PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

Virtualizing JBoss Enterprise Middleware with Azul

Shyam Pillalamarri
VP Engineering, Azul Systems
Stephen Hess
Sr. Director, Product Management, Red Hat

June 25, 2010

Agenda

Java Virtualization

Current Limitations

Azul – The Company

JBoss + Azul Solution

Benefits of the Solution

Q & A

Java Virtualization

State of the Union

Definition – transparently creating a virtual entity as opposed to a physical entity (Network, Storage, Server, OS, Application Server, Application etc.)

Goal – ease of manageability through centralized administrative tasks while providing for scalability and resiliency

Current State – most of the components mentioned above are virtualized to some extent or the other

Current Limitations – goal has been met on the lower layers of the system stack (closer to hardware), NOT as much at the higher layers

Java Virtualization
Current Limitations

For Java Applications, current state of virtualization provides

- Scalable virtualized storage
- Scalable virtualized compute BUT
- Most important resource - Memory is not scalable, is rigid AND
- Dependent on the OS and server hardware

Clearest Sign of this issue –

- Most commonly used heap sizes for Java applications has remained in the 2-4GB range for the past decade!

Java Application Memory Needs
Common Myths

- 64 bit JVMs solve the problem
  - Not an addressing limit but practical issue with GC Pauses
- Light Weight Frameworks are the answer
  - Address the code complexity issue and in fact increase the memory footprint
- My servers don't have much memory
  - Sweet spot for commodity servers is now between 64-128GB; will require 40-50 JVMs to fill it up!
- Who needs more Memory?
  - Every application wants to do more customization (session state), more caching, needs to handle more data (Web2.0, mobile, dynamic XML)

Java Deployment Challenges
Inherent limitations of the Java Platform

….Java platforms are

Rigid

- Instance footprint is fixed at the launch time

Non-elastic

- Can’t dynamically scale; leads to poor utilization and fragility

Not optimized

- Unable to keep pace with commodity server capacities (e.g. cores & mem)

Unstable

- Already at their limits and complex & costly to scale out

Azul Systems

Harnessing the power of virtualization

Founded in 2002 and shipping 4rd generation product

Privately held with offices around the globe

Recognized leader with award-winning technology

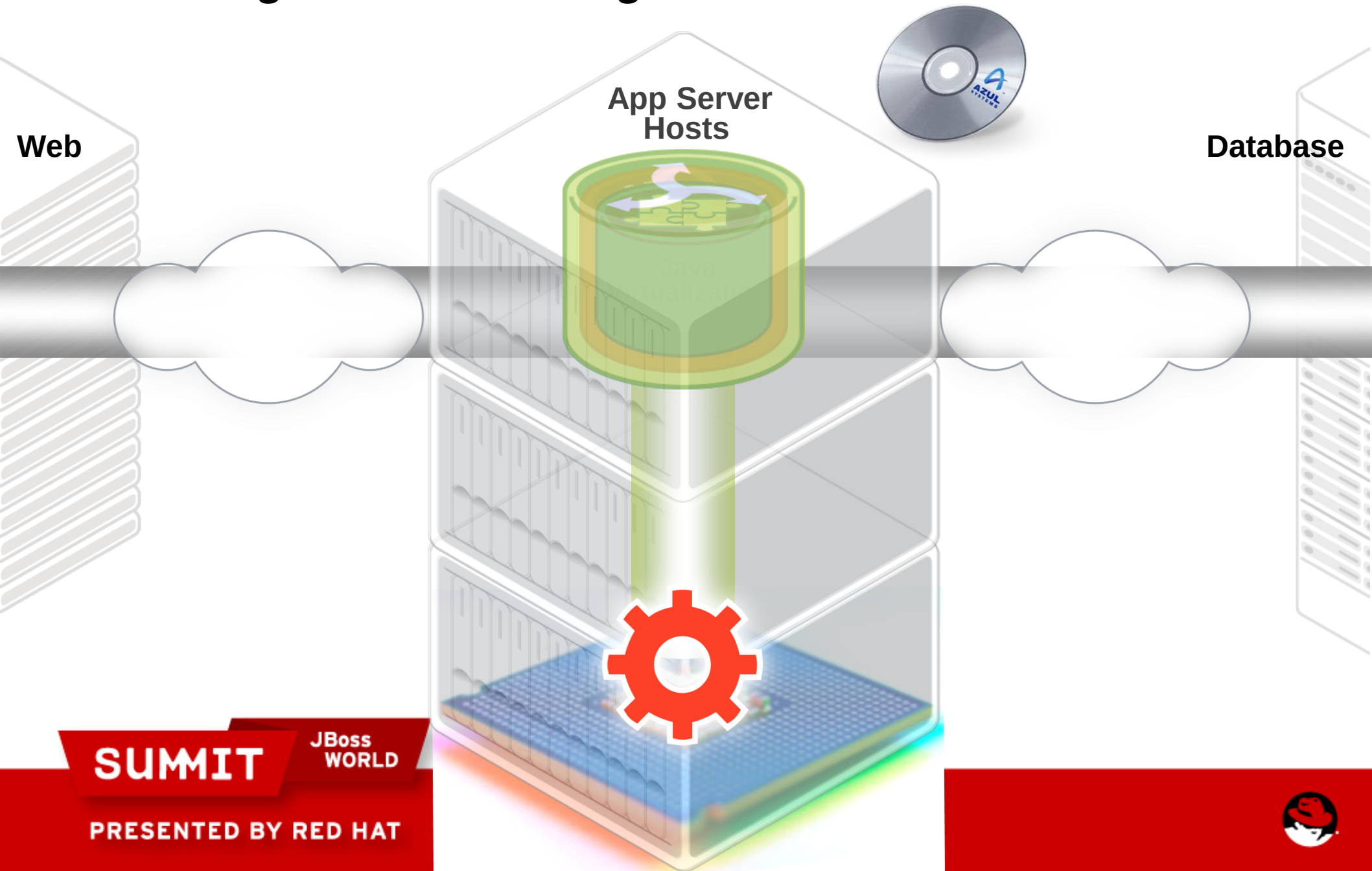
Numerous industry firsts:

- Generational pauseless garbage collection
- Elastic memory
- OS-agnostic Java virtualization

Proven, mission-critical deployments in global 2000 accounts

Java Virtualization

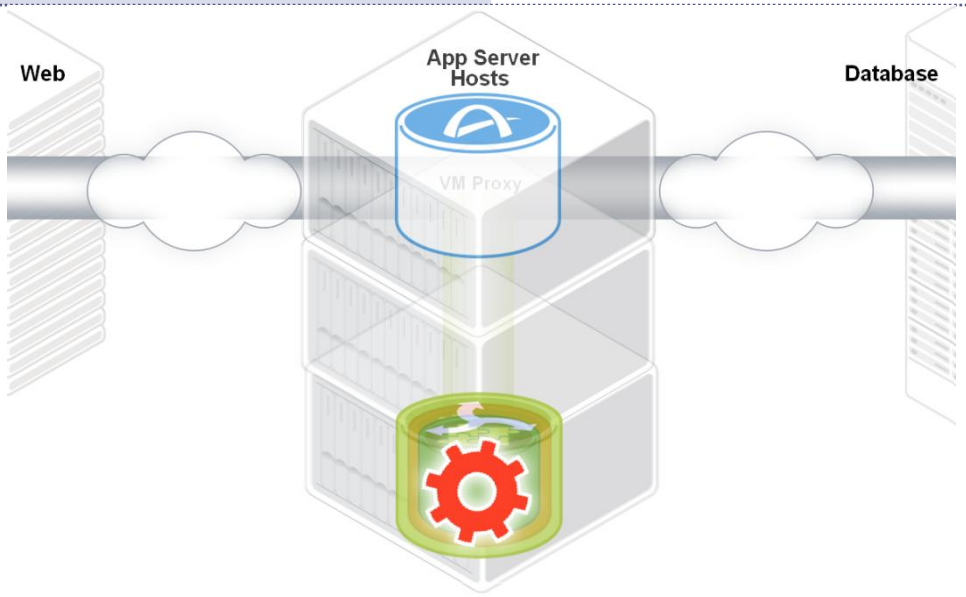
Liberating Java from its rigidities



Java Virtualization

Bare-metal, proxy architecture

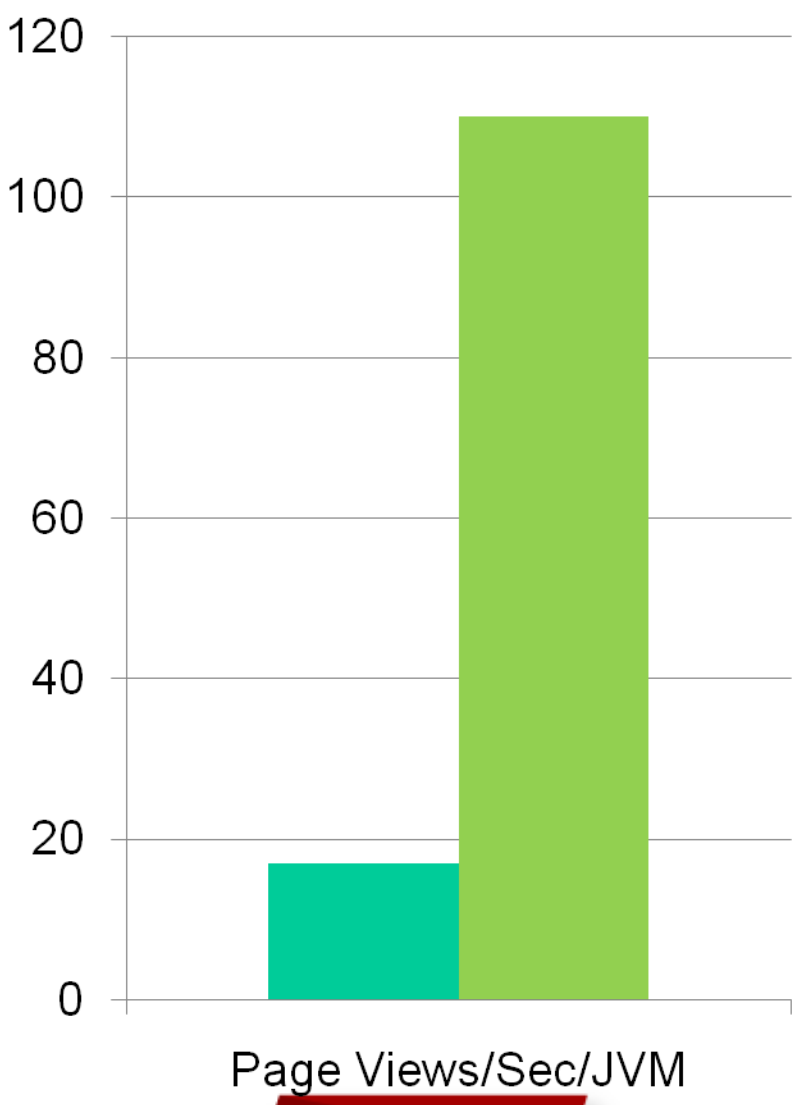
Allows you to…	So that you can…
Use and manage application resources outside the constraints of guest Operating Systems	**Enable application elasticity and maximize performance, reliability, visibility and manageability**

