java ee
and
microservices

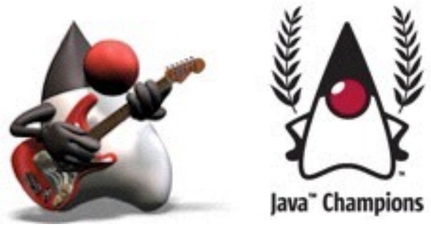
adam-bien.com

I'm working as Java EE/SE developer, consultant, sometimes author, speaker and trainer with Java since 1995...

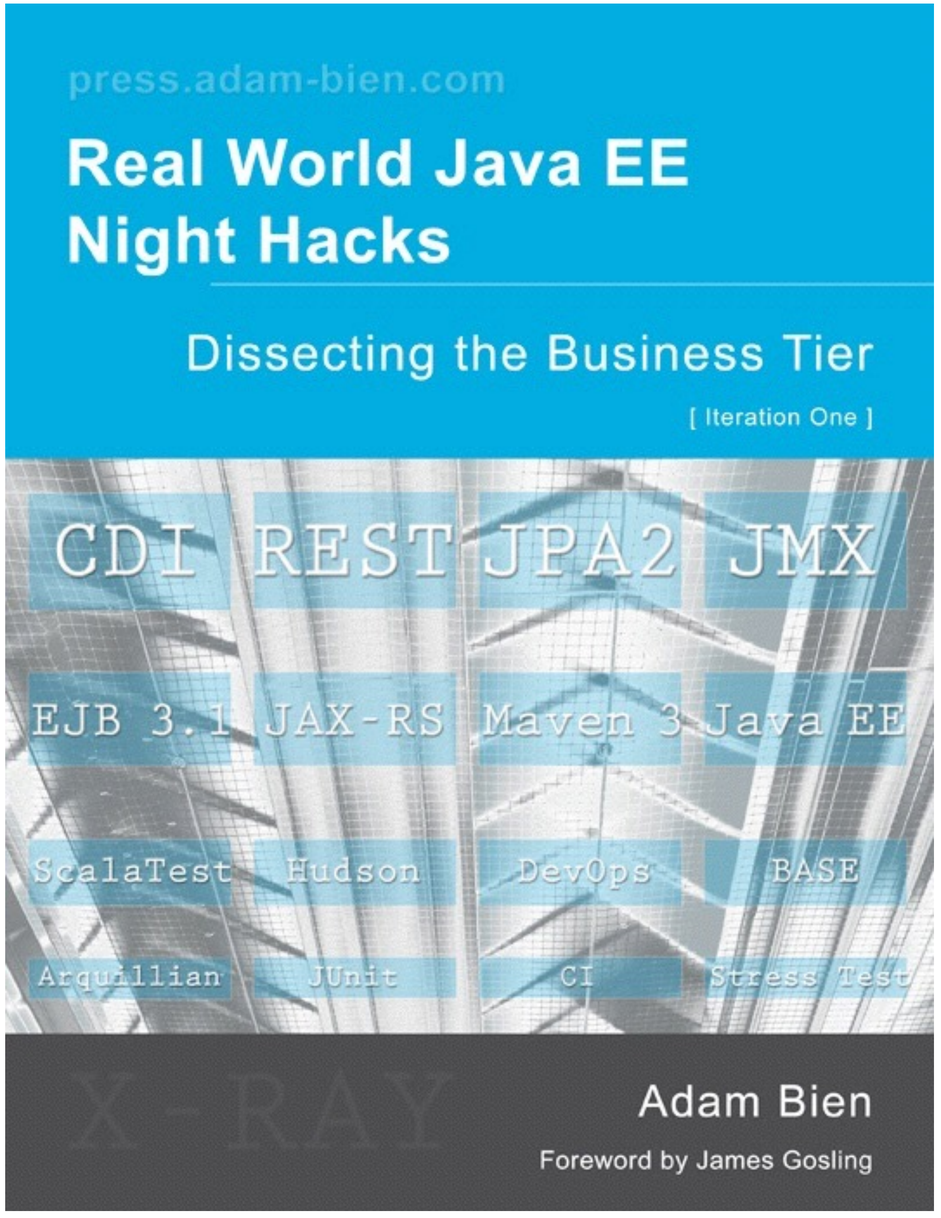
...and still really enjoying it!

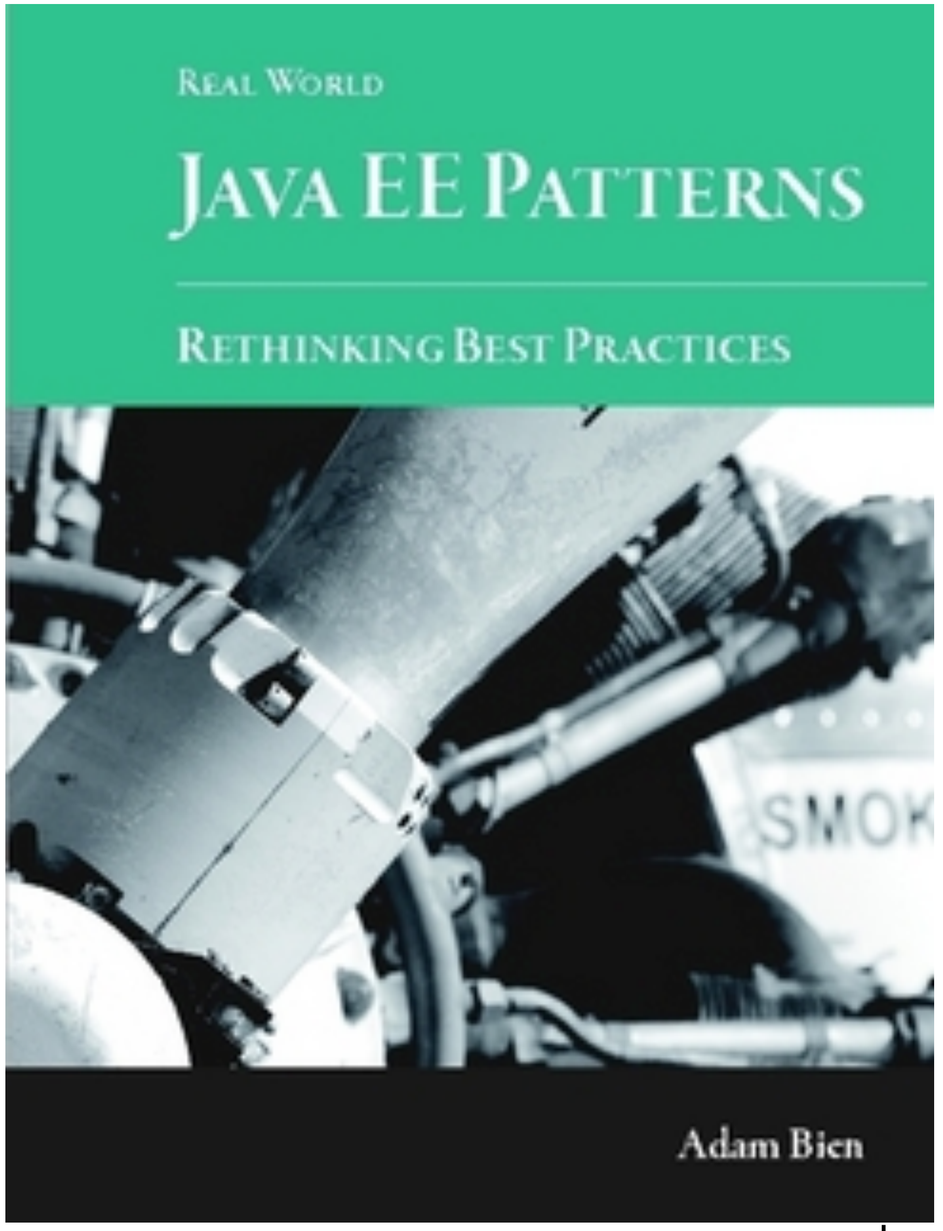
Java Programming Language rocks!

