PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

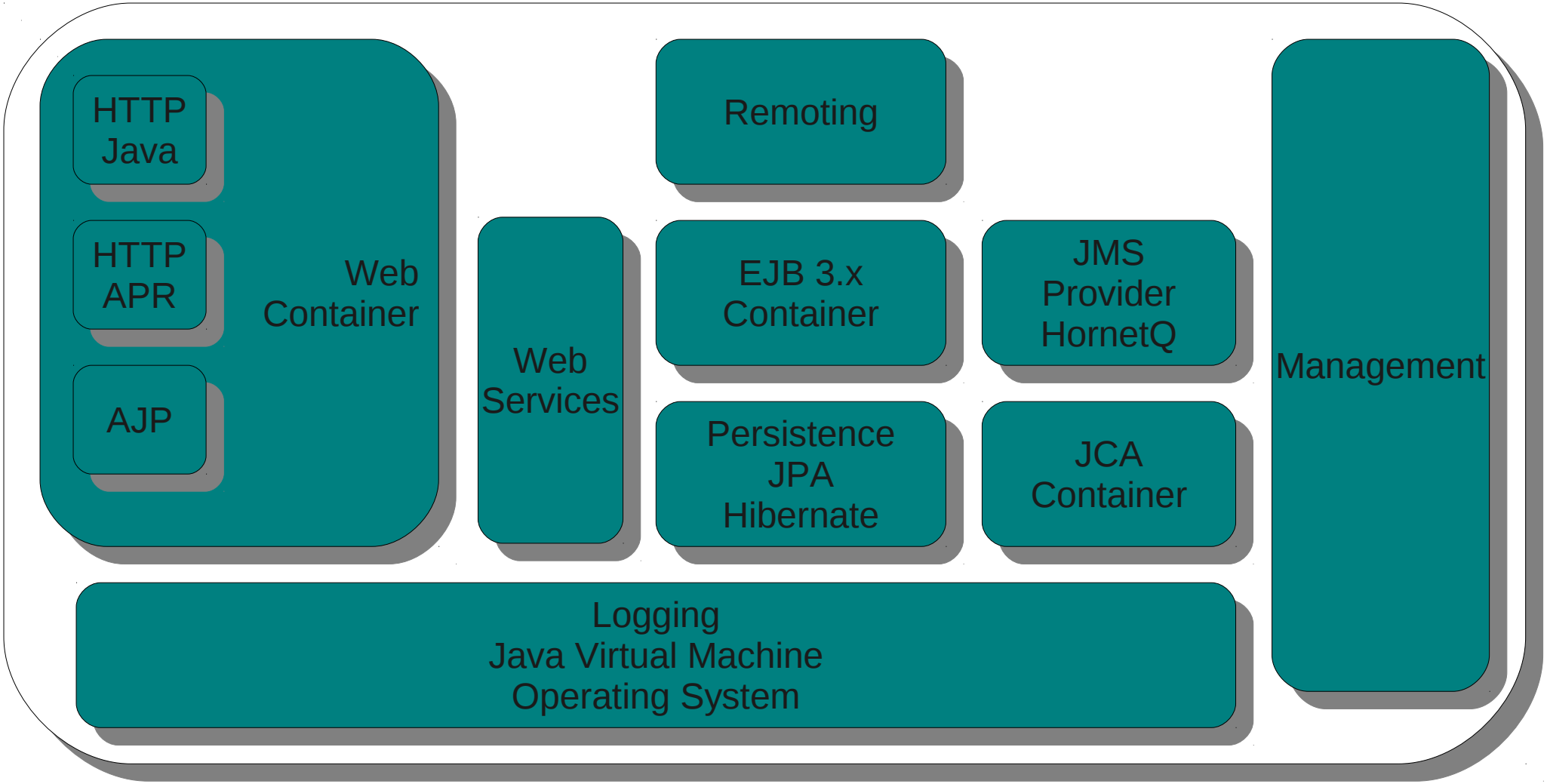
Accelerate Your JBoss Enterprise Middleware

Andrig (Andy) T. Miller
Global Platform Directory, Red Hat, Inc.
06.28.2012

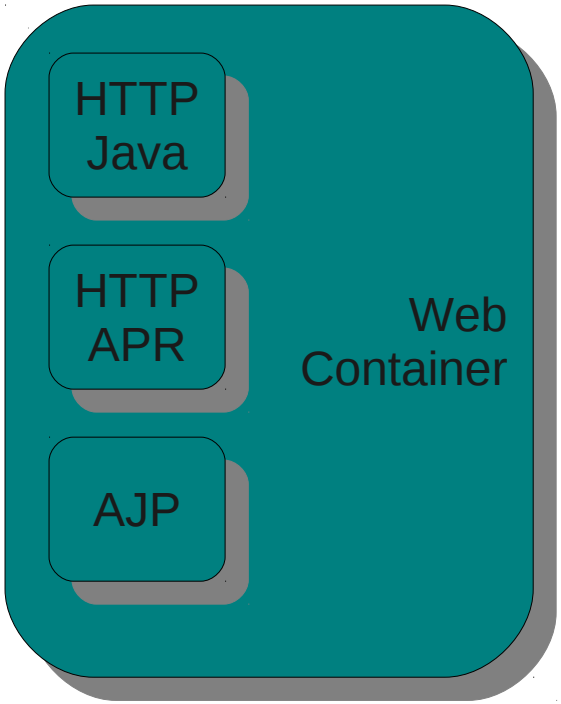
Accelerate Your JBoss Enterprise Middleware – Agenda

- Increase throughput!
  - Remove bottlenecks through configuration.
    - Pools of all kinds, Caching, Logging, Batching, Java Virtual Machine, Operating system and Monitoring.

Accelerate Your JBoss Enterprise Middleware – View from the Top of EAP 6

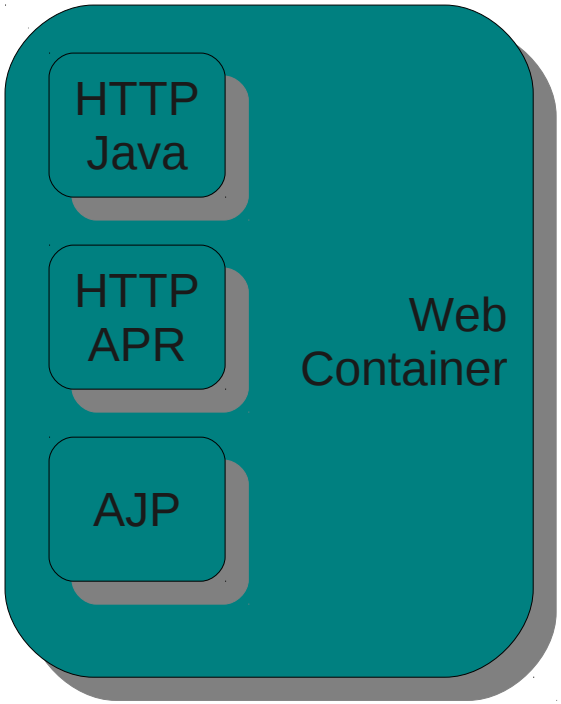


Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web



- Use case – fairly low number of connections, but very high concurrency.
  - Use Java connector (blocking I/O).
- Use case – high number of connections, but low concurrency
  - Use APR (non-blocking I/O) (Native to OS).
- Use case - Front end Apache HTTPD, with perhaps load balancing, like mod_cluster.
  - Use AJP connector.
- Important configuration parameters:
  - max-connections
    - Sets the maximum number of concurrent connections.
    - When using the Java connector, it also sets the size of the thread pool (unless using the next parameter, which defines an executor).
  - executor
    - Defines the name of a configured “executor”, or thread pool in JBoss Threads.
    - Can be used for all the connectors.
  - native
    - Determines whether you use the APR connector or Java.

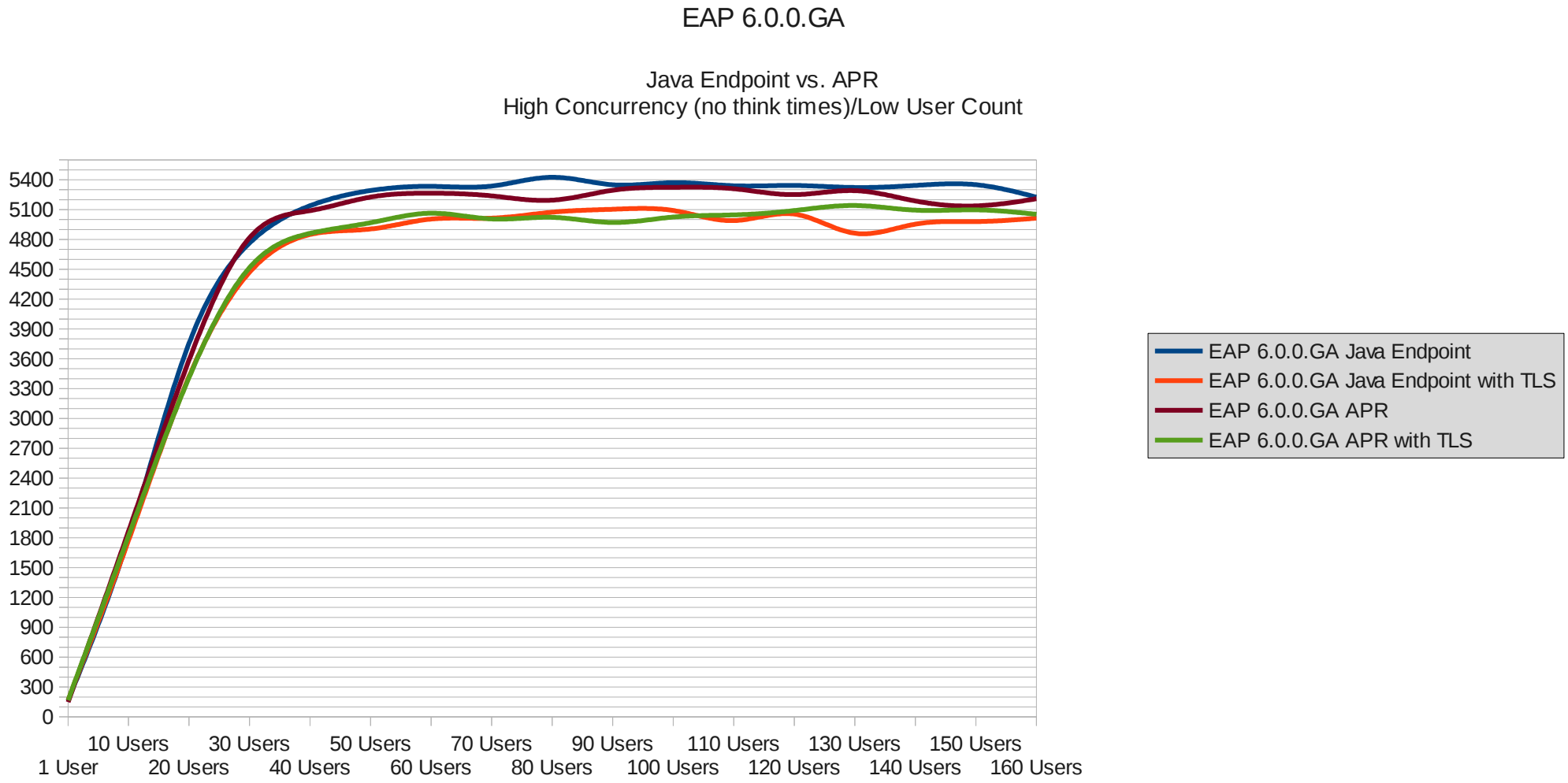
Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web



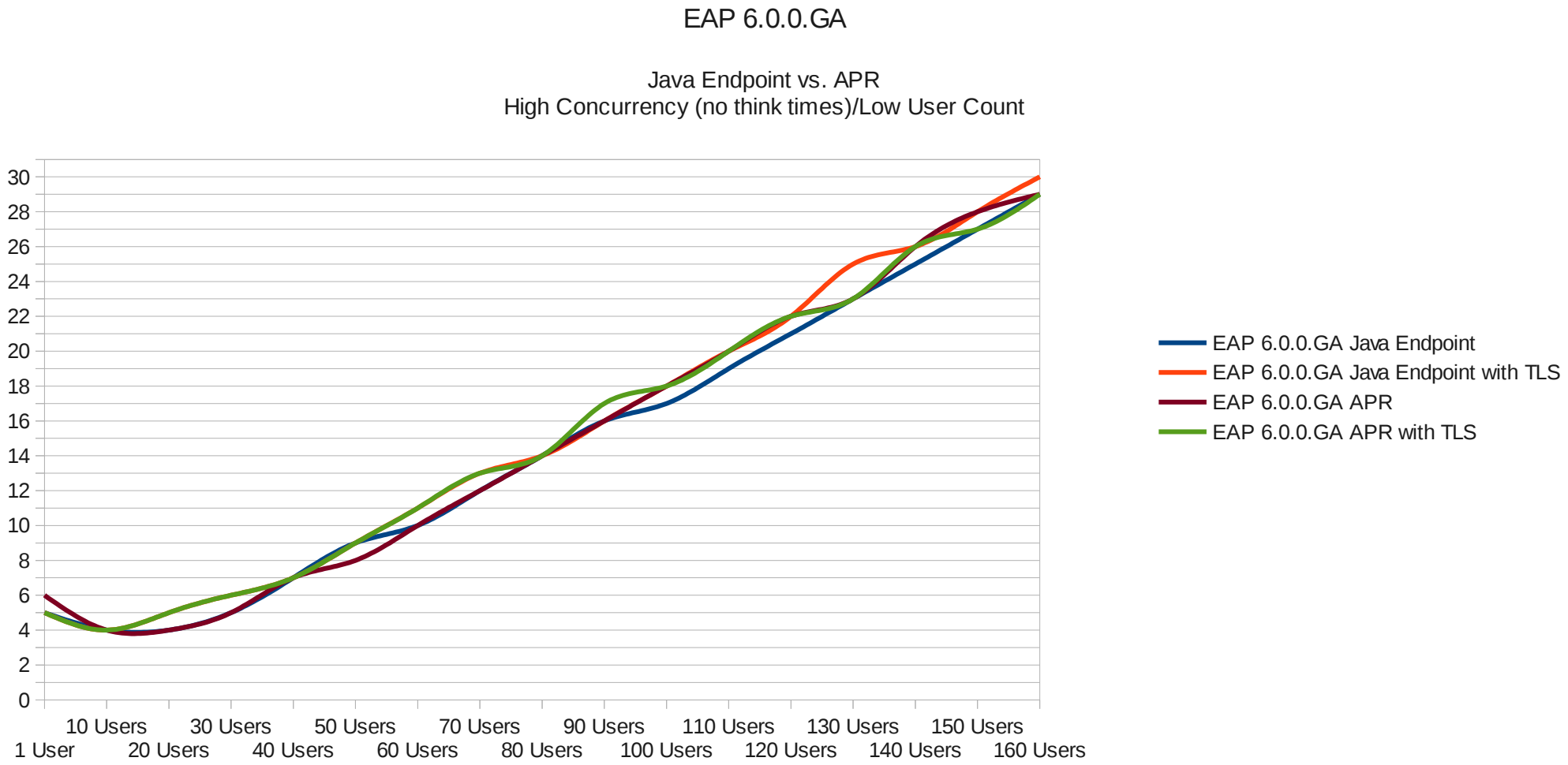
•What's a low number of connections vs. a high number of connections?