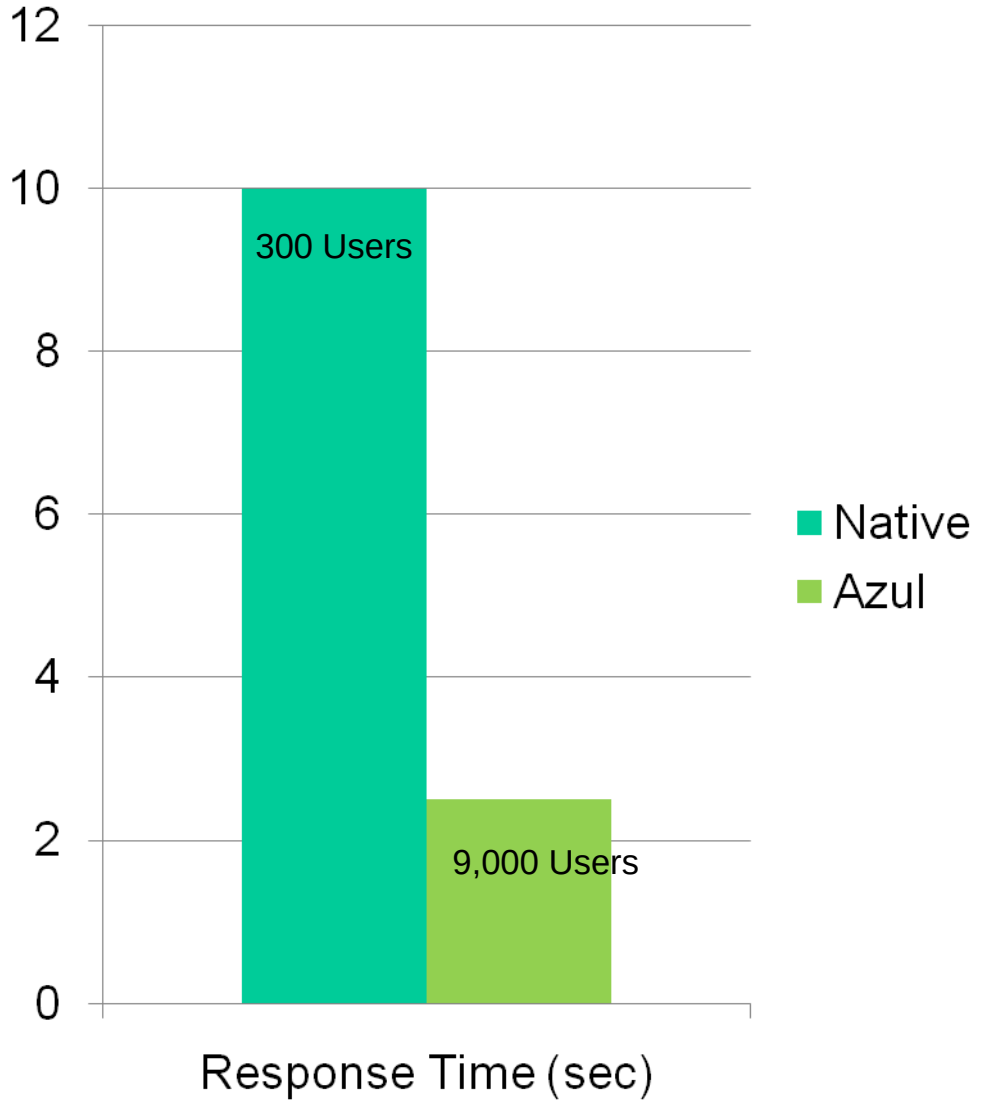


Features	**OS Agnostic**	**Solaris, AIX, Windows, Linux, HP-UX**
	Non-x86 Java workloads	**Allows for heterogeneous consolidation onto commodity HW**
	Transparent to Security or HA Configs	**Completely Transparent to App**

