LISTEN.
THINK.
SOLVE.℠

An Embedded Service Platform for Uninterruptible Processing

Lessons Learned

Tim Biernat
Paul Schmirler

Rockwell Automation

Agenda

- Industrial Automation
- Decisions, Decisions
  - Which Java? What Container? Datastore?
- OSGi
- Embedded Database
- Demo
- Some Challenges
  - Performance
  - Flash Memory
  - Hardening
  - Troubleshooting
- Q & A

Introductions

Rockwell Automation

- 20,000 employees, $4.8 billion sales
-
- Automotive, Food & Beverage, Pharma, Material Handling, Mining, Oil & Gas, Electronics, and more
- Components, motor drives, industrial control and information systems

- Tim Biernat
  - Worked with General Dynamics, Motorola, IBM and SoftwareMentor
  - Interests: java, distributed computing, real-time fault-tolerant systems
- Paul Schmirler
  - Worked with eFunds, Eagle Technology
  - Interests: mobile computing, cloud computing

Industrial Automation Primer

Many kinds of production processes

- Discrete (auto assembly)
- Batch (beer brewing)
- Continuous (metal production)

- Challenging Environments
  - Hot, cold, dusty, wet, EM, G shock
- Safety Concerns

Manufacturing is extremely competitive

- Downtime unacceptable

- Long-lived systems
  - 15 to 20+ years in service is not uncommon
  - Maintenance, spares, support can be a real challenge

Where will my Linux kernel be 10 years from now?

Software in Industrial Automation

Growing role

- Visualizing, communicating, integrating, controlling, monitoring
- Historically dominated by MS tech: Windows OS, OPC (OLE for Process Control), D/COM, VBA
- Desire to connect factory with enterprise

- Many different platforms
  - Cloud, Virtual, PC, embedded (ARM, x86)
  - Windows, Linux, RTOS
- Java is compelling
  - 3Ps: Portable, Productive, Performant
  - Large open source palette
- Java challenges
  - Largish footprint for many embedded applications
  - Need for fast response, deterministic execution
  - High level of abstraction ➔ isolated from hardware

Decisions: Which Java?

Requirements

- embedded, headless
- target multiple hardware architectures
- full SE (Standard Edition) APIs
- performant as native code

•

- OpenJDK, Oracle SE, Oracle Embedded, proprietary

- Discoveries
  - OpenJDK JIT still immature on ARM
  - benchmarks indicated decent Oracle Embedded performance
  - Oracle Embedded Java is JRE only, not JDK; client JVM only

Benchmarking

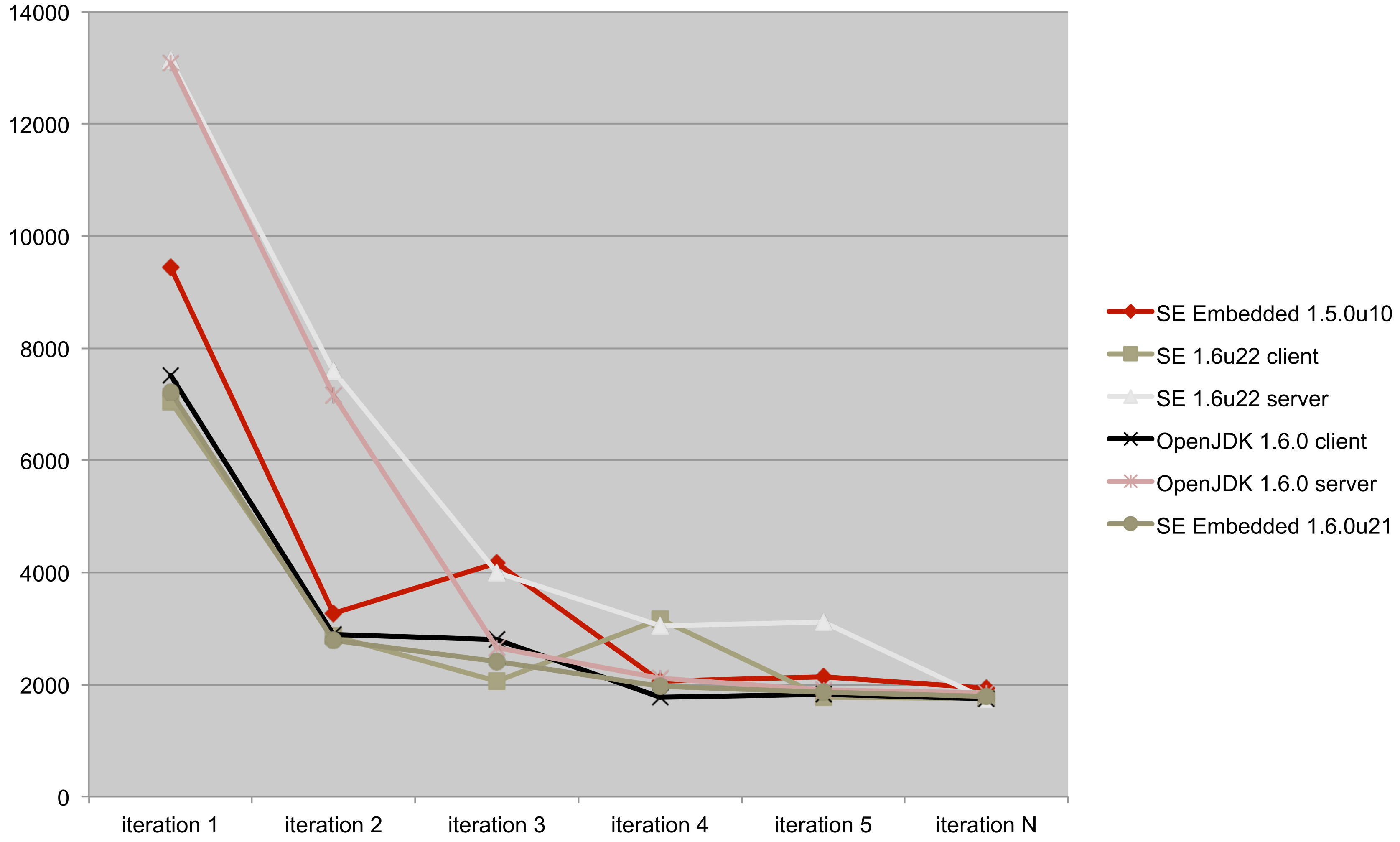
Desire to evaluate various hardware/JVM combinations

- X86, ARM
- OpenJDK, Oracle SE, Embedded JREs

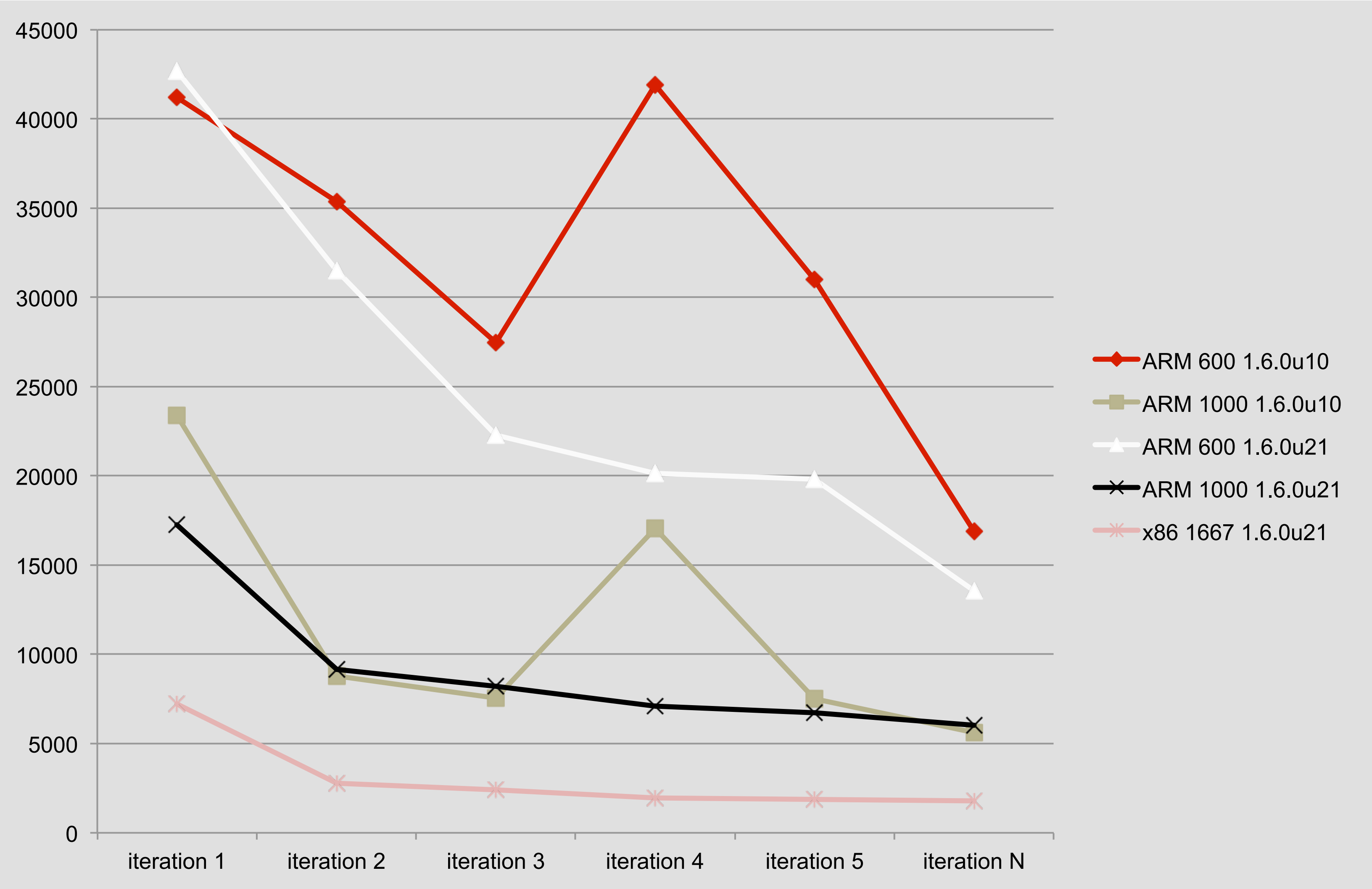
- Representative applications
  - service provider
  - embedded datastore
  - IDE
- DaCapo (http://dacapobench.org/