- In our testing, on a two socket Nehalem based server (8 cores with hyper-threading), with 24 GB of RAM, the number of connections that perform well for the Java blocking I/O connector is approximately 2,000 users.
  - After 2,000 users you start to see large degradation in response times, and it completely falls off a cliff at 5,000 users.
- On the same test server, using the APR connector (native code and non-blocking I/O), there is no appreciable degradation 3,500 users, and scales nicely through 5,000 users.
  - This is under the use case where you have high numbers of users, but low concurrency. The test used client think times randomly, but evenly distributed between 1 and 4 seconds.

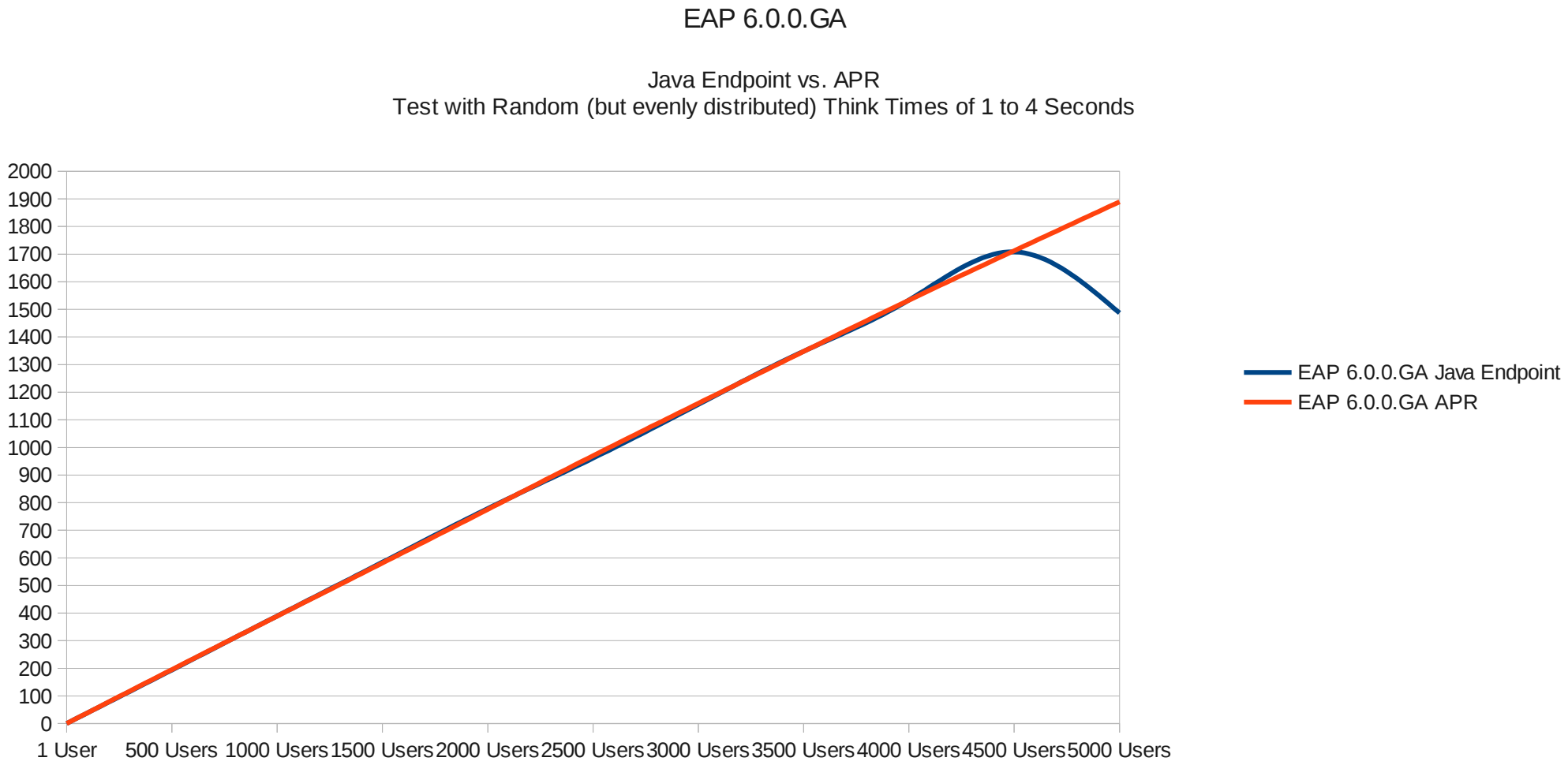
Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web



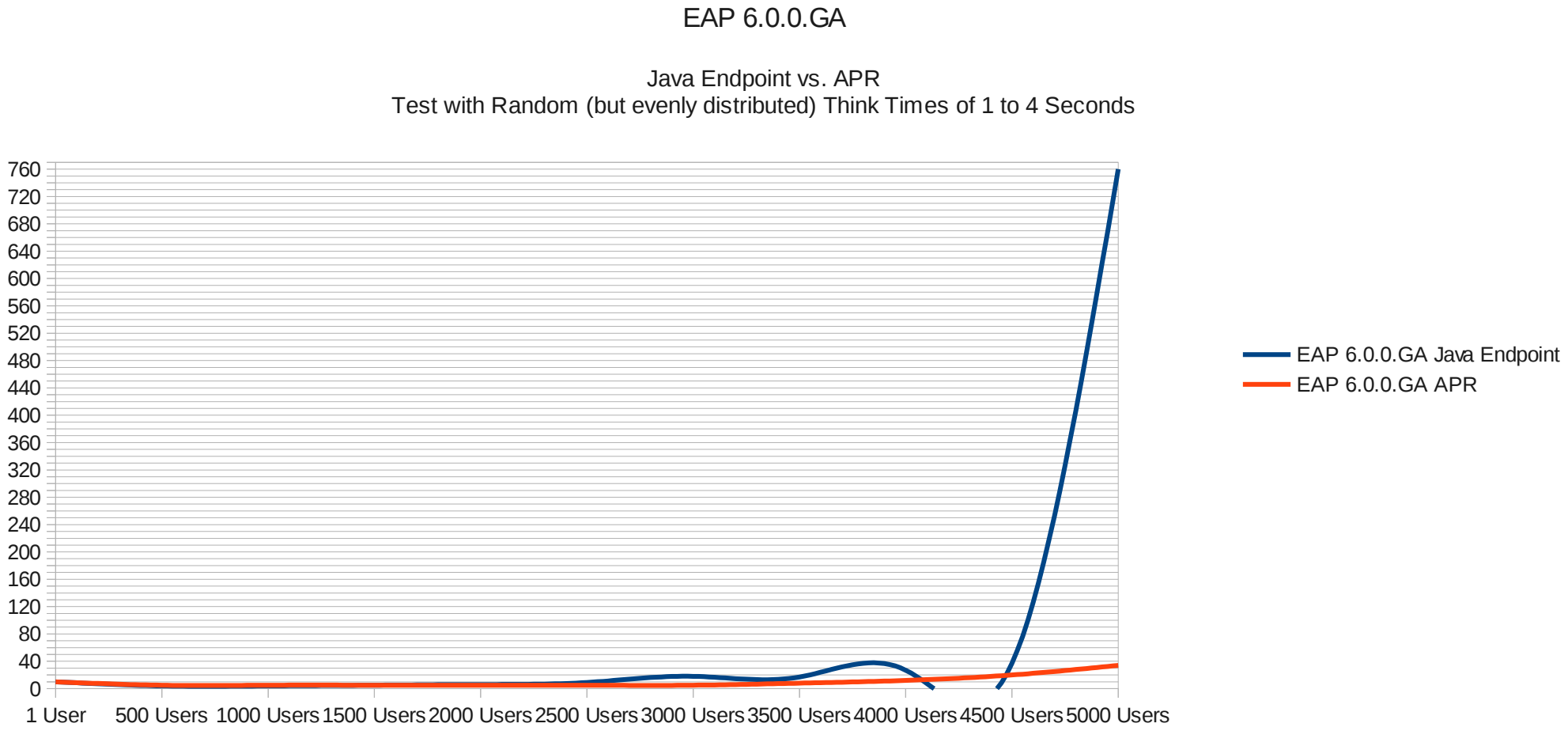
Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web



Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web



Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web



Accelerate Your JBoss Enterprise Middleware – Web Container – JBoss Web

```
<subsystem xmlns="urn:jboss:domain:threads:1.1">
    <unbounded-queue-thread-pool name="JBossWeb">
        <max-threads count="487"/>
        <keepalive-time time="75" unit="minutes"/>
    </unbounded-queue-thread-pool>
</subsystem>
...
<subsystem xmlns="urn:jboss:domain:web:1.1" default-virtual-server="default-
host" native="true">
    <connector name="http" protocol="HTTP/1.1" scheme="http" socket-
binding="http" enable-lookups="false" executor="JBossWeb" max-
connections="3260"/>
    <virtual-server name="default-host" enable-welcome-root="true">
        <alias name="jbosstesting.miller.org"/>
    </virtual-server>
</subsystem>
```


Accelerate Your JBoss Enterprise Middleware – Web Services

•Web service stack is based on both the original JBoss Web Services native code, specifically for JAX-RPC, and the Apache CXF, and is very high performing out-of-the-box.

- Two things to consider from a configuration standpoint.
- HTTP thread pool will be the thread pool that is used to invoke web service end points.
- Asynchronous web services, e.g. Web service end points annotated with @OneWay, have an internal thread pool, since once invoked they need to return to the client right away.
  - This internal thread pool, can be configured through the descriptor for the web service.
    - Important configuration parameters in jboss-webservices.xml:
      - maxQueueSize
      - lowWaterMark
      - highWaterMark
      - initialThreads

Accelerate Your JBoss Enterprise Middleware – Web Services

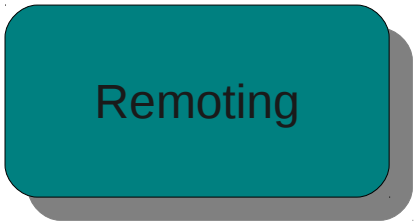
```
<?xml version="1.1" encoding="UTF-8"?>
<webservices
  xmlns="http://www.jboss.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.2"
  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee">

  <property>
    <name>cxf.queue.default.maxQueueSize</name>
    <value>500</value>
  </property>
  <property>
    <name>cxf.queue.default.highWaterMark</name>
    <value>300</value>
  </property>
  <property>
    <name>cxf.queue.default.lowWaterMark</name>
    <value>30</value>
  </property>
  <property>
    <name>cxf.queue.default.initialThreads</name>
    <value>150</value>
  <property>
  <property>
    <name>cxf.queue.default.dequeueTimeout</name>
    <value>120000</value>
  </property>
  ...
</webservices>
```


Accelerate Your JBoss Enterprise Middleware – Web Services

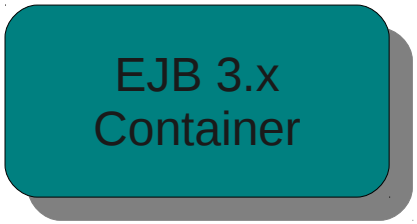
- A couple of additional comments on the configuration parameters
  - If you exceed the queue size, requests will start to be executed on the calling thread.
    - The calling thread will be the HTTP thread, and hence the web service will cease to be asynchronous, but synchronous, losing all the benefits of having annotated the web service with @OneWay to begin with.
  - The initial threads parameter causes that number of threads to be started at the time the AutomaticWorkerQueueImpl is started, so setting to something reasonable is a good idea, considering threads won't be started until the web service is invoked, and if that is under load, you could have some severe startup performance issues, while threads are being started.
    - If you don't set the initial threads, but set a lowWaterMark, the lowWaterMark will be used as the value of pre-starting threads in the pool.

Accelerate Your JBoss Enterprise Middleware – Remoting



- EAP 6 contains an entirely new remoting layer, written from scratch.
  - This new remoting layer is built on a JBoss project called XNIO, which as you can probably surmise, is based on the JDK's NIO capabilities.
  - It's a completely non-blocking I/O design.
  - Configuration is done through the subsystems that actually use remoting.
    - The EJB 3 container is the primary user of the remoting layer in EAP 6.
    - So, we will talk configuration of this as we talk about the EJB 3.x container.

Accelerate Your JBoss Enterprise Middleware – EJB 3.x Container



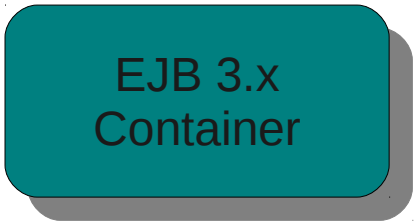
•With EAP 6, our EJB 3 container has also had significant work done to it.