Breaking Java Scale Barriers

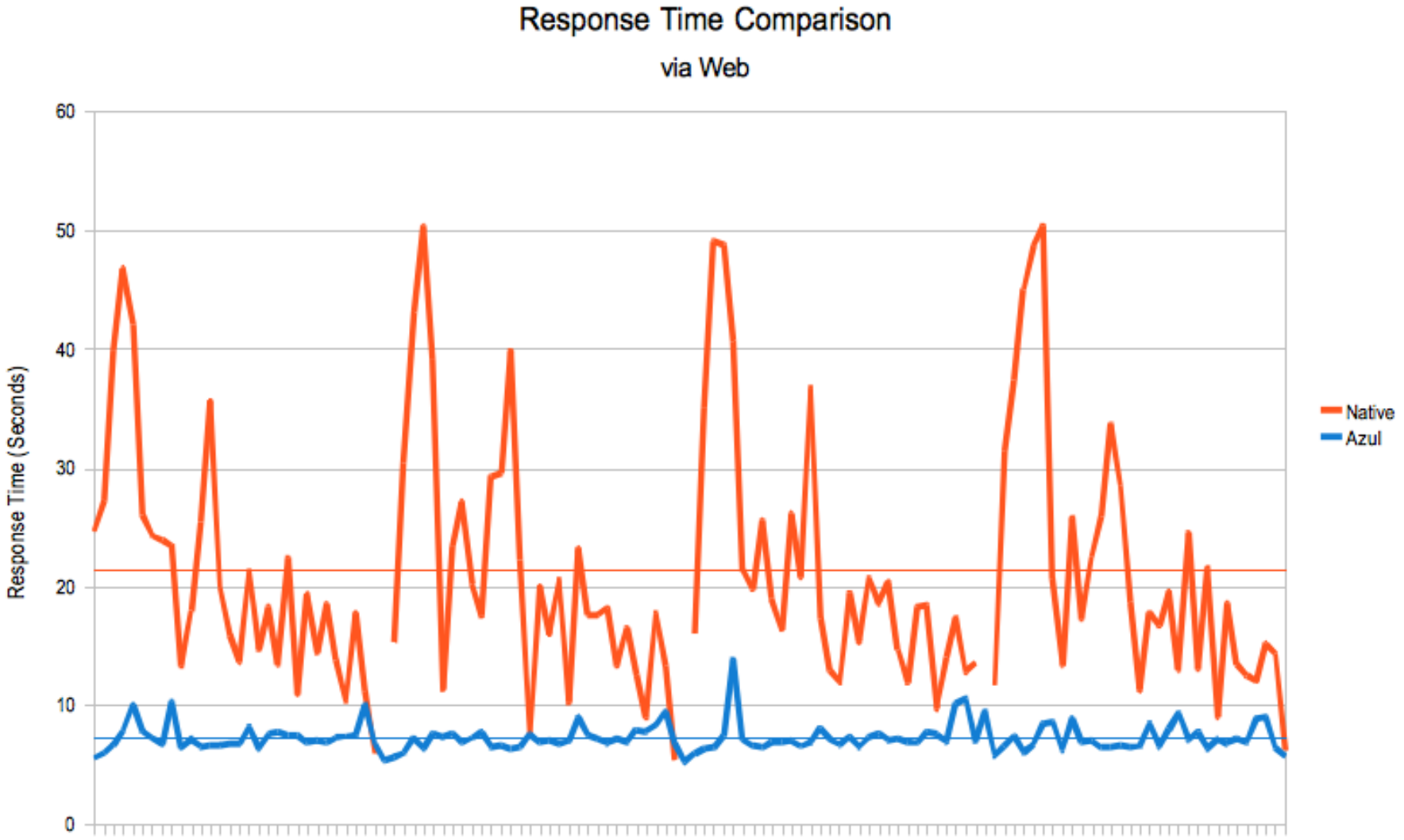
JBoss Portal on the Azul Solution



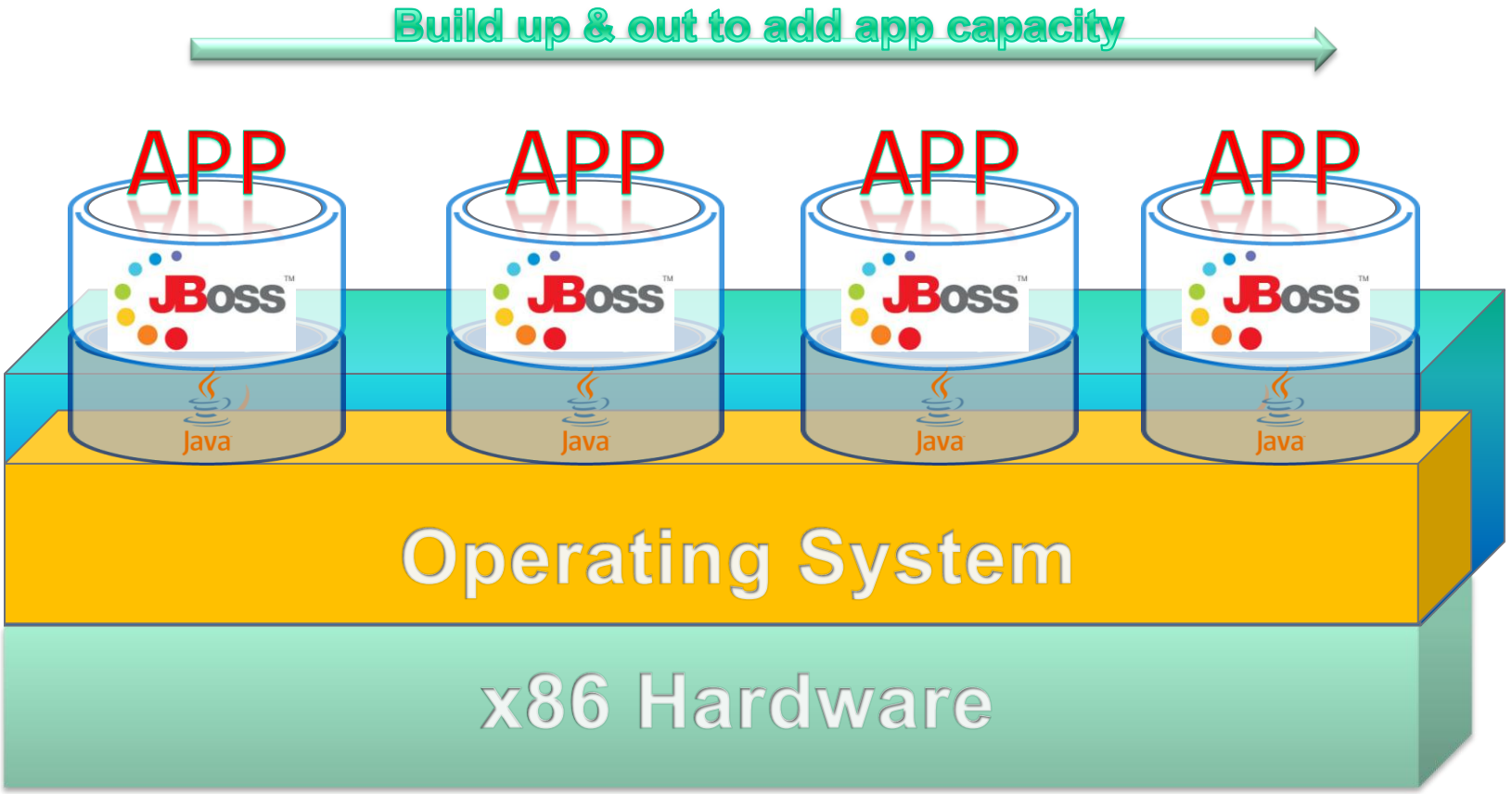


Breaking Java Scale Barriers