workshops.adam-bien.com



adam-bien.com

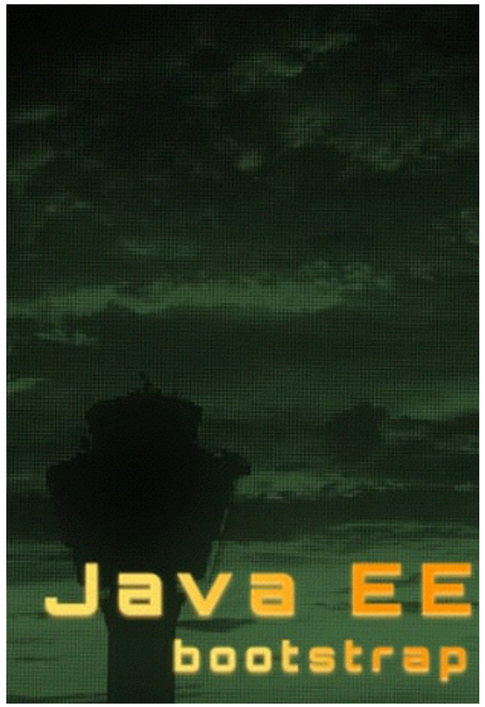


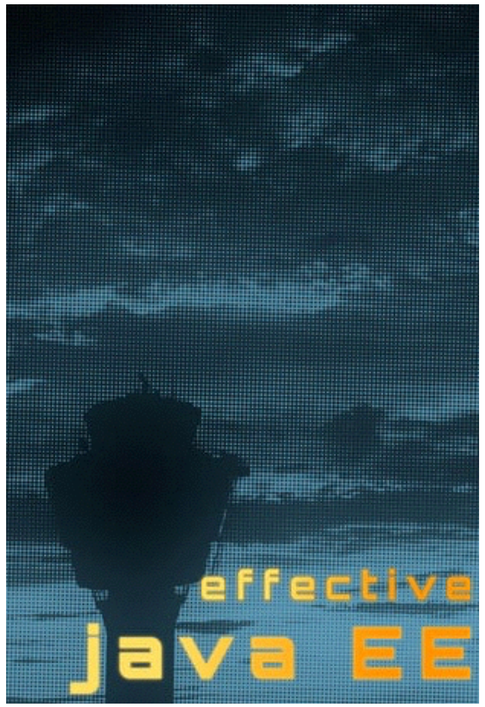


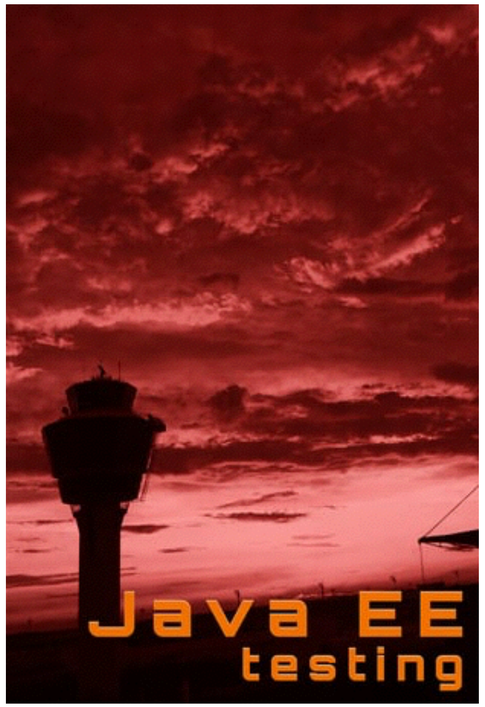
adam-bien.com

airhacks.TV

adam-bien.com


Java EE
bootstrap


effective
java EE


Java EE
testing


Java EE
microservices

airhacks.io

Forgotten Wisdom

1 The network is reliable.

2 Latency is zero.

3 Bandwidth is infinite.

4 The network is secure.

5 Topology doesn't change.

6 There is one administrator.

7 Transport cost is zero.

8 The network is homogeneous.

http://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

“We argue that objects that interact in a distributed system need to be dealt with in ways that are intrinsically different from objects that interact in a single address space. These differences are required because distributed systems require that the programmer be aware of latency, have a different model of memory access, and take into account issues of concurrency and partial failure. We look at a number of distributed systems that have attempted to paper over the distinction between local and remote objects, and show that such systems fail to support basic requirements of robustness and reliability. These failures have been masked in the past by the small size of the distributed systems that have been built. In the enterprise-wide distributed systems foreseen in the near future, however, such a masking will be impossible…”

A Note on Distributed Computing (1994), Jim Waldo

Idempotence

Is the property of certain operations in mathematics and computer science, that can be applied multiple times without changing the result beyond the initial application. (…)

The request methods of the Hypertext Transfer Protocol (HTTP) computer protocol are a common example of idempotence, in that data retrieval operations can be performed without changing or otherwise affecting the data.

Understanding CAP in Java EE context

CAP vs. FLP (Fischer, Lynch and Paterson)

In an asynchronous network where messages may be delayed but not lost, there is no consensus algorithm that is guaranteed to terminate in every execution for all starting conditions, if at least one node may fail-stop.

Don’t Distribute

Monoliths vs. Microservices

Self-Contained Services

Old Java World

Producer Consumer Problem

In computing, the producer–consumer problem[1][2] (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate a piece of data, put it into the buffer and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer) one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

http://en.wikipedia.org/wiki/Producer–consumer_problem

Back Pressure

The term is also used analogously in the field of information technology to describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data; the transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information. It also refers to an algorithm for routing data according to congestion gradients.

http://en.wikipedia.org/wiki/Back_pressure

Poison Message

A poison message is one which cannot be processed by a receiving MDB application. If a poison message is encountered, the JMS MessageConsumer and ConnectionConsumer objects can requeue it.

Sometimes, a badly-formatted message arrives on a queue. In this context, badly-formatted means that the receiving application cannot process the message correctly. Such a message can cause the receiving application to fail and to back out this badly-formatted message. The message can then be repeatedly delivered to the input queue and repeatedly backed out by the application. These messages are known as poison messages. The JMS MessageConsumer object detects poison messages and reroutes them to an alternative destination.

http://www-01.ibm.com/support/knowledgecenter/#!/SSFKSJ_7.0.1/com.ibm.mq.csqzaw.doc/jm35205_.htm

Leasing

Leasing is a key concept in the JiniTM architecture; in general, Jini technology-enabled services (Jini services) grant access to a resource only for as long as the clients of those Jini services actively express interest in the resource being maintained. **This pattern is in contrast to many other systems, in which access to a resource is granted until the client explicitly releases the resource. Using a leasing model generally makes a distributed system more robust by allowing stale information and services to be cleaned up, but it also places additional requirements on clients and services.**

https://river.apache.org/doc/specs/html/lrs-spec.html

Smart Stubs

Smart stubs are direct replacements for the original stubs that perform the same function as the originals with additional behavior. They can be added and removed easily with no code changes within the client application or the server object. The additional behavior may be the timing of method calls, or caching of data so that each client request does not necessarily result in a costly invocation across the network.

http://www.javaworld.com/article/2076163/soa/the-smart-approach-to-distributed-performance-monitoring-with-java.html

Microservices Motivation

Motivation

- Faster Delivery
- Lifecycle Independency
- Loose Coupling
- Interoperability

Consequences

- Additional complexity
- Everything is distributed
- [Asynchronous communication]
- DRY is an anti-pattern
- Higher development costs
- Devops are necessary

More Developers,
Faster Development,
Substantially More Expensive

Microservices

Definition

"In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies."

http://martinfowler.com/articles/microservices.html

What would be the opposite?

Definition

Microservices is a software architecture design pattern, in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small, highly decoupled and focus on doing a small task.

http://en.wikipedia.org/wiki/Microservices

Hardware

Sun E10K

E10K Server - Enterprise 10000 Server - Sun Microsystems

Sun Enterprise E10000 Server. This is a complete server with all system boards, power supplies, fan trays etc. It was installed at AOL and was in good working order until it was deinstalled. The CPU's and memory have been removed, but this is a great machine for parts. If you are suporting E10K servers, this is a perfect spare.Click here for technical information on the Sun Fire E10K server

- 30-Day Warranty on all hardware
- Great for spare parts. Complete list of installed options is available upon request.
- Can ship today
- in stock. Call for availability on quantities over 1

Delta Marketing Group, Inc. since 1986

We would also be happy to take your order or answer any questions over the phone
480-367-6698

