own dedicated stack of technology through virtual machines allocated from a pool of of shared hardware.

(c) Shared-everything approach: focuses on providing the full application execution platform stack to multiple tenants.

The advantage of the shared-everything approach is that it allows multitenant services to be offered natively, whereby the application and the platform technology are designed and implemented for a specific cloud-computing context and cloud-user requirements. By offering greater flexibility in provisioning, customizing and optimizing applications, native multitenancy meets the demands of on-demand computing more effectively and efficiently.

Since the shared-everything approach demands a discontinuity in application programming models, platforms and skills, it has become the preferred choice for new workloads designed from the ground up for cloud contexts. To avoid creating their own cloud-enabling multitenancy support (shared-nothing custom multitenancy), independent software vendors (ISVs) are seeking commercial cloud-enabled application platforms offered as a product (PaaS) or as a service (IaaS). Existing Java application providers and users cannot access the benefits of this shared-everything approach when they use current JVMs and Java platform software.

HOSTING MULTIPLE JAVA APPLICATIONS WITHIN A SINGLE JVM

The need to increase hosting efficiency and deployment density for Java applications hosted on virtualized infrastructure has driven the search for solutions to hosting multiple Java applications within a single JVM. Several shared-Java multitenancy approaches have been explored, but with only limited success.

Shared-Java multitenancy techniques to date have generally involved exposing a restricted subset of JVM functionality for use by user applications. Sometimes called "JRE White List", this list defines the specifically approved Java classes and language features that are permitted for use by application developers targeting deployment on such services. These requirements impose considerable restrictions and caveats on application developers and their resulting programs, effectively undermining the write-once-run-anywhere promise of Java Platforms.

A BREAKTHROUGH MULTITENANT SOLUTION FROM WARATEK

Waratek is the first Java virtualization technology that allows Java applications, of any type, to run natively and restriction free in multitenant virtual configurations on the cloud.

After an extensive examination of the problems associated with Java, virtualization and cloud computing, Waratek developed a breakthrough solution. With over 50 patents granted, and 100 more pending, Waratek has developed an innovative JVM designed specifically for cloud computing.

Waratek engineers understood that for traditional Java applications to run successfully on the most sophisticated cloud infrastructures, a totally new approach to JVM design and operation was necessary. Waratek's solution – its

newly designed Cloud Virtual Machine (VM) for Java - is the world's first technology solution that renders all prior Java applications immediately compatible with the most advanced features and functionality of cloud computing. Waratek® the new Cloud VM for Java meets the clear need for a higher-density hosting model for Java applications on virtualized infrastructure, where multiple mutually-distrusting applications can be safely and securely hosted side-by-side within a single Java virtual container (JVC) of the Waratek Cloud VM for Java.

Waratek's Java virtualization solution set gives virtualized cloud infrastructure running Java applications similar performance, reliability, scalability, and manageability characteristics as high-end operation critical servers - but without sacrificing binary compatibility with existing Java applications.

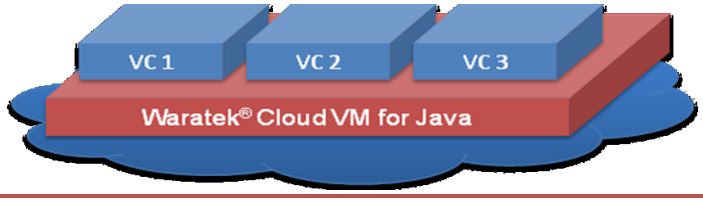
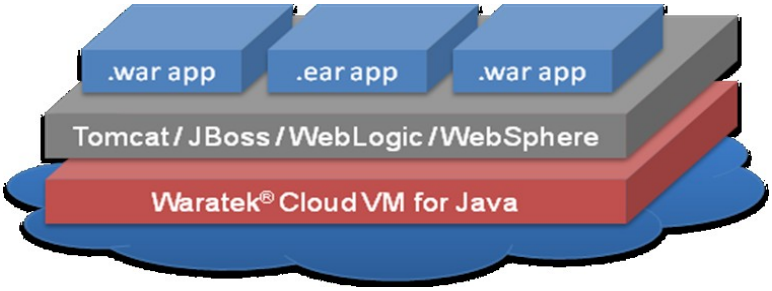
Waratek's Java virtualization solution is made possible by a unique multitenant virtual container architecture that virtualizes the Java Platform and delivers Java-as-a-Service (JaaS), running existing Java/J2EE platform software as a multitenant cloud service without code change. Just like an x86 Virtual Machine (VM) presents a virtual image of a shared physical server, each JVC provides a controlled and isolated virtual image of a shared JVM. By building native support for VCs inside the JVM, ultra-elastic and highly granular multitenancy for the Java Platform becomes possible without losing Java's write-once-run-anywhere binary compatibility guarantee.