external concurrency, multiple iterations, JVM warm up

x86 JVMs – IDE Bench



ARM & x86 – IDE Bench



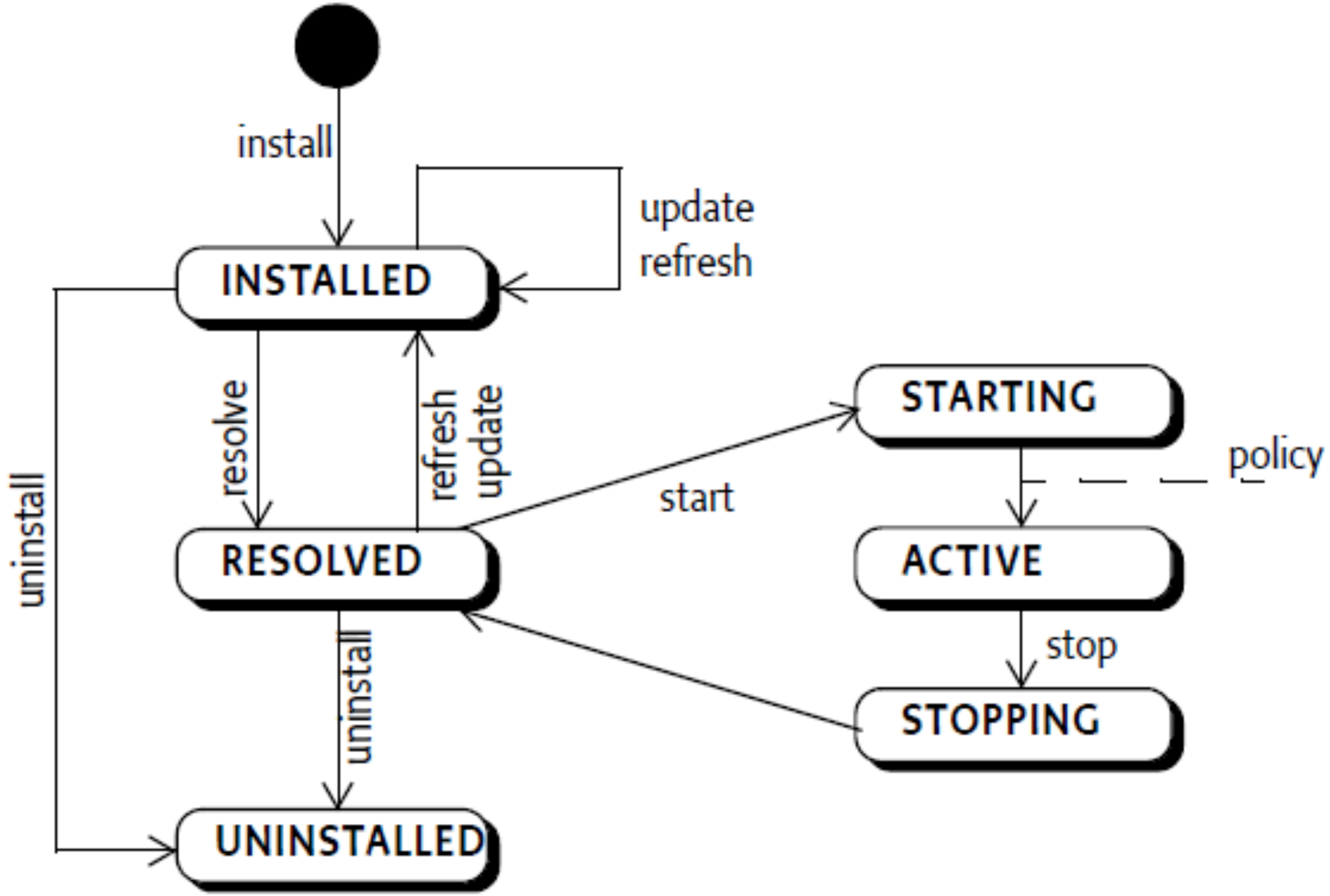
Decisions: Which Container?

- Requirements
  - embeddable (lightweight, proven, manageable)
  - modular deployment
  - dynamic “hot” deploy, update; concurrent software versions
  - Dependency Injection (DI) to minimize hardwiring, reduce coupling
  - support for native code
- Options
  - iPOJO, Guice, SpringDM, straight OSGi
- Discoveries
  - SpringDM is feature full, well integrated DI support
    - but fairly steep learning curve
  - OSGi a good fit
    - lightweight, mature and manageable
    - mature implementations (using Equinox, Felix DI was incomplete)
    - course-grained DI support, introduce full DI framework later

OSGi

- Modularity
  - Bundle – Physical and logical unit of modularity
  - Classloader model – Classloader per bundle
  - Imports/Exports – Restricts visibility to public API
  - Identity – Bundle-SymbolicName + Bundle-Version
  - Native code – Embedded in bundle / multi-platform
- Lifecycle
  - Dynamic – Independent of JVM
  - States – Installed, Resolved, Starting, Active, Stopping, Uninstalled
  - Activators – Hook for lifecycle events / access to OSGi framework
- Services
  - Decoupled, Dynamic, Pluggable
- Management Console

