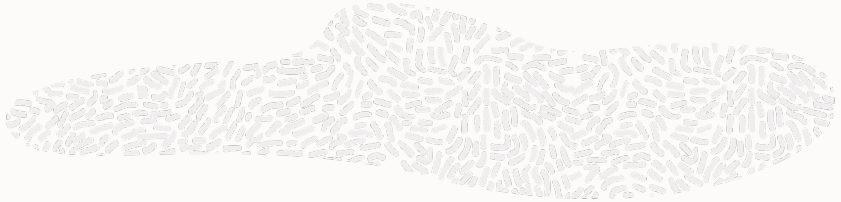

ORACLE
Java Benchmarking
Bernard Traversat,
VP of Development, Java Platform Group
Oracle

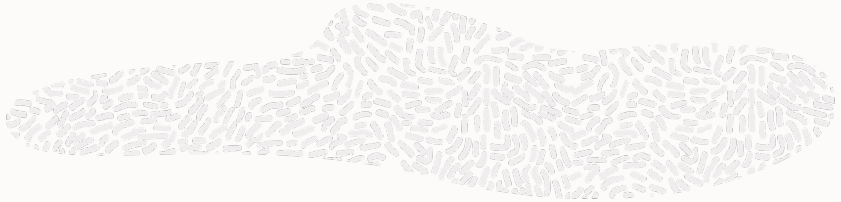
Java Benchmarking and Infrastructure

Agenda



- Challenges
- Process for testing weekly promoted builds
- Overview of benchmarks

Large Platform Support Matrix



CPU
- Intel & AMD
- ARM64 (more diversity between ARM64 CPU (Ampere, Graviton, Nvidia, Apple M4/5, Huawei, etc)
- Legacy & new architecture (RICS-V)

OS
- Windows
- Linux(s)
- macOS
- Solaris

UI
- Graphic cards

I/O
- Network NIC
- Filesystem

Performance Benchmarks

Metrics of interest

- Execution time
- Throughput
- Latency
- Memory usage
- CPU utilization
- GC (Garbage Collection) behavior
- Threading (FJP)

Why it matters?

- Detect performance regressions
- Compare algorithms
- Track efficiency
- Tune JVM, JIT and GC
- Validate system scalability

Java Benchmarking

Java adds significant benchmarking complexity due to the JVM runtime

Key challenges:

- JIT (Just-In-Time) compilation
  - Multi-tier compilation
- Garbage Collection pauses
- JVM warm-up behavior
- Dead-code elimination
- CPU memory/cache effects
- OS scheduling noise (Thread scheduling, native memory allocation)
- Many benchmarks often produce misleading results

JPG Performance Benchmarks

Multiple harnesses depending on the workload and intent

RefWorkload – Developed at Sun in the earliest days of Java perf testing
- Works on all platforms but huge complex script infrastructure
- Uses the quirky MKS Unix simulator on Windows
- Per-platform custom tools for collecting footprint from the OS
- Oracle internal framework

JMH – Java Microbenchmark Harness
- Developed by OpenJDK committers going back to 2012-ish
- OpenJDK project – the standard way to add micros into the JDK repo

Perfstartup – startup time harness using Linux perf developed in JDK 10 days for AppCDS/AOT etc
- Oracle internal framework

JPG Performance Benchmarks

SPECjbb2015 – the “reference” benchmark for Java and systems
- Emits 2 scores, Max and Critical jops
  - Max – best overall throughput
  - Critical – throughput meeting response time goals
- Good for testing GC pause time, fork/join pool, threads changes

SPECjvm2008 – compute intensive algorithms
- Good for testing compiler optimizations
- Generally run the machine 100% busy, threads in parallel doing some matrix math, etc
- A couple have locking/synchronization but mostly parallel work

Startup/Footprint – older startup time harness
- Per platform custom tools to measure footprint
- Startup times are more coarse than newer “perfstartup” harness
- Using NMT with footprint since about a year ago

JPG Performance Benchmarks

Dacapo benchmarks

Dacapo is a collection of open Java library workloads with smaller data sets

Dacapo has been very useful for testing JVM locking/synchronization changes

Contention, recursive locking, throws while holding a lock, etc

Running the 2009-MR1 version along with the new Nov 2023 release

Each workload runs in its own class loader for isolation - some examples:

Apache lucene small workloads
H2 – database in Java
Xalan – Apache XSLT
Pmd – source code analyzer
Fop – emits PDF

https://dacapobench.sourceforge.net/benchmarks.html

Some older benchmarks no longer work in more recent releases and are removed from promo testing

JPG Performance Benchmarks and Infrastructure

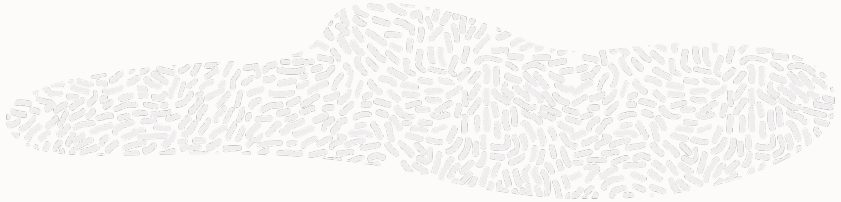
Contributed OpenJDK JMH benchmarks

JMH (Java Microbenchmark Harness) is developed by the OpenJDK contributors to avoid common benchmarking mistakes automatically