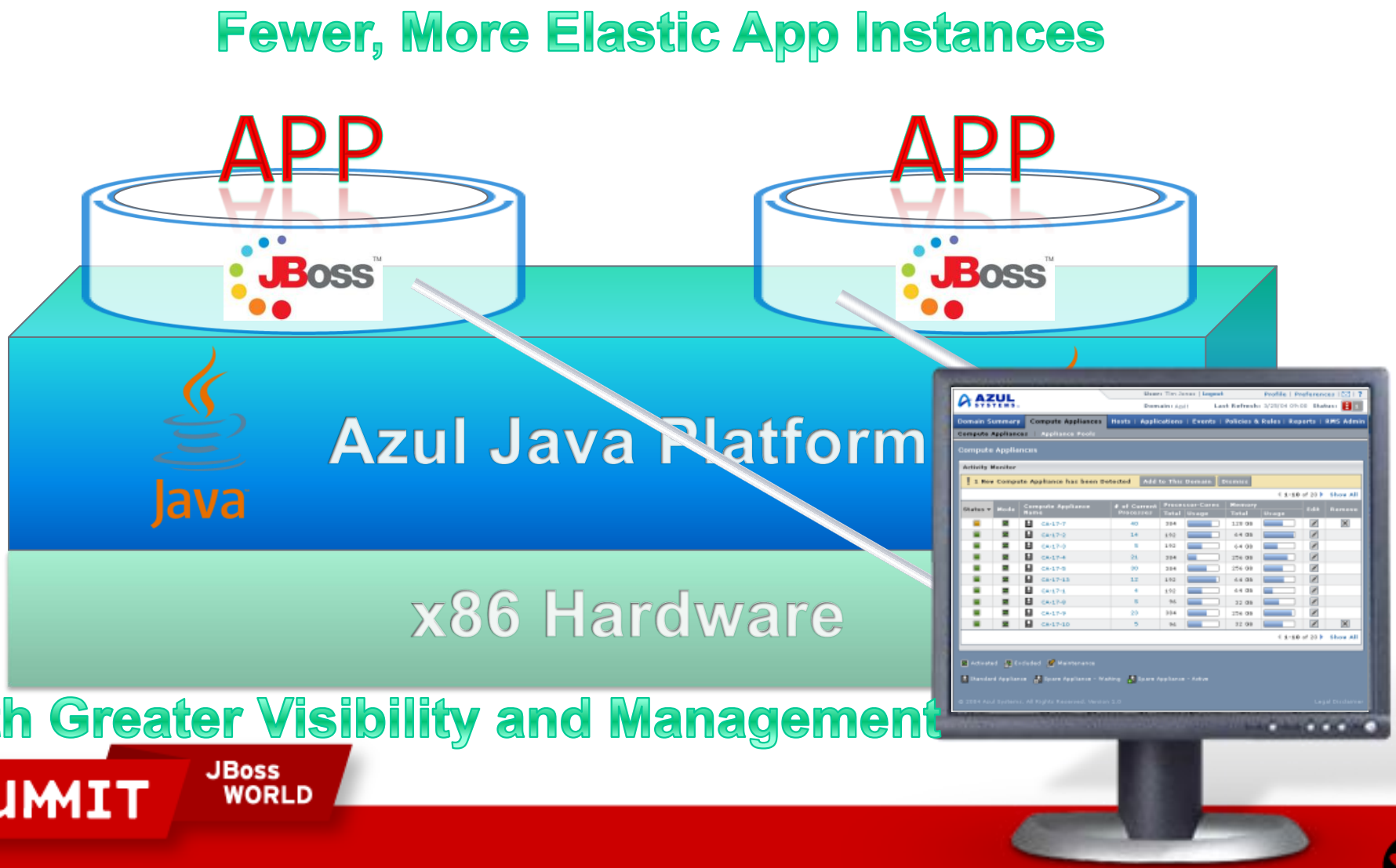
Giving app the resources they need



Existing Deployments

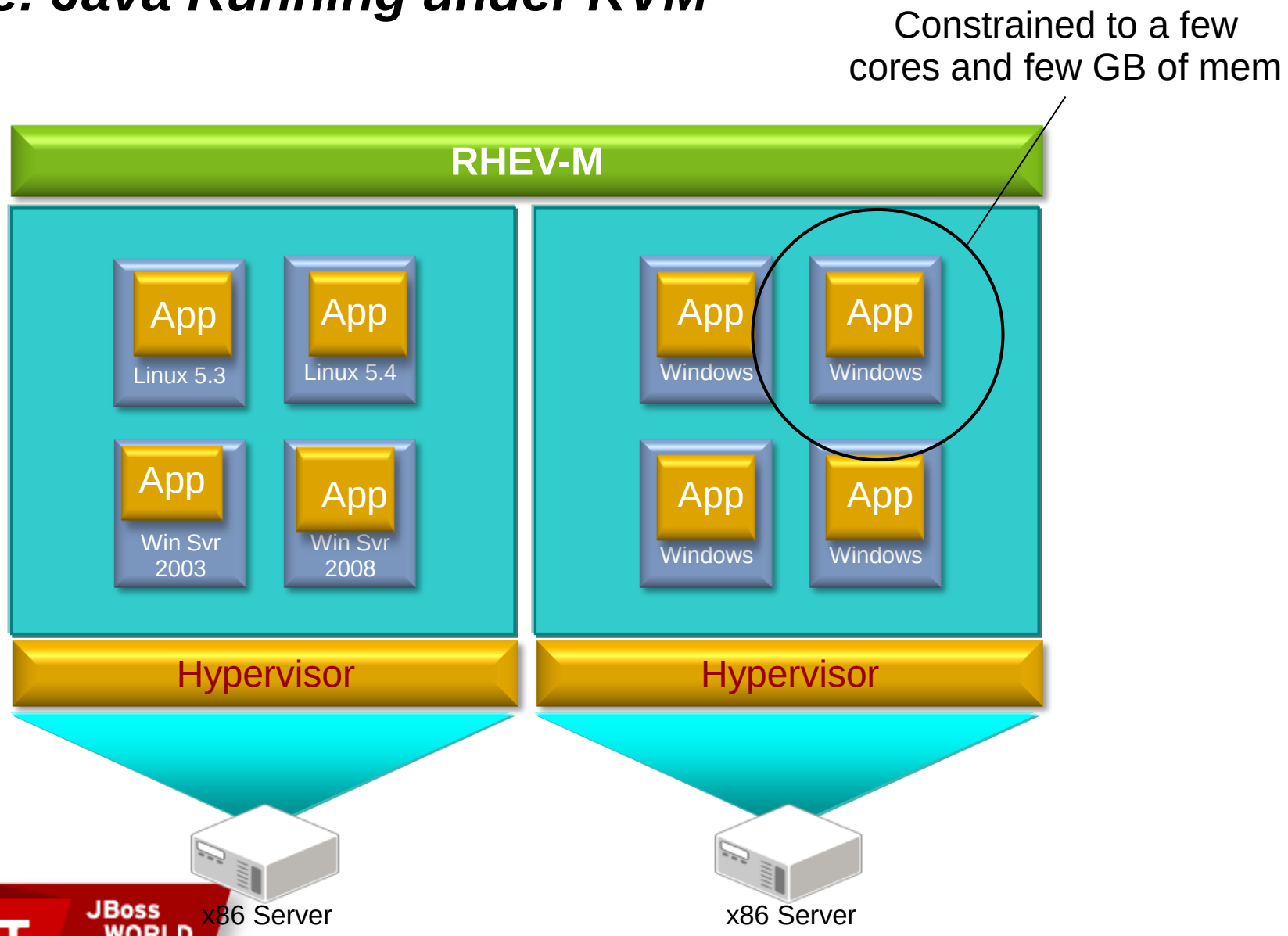


Simplifying Deployments