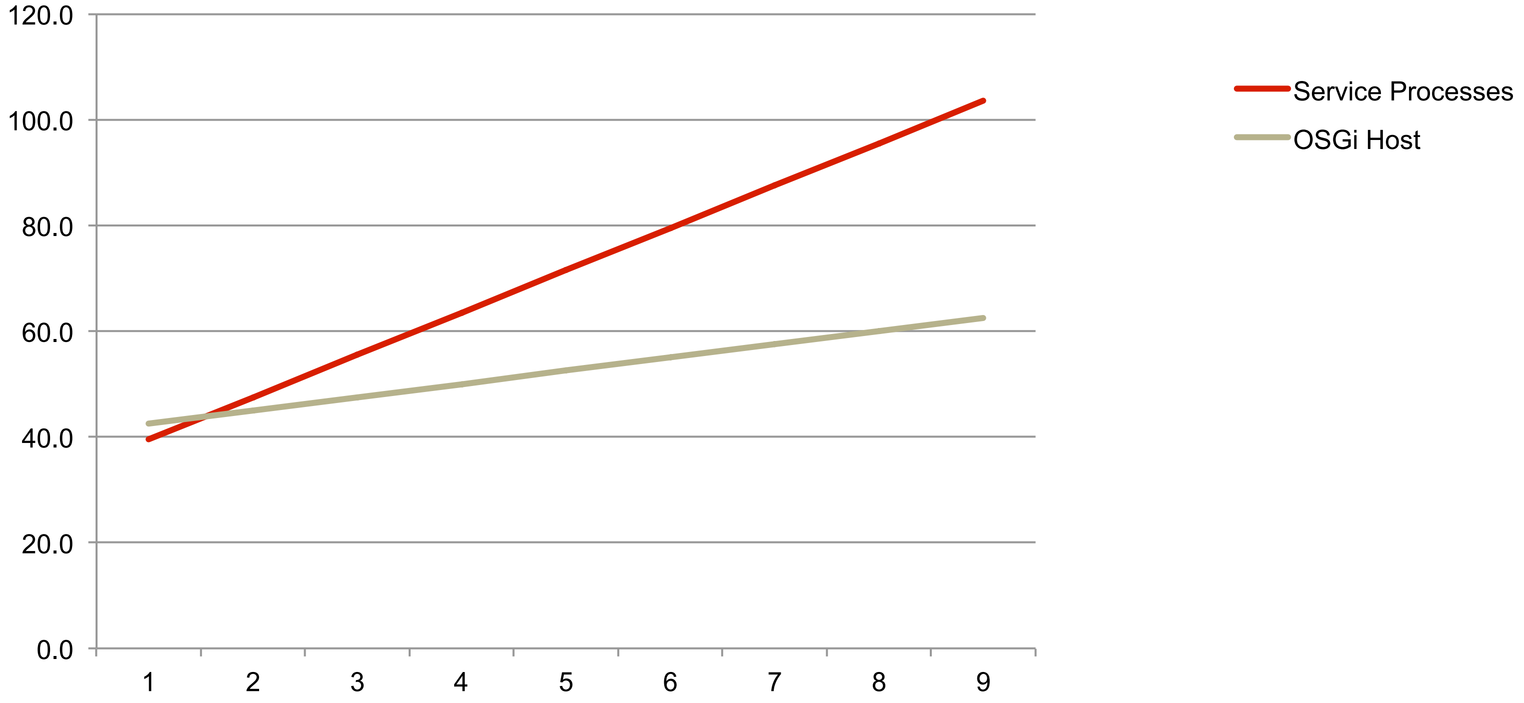
OSGi LIfecycle



ref. OSGI Service Platform Release 4, The OSGi Alliance

OSGi Footprint

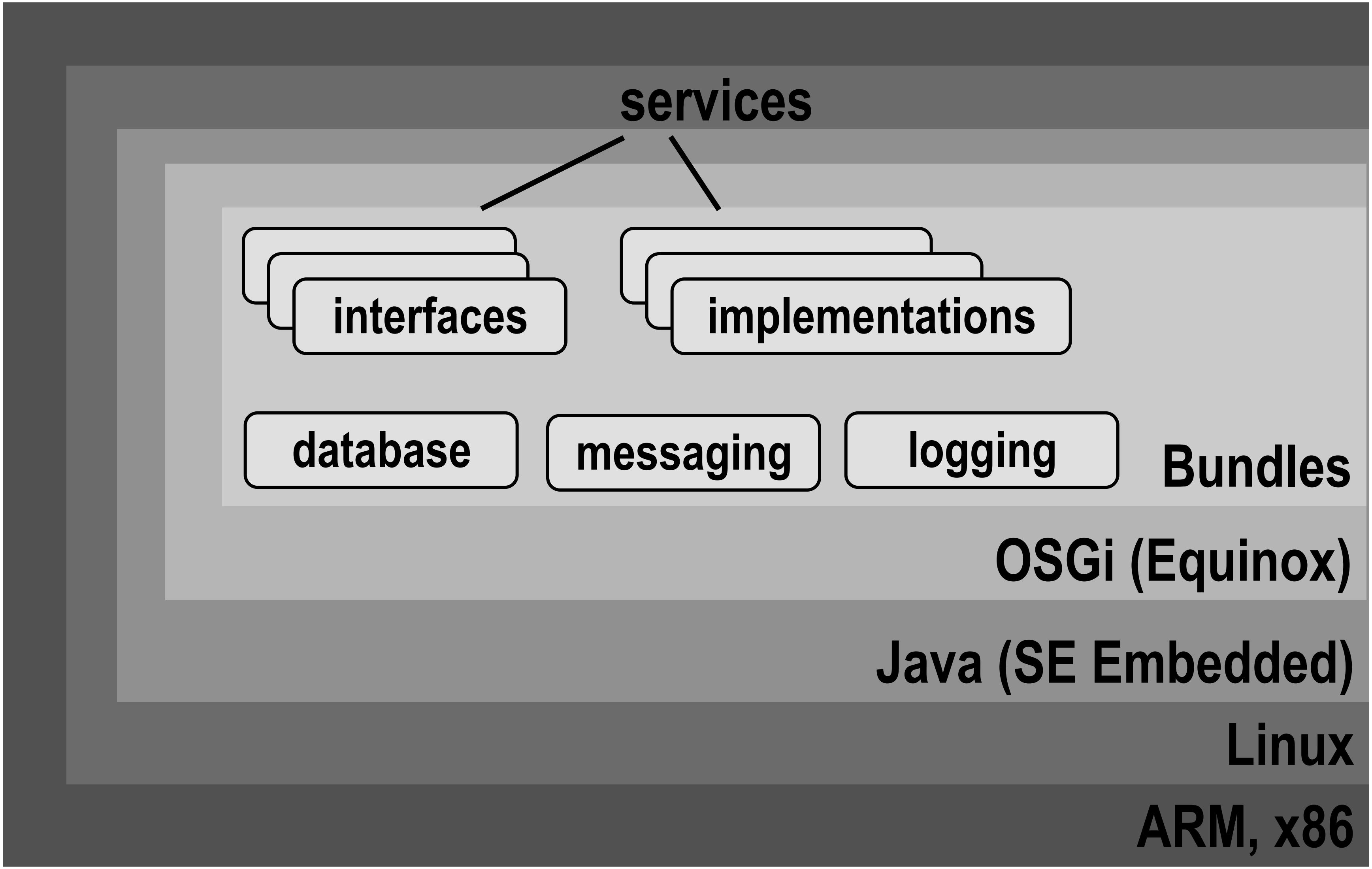
- Single OSGi JVM vs. multiple service provider JVMs
  - shared Java code loaded once
  - shared 3rd party libraries loaded once
  - shared native code loaded once
  - additional one time OSGi runtime overhead: 2 - 3 MB



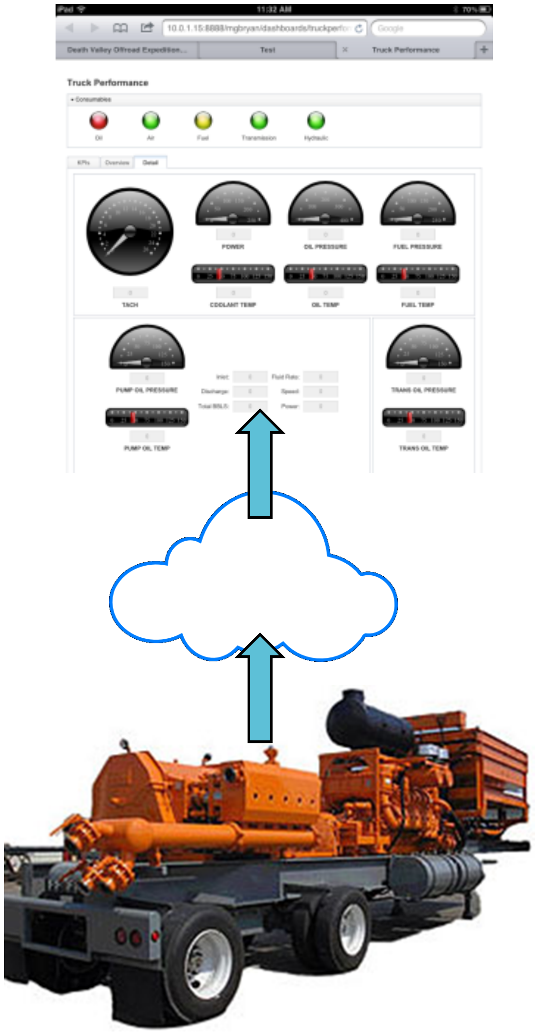
H2 Database

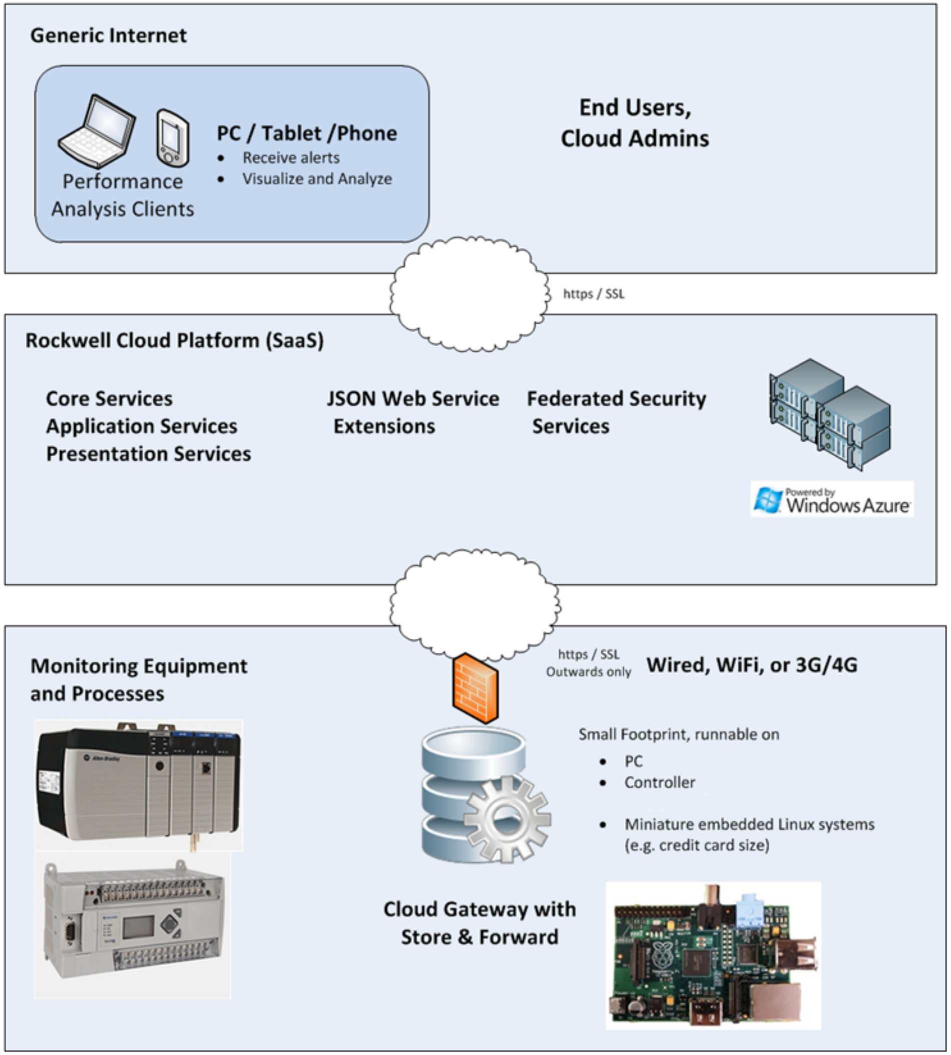
- Small footprint (1MB jar), pure Java solution
- Capable with 16MB of heap
- Best performance in class (vs. Derby, HSQLDB, PostgreSQL)
- Well-tested, good support ecosystem
- Other features
  - standard SQL support
  - dual open source license
  - fully transactional
  - highly tunable (buffers, cache, sync)
  - embedded and client-server modes
  - user-defined functions and stored procedures (in Java!)
  - built in full-text search or Lucerne support
  - built in profiling and performance statistics
  - engine-level encryption (2-3X slower)

Stack


services
interfaces
implementations
database
messaging
logging
Bundles
OSGi (Equinox)
Java (SE Embedded)
Linux
ARM, x86

Cloud Architecture





Generic Internet

Performance Analysis Clients

PC / Tablet /Phone
- Receive alerts
- Visualize and Analyze

End Users, Cloud Admins

https / SSL

Rockwell Cloud Platform (SaaS)