- It's using the new remoting layer we talked about in the previous slide.
- It's been simplified in terms of pooling.
- Key configuration parameters are:
  - maxPoolSize (bean instance pools – MDB/SLSB)
    - InstanceAcquisitionTimeout
  - channelCreationOptions
    - WORKER_READ_THREADS, WORKER_WRITE_THREADS, MAX_INBOUND_MESSAGES, MAX_OUTBOUND_MESSAGES
  - Thread pool
    - max-threads
    - Keepalive-time
  - in-vm-remote-interface-invocation

Accelerate Your JBoss Enterprise Middleware – EJB 3.x Container

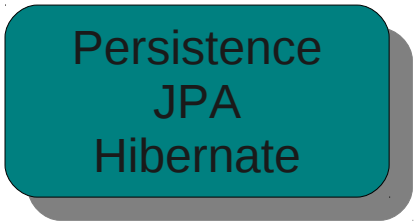
```
<pools>
    <bean-instance-pools>
        <strict-max-pool name="slsb-strict-max-pool" max-pool-size="1300" instance-acquisition-timeout="1" instance-acquisition-timeout-unit="MILLISECONDS"/>
        <strict-max-pool name="mdb-strict-max-pool" max-pool-size="180" instance-acquisition-timeout="1" instance-acquisition-timeout-unit="MILLISECONDS"/>
        </bean-instance-pools>
 </pools>
...
<remote connector-ref="remoting-connector" thread-pool-name="default">
    <channel-creation-options>
        <option name="WORKER_READ_THREADS" value="2" type="xnio"/>
        <option name="WORKER_WRITE_THREADS" value="2" type="xnio"/>
        <option name="MAX_INBOUND_MESSAGES" value="165" type="remoting"/>
        <option name="MAX_OUTBOUND_MESSAGES" value="165" type="remoting"/>
        </channel-creation-options>
</remote>
...
<thread-pools>
    <thread-pool name="default">
        <max-threads count="165"/>
        <keepalive-time time="75" unit="minutes"/>
    </thread-pool>
</thread-pools>
...
<in-vm-remote-interface-invocation pass-by-value="false"/>
```


Accelerate Your JBoss Enterprise Middleware – EJB 3.x Container



•Important notes:
- Default maxSession for Message Driven Beans (MDB), so regardless of how large you set the pool size, only 15 will execute concurrently, unless the maxSession is changed.
  - Of course, that is per MDB, so you could have a pool that is larger, and encompasses all the MDB's in the application, each only needing 15 or less to run concurrently.
- The pool for stateless session beans needs to be sized, based on the number of stateless session beans in the application, and the concurrency rate in which those beans are invoked.
  - e.g. Your application has 10 unique stateless session beans, and all 10 are invoked at the same rate, and that rate is 10 per second, and the response times of those invocations are 1 second each.
    - This yields a pool size of at least 100.
    - Decreases in response times, or increases in concurrency is what drives the size.

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate



•With EAP 6, our persistence strategy is the same, but with a newer version of Hibernate, as the JPA provider, Hibernate 4.1.x.

- Besides supporting the new Java EE 6 JPA specification, most of the changes are internal, and there is little difference from a configuration standpoint (at least what we will talk about here).
- Key topics for persistence:
  - Second-level cache (based on Infinispan in EAP 6):
    - Entity Caching
    - Query Caching
  - Batching
  - Fetch sizes
  - Batch inserts

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

- Entity Cache Keys:
  - Read/Write Ratio
    - Mostly read, with very little writes (inserts and/or updates)
  - Query type
    - entitymanager.find(class, pk)
    - Cacheable query
      - A cacheable query is one that always returns the exact same result.
  - Cache Concurrency Strategy
    - READ_ONLY
      - The read only strategy applies to entities that are only read, or read and inserted, but not updated.
    - TRANSACTIONAL
      - The transactional strategy is required if there are updates to the entity being cached.
  - Data Size
    - There is only so much heap space to play with, and extremely large sets of entities may suffer from low cache hit rates just because of the number of entities involved.
  - Access Pattern
    - You may have a large set of entities, but if the access pattern is such that a small subset of them are accessed very often, you may still derive benefits from caching them.

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

- Query Cache Keys:
  - Query type
    - Cacheable query
      - A cacheable query is one that always returns the exact same result.
      - Most of the time this means the application has either created a query that is using the primary key, or the query results are always the same because the entity is only read, and there are no inserts or updates.
  - The query cache only stores the keys of the result set!
    - This means you “MUST” also cache the entity in the entity cache.
  - While a query may be cacheable, if the entity is not a good candidate to be cached, then the query cache should not be used.
  - Sizing of the query cache is based on the number of unique combinations of parameters in the query (all still must result in the same result set).

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

- Key configuration parameters:
  - Entity Cache
    - Entity cache types
      - local-cache, invalidation-cache, replicated-cache
      - transaction mode
      - eviction-strategy
        - Defaults to LRU (Least recently used), but Infinispan gives a new algorithm called LIRS (Low Inter-reference Recency Set), which performs better in my tests.
      - max-entries
        - How many entries the cache can hold. Sizing this is based on the number of entities to be cached, and perhaps a subset of them based on the access pattern.
      - expiration
        - max-idle and lifespan
          - Lifespan causes eviction regardless of whether it has been idle or not.
          - Max-idle, without specifying lifespan, will cause entities that have not been accessed in that time, to be eligible to be evicted.

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

```
<cache-container name="hibernate" default-cache="local-query"
module="org.jboss.as.jpa.hibernate:4">
...
    <local-cache name="local-query">
        <transaction mode="NONE"/>
        <eviction strategy="LIRS" max-entries="180"/>
        <expiration max-idle="1200000" lifespan="1200000"/>
    </local-cache>
...
</cache-container>
```


Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

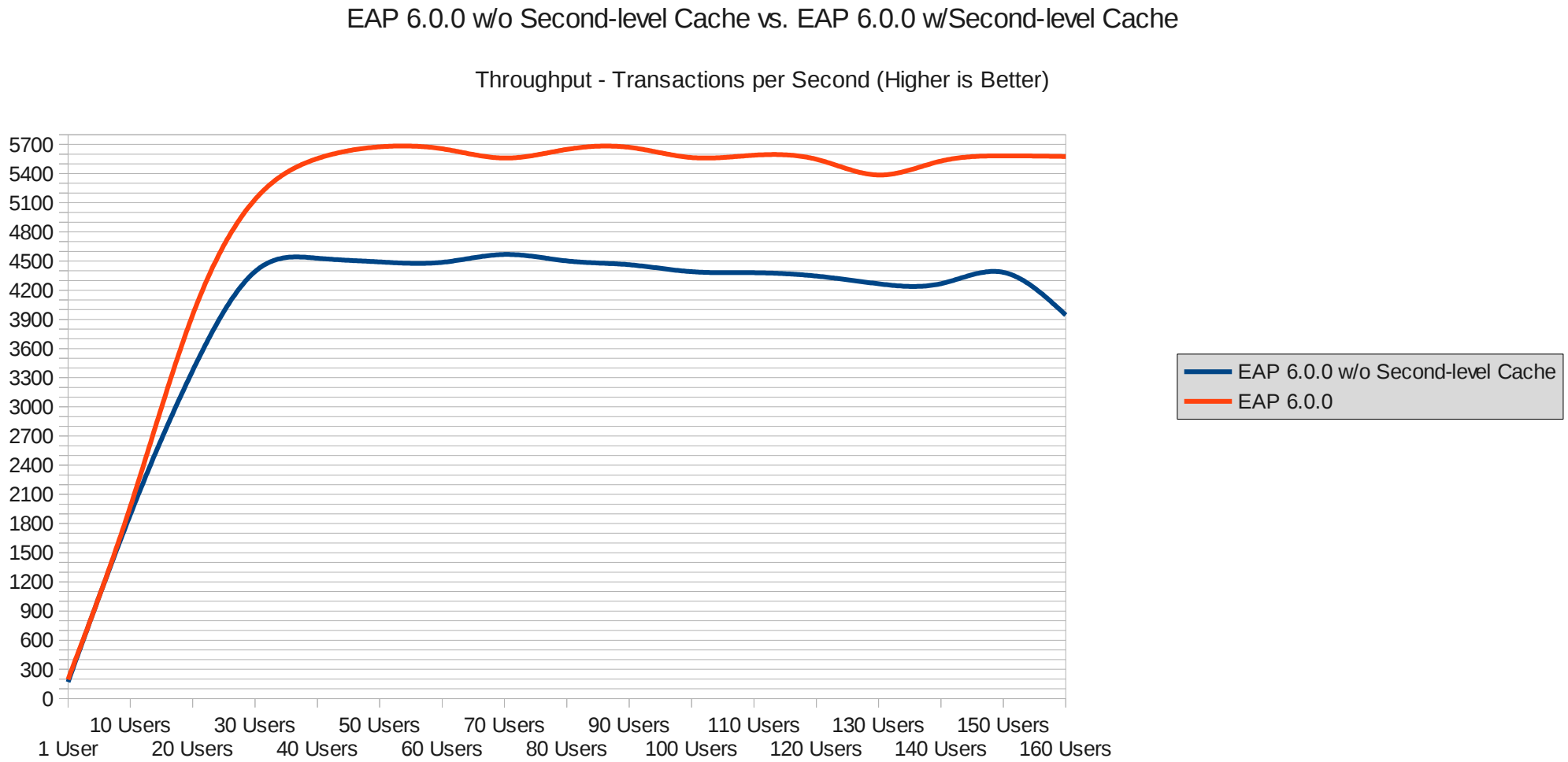
Persistence
JPA
Hibernate

- Key configuration parameters:
  - Query Cache
    - Entity cache types
      - local-cache is really the only cache type that makes sense for a query cache. You don't need to invalidate a cached query on another node, as the result set is always the same. You also don't want the overhead of replicating the cache, as other nodes will execute the query once, and cache it anyway.
      - eviction-strategy
      - max-entries
      - expiration
        - max-idle and lifespan

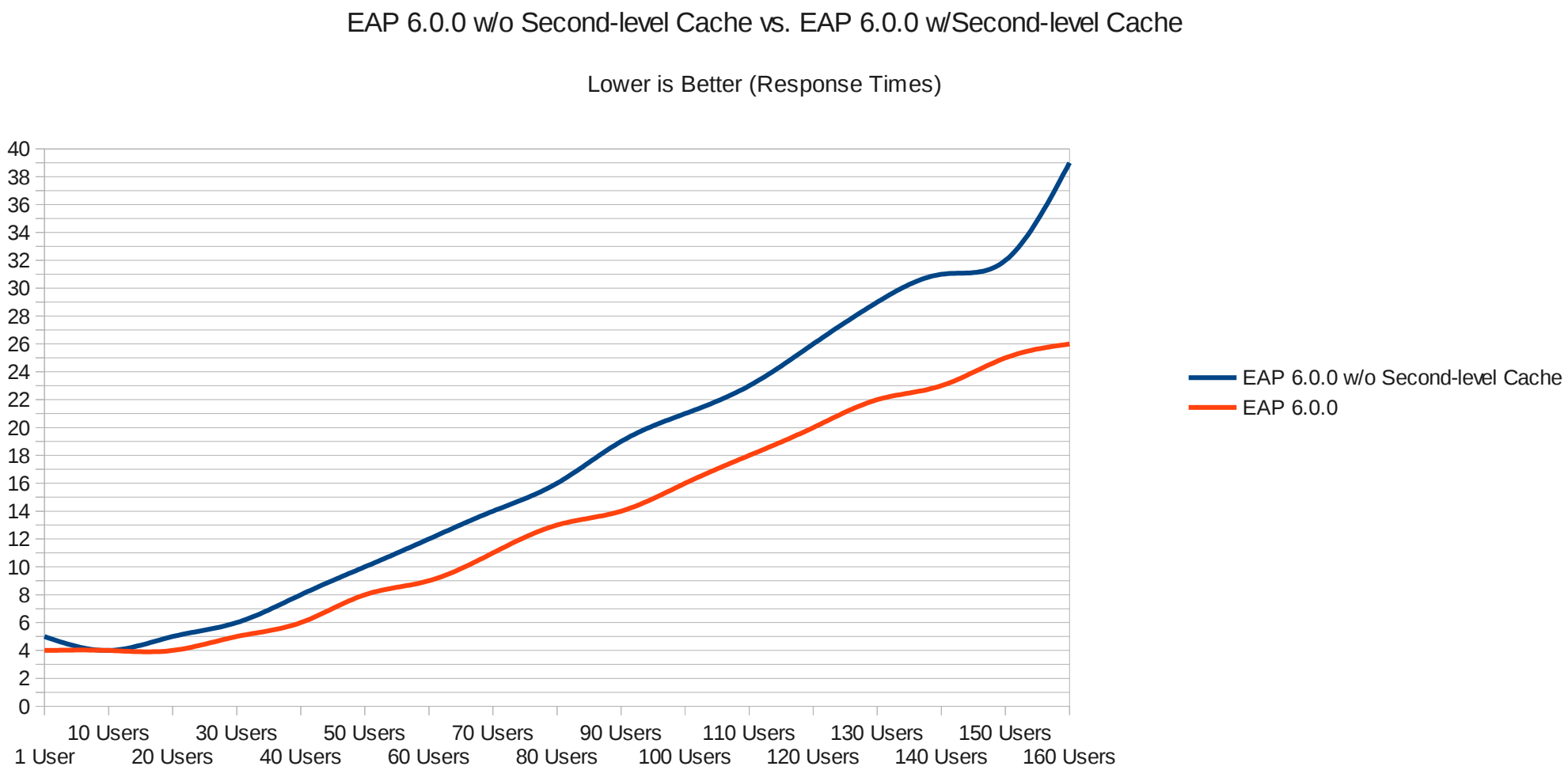
Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

```
<cache-container name="hibernate" default-cache="local-query"
module="org.jboss.as.jpa.hibernate:4">
    <local-cache name="entity">
        <transaction mode="NON_XA"/>
        <eviction strategy="LIRS" max-entries="17030000"/>
        <expiration max-idle="1200000" lifespan="1200000"/>
    </local-cache>
...
</cache-container>
```