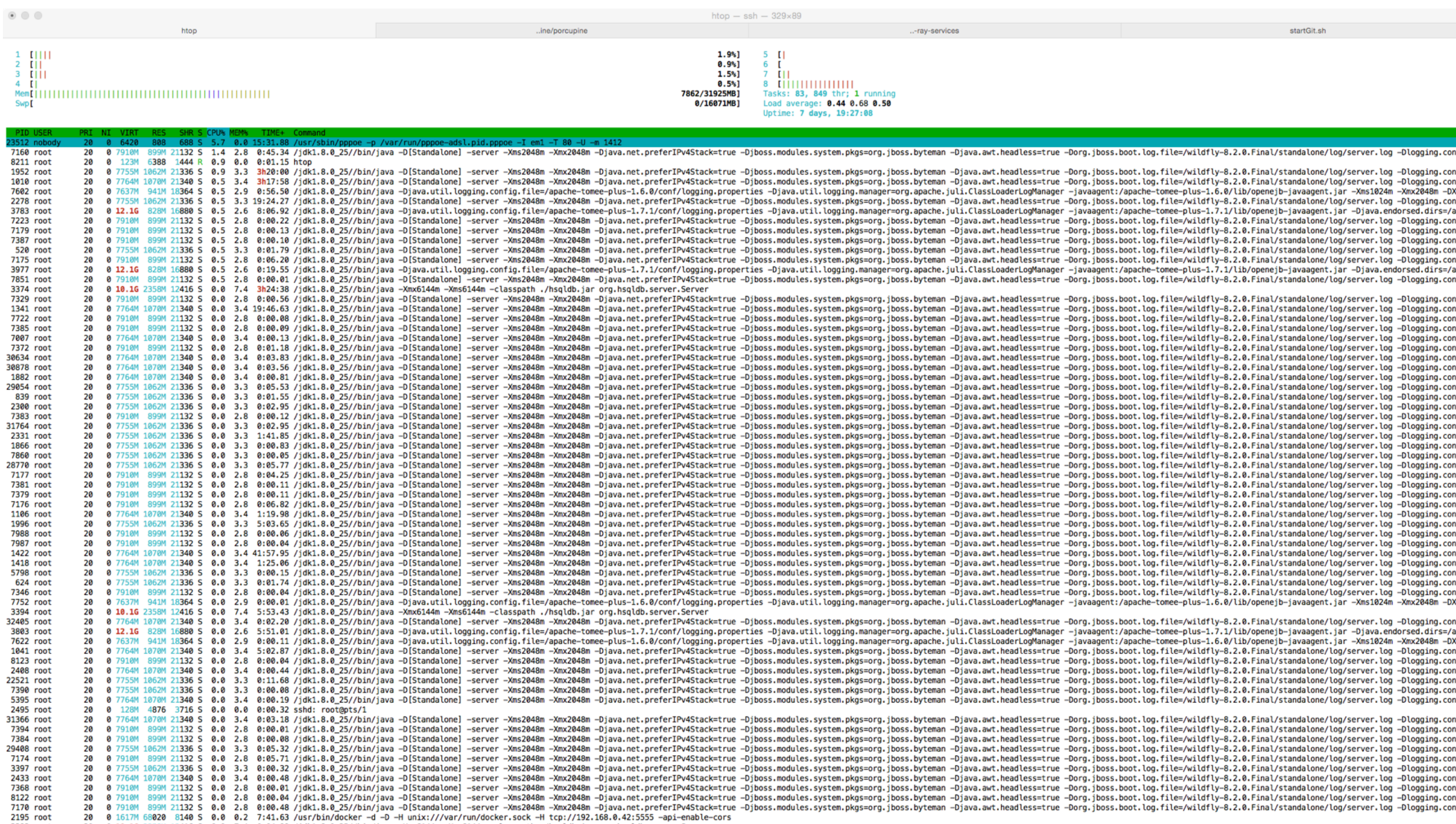
For Sales dial 1

or
Michael Hankerson
extension 101

Manufactured By: Sun Microsystems

Print Product Description | Email a Friend | Sign-up for Monthly Specials | Update My Profile

List Price:$1,200,000.00
Qty 1 qty:
Add To Cart
$1,995.00


htop — ssh — 329×89
htop
..ine/porcupine
..-ray-services
startGit.sh
1 [||||
2 [||
3 [|||
4 [|
Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Swp[
1.9%]
0.9%]
1.5%]
0.5%]
7862/31925MB]
0/16071MB]
5 [|
6 [
7 [||
8 [||||||||||||||||
Tasks: 83, 849 thr; 1 running
Load average: 0.44 0.68 0.50
Uptime: 7 days, 19:27:08
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
23512 nobody 20 0 6420 808 688 S 5.7 0.0 15:31.88 /usr/sbin/pppoe -p /var/run/pppoe-adsl.pid.pppoe -I em1 -T 80 -U -m 1412
7160 root 20 0 7910M 899M 21132 S 1.4 2.8 0:45.34 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
8211 root 20 0 123M 6388 1444 R 0.9 0.0 0:01.15 htop
1952 root 20 0 7755M 1062M 21336 S 0.9 3.3 3h20:00 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1010 root 20 0 7764M 1070M 21340 S 0.5 3.4 3h17:58 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7602 root 20 0 7637M 941M 18364 S 0.5 2.9 0:56.50 /jdk1.8.0_25//bin/java -Djava.util.logging.config.file=/apache-tomee-plus-1.6.0/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -javaagent:/apache-tomee-plus-1.6.0/lib/openejb-javaagent.jar -Xms1024m -Xmx2048m -DX
2278 root 20 0 7755M 1062M 21336 S 0.5 3.3 19:24.27 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
3783 root 20 0 12.1G 828M 16880 S 0.5 2.6 8:06.92 /jdk1.8.0_25//bin/java -Djava.util.logging.config.file=/apache-tomee-plus-1.7.1/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -javaagent:/apache-tomee-plus-1.7.1/lib/openejb-javaagent.jar -Djava.endorsed.dirs=/a
7223 root 20 0 7910M 899M 21132 S 0.5 2.8 0:00.22 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7179 root 20 0 7910M 899M 21132 S 0.5 2.8 0:00.13 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7387 root 20 0 7910M 899M 21132 S 0.5 2.8 0:00.10 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
520 root 20 0 7755M 1062M 21336 S 0.5 3.3 0:01.79 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7175 root 20 0 7910M 899M 21132 S 0.5 2.8 0:06.20 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
3977 root 20 0 12.1G 828M 16880 S 0.5 2.6 0:19.55 /jdk1.8.0_25//bin/java -Djava.util.logging.config.file=/apache-tomee-plus-1.7.1/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -javaagent:/apache-tomee-plus-1.7.1/lib/openejb-javaagent.jar -Djava.endorsed.dirs=/a
7851 root 20 0 7910M 899M 21132 S 0.5 2.8 0:00.01 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
3374 root 20 0 10.1G 2358M 12416 S 0.0 7.4 3h24:38 /jdk1.8.0_25//bin/java -Xmx6144m -Xms6144m -classpath ./hsqldb.jar org.hsqldb.server.Server
7329 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.56 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1341 root 20 0 7764M 1070M 21340 S 0.0 3.4 19:46.63 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7722 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.08 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7385 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.09 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7007 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:00.13 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7372 root 20 0 7910M 899M 21132 S 0.0 2.8 0:01.18 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
30634 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:03.83 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
30878 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:03.56 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1882 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:00.81 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
29054 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:05.53 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
839 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:01.55 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
2300 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:02.95 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7383 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.12 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
31764 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:02.95 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
2331 root 20 0 7755M 1062M 21336 S 0.0 3.3 1:41.85 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1866 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:00.83 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7860 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:00.05 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
28770 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:05.77 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7177 root 20 0 7910M 899M 21132 S 0.0 2.8 0:04.25 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7381 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.11 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7379 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.11 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7176 root 20 0 7910M 899M 21132 S 0.0 2.8 0:06.82 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1106 root 20 0 7764M 1070M 21340 S 0.0 3.4 1:19.98 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1996 root 20 0 7755M 1062M 21336 S 0.0 3.3 5:03.65 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7988 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.06 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7987 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.04 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1422 root 20 0 7764M 1070M 21340 S 0.0 3.4 41:57.95 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
1418 root 20 0 7764M 1070M 21340 S 0.0 3.4 1:25.06 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
5798 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:00.15 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
624 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:01.74 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7346 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.04 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7752 root 20 0 7637M 941M 18364 S 0.0 2.9 0:00.01 /jdk1.8.0_25//bin/java -Djava.util.logging.config.file=/apache-tomee-plus-1.6.0/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -javaagent:/apache-tomee-plus-1.6.0/lib/openejb-javaagent.jar -Xms1024m -Xmx2048m -DX
3394 root 20 0 10.1G 2358M 12416 S 0.0 7.4 5:53.43 /jdk1.8.0_25//bin/java -Xmx6144m -Xms6144m -classpath ./hsqldb.jar org.hsqldb.server.Server
32405 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:02.20 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
3803 root 20 0 12.1G 828M 16880 S 0.0 2.6 5:51.01 /jdk1.8.0_25//bin/java -Djava.util.logging.config.file=/apache-tomee-plus-1.7.1/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -javaagent:/apache-tomee-plus-1.7.1/lib/openejb-javaagent.jar -Djava.endorsed.dirs=/a
7622 root 20 0 7637M 941M 18364 S 0.0 2.9 0:00.11 /jdk1.8.0_25//bin/java -Djava.util.logging.config.file=/apache-tomee-plus-1.6.0/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -javaagent:/apache-tomee-plus-1.6.0/lib/openejb-javaagent.jar -Xms1024m -Xmx2048m -DX
1041 root 20 0 7764M 1070M 21340 S 0.0 3.4 5:02.87 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
8123 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.04 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
2408 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:00.44 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
22521 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:11.68 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7390 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:00.08 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
5395 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:00.19 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
2495 root 20 0 128M 4876 3716 S 0.0 0.0 0:00.32 sshd: root@pts/1
31366 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:03.18 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7394 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.01 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7384 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.08 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
29408 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:05.32 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7174 root 20 0 7910M 899M 21132 S 0.0 2.8 0:05.71 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
3397 root 20 0 7755M 1062M 21336 S 0.0 3.3 0:00.32 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
2433 root 20 0 7764M 1070M 21340 S 0.0 3.4 0:00.48 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7368 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.01 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
8122 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.04 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
7170 root 20 0 7910M 899M 21132 S 0.0 2.8 0:00.48 /jdk1.8.0_25//bin/java -D[Standalone] -server -Xms2048m -Xmx2048m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Dorg.jboss.boot.log.file=/wildfly-8.2.0.Final/standalone/log/server.log -Dlogging.con
2195 root 20 0 1617M 68020 8140 S 0.0 0.2 7:41.63 /usr/bin/docker -d -D -H unix:///var/run/docker.sock -H tcp://192.168.0.42:5555 -api-enable-cors