Core Services
Application Services
Presentation Services

JSON Web Service Extensions

Federated Security Services

Powered by Windows Azure

Monitoring Equipment and Processes

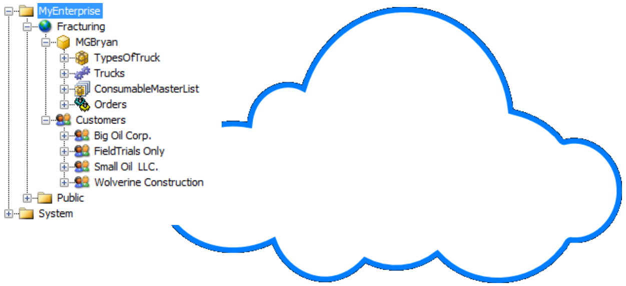
https / SSL
Outwards only

Wired, WiFi, or 3G/4G

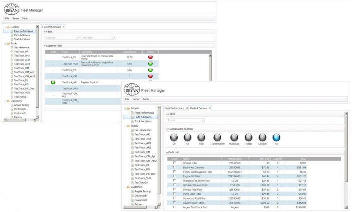
Small Footprint, runnable on
- PC
- Controller
- Miniature embedded Linux systems (e.g. credit card size)

Cloud Gateway with Store & Forward

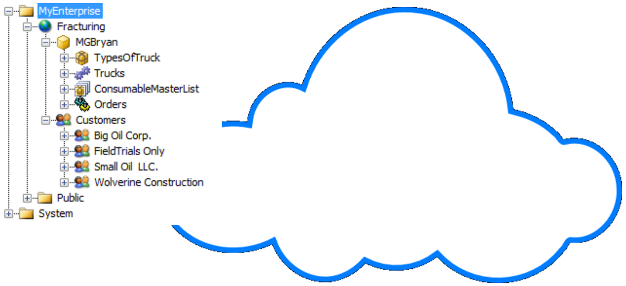
Demo - Cloud Gateway



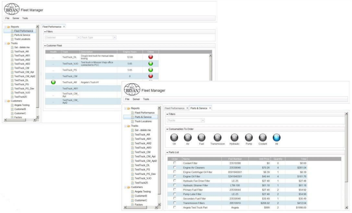
- MyEnterprise
  - Fracturing
    - MGBryan
      - TypesOfTruck
      - Trucks
      - ConsumableMasterList
      - Orders
    - Customers
      - Big Oil Corp.
      - FieldTrials Only
      - Small Oil LLC.
      - Wolverine Construction
  - Public
- System

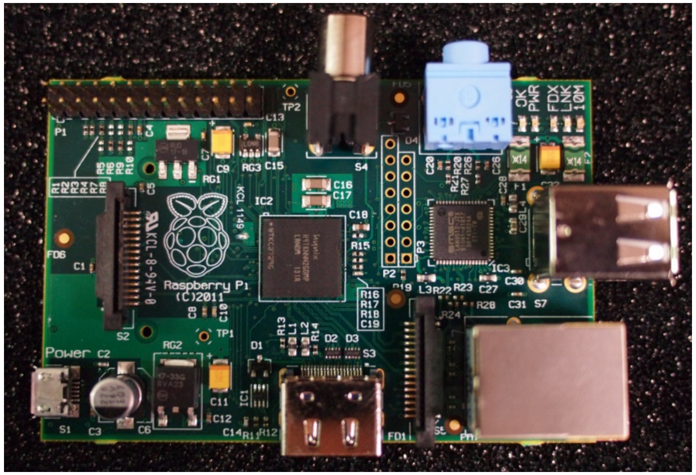


Cloud Gateway

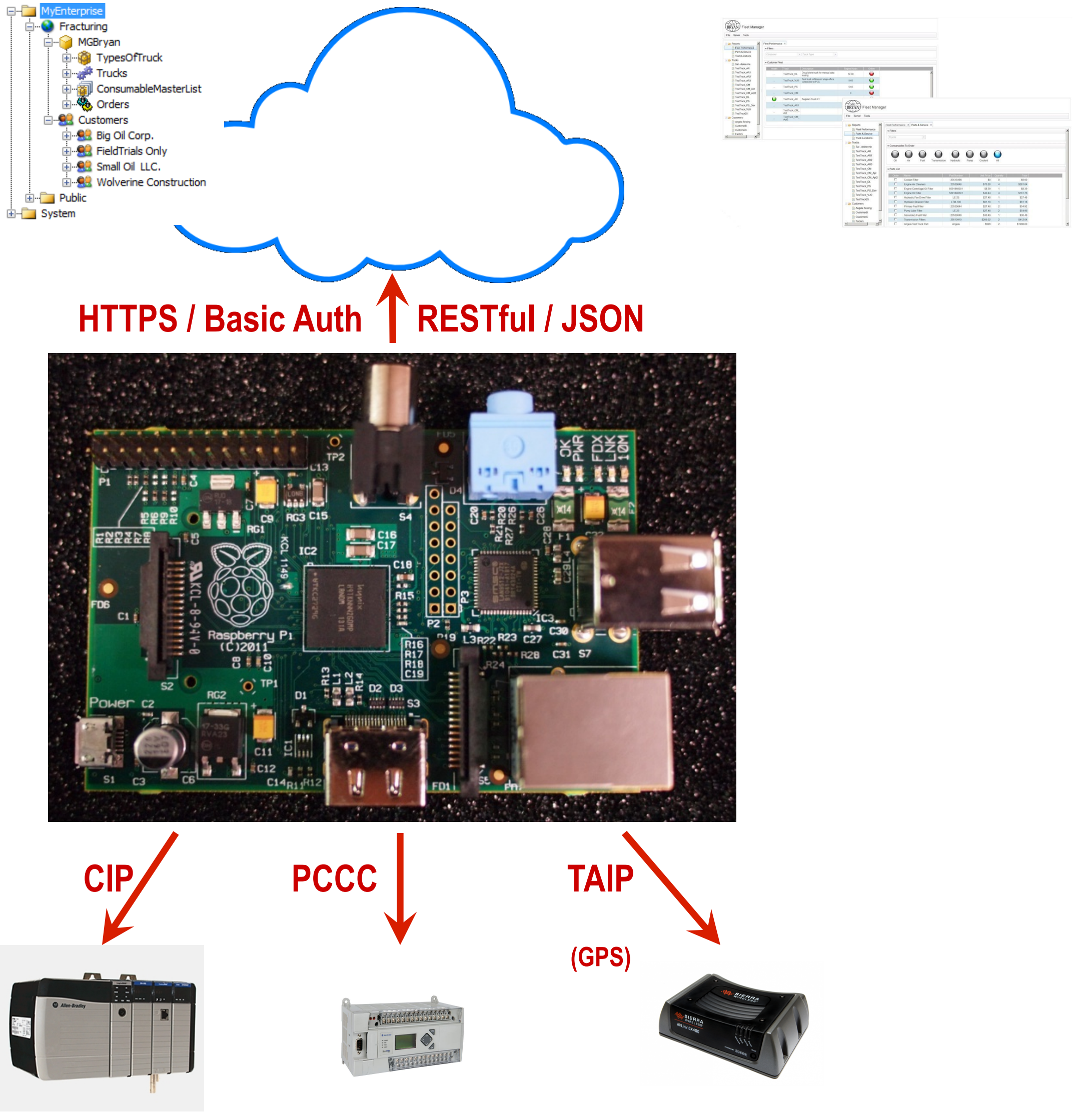


- MyEnterprise
  - Fracturing
    - MGBryan
      - TypesOfTruck
      - Trucks
      - ConsumableMasterList
      - Orders
    - Customers
      - Big Oil Corp.
      - FieldTrials Only
      - Small Oil LLC.
      - Wolverine Construction
  - Public
- System





Cloud Gateway



- MyEnterprise
  - Fracturing
    - MGBryan
      - TypesOfTruck
      - Trucks
      - ConsumableMasterList
      - Orders
    - Customers
      - Big Oil Corp.
      - FieldTrials Only
      - Small Oil LLC.
      - Wolverine Construction
  - Public
- System