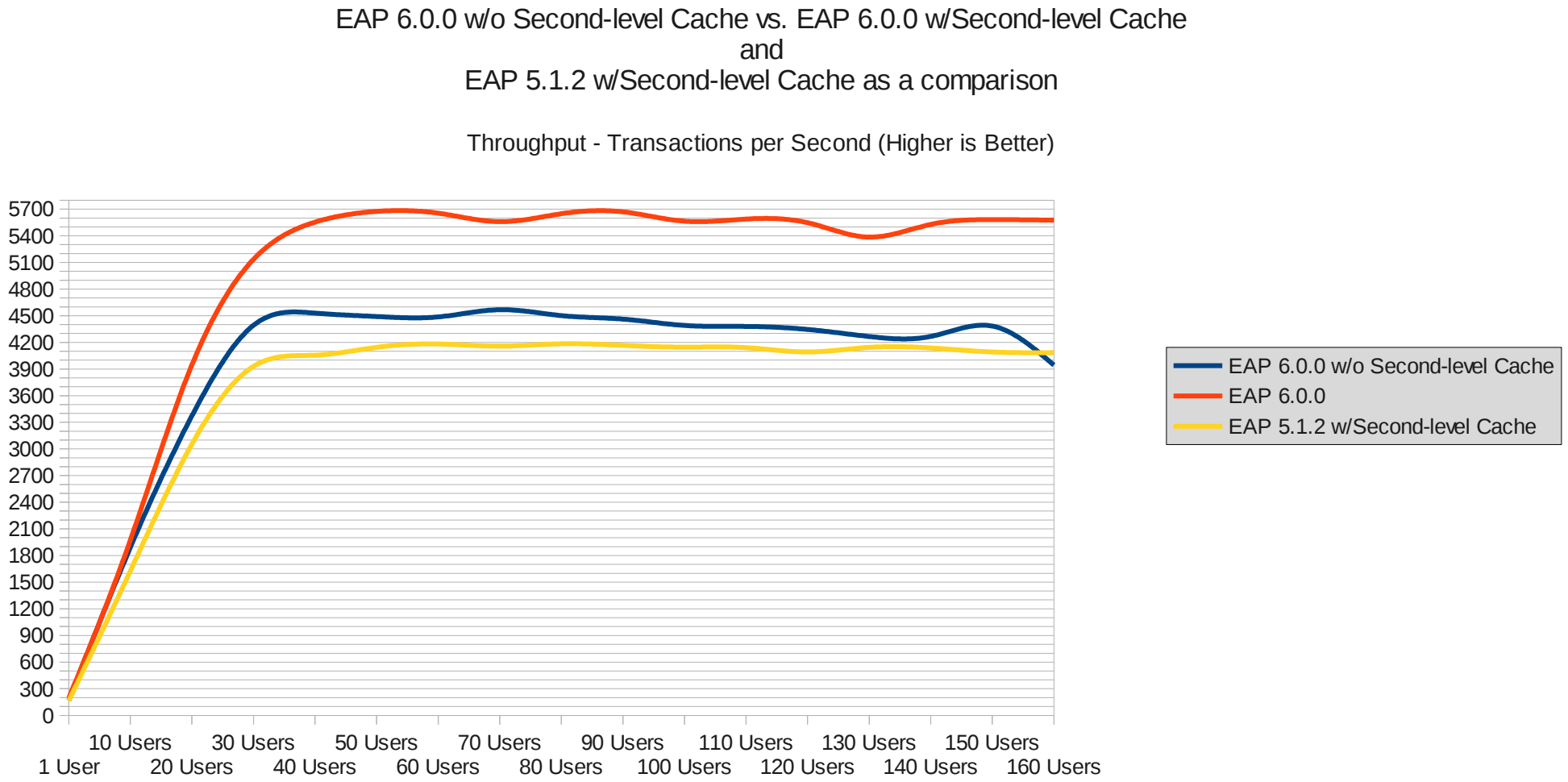

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate



Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate



Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate



Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

- Batching
  - What is is?
    - Well, this is Hibernate's ability to batch a set of SQL statements to be executed and send them together to the database.
    - This reduces latency, and optimizes the network between the application server and the database server.
    - Is specified through a Hibernate property:
      - `<property name="hibernate.jdbc.batch_size" value="20"/>`
    - Sizing this parameter requires good knowledge of how many inserts, deletes, updates, etc. there typically are in your transactions.
    - You can put this property in your persistence.xml.

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

- Fetch sizes
  - In Hibernate you can specify how many rows to return from the database at one time for queries that return more than one row.
  - If you return one row at a time, and there are tens, hundreds, or even thousands, its going to increase response times, but also add lots of latency for the rounds trips to and from the database.
  - You specify this through a Hibernate property:
    - `<property name="hibernate.jdbc.fetch_size" value="20"/>`
  - Sizing of this parameter requires knowledge of the application queries and the typical usage of the result set.
    - e.g., in the application that is represented by the graphs for the second-level cache results, the fetch size is set to 20, even though one of the main queries may return as many as 500 rows.
      - The reason for 20, instead of 500, is that the result set is managed by a stateful session bean that paginates by 20 rows at a time. Understanding the frequency in which a user will even go past the first 20 rows is important!
  - Like the batch size, this can be set in the persistence.xml.

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

- Batch Inserts
  - What do I mean by batch inserts?
    - This is when you can take multiple insert statements, that would normally be sent to and executed by the database one at a time, and make it into a single insert statement.
    - e.g.
      - `Insert into Table (id, val1, val2, val3) values ('x', 'x', 'x', 'x');`
      - `Insert into Table (id, val1, val2, val3) values ('y', 'y', 'y', 'y');`
      - `Insert into Table (id, val1, val2, val3) values ('z', 'z', 'z', 'z');`
      - And turn it into:
        - `Insert into Table (id, val1, val2, val3) values('x', 'x', 'x', 'x'), ('y', 'y', 'y', 'y'), ('z', 'z', 'z', 'z');`
      - This feature is dependent on two things. This first being a Hibernate property:
        - `<property name="hibernate.order_inserts" value="true"/>`
      - The second being the JDBC driver's capabilities to rewrite the statement:
        - e.g., the MySQL JDBC driver has a connection property called:
          - rewriteBatchedStatements

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

```
...
<datasource jndi-name="java:/MySqlDS" pool-name="MySqlDS" use-
ccm="false">
    <connection-url>
        jdbc:mysql://localhost:3306/EJB3
     </connection-url>
     <connection-property name="maintainTimeStats">
         false
     </connection-property>
     <connection-property name="rewriteBatchedStatements">
         true
     </connection-property>
...
```

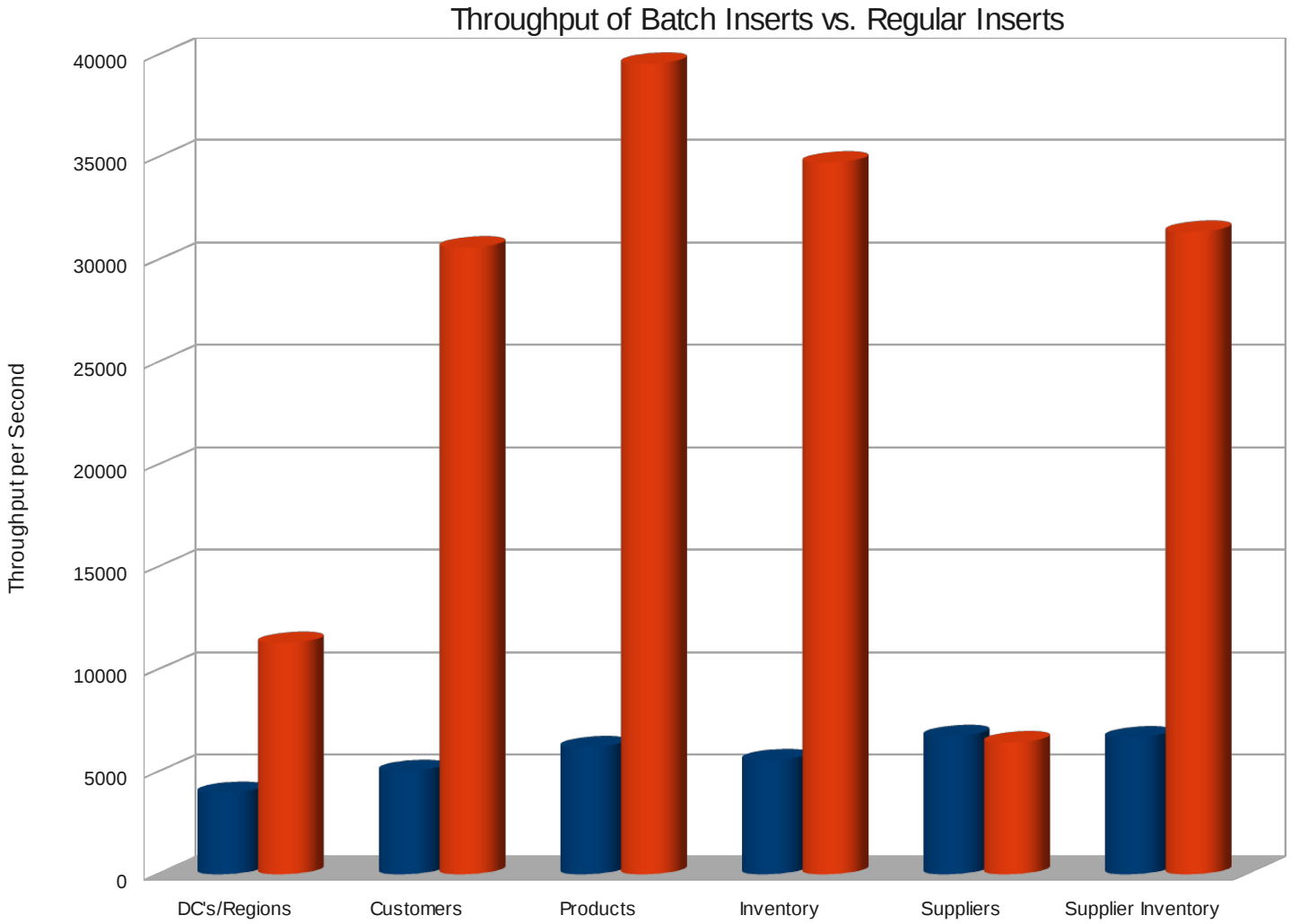

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate

Persistence
JPA
Hibernate

•Important notes on batch inserts:

- You must both specify the ordered inserts parameter to Hibernate, and have a JDBC driver that can rewrite the statements into one.
- The Hibernate property enables the JDBC driver to detect that fact that all the inserts are to the same table, and can be rewritten.
- The Hibernate property orders the inserts statements by their primary key values (actually uses the entities hash code).
- If your JDBC driver does not have the ability to rewrite the statements into one, then throughput will suffer.
  - You are adding the sort overhead before executing the statements, so without benefit of the rewrite of the statement, that shortens the insert time, you just make things take longer.
- This capability does show benefits in OLTP applications, as well as batch applications, but careful testing is in order for OLTP workloads.

Accelerate Your JBoss Enterprise Middleware – Persistence JPA/Hibernate



Accelerate Your JBoss Enterprise Middleware – JCA Container

- For the JCA container we are going to concentrate on data sources.
  - The JCA container is responsible for the integration of our data sources into the application server and provides services for them.
  - The three areas we will concentrate on are:
    - Database connection pooling, the Cached Connection Manager, and Prepared Statement Caching.
  - The size of the database connection pool is directly related to the concurrent execution of queries across your application.
    - To small a pool, and you add to your response times.
    - The default timeout for a database connection is 30 seconds!
      - This is a long time to wait, and you won't get the log message of a timeout until you have real problems with application performance.
  - The Cached Connection Manager provides a debugging capability for leaked database connections.
    - Unless you are doing your own JDBC code, this is typically not needed.
    - It's much easier to turn off in EAP 6, than it was in EAP 5.

Accelerate Your JBoss Enterprise Middleware – JCA Container

```
...
<datasource jndi-name="java:/MySqlDS" pool-name="MySqlDS" use-ccm="false">
    <connection-url>
        jdbc:mysql://localhost:3306/EJB3
     </connection-url>
...
<pool>
    <min-pool-size>200</min-pool-size>
    <max-pool-size>250</max-pool-size>
    <prefill>true</prefill>
</pool>
...
<statement>
    <prepared-statement-cache-size>100</prepared-statement-cache-size>
    <share-prepared-statements>true</share-prepared-statements>
</statement>
...
```


Accelerate Your JBoss Enterprise Middleware – JCA Container

- Important notes on data source configuration:
  - The shared-prepared-statements parameter, when set to true, along with the cache being non-zero, will reuse the same prepared statement if its executed more than once in the same transaction.
    - This may or may not happen in your application, but I have seen lots of applications that do this.
  - Sizing of the prepared statement cache is based on the number of prepared statements your application has within it.
    - Of course, this is by data source, so if you application uses multiple data sources (in JPA, multiple persistence units), then you would configure a cache for each one, and it would be sized based on the prepared statement count for that individual data source.

Accelerate Your JBoss Enterprise Middleware – JMS Provider - HornetQ

- With the introduction of EAP 5.1.2, we had two JMS providers.
  - The default is still JBoss Messaging, but we added HornetQ, the successor to JBoss Messaging as a supported JMS provider.
    - HornetQ is a very high performance, and highly reliable JMS provider.
      - In fact, it holds the world record SPECjms2007 result:
        - http://www.spec.org/jms2007/results/res2011q2/
    - With EAP 6, JBoss Messaging is no longer provided, and the sole JMS provider is HornetQ.
    - There is no longer a database backend for persistence. Instead, there is a high performance journal maintained on disk.

Accelerate Your JBoss Enterprise Middleware – JMS Provider - HornetQ

- Key configuration parameters:
  - journal-type
    - ASYNCIO, NIO
      - The ASYNCIO option specifies using native ASYNC I/O capabilities, plus opens the file using DIRECT I/O, which bypasses the file system buffer cache.
      - The NIO option uses the JDK's NIO API's to write to the journal.
  - journal-directory
    - The placement of the journal files is important, as the default will be relative to the install of the application server, and that may not be the best performing file system to place your persistent messages on.
  - Pooled connection factory:
    - transaction mode
      - Whether to use XA transactions or local transactions.
    - min-pool-size, max-pool-size
      - The session pool size.
      - The sizing of this depends on the number of concurrent MDB's your application may be executing, and relates to the maxSession on those MDB's, or if you are using the JMS api directly, the number of concurrent messages being processed.