Most cited example

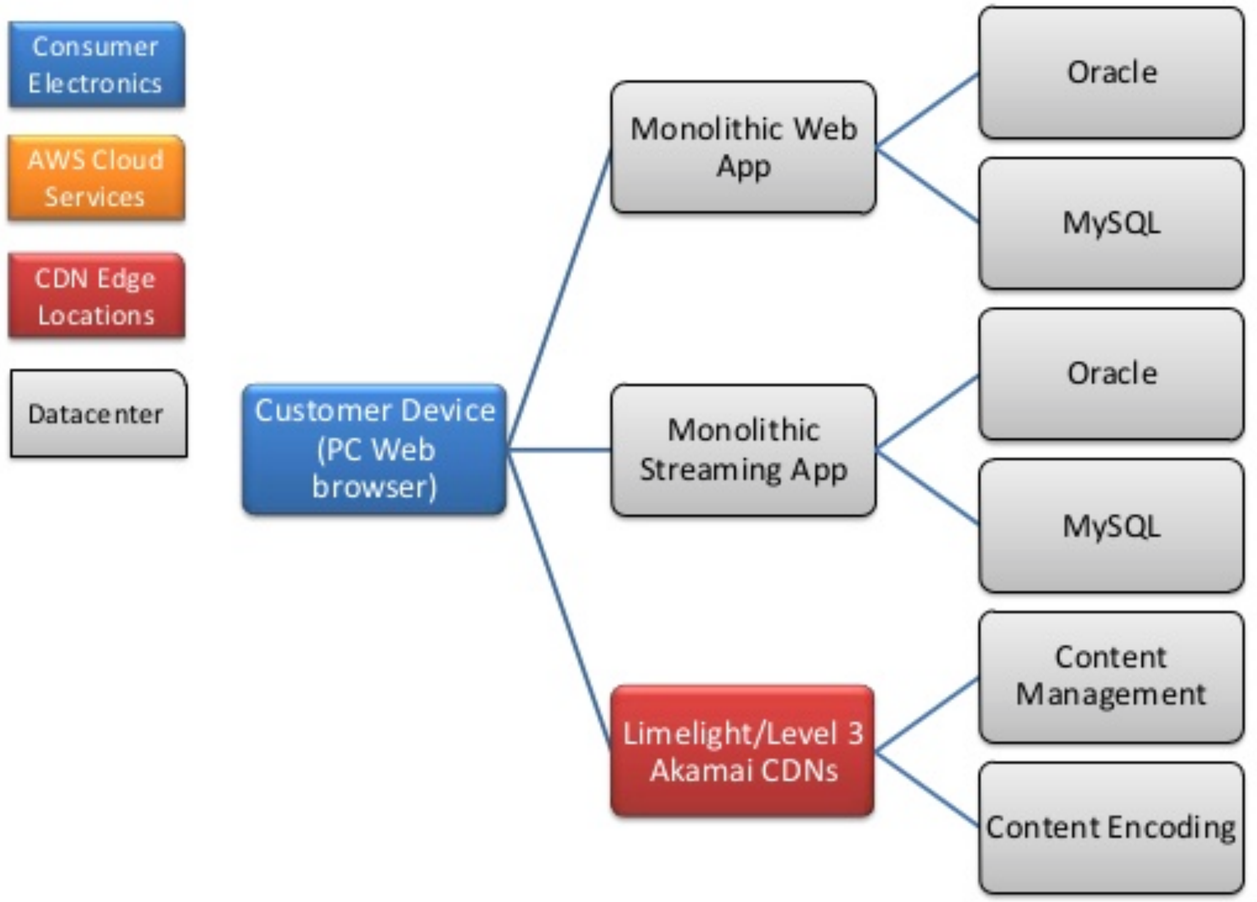
Who are the 6 million people still getting Netflix by mail? I'm one of them

Sure, the streaming revolution has come. But that doesn't mean there's no upside to the postie handing you a movie

Netflix DVD packets, ready to be mailed out to 6.26 million diehards. Photograph: Justin Sullivan/Getty

In 2014, Netflix spent $0 on marketing its DVD business. That's down from 2013, when it spent a whopping $292,000 on DVD marketing. For comparison, the firm's marketing for its streaming service cost $65m last quarter alone.

How Netflix Used to Work



http://de.slideshare.net/adrianco/yowworkshop-131203193626phpapp01-1

Netflix Scale

- Tens of thousands of instances on AWS
  - Typically 4 core, 30GByte, Java business logic
  - Thousands created/removed every day
- Thousands of Cassandra NoSQL nodes on AWS
  - Many hi1.4xl - 8 core, 60Gbyte, 2TByte of SSD
  - 65 different clusters, over 300TB data, triple zone
  - Over 40 are multi-region clusters (6, 9 or 12 zone)
  - Biggest 288 m2.4xl – over 300K rps, 1.3M wps

http://de.slideshare.net/adrianco/yowworkshop-131203193626phpapp01-1

Docker

Default Java EE Architecture

Some Facts

No Interfaces, No XML

95% Business Code

Few annotations

Just POJOs

Straightforward testing

One team, one WAR, one standalone server

More teams, multiple WARs, multiple servers

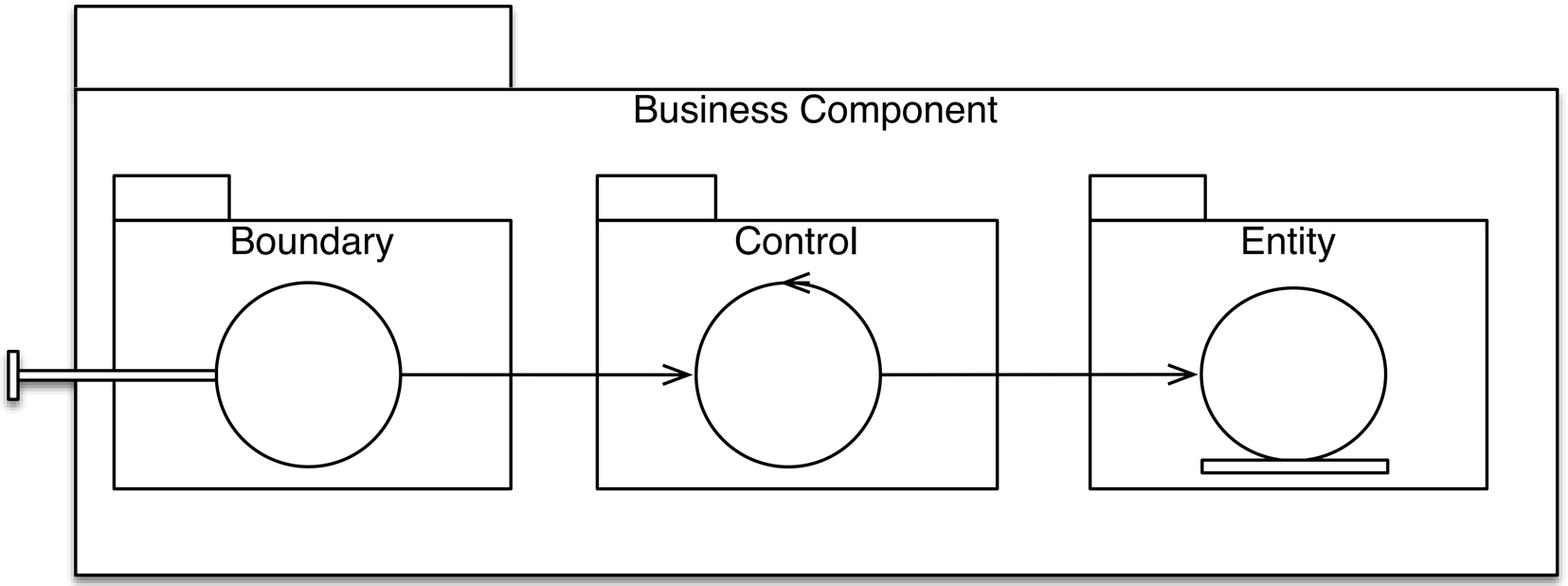
One team with specific requirements -> multiple WARs, multiple servers.

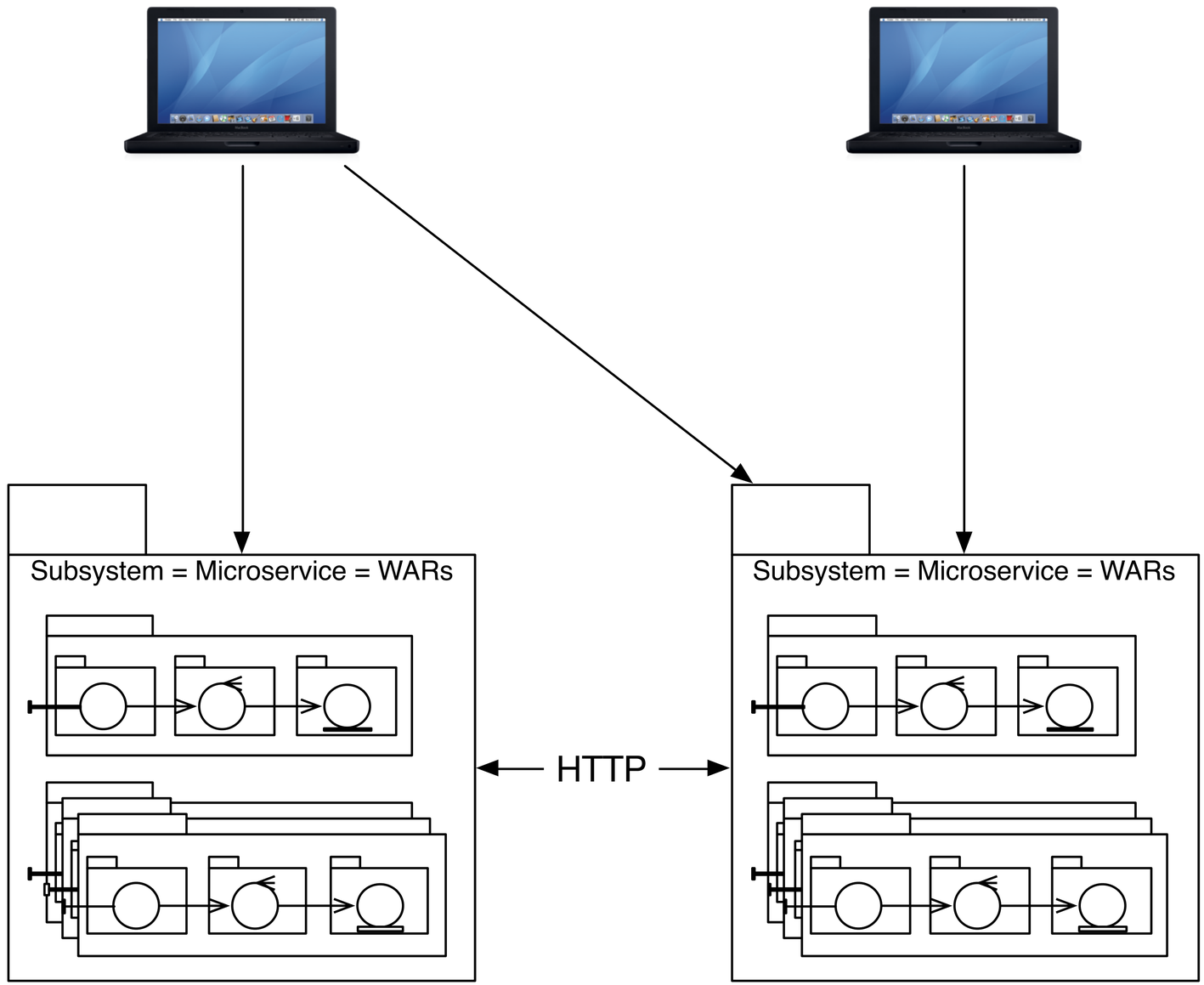
“Don’t make me think”