HTTPS / Basic Auth

RESTful / JSON

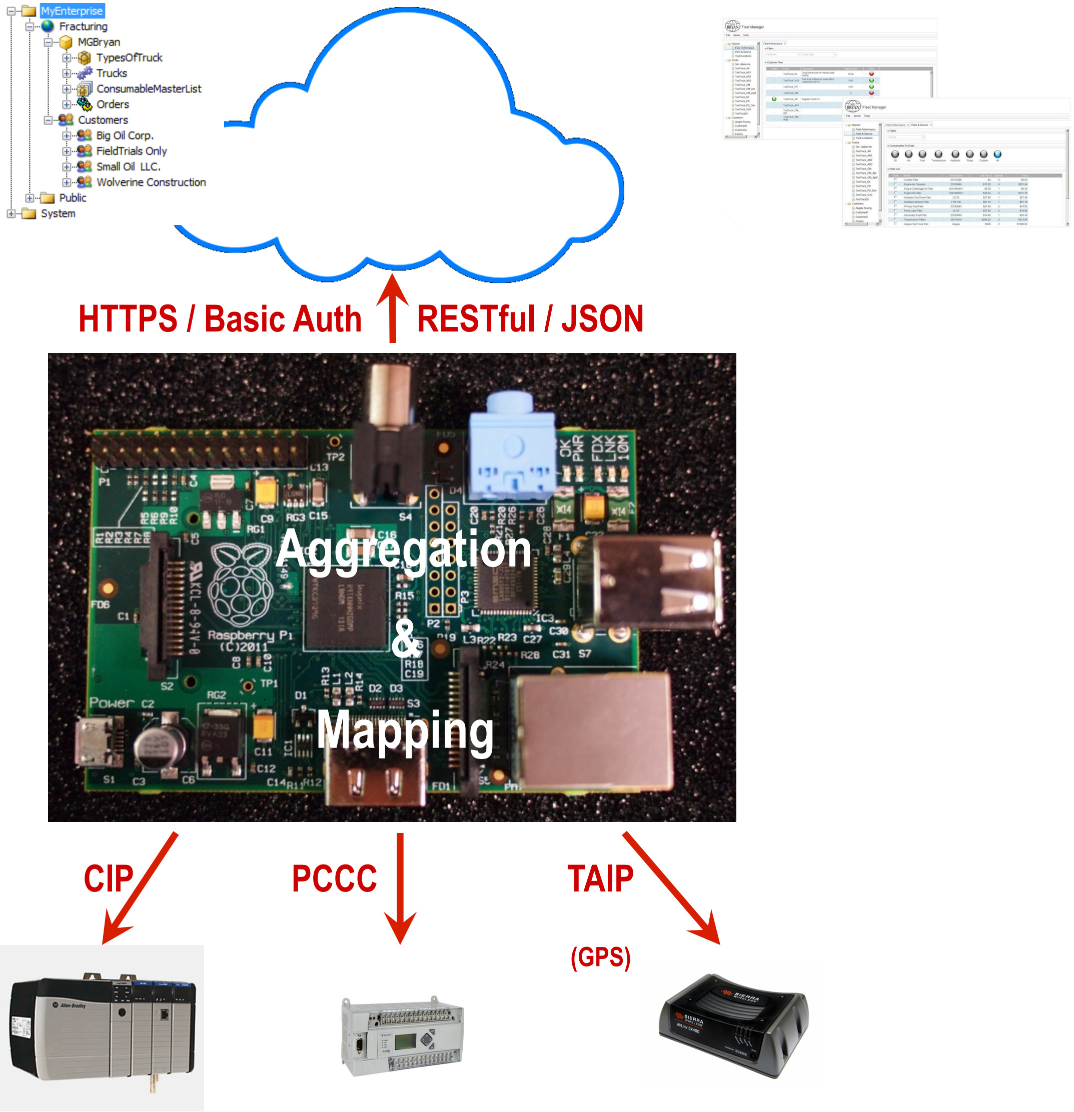
CIP

PCCC

TAIP

(GPS)

Cloud Gateway



- MyEnterprise
  - Fracturing
    - MGBryan
      - TypesOfTruck
      - Trucks
      - ConsumableMasterList
      - Orders
    - Customers
      - Big Oil Corp.
      - FieldTrials Only
      - Small Oil LLC.
      - Wolverine Construction
  - Public
- System

HTTPS / Basic Auth

RESTful / JSON

Aggregation

&

Mapping

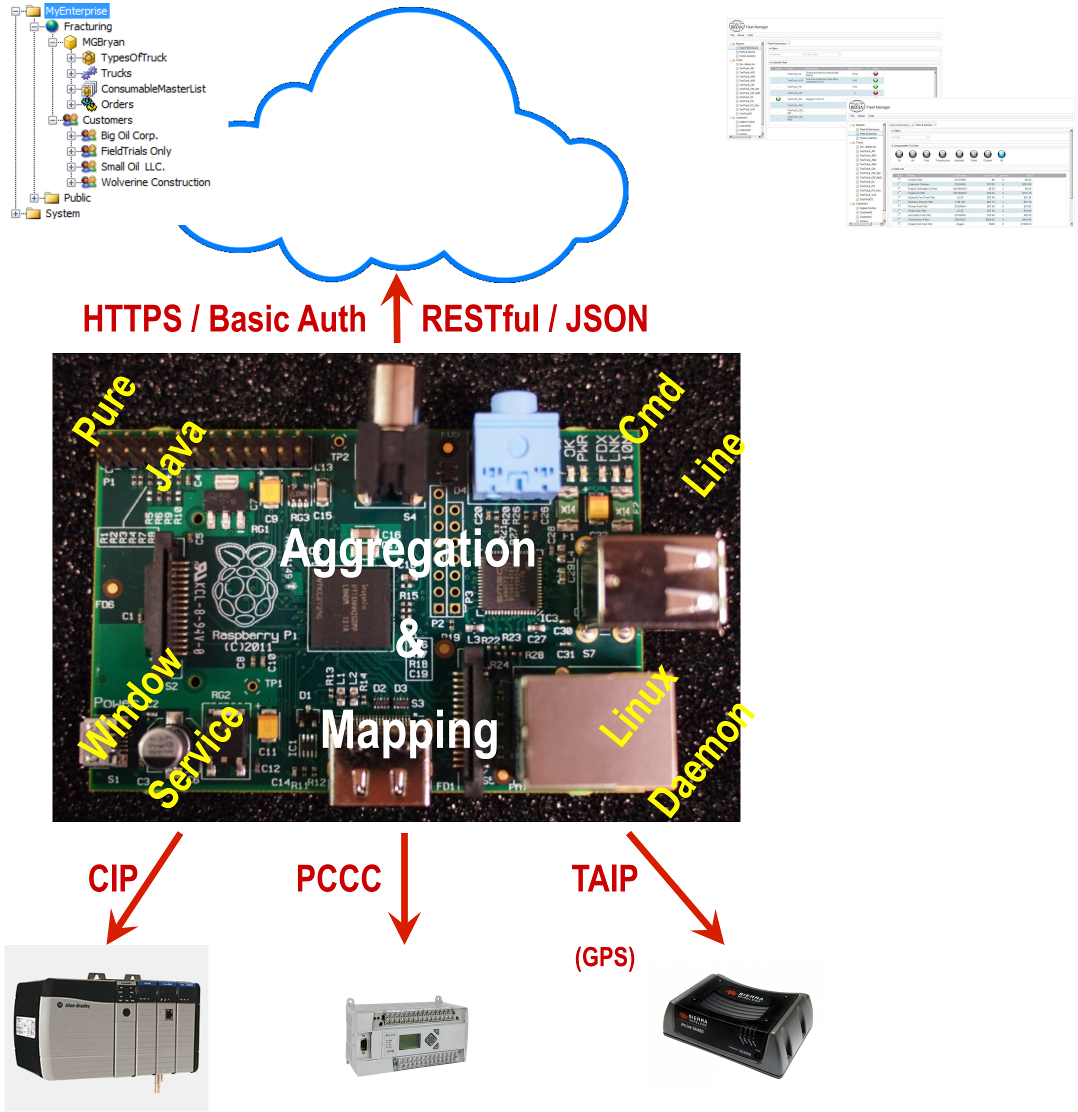
CIP

PCCC

TAIP

(GPS)

Cloud Gateway



- MyEnterprise
  - Fracturing
    - MGBryan
      - TypesOfTruck
      - Trucks
      - ConsumableMasterList
      - Orders
    - Customers
      - Big Oil Corp.
      - FieldTrials Only
      - Small Oil LLC.
      - Wolverine Construction
  - Public
- System

HTTPS / Basic Auth RESTful / JSON

Pure Java

Cmd Line

Aggregation & Mapping

Window Service

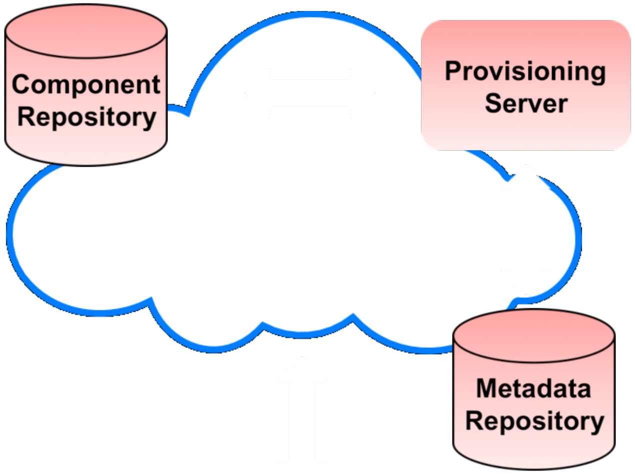
Linux Daemon

CIP

PCCC

TAIP (GPS)

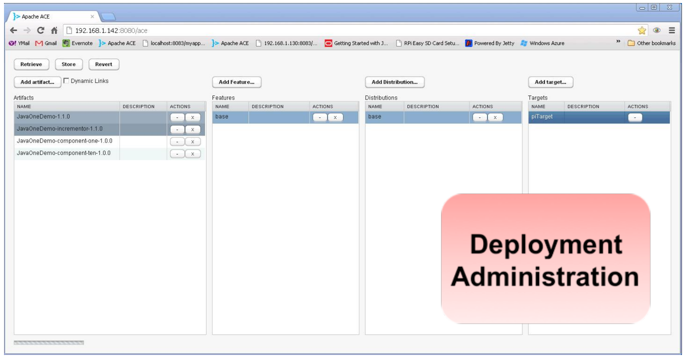
Provisioning the Cloud Gateway



Component Repository

Provisioning Server

Metadata Repository



Artifacts

NAME	DESCRIPTION	ACTIONS
JavaOneDemo-1.1.0		- x
JavaOneDemo-incrementor-1.1.0		- x
JavaOneDemo-component-one-1.0.0		- x
JavaOneDemo-component-ten-1.0.0		- x

Features

NAME	DESCRIPTION	ACTIONS
base		- x

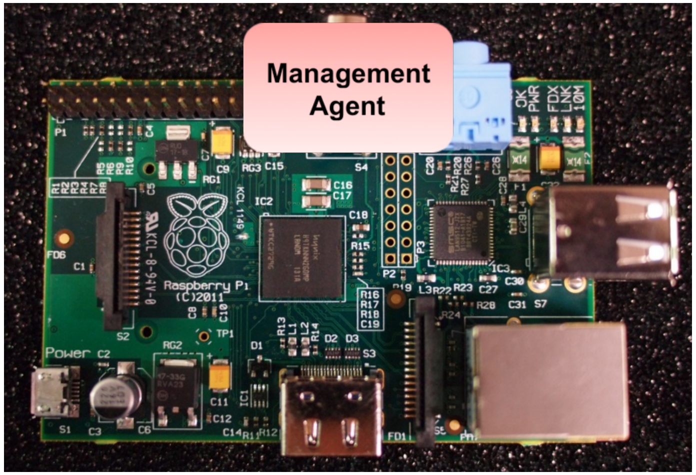
Distributions

NAME	DESCRIPTION	ACTIONS
base		- x

Targets

NAME	DESCRIPTION	ACTIONS
piTarget		-

Deployment Administration



Management Agent

Apache Ace

- Management Agent
  - Identification, Discovery, Scheduler, Deployment, Audit Logs
- Deployment Administration
  - Versioned, Transactional, Deltas, Digital Signatures, Extensible
- Provisioning Server
  - Maps components to targets
  - Only stores metadata
- Component Repository
  - OBR, Maven, anything reachable by URL
  - Can be deployed separately from the server
  - May be replicated