Waratek, the Cloud VM for Java, solves the problem of Java performance and compatibility with the cloud by re-engineering the core functionality of how Java operates within the JVM itself. Waratek delivers all of the advanced features of cloud computing to all existing Java applications at run-time, eliminating the need for companies to undertake the huge cost and risks of modifying their existing applications for cloud computing compatibility.

The following figure highlights some of the core features of Waratek, the Cloud VM for Java.

FIGURE 1

WARATEK® - THE CLOUD VM FOR JAVA: FEATURE HIGHLIGHTS

GENUINE MULTITENANT JVM FOR THE CLOUD	• Waratek, the Cloud VM for Java, introduces Java Virtual Containers (JVCs) • A JVC is a meta-circular VM within the JVM which shares the host JVM environment with other JVCs • A JVC is extremely lightweight: <1MB overhead per JVC 
FINE-GRAINED RESOURCE CONTROL	• A single JVM can host tens or hundreds of JVCs: from 1MB up to many GBs • Every aspect of JVC execution is isolated: CPU priority, memory limits, bandwidth etc • Every aspect of JVC use is metered: CPU use in Hz, memory/IO use in Bytes • JVCs provide computing power like electricity kW-h: CPU consumption in GHz-h, memory consumption in GB-h
DELIVERS JAAS (JAVA-AS-A-SERVICE)	• Runs existing Java/J2EE platform software as a multitenant cloud service without code change: binary compatible with existing apps and platforms • Every .war/.ear app gets its own JVC 
PROVIDES AN OPEN FRAMEWORK FOR EXTENSIBLE JVCS	• JVC framework is extensible and customizable: enhanced JVCs can be created using Waratek APIs • Special-purpose JVCs are currently in active development: Scala JVC, Clojure JVC, Python JVC, Ruby JVC, Groovy JVC

CONCLUSION

While multitenancy has become a feature of mainstream cloud and enterprise computing solutions, existing Java application servers cannot deliver multitenancy support to cloud-based applications.

Given the industry consensus that mainstream enterprises, as well as small, medium and large organization in the public, private and third sectors, will eventually adopt cloud computing, it is clear that computing in the cloud require technologies and architectures that are natively designed for the cloud.

It is also clear that applications offered as cloud services require multitenancy support in their enabling platform technologies. In the absence of application platforms with this capability, companies must currently choose which approach to use to support multiple tenants in the cloud.

In terms of Gartner's Reference Architecture for Multitenancy, the Waratek Cloud VM for Java is the only *shared-everything* solution in the market capable of offering an immediate answer to the complexities of using Java in the cloud.

Designed specifically for the cloud, Waratek the Cloud VM for Java natively delivers services that enable the migration of existing Java enterprise workloads to cloud environments. These services include high density multitenant provisioning, highly granular metering of CPU cycles and memory utilization, real time resource allocation and prioritization, real time instant scalability and granular elasticity, and binary compatibility with existing Java applications.

For more information about the Waratek Cloud VM for Java solution, please visit us at www.waratek.com or contact us at enquiries@waratek.com.