BCE / ECB

Conventions are paramount, configuration is optional

Business Driven Design


Business Component
Boundary
Control
Entity


Subsystem = Microservice = WARs
HTTP
Subsystem = Microservice = WARs

The Default Java EE Deployment

Self-contained WARs

One OS, One Server, One War

No Dependencies

No JARs

Installation

Download size: ~100MB

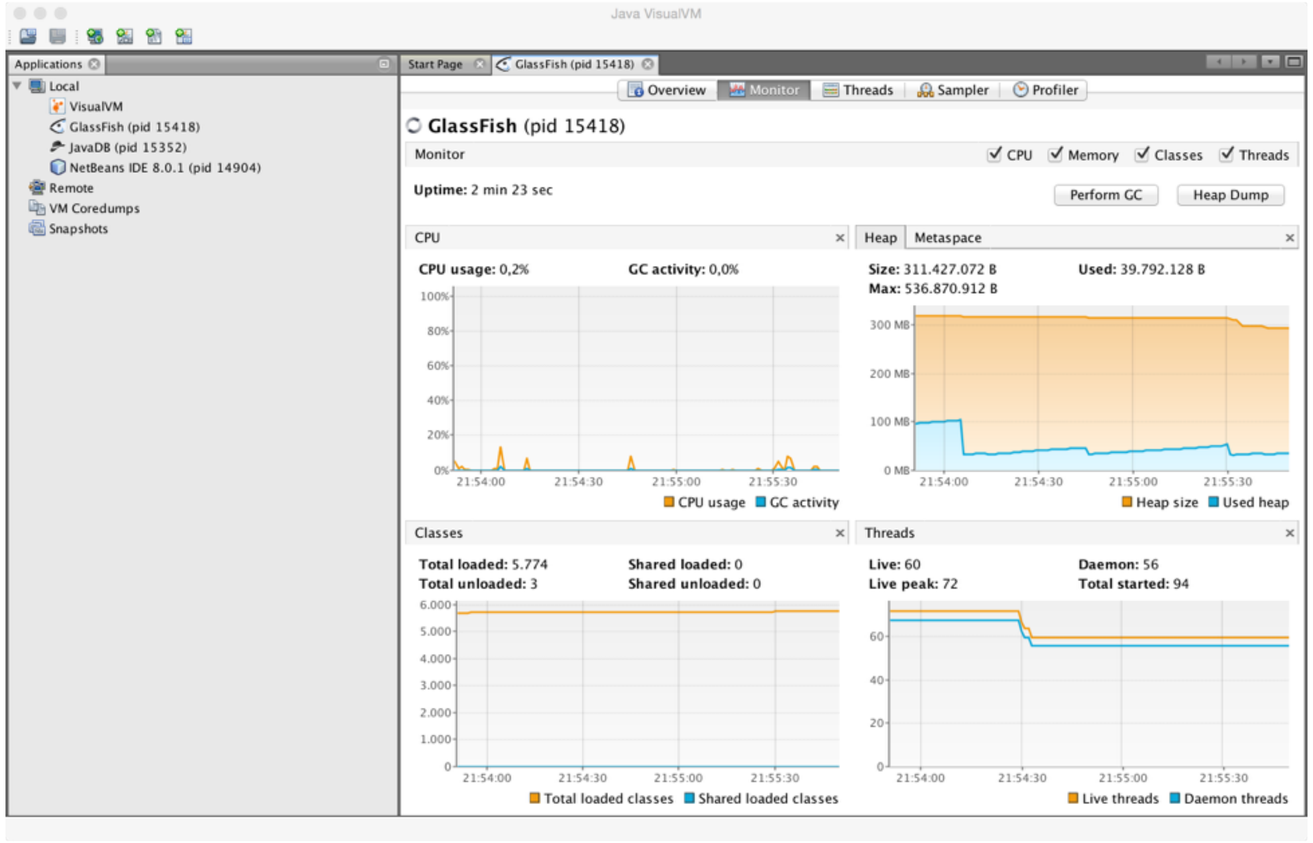
Installation: unzip

Start: Shell script execution

Start time: 1s - 3secs

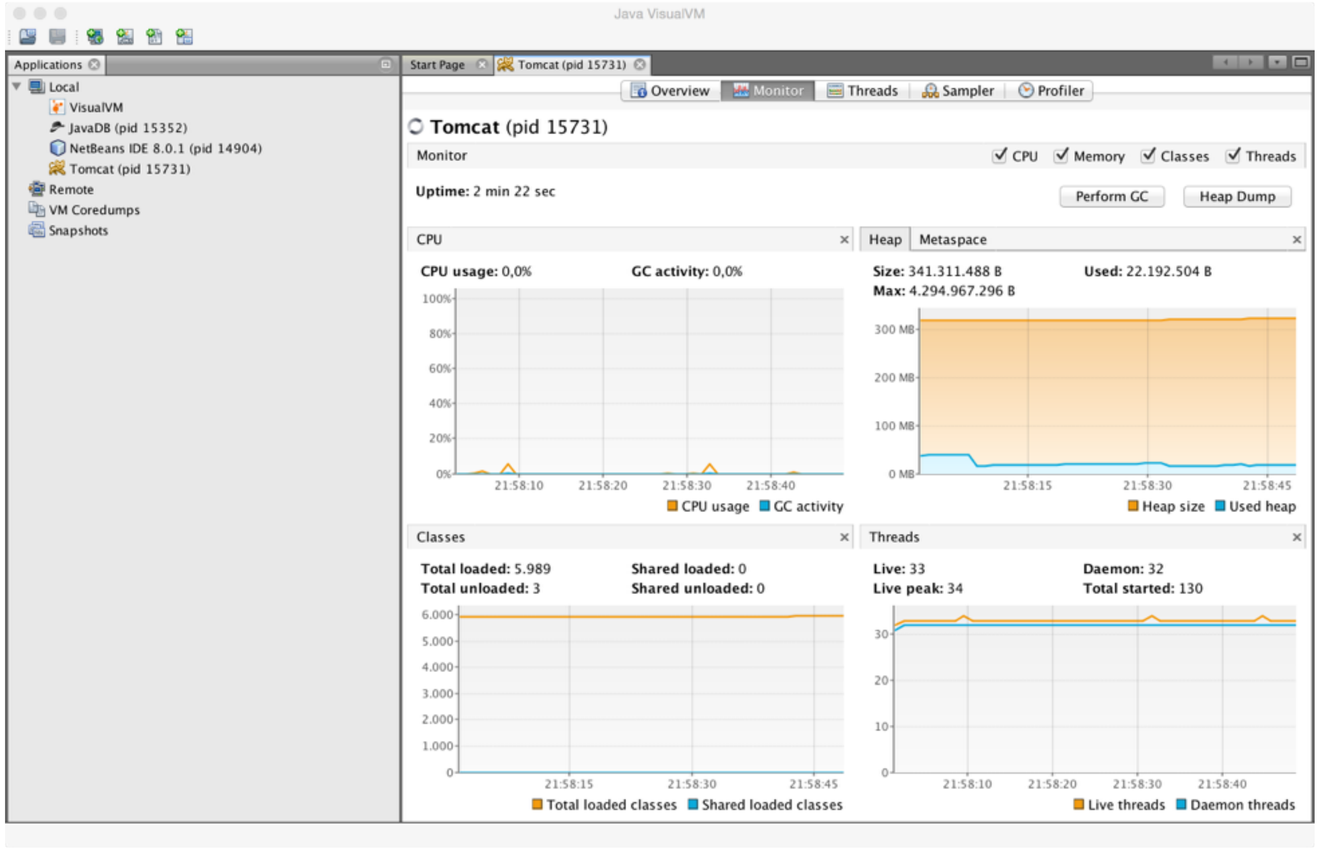
Pure Java is default.