Key features:

- JVM warmup control
- Multiple benchmark iterations
- Forked JVM processes
- Dead-code elimination prevention
- Accurate statistical output

Automatic Heap Sizing



Most GC tuning no longer required

- Modern GCs much more adaptive
- G1 uses a pause time goal
- ZGC “only” needs you to set a heap size

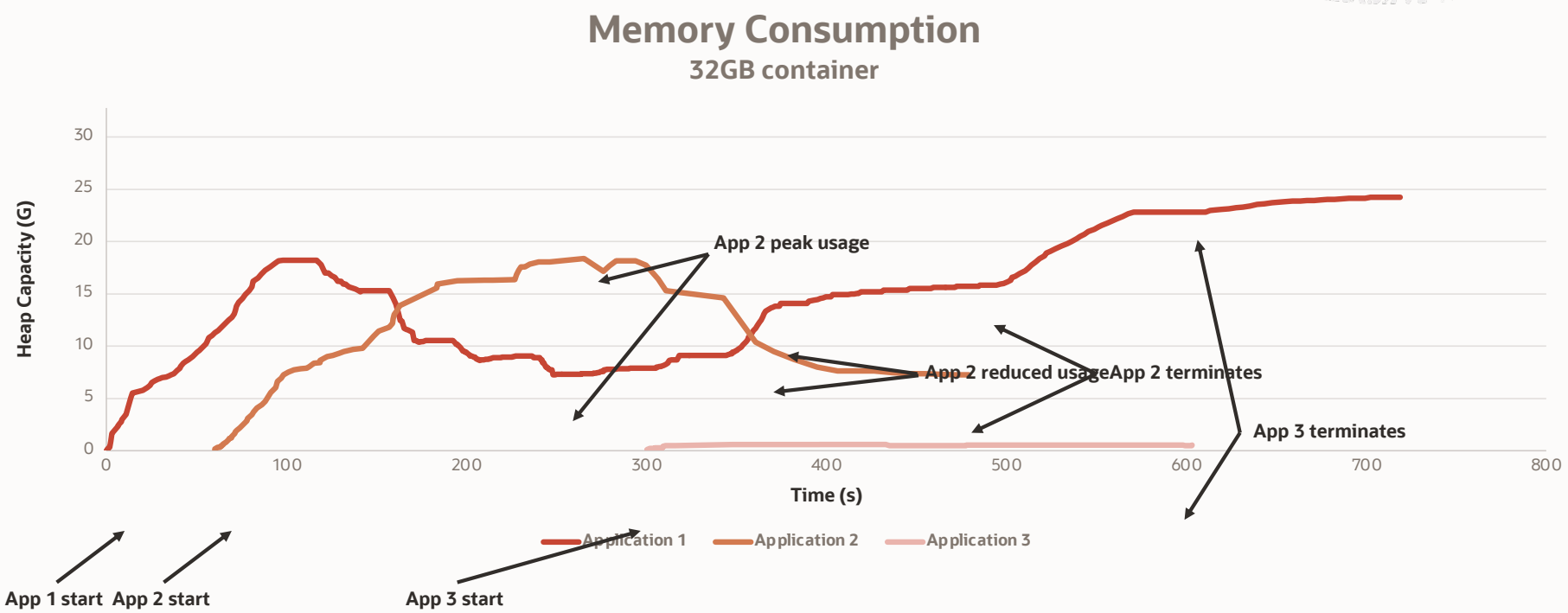
Setting the heap size is not that easy

- Are there other processes running?
- How much memory is needed by the JVM?
- Is the application using native memory?

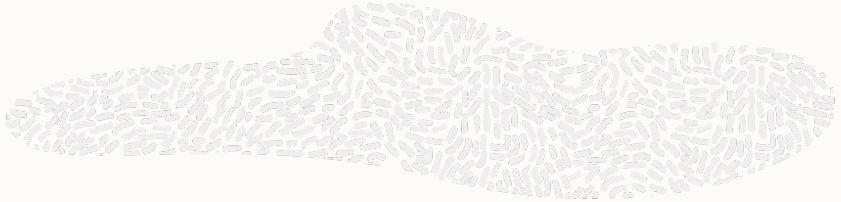
Let’s size the heap automatically

- Monitor machine/container usage
- Adapt to changes in the environment
- JEP Faster Startup and Warmup with ZGC

Automatic Heap Sizing



AI & Benchmarking



AI can help analyze performance data at scale

- Regression detection
- Anomaly detection
- Performance prediction
- Workload simulation
- Root cause analysis

AI can identify performance patterns across thousands of benchmarks.