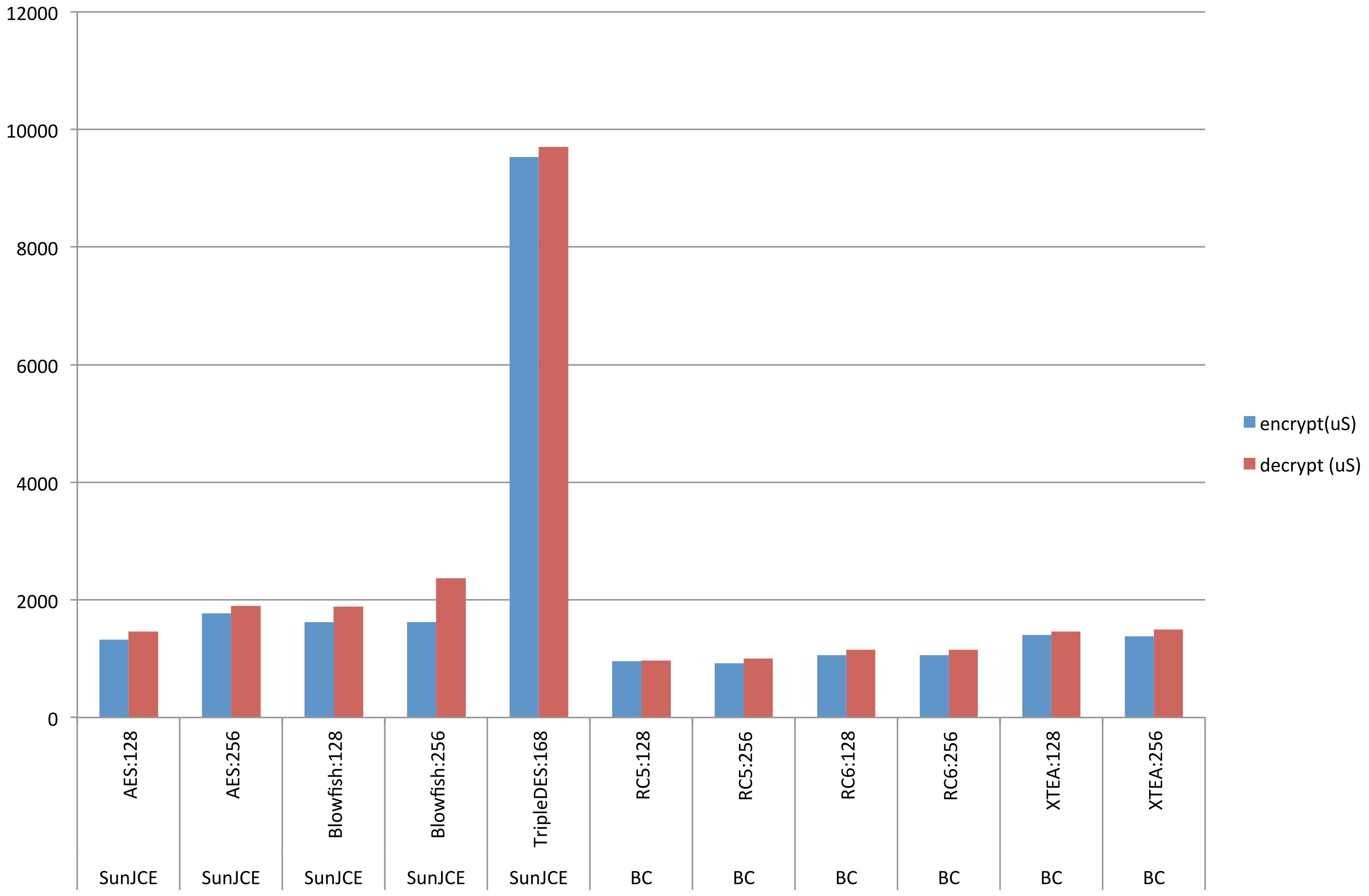
Challenges - Performance

Startup Time

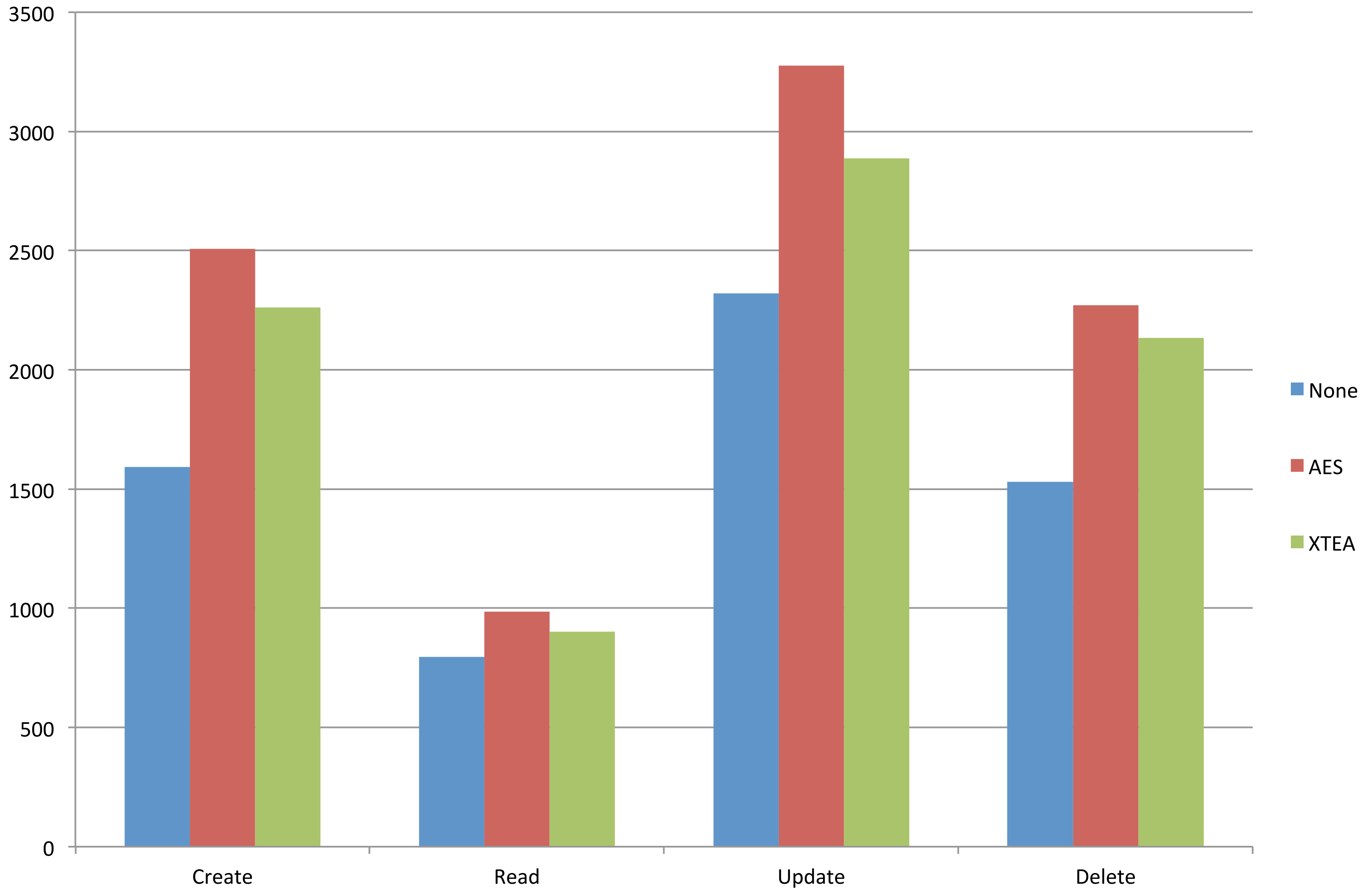
- How to speed up OSGi initialization?
    - Use bundle cache
    - Deferred startup
    - Concurrent startup: bundle activator threads
    - Strict bundle loading faster than dynamic
- Jar consolidation

- Runtime
    - Deterministic Response Time
        - Impeded by periodic processing or DB transaction log
        - Impacted by various DB housekeeping chores
    - Limit use of JNI
- Managing Memory
    - Consider more aggressively releasing heap to OS

Java Cipher Performance on ARM



H2 Encryption Performance



Challenges – Flash Memory

- Automation systems are long lived
- Flash memory wears out: limited # of write/erase cycles
- Write cycles must be considered
- Database
  - updates
  - index generation
  - transaction logs
  - Housekeeping
- Swap
- Temp file systems
- Trace / log data
- Diagnostics