Accelerate Your JBoss Enterprise Middleware – JMS Provider - HornetQ

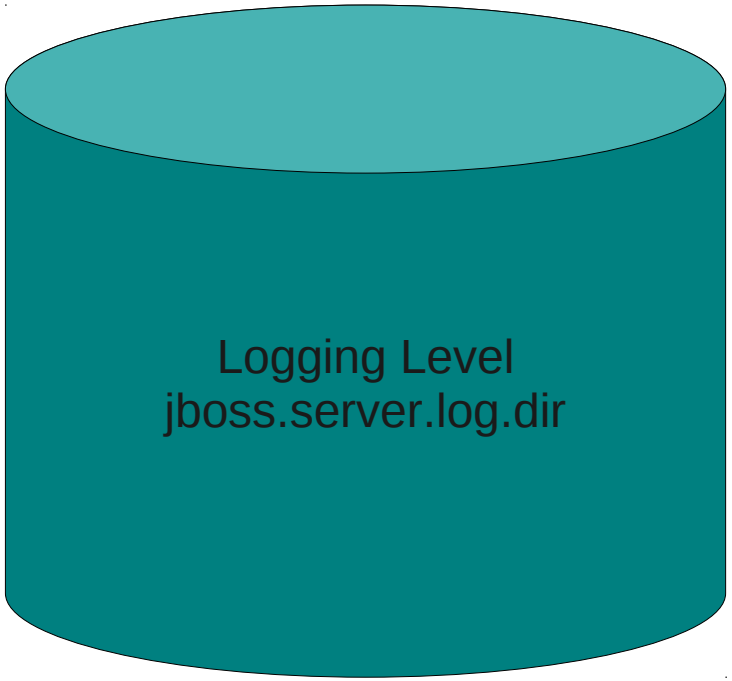
```
<hornetq-server>
...
    <journal-type>ASYNCIO</journal-type>
    <journal-directory>
       <path>/some/absolute/path</path>
    </journal-directory>
...
<pooled-connection-factory name="hornetq-ra">
    <transaction mode="xa"/>
    <min-pool-size>180</min-pool-size>
    <max-pool-size>198</max-pool-size>
...
```


Accelerate Your JBoss Enterprise Middleware – JMS Provider - HornetQ

- Important Notes on configuration items:
  - The path element requires an absolute path name, or it will be relative to the **`jboss.server.data.dir`** value that is globally defined for the application server.
  - The ASYNCIO journal option is by far the highest performing option.
    - This option uses a native code library, since Java cannot use native OS asynchronous I/O (at least not JDK 6), plus it opens the files using O_DIRECT, which no JDK supports.
      - This native code only operates on the Linux platform.
      - In order for this native code layer to load properly, there is also a dependency on having libaio installed.
  - For the transaction mode, which defaults to XA, you can set it to local or none.
    - The default is XA, because in most applications, transactions span MDB's and other components that use a database. You will in most cases having a transaction span two resource manager's as a result.
    - Be very careful about setting this to local or none. Only do this if you are sure the messaging interactions are in their own transactions, or you don't have any persistent messages.

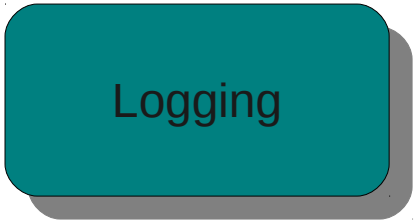
Accelerate Your JBoss Enterprise Middleware – Logging

Logging



if debugEnabled()...

Accelerate Your JBoss Enterprise Middleware – Logging



- Important Notes on logging:
  - There is an entirely new logging implementation in EAP 6.
  - This logging implementation is very high performance, supports internationalization/localization (i18n/L10n), and is available as a public API for applications to use.
  - It doesn't suffer from the problems of some of the other logging implementations where they don't check the log level until they are ready to try and write to the log file.
    - JBoss logging has a fail fast design for the log level.
    - This makes most of the need to wrap statements with debugEnabled() calls, or traceEnabled() calls go away.
      - Having said that, there are still cases where you may be doing something expensive in your logging code, like Object.toString(), and the statements are still there to wrap those log statements so they don't get expensive.

Accelerate Your JBoss Enterprise Middleware – Management

- So, why do we talk about “management”, like other subsystems within EAP?
  - Management is now a built in part of the application server. It's not something external.
  - The management capability of the application server provides a command-line interface (CLI), a domain model for each subsystem and the system as a whole, that is described through XML schema, and a management console that sits on top of both.
  - The CLI is completely scriptable and is accessible over the network (with proper security credentials).
- So, what does this have to do with performance?
  - Well, there are some really nice performance related metrics that are exposed through the management capability.
  - We will talk about two of them, specifically data source metrics, and JPA metrics.
    - There are others, but time is short, and these will probably get used the most heavily out of everything that is available.

Accelerate Your JBoss Enterprise Middleware – Management


```
[root@jbosstesting bin]# ls
add-user.bat        appclient.sh     domain.sh      JBossPublicKey.RSA  product.conf
 standalone.conf.bat  wsconsume.sh
add-user.sh         client           init.d         jconsole.bat        run.bat
 standalone.sh        wsprovide.bat
appclient.bat       domain.bat       jboss-cli.bat  jconsole.sh         run.sh
 vault.bat            wsprovide.sh
appclient.conf      domain.conf      jboss-cli.sh   jdr.bat             standalone.bat
 vault.sh
appclient.conf.bat  domain.conf.bat  jboss-cli.xml  jdr.sh              standalone.conf
 wsconsume.bat
[root@jbosstesting bin]# ./jboss-cli.sh
You are disconnected at the moment. Type 'connect' to connect to the server or 'help'
for the list of supported commands.
[disconnected /]connect jbosstesting.miller.org
Authenticating against security realm: ManagementRealm
Username: admin
Password:
[standalone@jbosstesting.miller.org:9999 /]
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 /]ls
core-service                          deployment                          extension
interface                             path                                socket-
binding-group
subsystem                             system-property                     launch-
type=STANDALONE
management-major-version=1            management-minor-version=2
name=jbosstesting
namespaces=[]                         process-type=Server                 product-
name=EAP
product-version=6.0.0.GA              profile-name=undefined              release-
codename=Steropes
release-version=7.1.2.Final-redhat-1  running-mode=NORMAL                 schema-
locations=[]
server-state=running
[standalone@jbosstesting.miller.org:9999 /]
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 /] cd deployment
[standalone@jbosstesting.miller.org:9999 deployment] ls
specj.ear
[standalone@jbosstesting.miller.org:9999 deployment] cd specj.ear
[standalone@jbosstesting.miller.org:9999 deployment=specj.ear] ls
subdeployment
subsystem
content=[{"path" => "deployments/specj.ear","relative-to" =>
"jboss.server.base.dir","archive" => true}]
enabled=true
name=specj.ear
persistent=false
runtime-name=specj.ear
status=OK
[standalone@jbosstesting.miller.org:9999 deployment=specj.ear] cd subdeployment
[standalone@jbosstesting.miller.org:9999 subdeployment] ls
specj.jar      specj.war      supplier.war
[standalone@jbosstesting.miller.org:9999 subdeployment] cd specj.jar
[standalone@jbosstesting.miller.org:9999 subdeployment=specj.jar] ls
subsystem
[standalone@jbosstesting.miller.org:9999 subdeployment=specj.jar] cd subsystem
[standalone@jbosstesting.miller.org:9999 subsystem] ls
ejb3          jpa           web           webservices
[standalone@jbosstesting.miller.org:9999 subsystem]
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 subsystem] cd jpa
[standalone@jbosstesting.miller.org:9999 subsystem=jpa] ls
hibernate-persistence-unit
[standalone@jbosstesting.miller.org:9999 subsystem=jpa] cd hibernate-persistence-unit
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-unit] ls
specj.ear/specj.jar#Loader     specj.ear/specj.jar#Mfg        specj.ear/specj.jar#Order
     specj.ear/specj.jar#Supplier
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-unit] cd
specj.ear\/specj.jar#Order
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order] ls
collection                                                  entity
entity-cache                                                query-cache
close-statement-count=0                                     collection-fetch-count=0
collection-load-count=0                                     collection-recreated-count=0
collection-remove-count=0                                   collection-update-count=0
completed-transaction-count=0                               connect-count=0
enabled=false                                               entity-delete-count=0
entity-fetch-count=0                                        entity-insert-count=0
entity-load-count=0                                         entity-update-count=0
flush-count=0                                               optimistic-failure-count=0
prepared-statement-count=0                                  query-cache-hit-count=0
query-cache-miss-count=0                                    query-cache-put-count=0
query-execution-count=0                                     query-execution-max-time=0
query-execution-max-time-query-string=undefined             scoped-unit-
name=specj.ear/specj.jar#Order
second-level-cache-hit-count=0                              second-level-cache-miss-count=0
second-level-cache-put-count=0                              session-close-count=0
session-open-count=0                                        successful-transaction-count=0
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order] :write-attribute(name=enabled,value=true)
{"outcome" => "success"}
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order]ls
collection                                          entity
entity-cache                                        query-cache
close-statement-count=0                             collection-fetch-count=0
collection-load-count=0                             collection-recreated-count=0
collection-remove-count=0                           collection-update-count=0
completed-transaction-count=0                       connect-count=0
enabled=true                                        entity-delete-count=0
entity-fetch-count=0                                entity-insert-count=0
entity-load-count=0                                 entity-update-count=0
flush-count=0                                       optimistic-failure-count=0
prepared-statement-count=0                          query-cache-hit-count=0
query-cache-miss-count=0                            query-cache-put-count=0
query-execution-count=0                             query-execution-max-time=0
query-execution-max-time-query-string=undefined     scoped-unit-
name=specj.ear/specj.jar#Order
second-level-cache-hit-count=0                      second-level-cache-miss-count=0
second-level-cache-put-count=0                      session-close-count=0
session-open-count=0                                successful-transaction-count=0
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order]
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order] ls
...
entity-delete-count=43279
entity-fetch-count=36689
entity-insert-count=57687
entity-load-count=9338637
entity-update-count=49389
flush-count=47440
optimistic-failure-count=0
prepared-statement-count=471308
query-cache-hit-count=282440
query-cache-miss-count=35381
query-cache-put-count=35381
query-execution-count=91501
query-execution-max-time=9989
query-execution-max-time-query-string=select COUNT(a) from CustomerInventory a
scoped-unit-name=specj.ear/specj.jar#Order
second-level-cache-hit-count=56526784
second-level-cache-miss-count=28403
second-level-cache-put-count=265045
session-close-count=669836
session-open-count=669844
successful-transaction-count=47441
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order]
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order] :read-resource(include-runtime=true,recursive=true)
{
        ...
        "query-cache-hit-count" => 594027L,
        "query-cache-miss-count" => 48702L,
        "query-cache-put-count" => 48702L,
        ...
        "second-level-cache-hit-count" => 118996815L,
        "second-level-cache-miss-count" => 37320L,
        "second-level-cache-put-count" => 360120L,
        ...
        "collection" => {
            "org.spec.jent.ejb.orders.entity.Customer.customerInventories" => {
                "collection-fetch-count" => 0L,
                "collection-load-count" => 405550L,
                "collection-recreated-count" => 0L,
                "collection-remove-count" => 1L,
                "collection-update-count" => 66516L
            },
            "org.spec.jent.ejb.orders.entity.Order.orderLines" => {
                "collection-fetch-count" => 14292L,
                "collection-load-count" => 48602L,
                "collection-recreated-count" => 22261L,
                "collection-remove-count" => 7980L,
                "collection-update-count" => 0L
            }
        },
        ...
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order] :read-resource(include-runtime=true,recursive=true)
{
...
"entity-cache" => {
            "org.hibernate.cache.internal.StandardQueryCache" => {
                "element-count-in-memory" => 84L,
                "hit-count" => 594214L,
                "miss-count" => 48705L,
                "put-count" => 48705L
            },
            "org.spec.jent.ejb.orders.entity.Item" => {
                "element-count-in-memory" => 26442L,
                "hit-count" => 119034411L,
                "miss-count" => 37323L,
                "put-count" => 360162L
            },
            "org.hibernate.cache.spi.UpdateTimestampsCache" => {
                "element-count-in-memory" => 4L,
                "hit-count" => 0L,
                "miss-count" => 0L,
                "put-count" => 0L
            }
        },
...
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 hibernate-persistence-
unit=specj.ear/specj.jar#Order] :read-resource(include-runtime=true,recursive=true)
{
...
"query-cache" => {
"SELECT_space_i_space_FROM_space_Item_space_i_space_WHERE_space_i.category_equal__colon
_category_space_ORDER_space_BY_space_i.id" => {
                "query-cache-hit-count" => 594116L,
                "query-cache-miss-count" => 48702L,
                "query-cache-put-count" => 48702L,
                "query-execution-average-time" => 31L,
                "query-execution-count" => 48702L,
                "query-execution-max-time" => 3639L,
                "query-execution-min-time" => 3L,
                "query-execution-row-count" => 9740400L,
                "query-name" => "SELECT i FROM Item i WHERE i.category=:category ORDER
BY i.id"
            },
...
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 /] ls
core-service                        deployment                          extension
interface                           path                                socket-
binding-group
subsystem                           system-property                     launch-
type=STANDALONE
management-major-version=1          management-minor-version=2
name=jbosstesting
namespaces=[]                       process-type=Server                 product-
name=EAP
product-version=6.0.0.GA            profile-name=undefined              release-
codename=Steropes
release-version=7.1.2.Final-redhat-1   running-mode=NORMAL              schema-
locations=[]
server-state=running
[standalone@jbosstesting.miller.org:9999 /] cd subsystem
[standalone@jbosstesting.miller.org:9999 subsystem] ls
cmp                 configadmin         datasources         deployment-scanner   ee
                ejb3
infinispan          jacorb              jaxr                jaxrs                jca
                jdr
jmx                 jpa                 jsr77               logging
mail                messaging
naming              osgi                pojo                remoting
resource-adapters   sar
security            threads             transactions        web
webservices         weld
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 subsystem] cd datasources
[standalone@jbosstesting.miller.org:9999 subsystem=datasources] ls
data-source      jdbc-driver      xa-data-source
[standalone@jbosstesting.miller.org:9999 subsystem=datasources] cd xa-data-source
[standalone@jbosstesting.miller.org:9999 xa-data-source] ls
SPECjLoaderDS     SPECjMfgDS        SPECjOrderDS      SPECjSupplierDS
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 xa-data-source] cd SPECjOrderDS
[standalone@jbosstesting.miller.org:9999 xa-data-source=SPECjOrderDS] ls
statistics                                          xa-datasource-properties
allocation-retry=undefined                          allocation-retry-wait-
millis=undefined
allow-multiple-users=undefined                      background-validation=undefined
background-validation-millis=undefined              blocking-timeout-wait-
millis=undefined
check-valid-connection-sql=undefined                driver-name=mysql
enabled=false                                       exception-sorter-class-
...
```


Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 xa-data-source] cd SPECjOrderDS
[standalone@jbosstesting.miller.org:9999 xa-data-source=SPECjOrderDS] ls
statistics                                          xa-datasource-properties
allocation-retry=undefined                          allocation-retry-wait-
millis=undefined
allow-multiple-users=undefined                      background-validation=undefined
background-validation-millis=undefined              blocking-timeout-wait-
millis=undefined
check-valid-connection-sql=undefined                driver-name=mysql
enabled=false                                       exception-sorter-class-
...
```

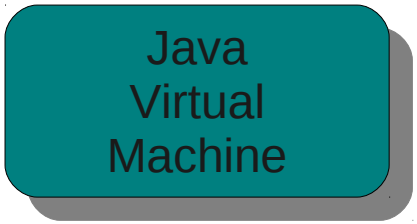

Accelerate Your JBoss Enterprise Middleware – Management


```
[standalone@jbosstesting.miller.org:9999 xa-data-source=SPECjOrderDS] :read-
resource(include-runtime=true,recursive=true)
{
    "outcome" => "success",
    "result" => {
        ...
        "driver-name" => "mysql",
        ...
        "jndi-name" => "java:/jdbc/SPECjOrderDS",
        "jta" => true,
        "max-pool-size" => 550,
        "min-pool-size" => 50,
        ...
        "pool-prefill" => true,
       ...
        "prepared-statements-cache-size" => 32L,
        ...
        "share-prepared-statements" => true,
        "spy" => false,
        ...
        "transaction-isolation" => "TRANSACTION_READ_COMMITTED",
        ...
        "use-ccm" => false,
       ...
        "statistics" => {
            "jdbc" => {
                "PreparedStatementCacheAccessCount" => "2820107",
                "PreparedStatementCacheAddCount" => "16984",
                "PreparedStatementCacheCurrentSize" => "7705",
                "PreparedStatementCacheDeleteCount" => "9279",
                "PreparedStatementCacheHitCount" => "2803123",
                "PreparedStatementCacheMissCount" => "0"
            },
            "pool" => {
                "ActiveCount" => "502",
```

Accelerate Your JBoss Enterprise Middleware – Management