Overhead: GlassFish v4



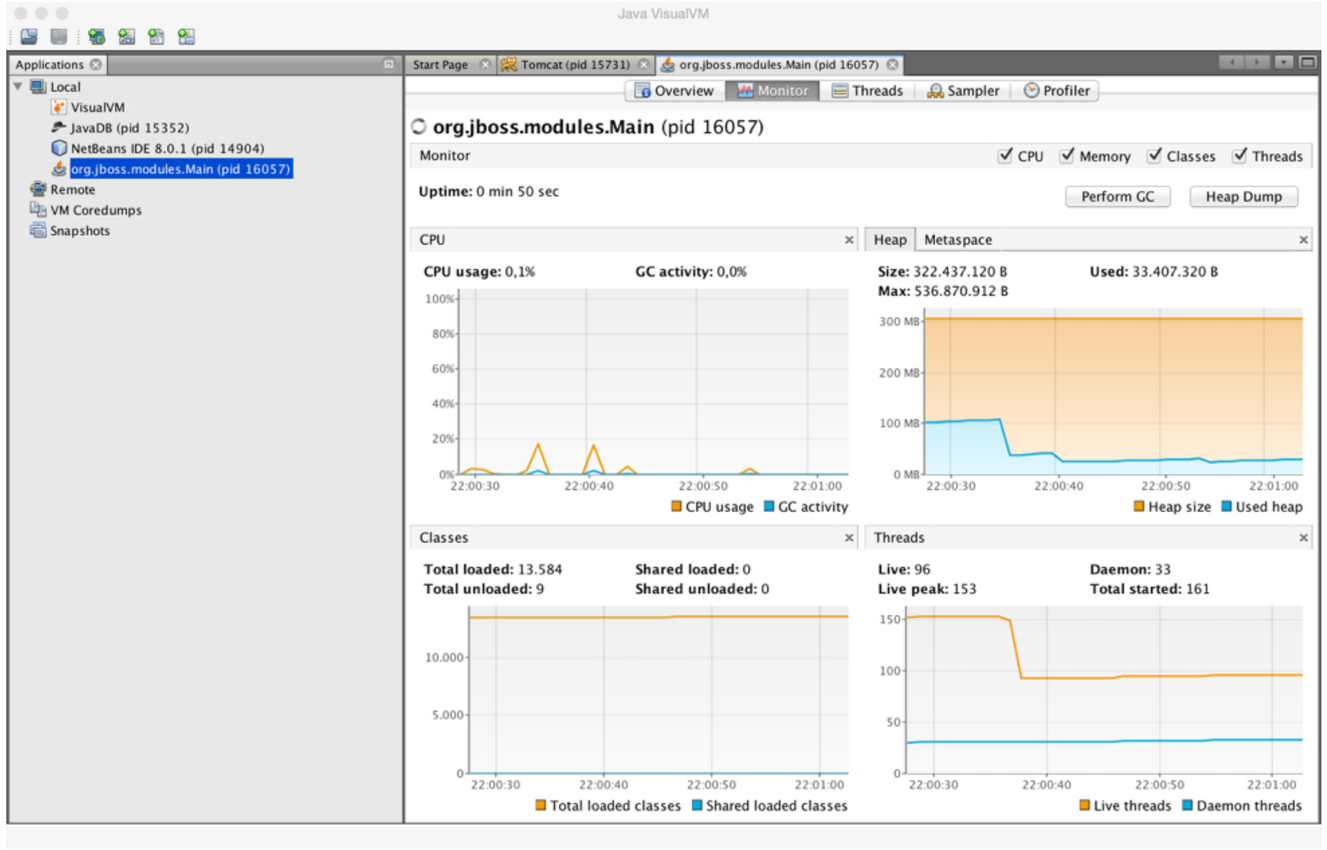
adam-bien.com

Overhead: TomEE 1.7.1



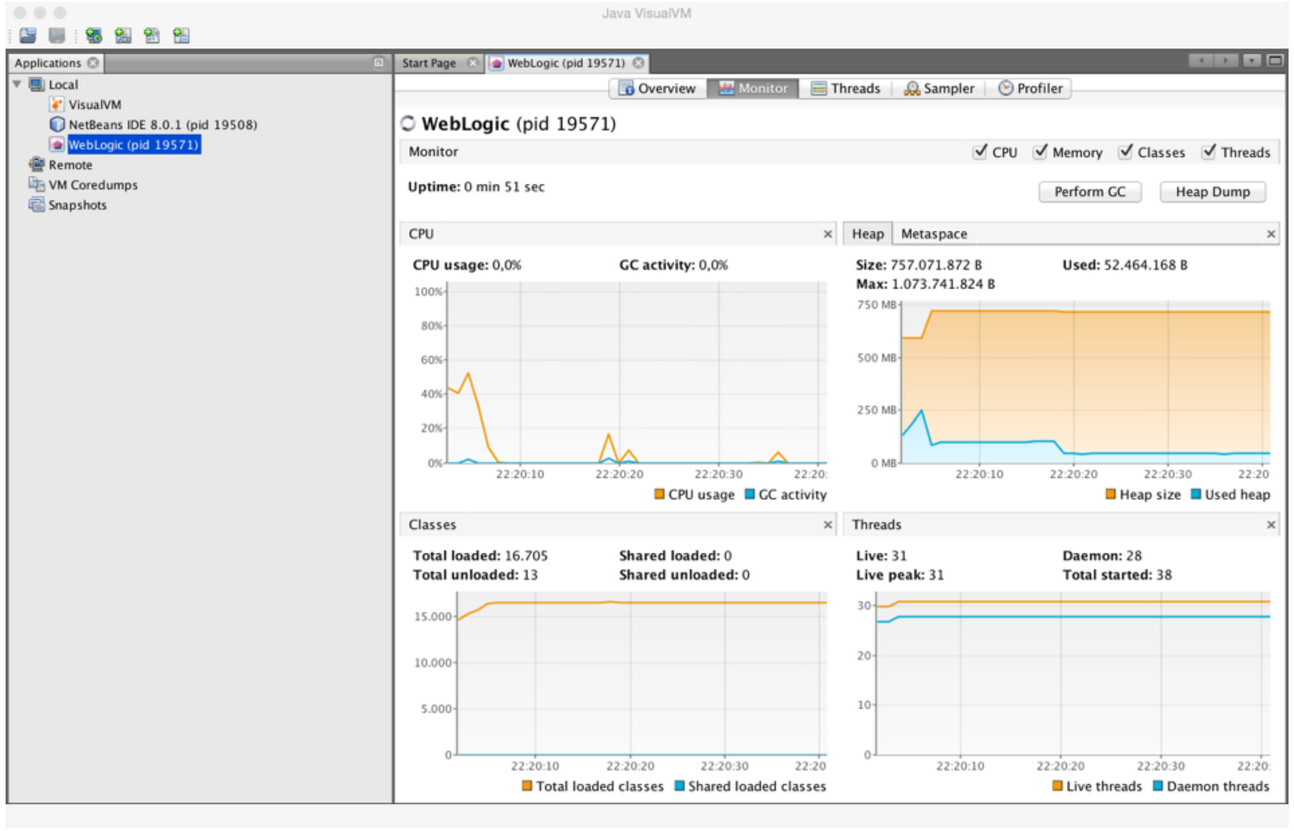
adam-bien.com

Overhead: Wildfly 8.1



adam-bien.com

Overhead: WSL12.1.3.0.0



adam-bien.com

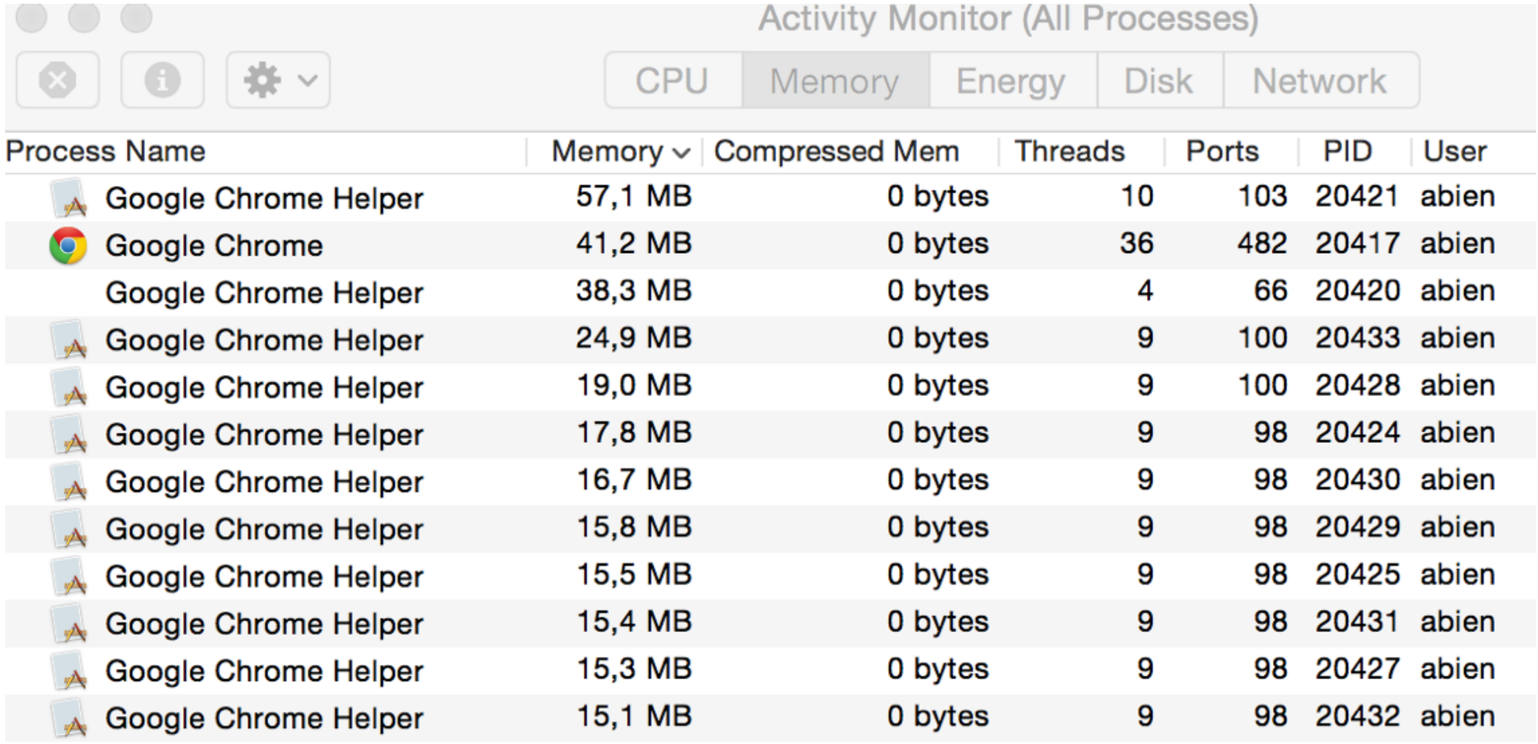
WebSphere Application Server V8.5.5.5

adam-bien.com

Overhead Summary

An idle browser consumes more memory, than an idle application server.

Empty Chrome



Process Name	Memory	Compressed Mem	Threads	Ports	PID	User
Google Chrome Helper	57,1 MB	0 bytes	10	103	20421	abien
Google Chrome	41,2 MB	0 bytes	36	482	20417	abien
Google Chrome Helper	38,3 MB	0 bytes	4	66	20420	abien
Google Chrome Helper	24,9 MB	0 bytes	9	100	20433	abien
Google Chrome Helper	19,0 MB	0 bytes	9	100	20428	abien
Google Chrome Helper	17,8 MB	0 bytes	9	98	20424	abien
Google Chrome Helper	16,7 MB	0 bytes	9	98	20430	abien
Google Chrome Helper	15,8 MB	0 bytes	9	98	20429	abien
Google Chrome Helper	15,5 MB	0 bytes	9	98	20425	abien
Google Chrome Helper	15,4 MB	0 bytes	9	98	20431	abien
Google Chrome Helper	15,3 MB	0 bytes	9	98	20427	abien
Google Chrome Helper	15,1 MB	0 bytes	9	98	20432	abien

adam-bien.com

WAR Packaging

unzip xyz.war 0,02s user 0,02s system 64% cpu 0,072 total

jar -xvf zyz.war 0,17s user 0,05s system 114% cpu 0,191 total

Java EE Protocols

HTTP (JAX-RS, Servlets)

Comet (Servlet, JAX-RS)

WebSockets

JMS (usually with JMS)

Built-in

Back Pressure

Circuit Breaker, Bulkheads, Timeouts

Monitoring (mostly via REST / JSON)

Customizable Threadpools, Isolation

Built-in

Elasticity: Request Queue

Throttling

In-process events

Aspects

(Async) HttpClient

Together with Java 8

Reactive programming

Seamless integration of synchronous methods

Reactive exception handling

Approaches

Timeouts

Bulkheads

Handshaking

Circuit Breaker

Compensative Actions

12factor.net

Reactive Programming

adam-bien.com

Reactive programming

From Wikipedia, the free encyclopedia

In computing, **reactive programming** is a programming paradigm oriented around data flows and the propagation of change. This means that it should be possible to express static or dynamic data flows with ease in the programming languages used, and that the underlying execution model will automatically propagate changes through the data flow.

For example, in an imperative programming setting, $a := b + c$ would mean that a is being assigned the result of $b + c$ in the instant the expression is evaluated. Later, the values of b and c can be changed with no effect on the value of a.

In reactive programming, the value of a would be automatically updated based on the new values.

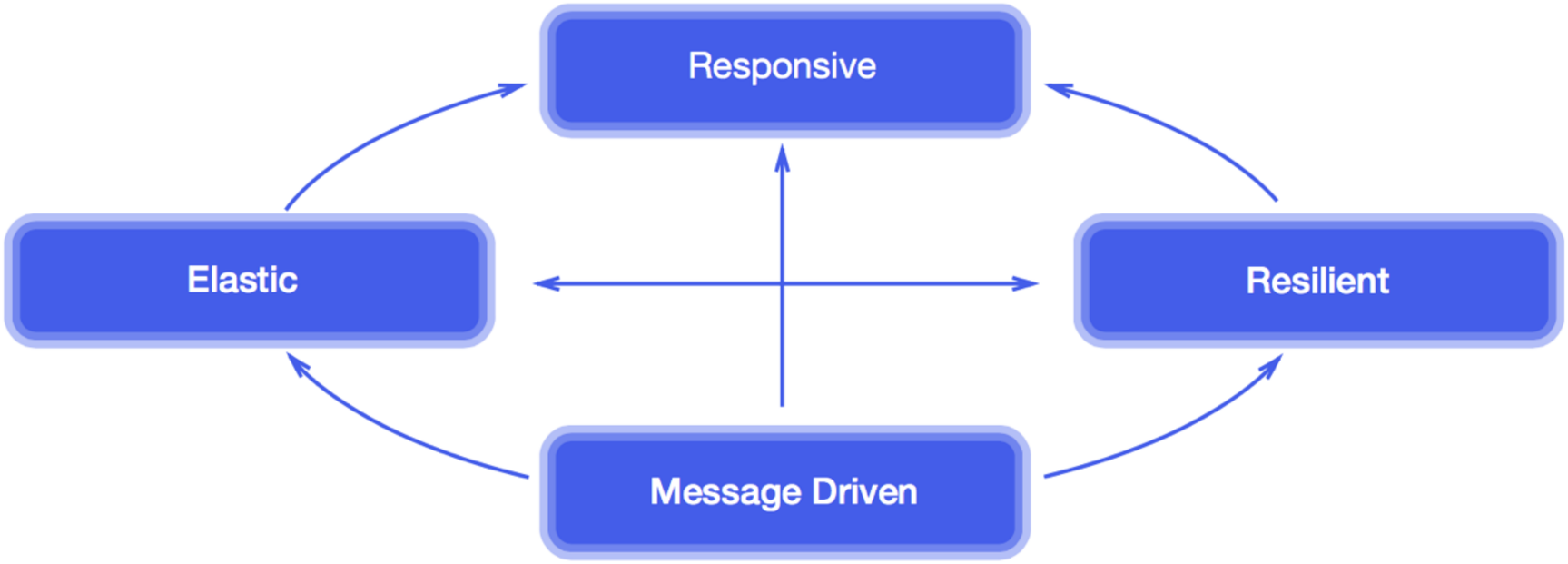
A modern spreadsheet program is an example of reactive programming. Spreadsheet cells can contain literal values, or formulas such as "=B1+C1" that are evaluated based on other cells. Whenever the value of the other cells change, the value of the formula is automatically updated.