Challenges - Hardening

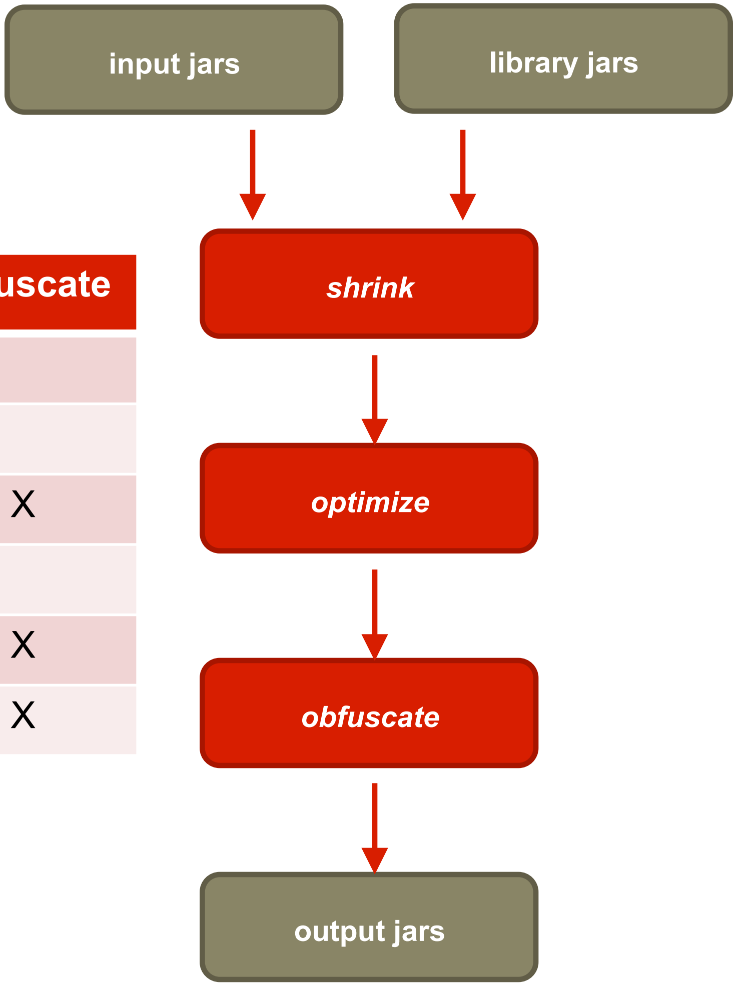
- How to protect Intellectual Property
- How to make it tamperproof
- How to handle secrets?
  - Device must have access to keys when disconnected
  - Hide in filesystem
  - Hide in code
  - Encrypt on dongle
- Build time techniques
  - Identify critical components
  - Use “Clean room” development
  - Code signing, encryption
  - Obfuscation
    - Challenge for debugging

Code Obfuscation

- Desire to protect code and intellectual property
- Approaches
  - Layout, Data, Control, Encryption
- Consequences
  - Difficult to debug
  - May execute more slowly, particularly with encryption
  - Bytecode manipulation may introduce unintended behavior, as when a dynamically loaded a class
  - Engineering work must be performed to determine which classes must be excluded from obfuscation.
  - Additional build steps may be required to process code.
- ProGuard
  - Flexible, open source obfuscator
  - Also optimizes execution, shrinks jar footprint ~ 1/3
  - Retracing file allows stack traces to be reconstructed

ProGuard jar sizing

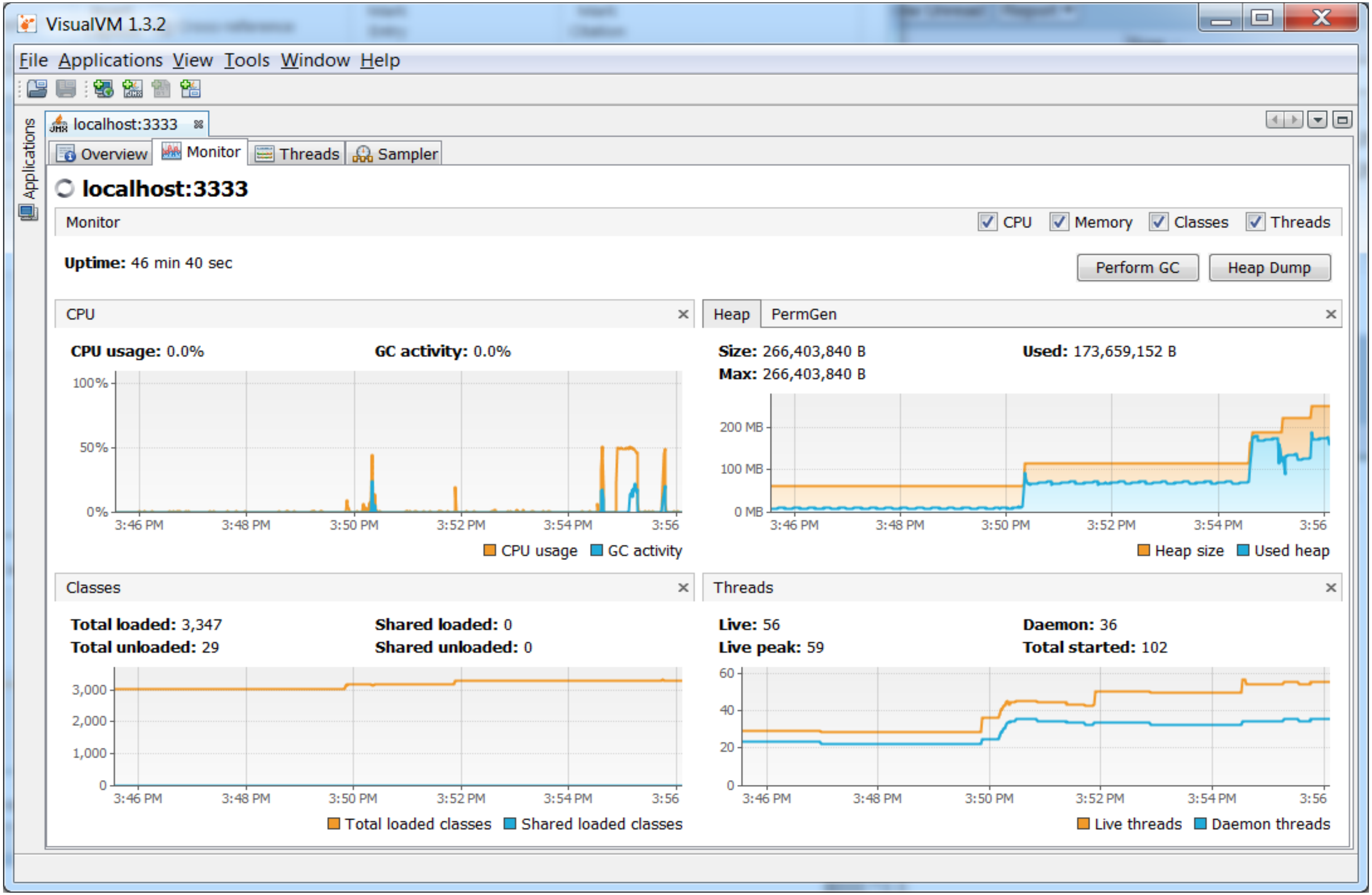
reduction (%)	shrink	optimize	obfuscate
8.8	X		
0*		X	
29.6			X
11.7	X	X	
34.9	X		X
36.2	X	X	X



Challenges - Troubleshooting

- Embedded Java provides a JRE not a JDK
- Typical JDK diagnostic tools are unavailable on the target
  - jmap, jps, jstat, etc.
- The JRE can support remote connection via JMX protocol
- Attach VisualVM from remote workstation
  - Note: run with same major Java version and arch
  - Monitor memory & CPU
  - Generate and analyze heapdumps
  - Profiling (sampling)

VisualVM Monitoring



localhost:3333

Uptime: 46 min 40 sec

CPU usage: 0.0%

GC activity: 0.0%

Size: 266,403,840 B

Used: 173,659,152 B

Max: 266,403,840 B

Total loaded: 3,347

Shared loaded: 0

Total unloaded: 29

Shared unloaded: 0

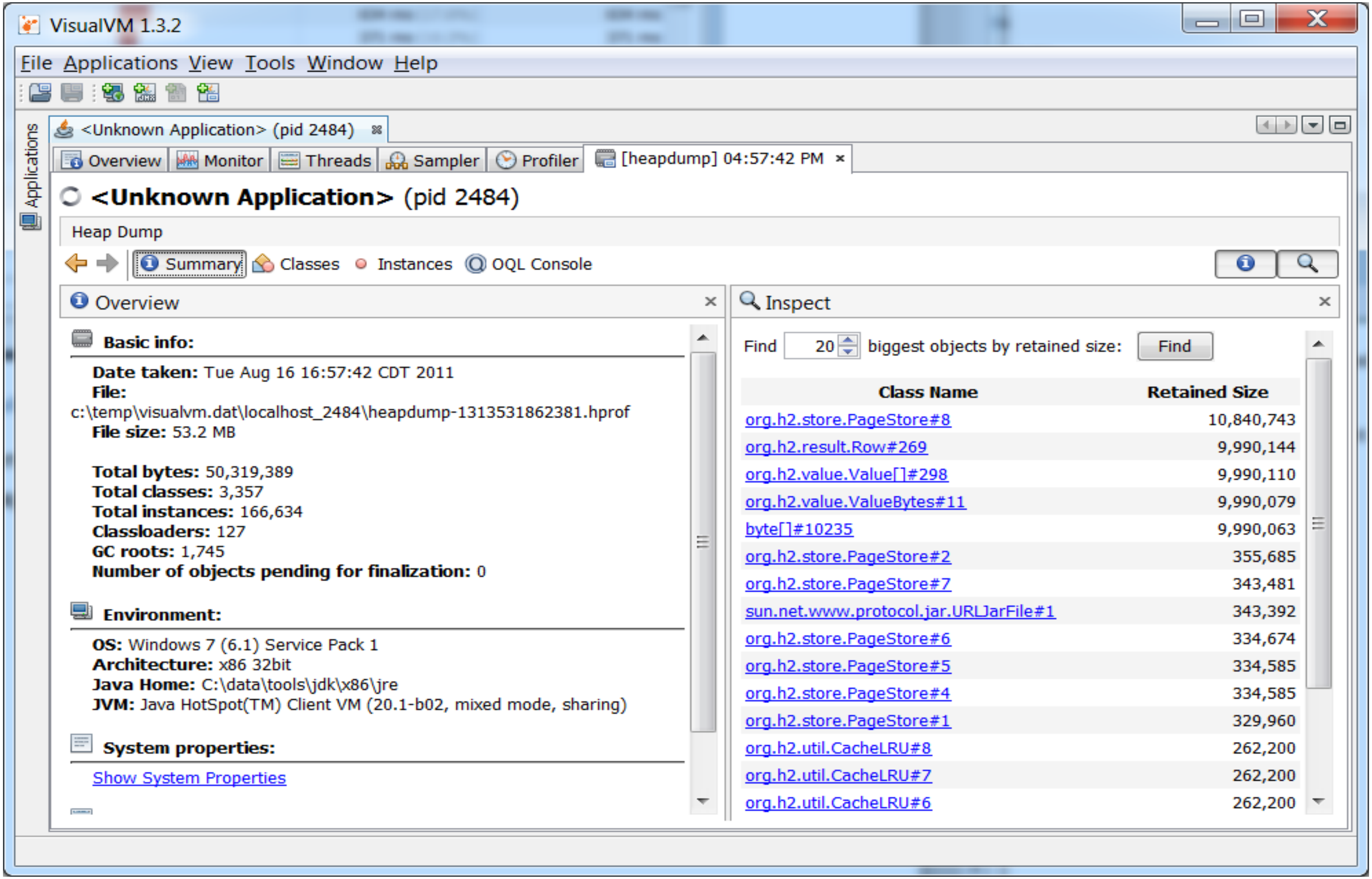
Live: 56

Daemon: 36

Live peak: 59

Total started: 102

VisualVM Heap Analysis



<Unknown Application> (pid 2484)