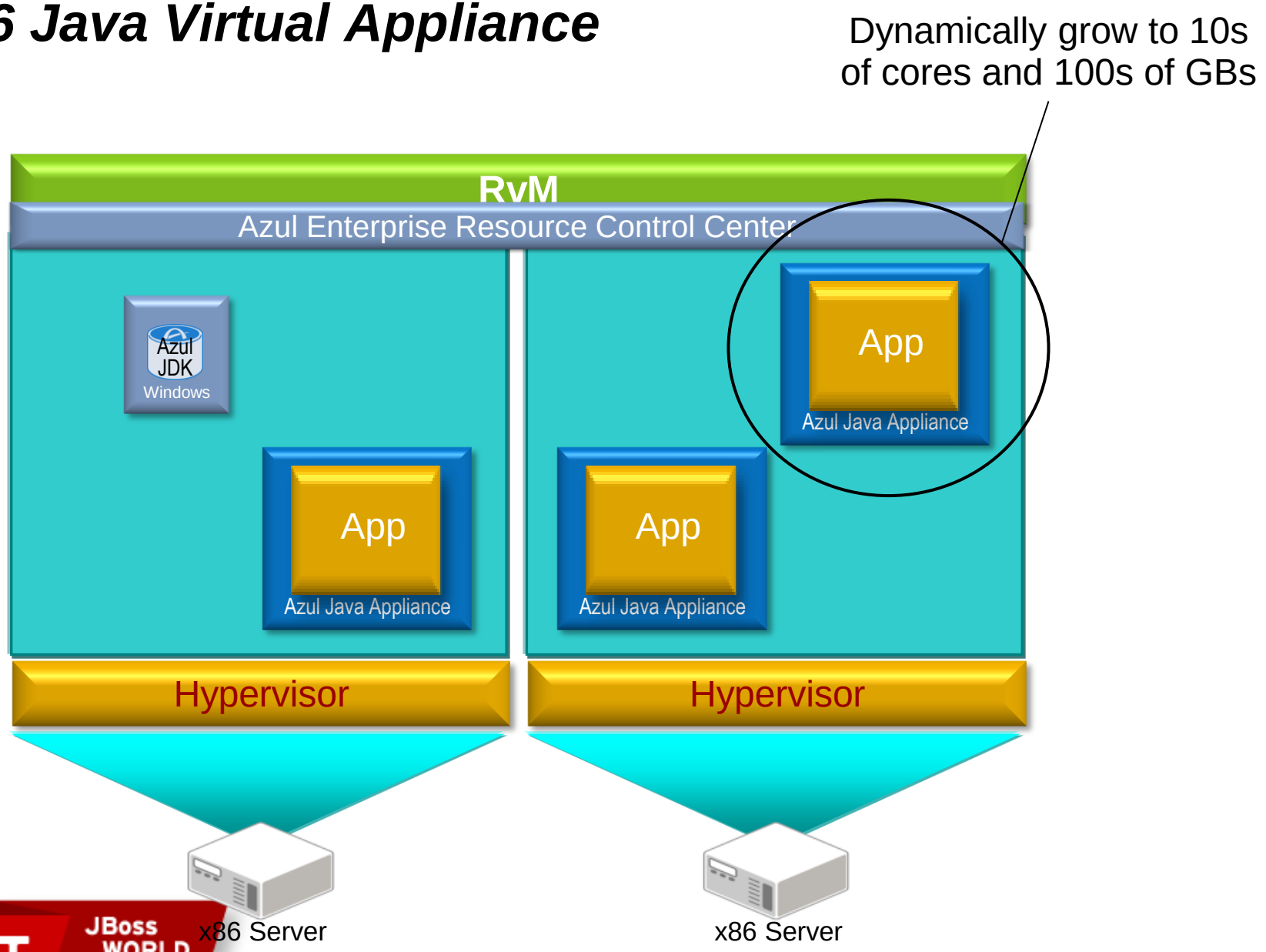
Traditional Virtualized Java Deployment

Example: Java Running under KVM

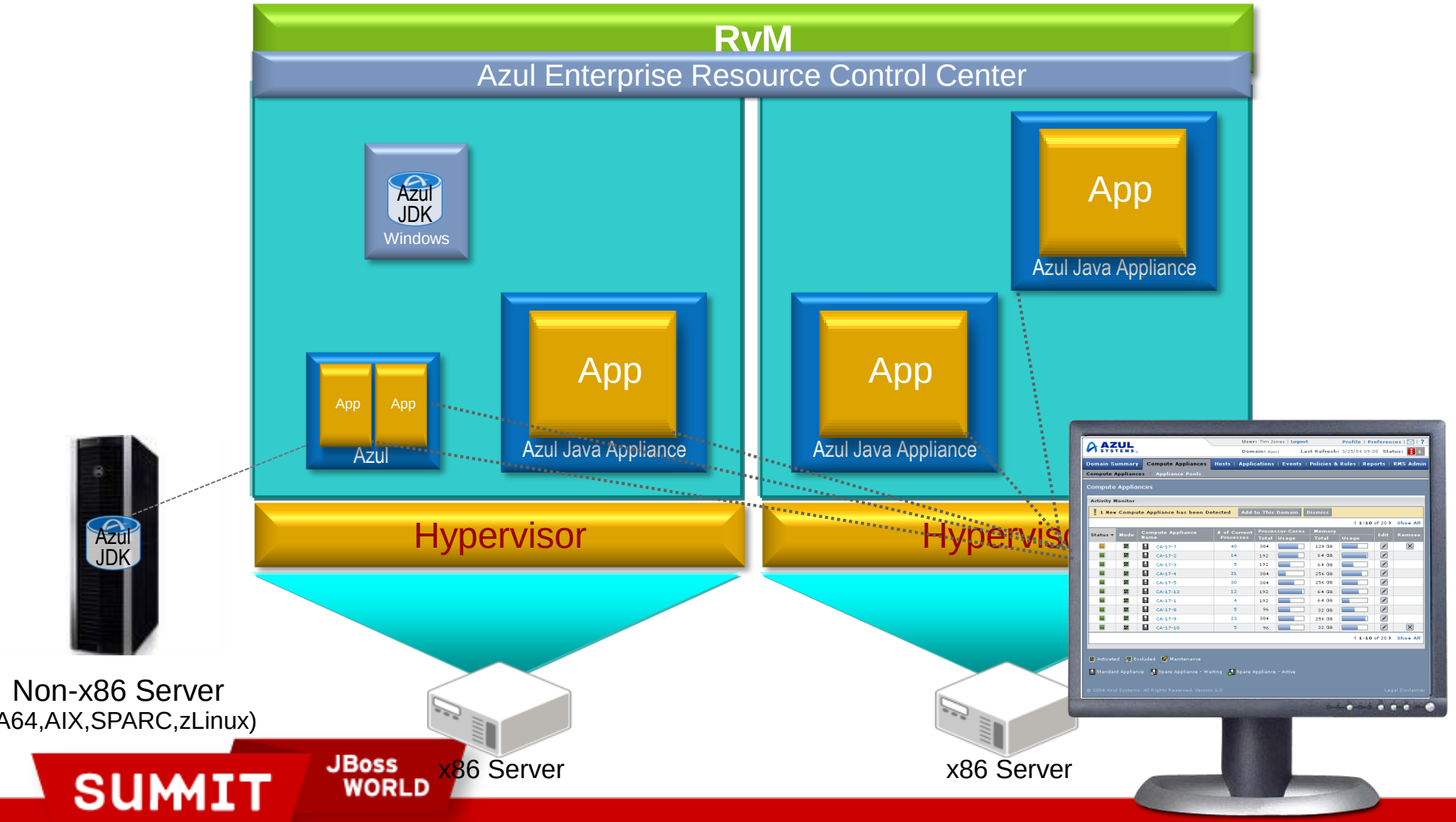