Another example is a hardware description language such as Verilog. In this case reactive programming allows us to model changes as they propagate through a circuit.

Reactive programming has foremost been proposed as a way to simplify the creation of interactive user interfaces, animations in real time systems, but is essentially a general programming paradigm.

http://en.wikipedia.org/wiki/Reactive_programming

Reactive Manifesto



http://www.reactivemanifesto.org

Responsive

The system responds in a timely manner if at all possible. Responsiveness is the cornerstone of usability and utility, but more than that, responsiveness means that problems may be detected quickly and dealt with effectively. Responsive systems focus on providing rapid and consistent response times, establishing reliable upper bounds so they deliver a consistent quality of service. This consistent behaviour in turn simplifies error handling, builds end user confidence, and encourages further interaction.

Resilient

The system stays responsive in the face of failure. This applies not only to highly-available, mission critical systems — any system that is not resilient will be unresponsive after a failure. Resilience is achieved by replication, containment, isolation and delegation. Failures are contained within each component, isolating components from each other and thereby ensuring that parts of the system can fail and recover without compromising the system as a whole. Recovery of each component is delegated to another (external) component and high-availability is ensured by replication where necessary. The client of a component is not burdened with handling its failures.

Elastic

The system stays responsive under varying workload. Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs. This implies designs that have no contention points or central bottlenecks, resulting in the ability to shard or replicate components and distribute inputs among them. Reactive Systems support predictive, as well as Reactive, scaling algorithms by providing relevant live performance measures. They achieve elasticity in a cost-effective way on commodity hardware and software platforms.

Message Driven

Reactive Systems rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation, location transparency, and provides the means to delegate errors as messages. Employing explicit message-passing enables load management, elasticity, and flow control by shaping and monitoring the message queues in the system and applying back-pressure when necessary. Location transparent messaging as a means of communication makes it possible for the management of failure to work with the same constructs and semantics across a cluster or within a single host. Non-blocking communication allows recipients to only consume resources while active, leading to less system overhead.