```
...
        "statistics" => {
            "jdbc" => {
                "PreparedStatementCacheAccessCount" => "2820107",
                "PreparedStatementCacheAddCount" => "16984",
                "PreparedStatementCacheCurrentSize" => "7705",
                "PreparedStatementCacheDeleteCount" => "9279",
                "PreparedStatementCacheHitCount" => "2803123",
                "PreparedStatementCacheMissCount" => "0"
            },
            "pool" => {
                "ActiveCount" => "502",
                "AvailableCount" => "550",
                "AverageBlockingTime" => "1",
                "AverageCreationTime" => "667",
                "CreatedCount" => "502",
                "DestroyedCount" => "0",
                "MaxCreationTime" => "1804",
                "MaxUsedCount" => "498",
                "MaxWaitTime" => "98",
                "TimedOut" => "0",
                "TotalBlockingTime" => "585",
                "TotalCreationTime" => "334901"
            }
        },
        ...
    }
}
[standalone@jbosstesting.miller.org:9999 xa-data-source=SPECjOrderDS]
```

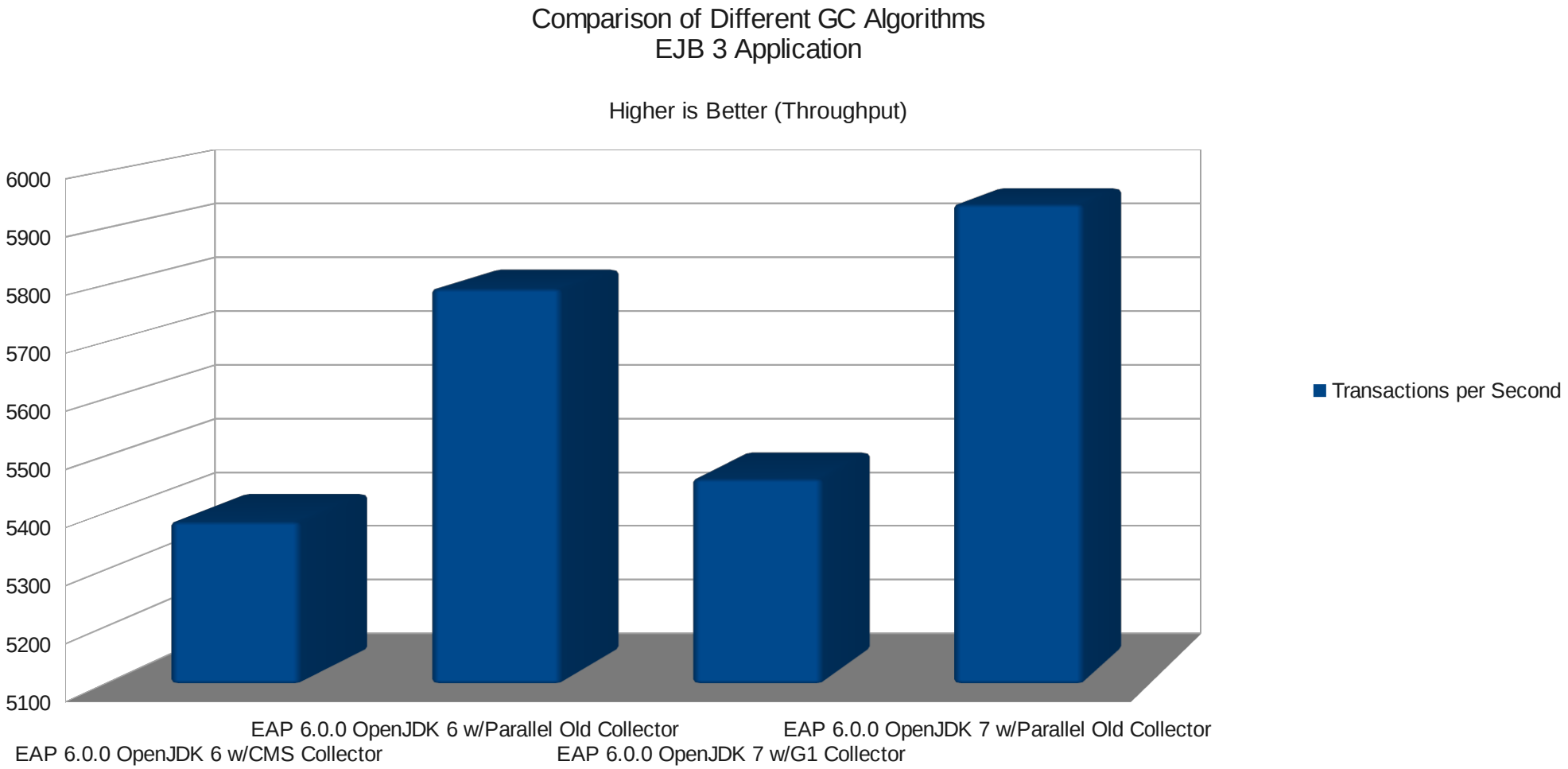

Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



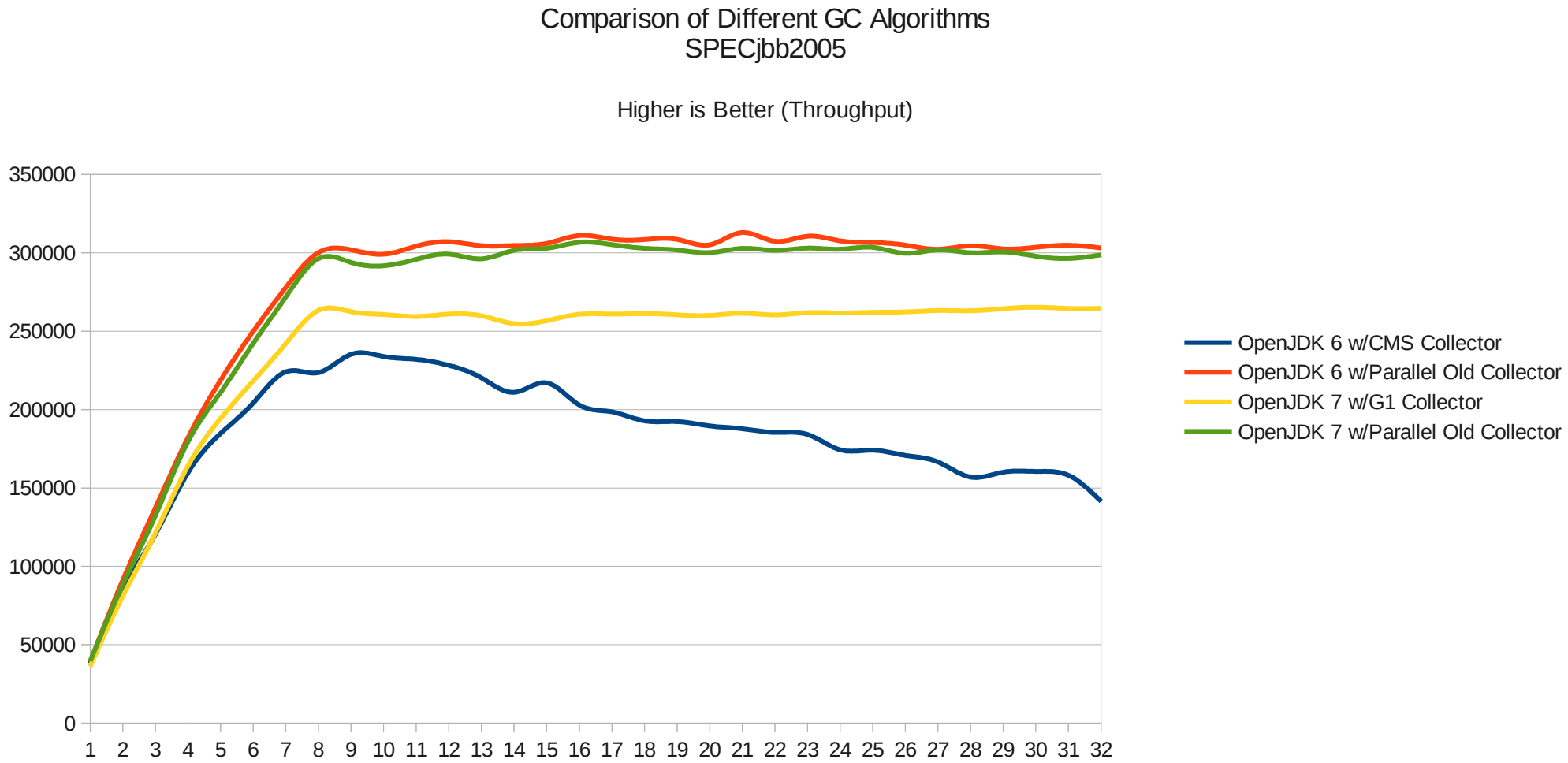
• The Java Virtual Machine is our key piece of software, as the entire platform is completely dependent on it.

- Tuning the JVM can be difficult.
- Start simple!
- Test one thing at a time!
- Understand your goals!
- Use the 64-bit JVM with Compressed Oops.
  - Seems like the 32-bit JVM is on its way out, with perhaps the exception of virtualized environments that have severe memory contraints.
- Key topics:
  - Garbage collection algorithms.
    - CMS vs. Throughput on JDK 6.
    - G1 vs. Throughput on JDK 7.
  - Compressed Oops.
  - Large Page Memory.

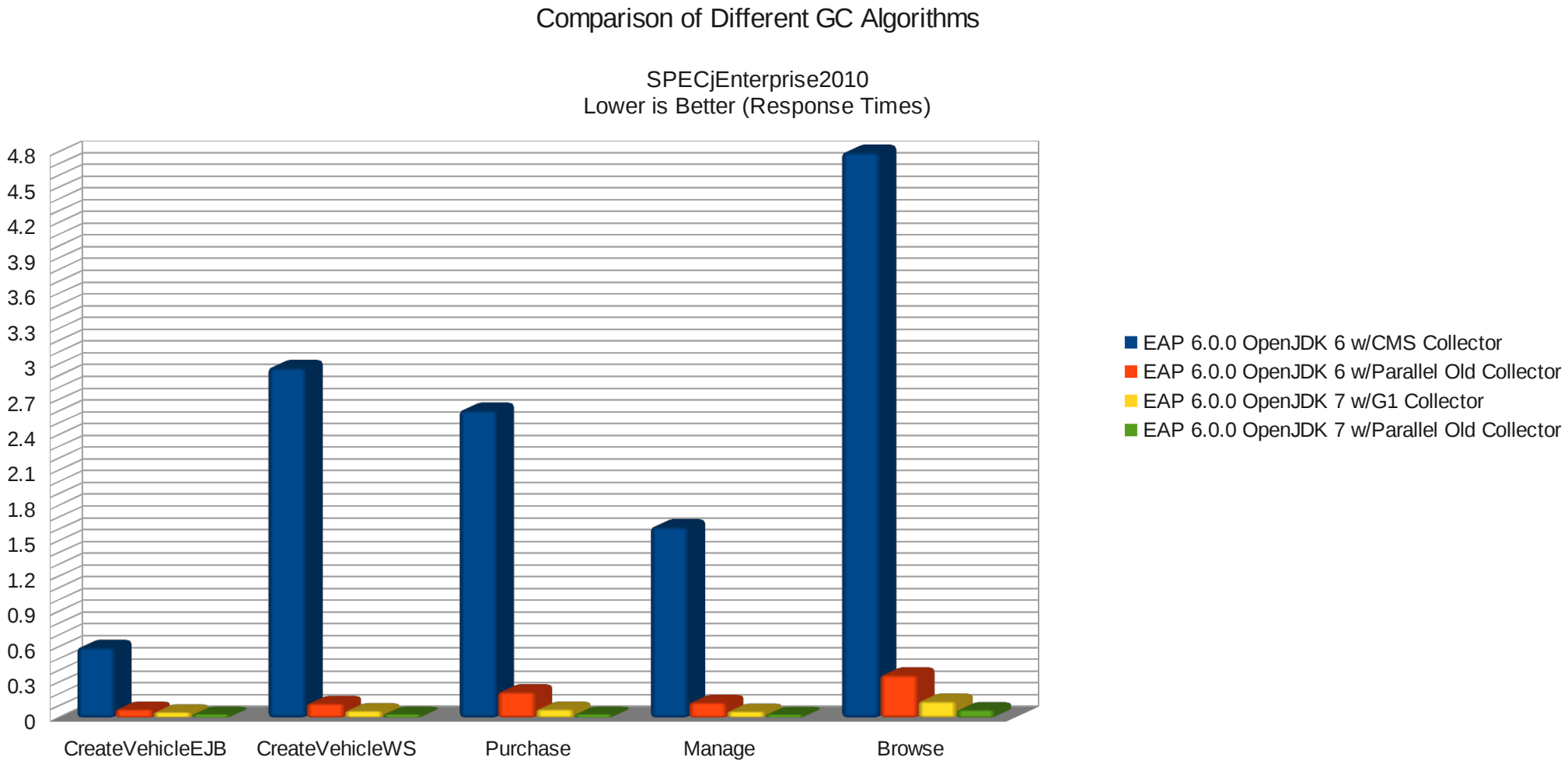
Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



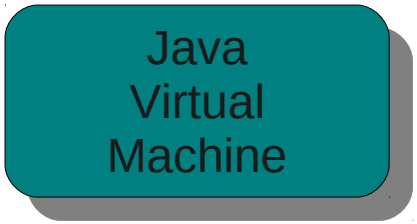
Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



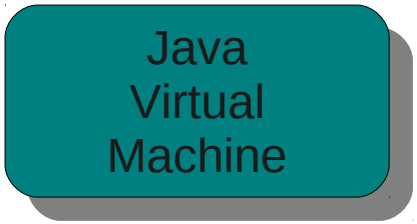
Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



- What are Compressed Oops?
  - An "oop", or ordinary object pointer. An oop is normally the same size as the word size of the server (64-bits on X86_64, and 32-bits on X86, as an example).
  - So, a compressed oop is an object pointer that instead of using the word size (e.g. 64-bits), instead is compressed to 32-bits, or as a 32-bit offset from the 64-bit heap base address.
    - This allows a heap size of about 32 gigabytes (because they are an offset, not a direct memory pointer).
  - Compressed oops is enabled by default in update 23 and above of JDK 6. For JDK 7 its on by default as long as the maximum heap size is less than 32 GB's.
    - EAP 6.0.0.GA sets the command-line option to enable compressed oops for any Hotspot based 64-bit virtual machine.
      - The flag is -XX:+UseCompressedOops.

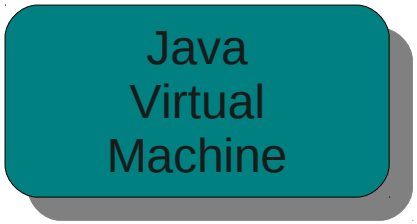
•All tests in this presentation have compressed oops on, as its on by default now.

Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



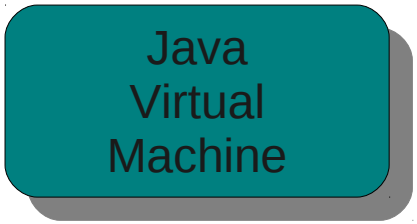
- What is large page memory?
  - The normal memory page size in the OS is usually 4k. When you consider the memory footprints of modern servers, there are an awful lot of memory pages that the OS has to manage.
  - Large page memory, is a larger memory page than 4k.
  - Typically, on X86_64 systems, this is 2 MB.
  - Newer X86_64 systems can support 1 GB page sizes.
  - Another attribute of large pages (called HugeTLB in Linux), is that they are locked into physical memory, and cannot be swapped to disk.
  - This is a great attribute for the JVM.
    - If you have ever experienced the JVM heap being swapped to disk, you know this is a situation that often leads to the JVM crashing.
- How do we take advantage of large page memory?

Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



- The Sun JVM, as well as OpenJDK, requires the following option, passed on the command-line, to use large pages:
  - -XX:+UseLargePages
- There is also something in Linux called “Transparent huge pages”.
  - Transparent huge pages allows the operating system (Linux) to evaluate memory usage of processes and dynamically move from regular pages to large pages (consolidates many 4k pages into a large page).
  - For some workloads this may be sufficient. For other, the static configuration, and usage will be better.
  - If you have transparent huge pages turned on in the Linux kernel, you should not specify the JVM argument to use large pages. Only if you opt for the static configuration, which I'll walk through next.
- ***From my testing so far, I would not recommend having both statically defined large pages, and transparent huge page support on at the same time!***
  - ***To turn off transparent huge pages, you can set a boot parameter in grub.conf as follows:***

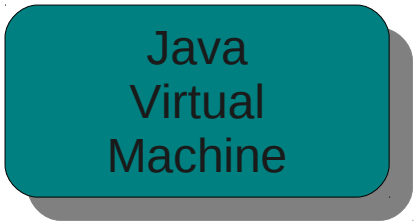
Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



• To turn off transparent huge pages, you can set a boot parameter in grub.conf as follows:

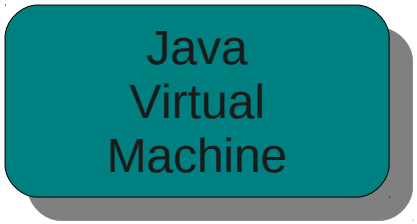
```
title Red Hat Enterprise Linux (2.6.32-220.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-220.el6.x86_64 ro root=UUID=f8196a3a-1f1a-47f3-9141-1d33e3da4454
rd_NO_LUKS rd_NO_LVM LANG=en_US.UTF-8 rd_NO_MD quiet SYSFONT=latarcyrheb-sun16 rhgb
crashkernel=auto  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM transparent_hugepage=never
        initrd /initramfs-2.6.32-220.el6.x86_64.img
```


Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



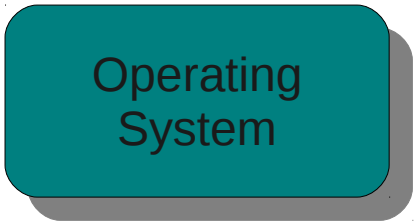
- The Oracle JVM, as well as OpenJDK, requires the following option:
  - -XX:+UseLargePages
    - The Oracle instructions leave it at that and you will most likely get the following error:
      - Failed to reserve shared memory (error-no=12).
  - Next, you set the following in /etc/sysctl.conf
    - kernel.shmmax = *n*
      - Where ***n*** is equal to the number of bytes of the maximum shared memory segment allowed on the system. You should set it to perhaps 3 times the amount of physical memory.
        - Setting this value smaller, may result in error-no=22 on startup of the JVM. This error, is “no space left on device”, and is a rather new phenomenon on the Linux kernel.
    - vm.nr_hugepages = *n*
      - Where ***n*** is equal to the number of large pages. You will need to look up the large page size in /proc/meminfo.
    - vm.huge_tlb_shm_group = *gid*
      - Where ***gid*** is a shared group id for the users you want to have access to the large pages.

Accelerate Your JBoss Enterprise Middleware – Java Virtual Machine



- Next, set the following:
  - In `/etc/security/limits.conf`
    - `<username> soft memlock n`
    - `<username> hard memlock n`
      - Where ***<username>*** is the runtime user of the JVM.
      - Where ***n*** is the number of pages from `vm.nr_hugepages` * the page size in KB from `/proc/meminfo`.
        - The value for ***n*** can also be **unlimited**.
  - You can now enter the command `sysctl -p`, and everything will be set and survive a reboot.
    - You can tell that the large pages are allocated by looking at `/proc/meminfo`, and seeing a non-zero value for `HugePages_Total`.
      - This may fail without a reboot, because when the OS allocates these pages, it must find contiguous memory for them.
  - **WARNING**: when you allocate large page memory, it is not available to applications in general and you system will look and act like it has that amount of memory removed from it!

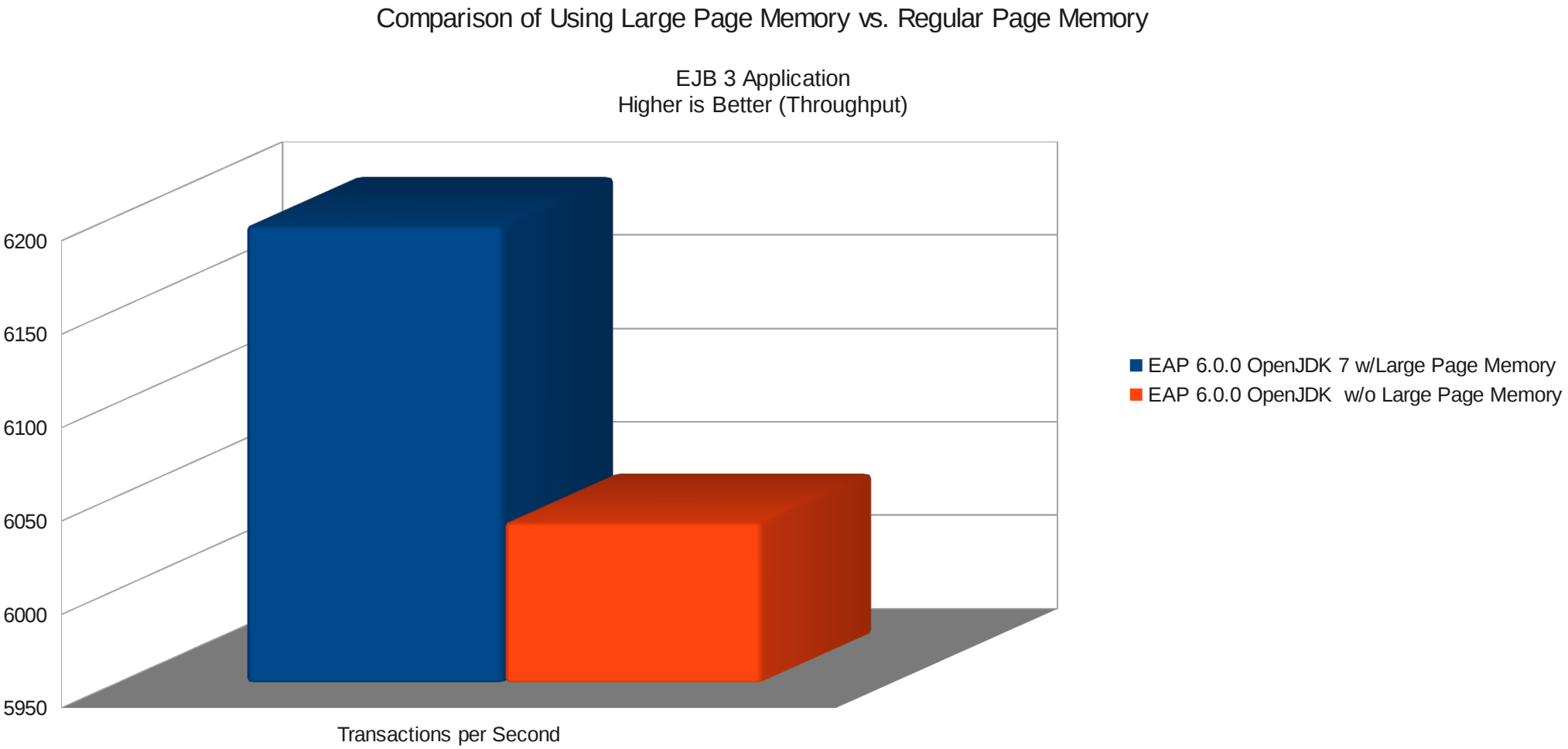
Accelerate Your JBoss Enterprise Middleware – Operating System



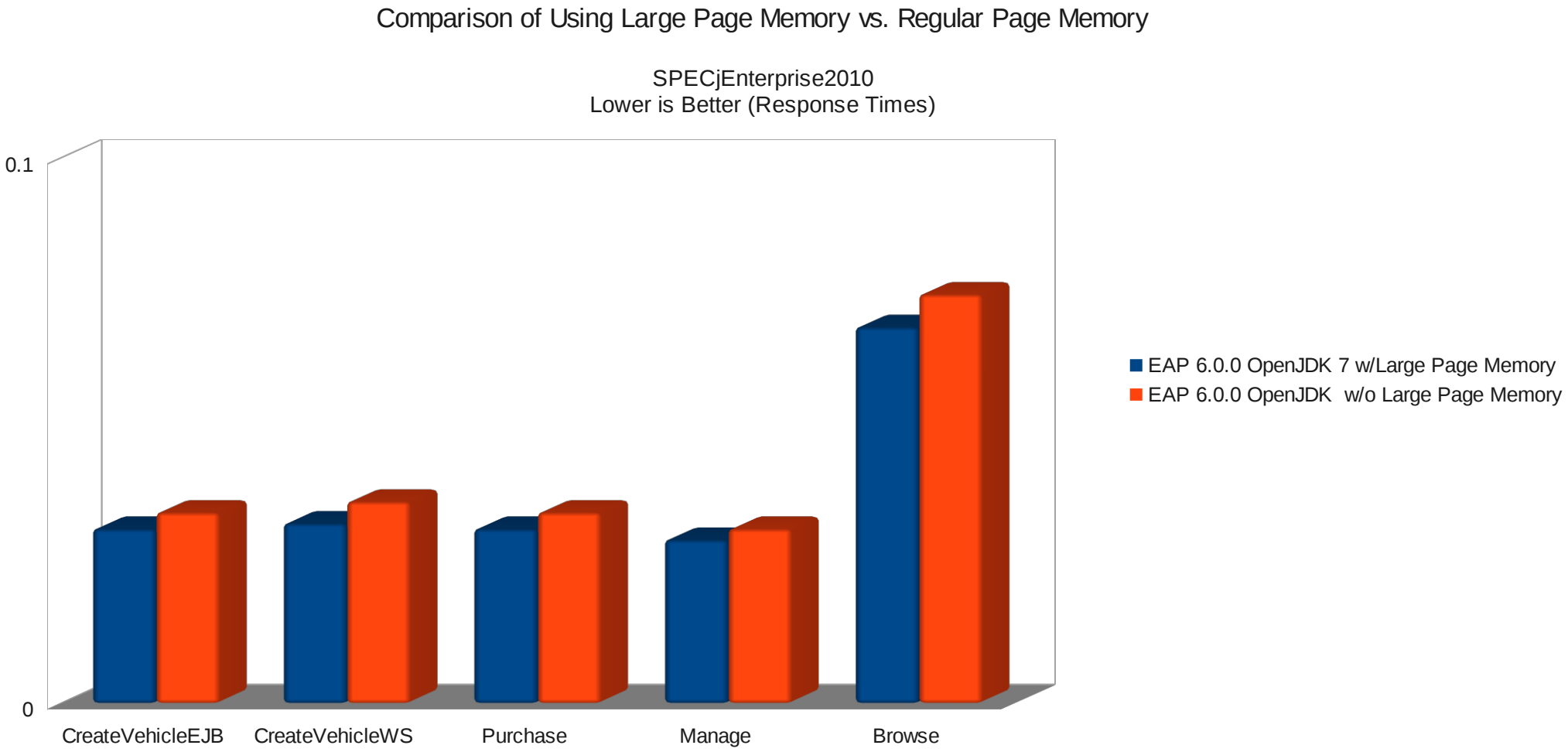
• In this case, when we talk about operating system, we are going to be talking specifically about Red Hat Enterprise Linux.

- All my testing was done on RHEL 6.2 (6.3 is out now, but I did not have time to upgrade my test server and rerun all my tests).
- We will go over large page memory setup.
- We will cover NUMA architecture, and how you can leverage it for EAP 6.

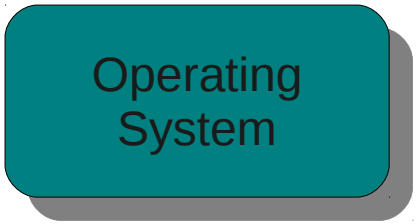
Accelerate Your JBoss Enterprise Middleware – Operating System



Accelerate Your JBoss Enterprise Middleware – Operating System

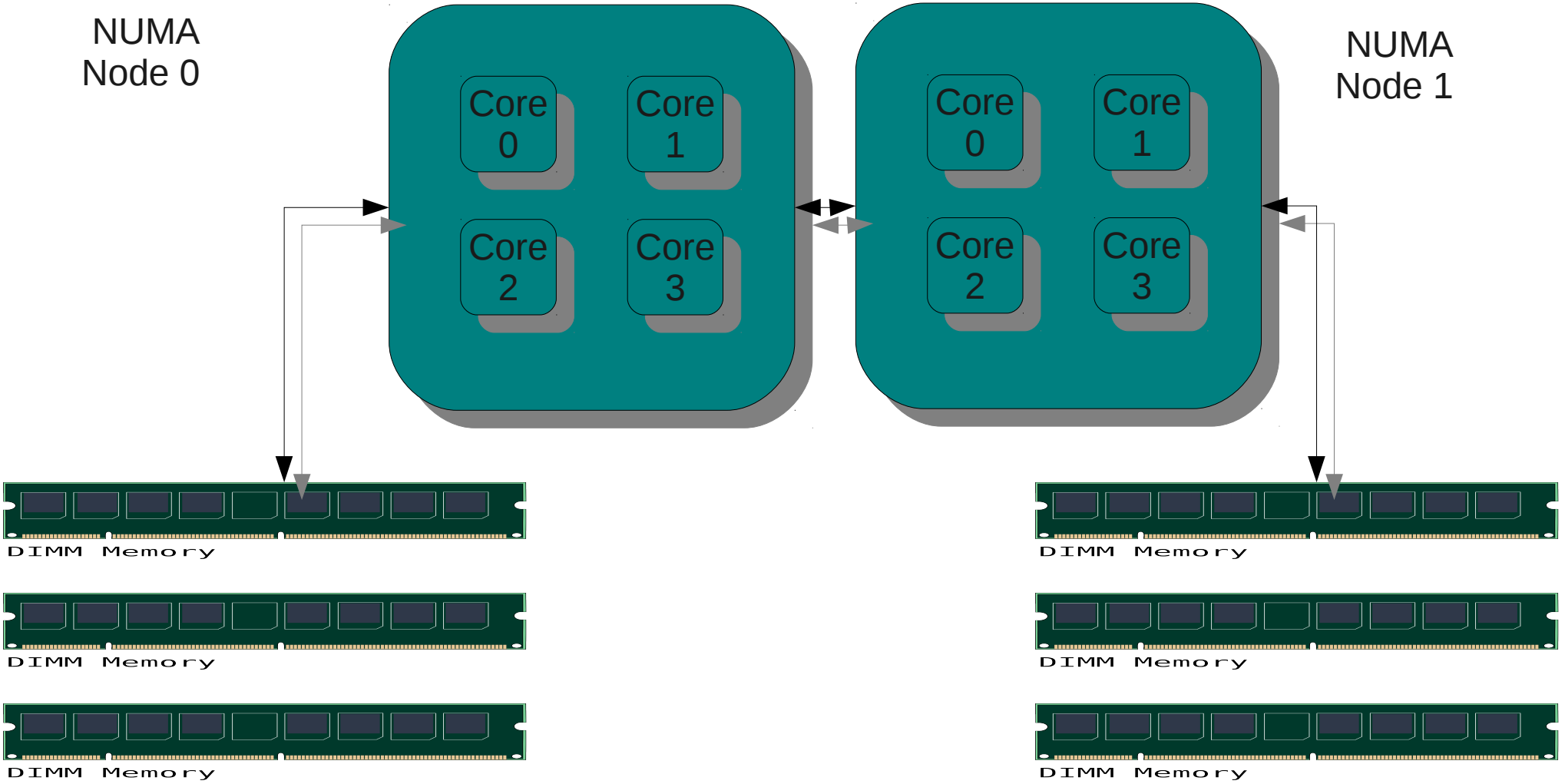


Accelerate Your JBoss Enterprise Middleware – Operating System

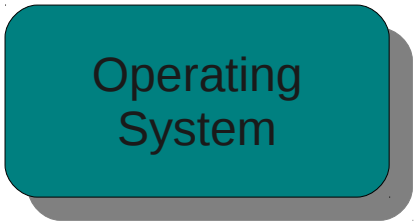


- What is NUMA?
  - It stands for non-uniform memory architecture.
  - Why is it important?
    - It's important because all newer x86 architecture servers are based on NUMA.
  - The memory architecture of a NUMA system is laid out so that each socket (with its cores) is attached directly to a single bank of memory.
    - Accessing this bank of memory has the lowest latency of all memory accesses.
    - To access memory from any other bank of memory, it has to go through the other sockets to access the memory adding significant latency.
    - The best performance comes from keeping processing running on a single socket, and having its memory needs satisfied through the local memory bank attached to that socket.

Accelerate Your JBoss Enterprise Middleware – Operating System



Accelerate Your JBoss Enterprise Middleware – Operating System



- How do you take advantage of NUMA?
  - First, you have to understand your NUMA hardware layout.
  - Second, you have to start the JVM with numactl.
  - Third, you have to supply numactl the policy information necessary to bind the process and its threads to the NUMA node, and its memory accesses to the local memory of that NUMA node.
  - Fourth, if you are using large page memory, you need to understand how many large pages are on each NUMA node.
  - Fifth, you have to understand the number of threads to use for GC, as the default JVM ergonomics will not apply.
    - In the test example I show, I set the number of GC threads to the number of virtual cores (including hyper-threading) that were on the NUMA node I bound the JVM too.
- So, let's take a look at the commands you have to issue to accomplish all five points above.

Accelerate Your JBoss Enterprise Middleware – Operating System