The Azul Elastic Java Platform

Azul x86 Java Virtual Appliance



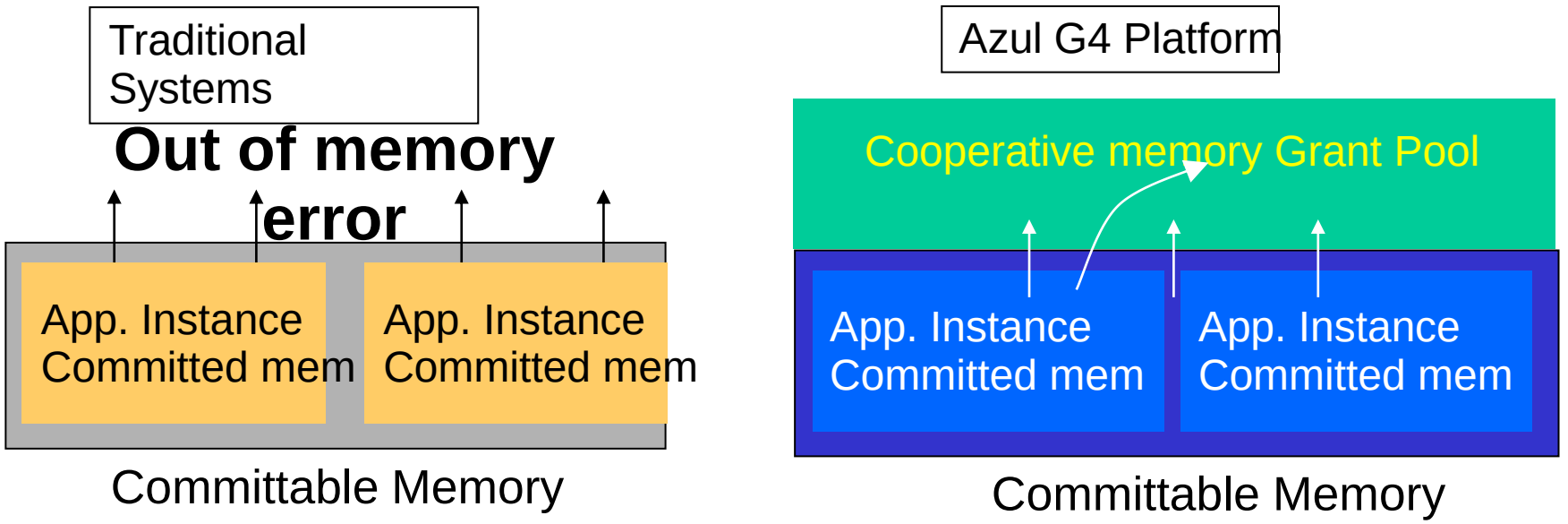
Azul Elastic Java Platform
Supporting x86 and non-x86 Java Workloads