Basic info:

Date taken: Tue Aug 16 16:57:42 CDT 2011
File:
c:\temp\visualvm.dat\localhost_2484\heapdump-1313531862381.hprof
File size: 53.2 MB

Total bytes: 50,319,389
Total classes: 3,357
Total instances: 166,634
Classloaders: 127
GC roots: 1,745
Number of objects pending for finalization: 0

Environment:

OS: Windows 7 (6.1) Service Pack 1
Architecture: x86 32bit
Java Home: C:\data\tools\jdk\x86\jre
JVM: Java HotSpot(TM) Client VM (20.1-b02, mixed mode, sharing)

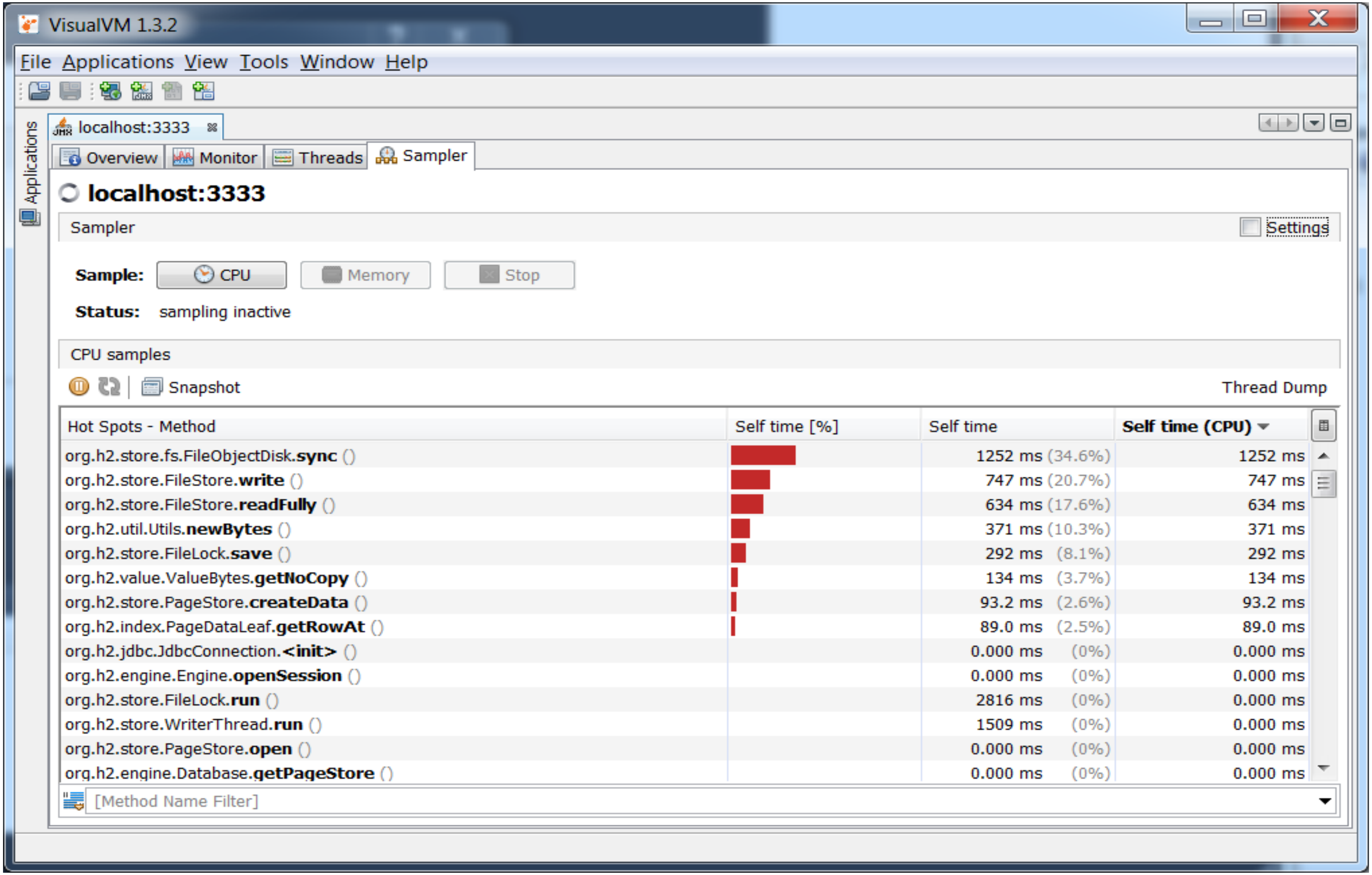
System properties:

Show System Properties

Find 20 biggest objects by retained size: Find

Class Name	Retained Size
org.h2.store.PageStore#8	10,840,743
org.h2.result.Row#269	9,990,144
org.h2.value.Value[]#298	9,990,110
org.h2.value.ValueBytes#11	9,990,079
byte[]#10235	9,990,063
org.h2.store.PageStore#2	355,685
org.h2.store.PageStore#7	343,481
sun.net.www.protocol.jar.URLJarFile#1	343,392
org.h2.store.PageStore#6	334,674
org.h2.store.PageStore#5	334,585
org.h2.store.PageStore#4	334,585
org.h2.store.PageStore#1	329,960
org.h2.util.CacheLRU#8	262,200
org.h2.util.CacheLRU#7	262,200
org.h2.util.CacheLRU#6	262,200

VisualVM Profiling



localhost:3333

Sampler

Sample: CPU Memory Stop

Status: sampling inactive

CPU samples

Hot Spots - Method	Self time [%]	Self time	Self time (CPU)
org.h2.store.fs.FileObjectDisk.**sync** ()		1252 ms (34.6%)	1252 ms
org.h2.store.FileStore.**write** ()		747 ms (20.7%)	747 ms
org.h2.store.FileStore.**readFully** ()		634 ms (17.6%)	634 ms
org.h2.util.Utils.**newBytes** ()		371 ms (10.3%)	371 ms
org.h2.store.FileLock.**save** ()		292 ms (8.1%)	292 ms
org.h2.value.ValueBytes.**getNoCopy** ()		134 ms (3.7%)	134 ms
org.h2.store.PageStore.**createData** ()		93.2 ms (2.6%)	93.2 ms
org.h2.index.PageDataLeaf.**getRowAt** ()		89.0 ms (2.5%)	89.0 ms
org.h2.jdbc.JdbcConnection.**<init>** ()		0.000 ms (0%)	0.000 ms
org.h2.engine.Engine.**openSession** ()		0.000 ms (0%)	0.000 ms
org.h2.store.FileLock.**run** ()		2816 ms (0%)	0.000 ms
org.h2.store.WriterThread.**run** ()		1509 ms (0%)	0.000 ms
org.h2.store.PageStore.**open** ()		0.000 ms (0%)	0.000 ms
org.h2.engine.Database.**getPageStore** ()		0.000 ms (0%)	0.000 ms

More Troubleshooting

- Development
  - Monitoring: memory leaks, GC
  - Sampling / Profiling: hotspots, latencies
- Production
  - Postmortem: system crashes, hangs
- Lots of troubleshooting tools available
  - JVM cmd line options like -XX:+HeapDumpOnOutOfMemoryError
  - JDK tools like *hprof*, *jmap, jstack* and *jhat* (not on Embedded JRE!)
  - JConsole and VisualVM
  - Native tools like *dtrace* (Solaris) and *strace* (Linux)
- Troubleshooting Guide for Java SE 6
  - http://www.oracle.com/technetwork/java/javase/tsg-vm-149989.pdf

Troubleshooting Scenarios

Memory Utilization	• *-verbose:class* displays classes loaded • *-XX:+HeapDumpOnOutOfMemoryError* generates a heap dump on OOME • stack dump summarizes heap memory usage • heap analysis shows detailed memory usage, potential leaks • monitoring exhibits runtime memory allocation and GC behaviors
Garbage Collection	• *-verbose:gc* displays GC statistics • monitoring exhibits runtime memory allocation and GC behaviors • heap analysis shows large instances counts and GC roots
Poor Performance	• stack dump summarizes memory usage, indicating potential memory utilization issues • monitoring exhibits runtime CPU usage, thread count & activity • monitoring exhibits runtime memory allocation and GC behaviors • stack dump indicates deadlocks
Hanging / Looping	• stack dump indicates busy threads, deadlocks (may need to force dump) • monitoring exhibits runtime CPU usage, thread count & activity
System Crash	• examine application and Fatal Error logs for clues • look in filesystem working dir for stack/heap dumps • restart JVM with monitoring enabled • restart JVM with *-XX:+HeapDumpOnOutOfMemoryError* option • restart JVM with *-XX:OnError* executing script to assist w/ debugging

Q & A