Patterns

adam-bien.com

Tolerant Reader

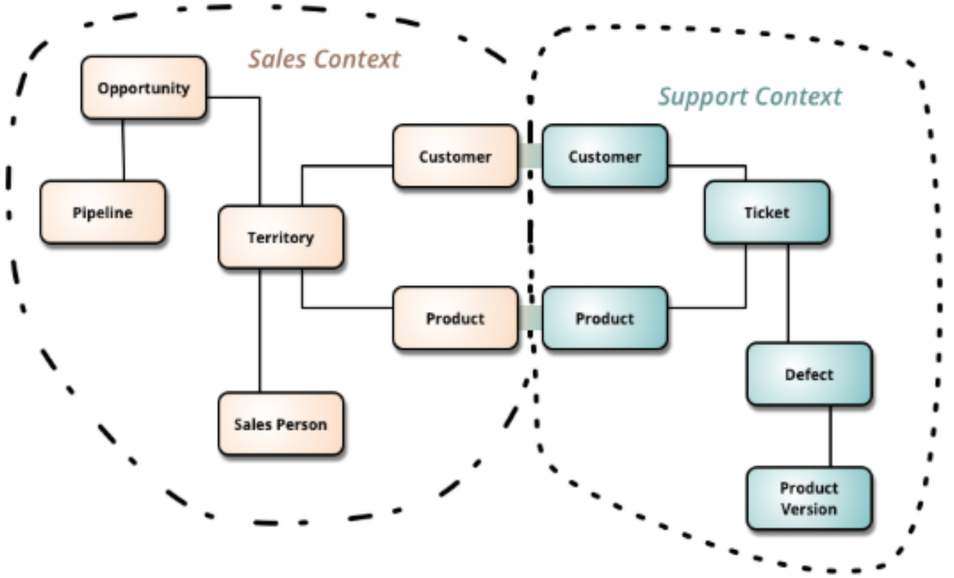
“be conservative in what you do, be liberal in what you accept from others”

-- Jon Postel

BoundedContext

Martin Fowler
15 January 2014

Bounded Context is a central pattern in Domain-Driven Design. It is the focus of DDD's strategic design section which is all about dealing with large models and teams. DDD deals with large models by dividing them into different Bounded Contexts and being explicit about their interrelationships.



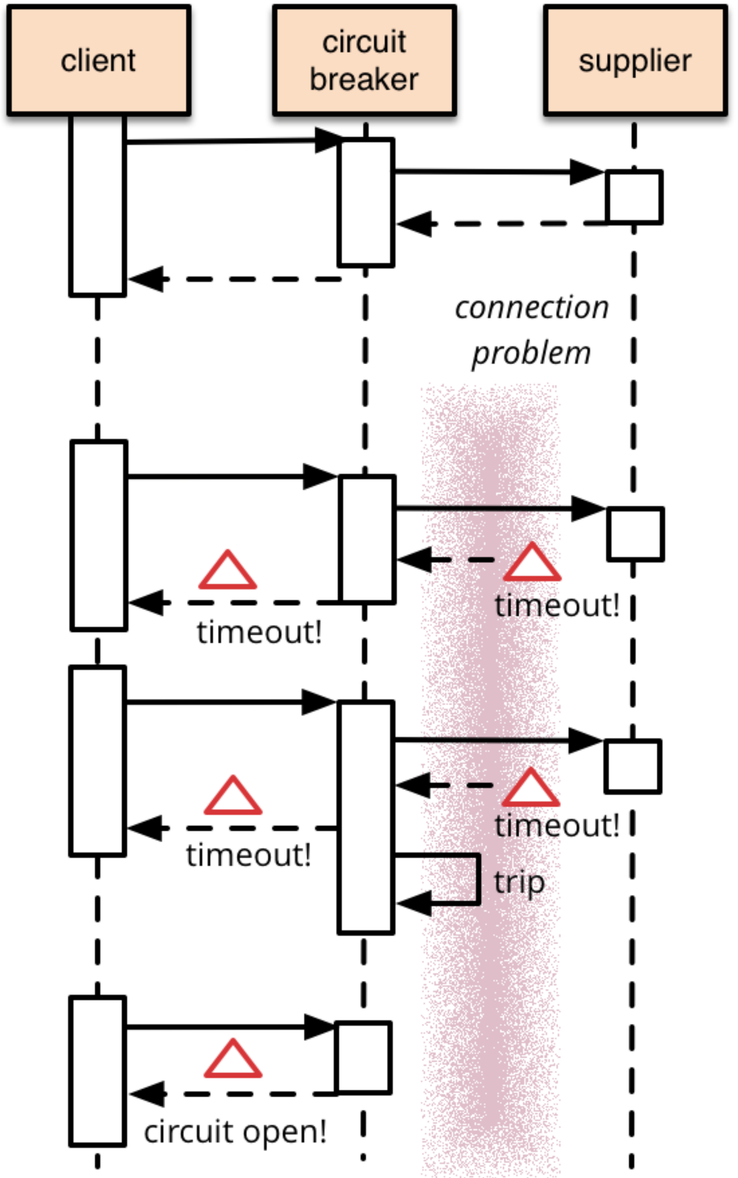
DDD is about designing software based on models of the underlying domain. A model acts as a UbiquitousLanguage to help communication between software developers and domain experts. It also acts as the conceptual foundation for the design of the software itself - how it's broken down into objects or functions. To be effective, a model needs to unified - that is to be internally consistent so that there are no contradictions within it.

CircuitBreaker

Martin Fowler
6 March 2014

It's common for software systems to make remote calls to software running in different processes, probably on different machines across a network. One of the big differences between in-memory calls and remote calls is that remote calls can fail, or hang without a response until some timeout limit is reached. What's worse if you have many callers on a unresponsive supplier, then you can run out of critical resources leading to cascading failures across multiple systems. In his excellent book Release It, Michael Nygard popularized the Circuit Breaker pattern to prevent this kind of catastrophic cascade.

The basic idea behind the circuit breaker is very simple. You wrap a protected function call in a circuit breaker object, which monitors for failures. Once the failures reach a certain threshold, the circuit breaker trips, and all further calls to the circuit breaker return with an error, without the protected call being made at all. Usually you'll also want some kind of monitor alert if the circuit breaker trips.



DDD_Aggregate

Martin Fowler

Aggregate is a pattern in Domain-Driven Design. A DDD aggregate is a cluster of domain objects that can be treated as a single unit. An example may be an order and its line-items, these will be separate objects, but it's useful to treat the order (together with its line items) as a single aggregate.

An aggregate will have one of its component objects be the aggregate root. Any references from outside the aggregate should only go to the aggregate root. The root can thus ensure the integrity of the aggregate as a whole.

Aggregates are the basic element of transfer of data storage - you request to load or save whole aggregates. Transactions should not cross aggregate boundaries.

DDD Aggregates are sometimes confused with collection classes (lists, maps, etc). DDD aggregates are domain concepts (order, clinic visit, playlist), while collections are generic. An aggregate will often contain mutliple collections, together with simple fields. The term "aggregate" is a common one, and is used in various different contexts (e.g. UML), in which case it does not refer to the same concept as a DDD aggregate.

For more details see the Domain-Driven Design book and other links in the DDD Community website.

Shared nothing architecture

A shared nothing architecture (SN) is a distributed computing architecture in which each node is independent and self-sufficient, and there is no single point of contention across the system. More specifically, none of the nodes share memory or disk storage.

http://en.wikipedia.org/wiki/Shared_nothing_architecture

Compare and Swap

Most systems provide an atomic compare-and-swap instruction that reads from a memory location, compares the value with an "expected" one provided by the user, and writes out a "new" value if the two match, returning whether the update succeeded. We can use this to fix the non-atomic counter algorithm as follows:

1. read the value in the memory location;
2. add one to the value
3. use compare-and-swap to write the incremented value back
4. retry if the value read in by the compare-and-swap did not match the value we originally read

Since the compare-and-swap occurs (or appears to occur) instantaneously, if another process updates the location while we are in-progress, the compare-and-swap is guaranteed to fail.

http://en.wikipedia.org/wiki/Linearizability

Fetch-and-increment

Many systems provide an atomic fetch-and-increment instruction that reads from a memory location, unconditionally writes a new value (the old value plus one), and returns the old value. We can use this to fix the non-atomic counter algorithm as follows:

1 Use fetch-and-increment to read the old value and write the incremented value back.

Using fetch-and increment is always better (requires fewer memory references) for some algorithms -- such as the one shown here -- than compare-and-swap,[1] even though Herlihy earlier proved that compare-and-swap is better for certain other algorithms that can't be implemented at all using only fetch-and-increment. So CPU designs with both fetch-and-increment and compare-and-swap (or equivalent instructions) may be a better choice than ones with only one or the other.

http://en.wikipedia.org/wiki/Linearizability

JXTA Ads

An Advertisement is an XML document which describes any resource in a P2P network (peers, groups, pipes, services, etc.). The communication in JXTA can be thought as the exchange of one or more advertisements through the network.

Adam Bien's Workshops

Munich (MUC) Airport Workshops, 13th, Winter Edition:

Java EE 7 Bootstrap, December 12th, 2016
also available as: [Streaming / Download Edition]

Effective Java EE 7, December 13th, 2016
also available as: [Streaming / Download Edition]

Java EE 7 Architectures, December 14th, 2016

Microservices With Java EE 7 and Java 8, December 15th, 2016
also available as: [Streaming / Download Edition]

Dedicated Virtual Workshops [on demand]

NEW:

Building HTML 5 Applications With React, October 24th, 2016

Thank You!

blog.adam-bien.com
twitter.com/AdamBien