```
[root@jbosstesting ~]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14
node 0 size: 12277 MB
node 0 free: 710 MB
node 1 cpus: 1 3 5 7 9 11 13 15
node 1 size: 12287 MB
node 1 free: 225 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```


Accelerate Your JBoss Enterprise Middleware – Operating System

```
if [ "x$LAUNCH_JBOSS_IN_BACKGROUND" = "x" ]; then
      # Execute the JVM in the foreground
      eval numactl --membind 0 –cpunodebind 0 \"$JAVA\" -D\"[Standalone]\" $JAVA_OPTS \
         \"-Dorg.jboss.boot.log.file=$JBOSS_LOG_DIR/boot.log\" \
         \"-Dlogging.configuration=file:$JBOSS_CONFIG_DIR/logging.properties\" \
         -jar \"$JBOSS_HOME/jboss-modules.jar\" \
         -mp \"${JBOSS_MODULEPATH}\" \
         -jaxpmodule "javax.xml.jaxp-provider" \
         org.jboss.as.standalone \
         -Djboss.home.dir=\"$JBOSS_HOME\" \
         -Djboss.server.base.dir=\"$JBOSS_BASE_DIR\" \
         "$@"
      JBOSS_STATUS=$?
   else
      # Execute the JVM in the background
      eval numactl --membind 0 –cpunodebind 0 \"$JAVA\" -D\"[Standalone]\" $JAVA_OPTS \
         \"-Dorg.jboss.boot.log.file=$JBOSS_LOG_DIR/boot.log\" \
         \"-Dlogging.configuration=file:$JBOSS_CONFIG_DIR/logging.properties\" \
         -jar \"$JBOSS_HOME/jboss-modules.jar\" \
         -mp \"${JBOSS_MODULEPATH}\" \
         -jaxpmodule "javax.xml.jaxp-provider" \
         org.jboss.as.standalone \
         -Djboss.home.dir=\"$JBOSS_HOME\" \
         -Djboss.server.base.dir=\"$JBOSS_BASE_DIR\" \
         "$@" "&"
      JBOSS_PID=$!
...
```

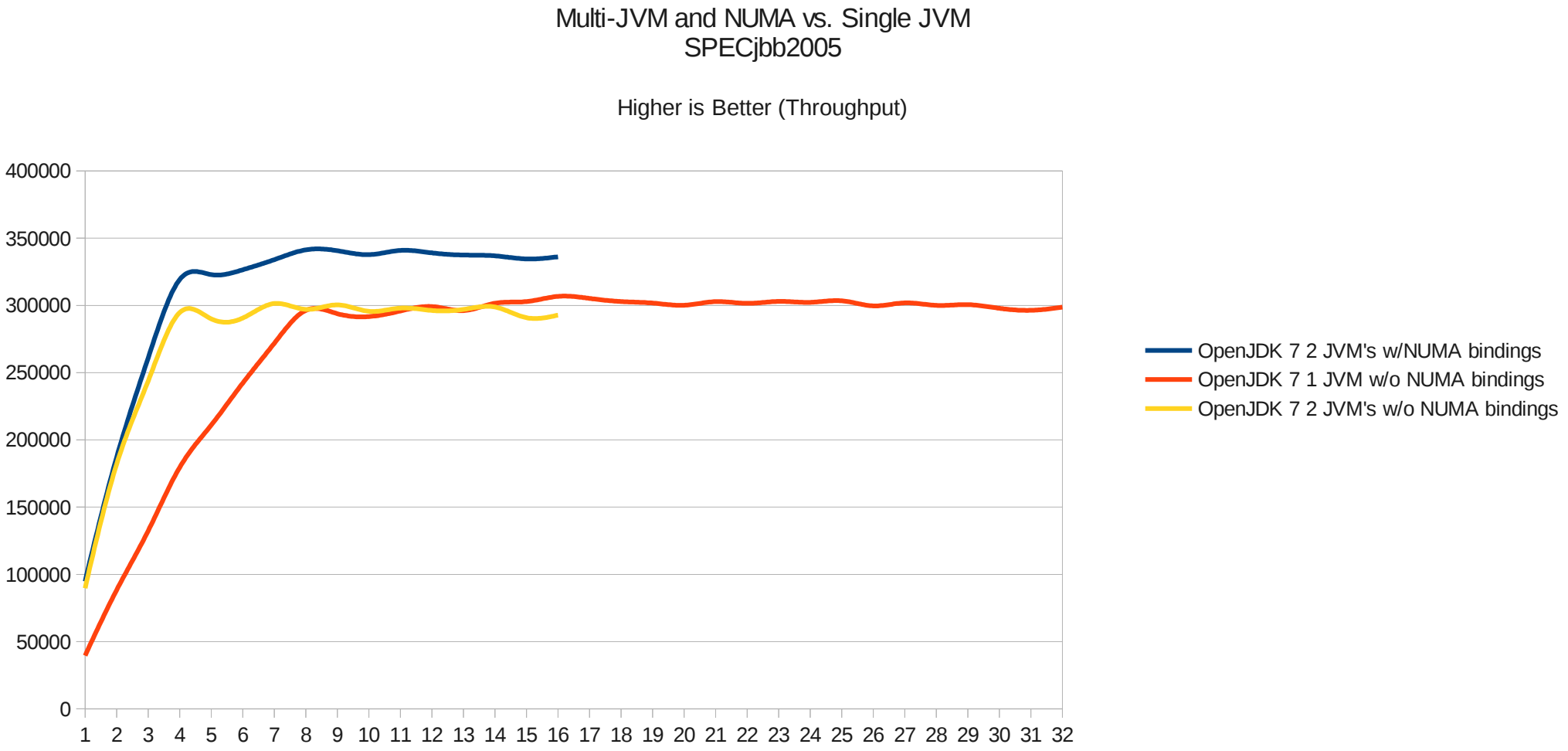

Accelerate Your JBoss Enterprise Middleware – Operating System

```
[root@jbosstesting hugepages-2048kB]# pwd
/sys/devices/system/node/node0/hugepages/hugepages-2048kB
[root@jbosstesting hugepages-2048kB]# cat nr_hugepages
5376
[root@jbosstesting hugepages-2048kB]#

...

[root@jbosstesting hugepages-2048kB]# pwd
/sys/devices/system/node/node1/hugepages/hugepages-2048kB
[root@jbosstesting hugepages-2048kB]# cat nr_hugepages
5376
[root@jbosstesting hugepages-2048kB]#
```


Accelerate Your JBoss Enterprise Middleware – Operating System



References

- This presentation has shown a lot of XML fragments. The following link is to the schema definitions for the application server in GITHUB:
  - https://github.com/jbossas/jboss-as/tree/master/build/src/main/resources/docs/schema

LIKE US ON FACEBOOK

www.facebook.com/redhatinc

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#redhat

READ THE BLOG

summitblog.redhat.com

GIVE US FEEDBACK

www.redhat.com/summit/survey