Elastic Memory

Better utilization AND greater resiliency

Allows you to…	So that you can…
Elastically right-size memory based on real-time app behavior; utilize shared memory headroom	**Improve application resiliency and uptime; faster time to deployment; higher resource utilization**

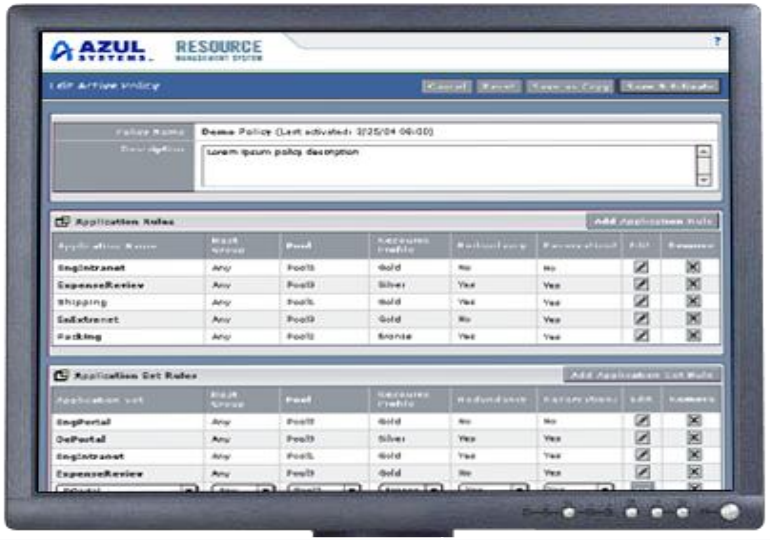


Features		
	Reduces "out of memory" Errors	**Increases App Resiliency**
	Increases Memory Utilization	**"soft landing" for memory leaks**
	Headroom shared across all apps	**Careful fine tuning eliminated**

Performance Monitoring in Production

Always on and Zero Overhead

Allows you to…	So that you can…
Collect real-time, fine-grain performance data in production & development environments with no application impact	**Dramatically improve problem resolution times; identify and eliminate performance issues**



Features		
	App Configuration Info	**Open Sockets and File data**
	Thread Visibility & Stack Trace	**System Call Profiling**
	Lock Contention	**Memory Usage and Object Stats**

Most Elastic Java Platform

Business Implications

Achieve consistently fast app response times

Improve customer experience and loyalty

Greater app availability, even during peaks

Room for future growth

Lower maintenance costs

IT Implications

Orders of magnitude improvements in response time and throughput

Robust and elastic foundation for all your Java applications

Simplified deployments with unmatched production-time visibility and management

Fast ROI

- Reduced TCO (>50%)
- Reduced JVM instance count

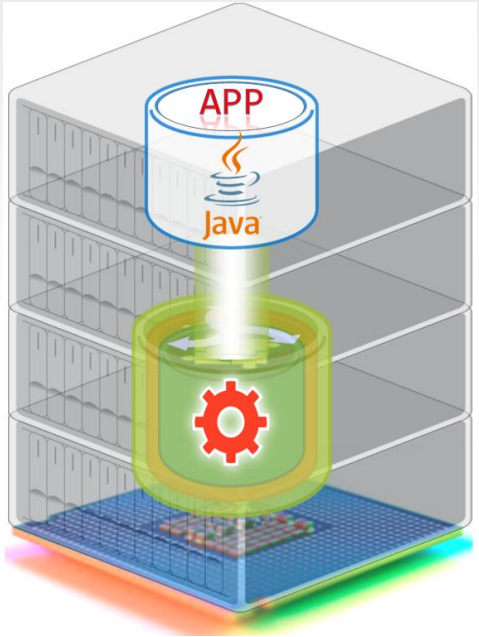
The Azul Elastic Java Platform

- ***Virtualized:*** Server and OS agnostic, transparently separates capacity from configuration
- ***Elastic:*** Smoothly scale up & down within available infrastructure
- ***Resilient:*** Tap shared headroom to survive unexpected demand, policy-based management & enforcement with isolation
- ***Efficient:*** Maximize use of available physical resources
- ***Visibility:*** Fine-grain instrumentation, always on, from development through production with zero overhead and without code changes
- ***Automated:*** Deliver resources automatically based on real-time demand

The Azul Elastic Java Platform

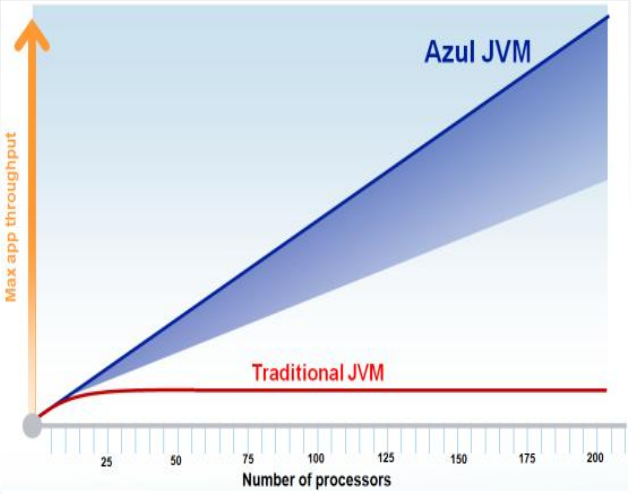
Liberate Java from the OS

- More Elastic
- Java-Optimized
- Highly Scalable
- More Resilient



Break the Scale Barriers

- Improved Utilization
- Better Response times
- Greater Scalability
- Higher Throughput



Simplify Deployments

- Reduced Complexity
- Better Manageability
- Greater Visibility
- Lower TCO

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#summitjbw

READ THE BLOG

http://summitblog.redhat.com/