FRANKFURT, GERMANY

Liferay Performance Tuning

Tips, tricks, and best practices

Michael C. Han
Liferay, INC

A Survey

Why?

- Considering using Liferay, curious about performance.
- Currently implementing and thinking ahead.
- Running Liferay in production but would like to improve.
- Living the dream. Just need a place to nap after lunch.

This talk covers...

1. Strategy
2. Holistic Performance
3. High Level Portal Tuning
4. Runtime Settings

This talk does not cover...

- Step by step of performance profiling
- Configuring your Liferay cluster
- Designing your High Availability environment
- Assessing the performance which users experience

Performance Strategy

What's your goal?

A few people, some cargo, some of the time

What's your goal?

Lots of people, 24/7, high throughput, reliable

What's your goal?

Big data, heavy process, serious integrations

When to consider performance...

Make it Work, Make it Right, Make it Fast

Development Lifecycle

- Architecture/System Design
- Implementation Phase
- Quality Assurance
- Deployment

Holistic Performance

Beyond the Portal

- Application server resources
- Database deployment architecture
- Content Delivery Networks
- Enterprise Services (search, web services, etc)

Content Delivery Networks

- CDNs Replicate content to servers closer to end user to reduce latencies
- Load static JS, images, etc. outside of Portal Application Server
- Reduces load on application servers

- Enable with *cdn.host.http* and *cdn.host.https*

Application Server Resources

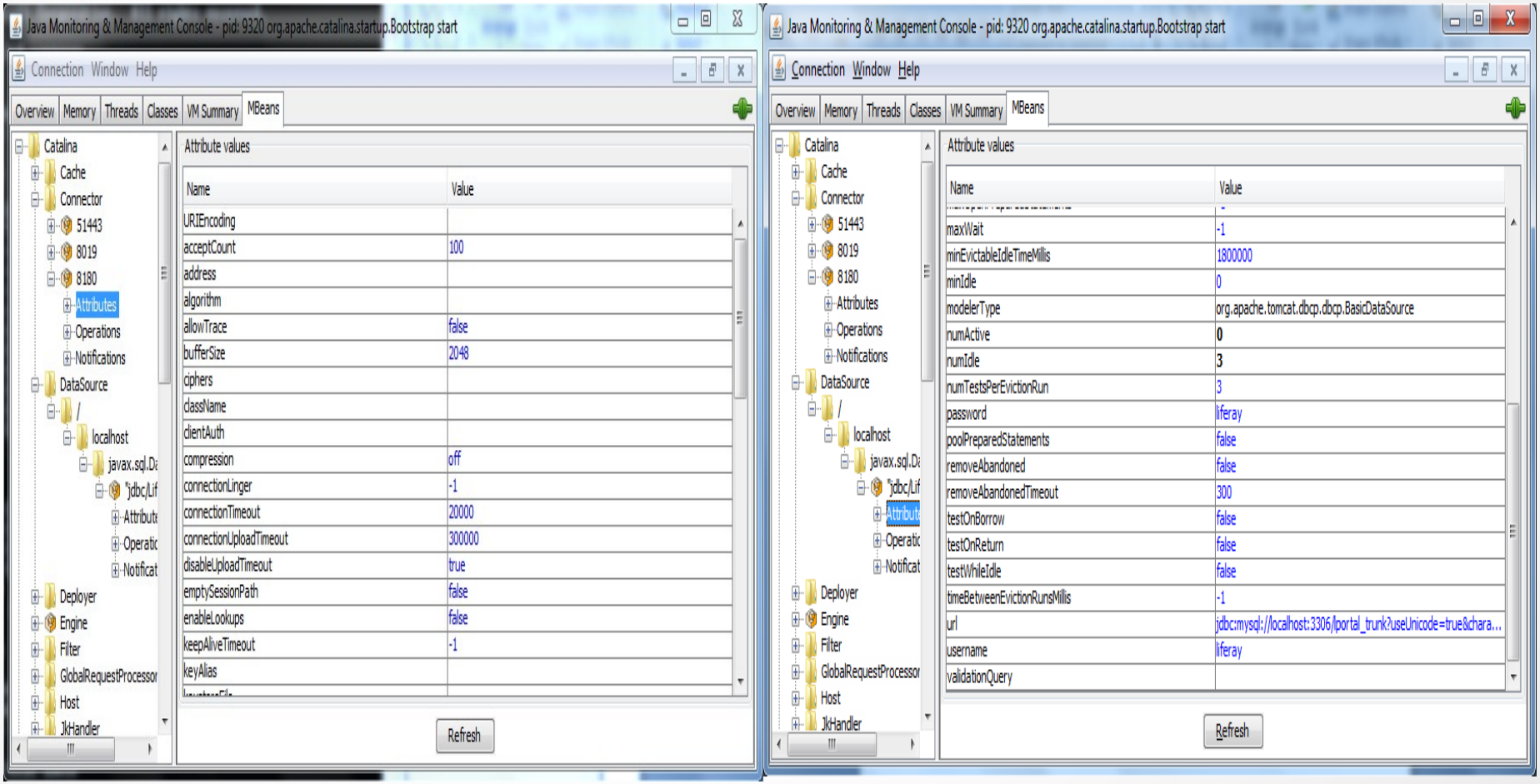
Monitor application server threads

- Do not rely upon “auto-sizing”, could lead to “auto-thrashing”
- Fast transactions imply 50-75 threads.
- No more than 200-300 threads

Monitor JDBC connections

- Initially size for 20 connections.
- Adjust according to monitored usage

Peek-a-boo - JConsole



Database Read-write Split

Benefits

- Optimize databases separately for reading and writing.
- Scale databases separately

Implementation

- Deploy two data sources, one read and one write
- Add META-INF/dynamic-data-source-spring.xml to list of spring configurations
- Enable replication between DB servers

Don't Neglect the Database

Database - MySQL

Buffer sizing to match size and load

- Key buffer
- Sort buffer
- Read buffer

Caches

- Query caches
- Thread caches

Database - Oracle

Oracle RAC and Oracle Name Service

Oracle Statistics Pack

Oracle buffer sizes (transaction and rollback logs, etc)

Database Sharding

Benefits

- Split data along logical divisions
- Common technique used by SaaS providers (e.g. Google Apps, Salesforce, Facebook, etc.)
- Liferay shards along portal instances

Implementation

- Configure an appropriate shard selector in portal.properties
- Configure list of shard data sources
- Add META-INF/shard-data-source-spring.xml to list of spring configurations

Database Sharding

Benefits

- Split data along logical divisions
- Common technique used by SaaS providers (e.g. Google Apps, Salesforce, Facebook, etc.)
- Liferay shards along portal instances

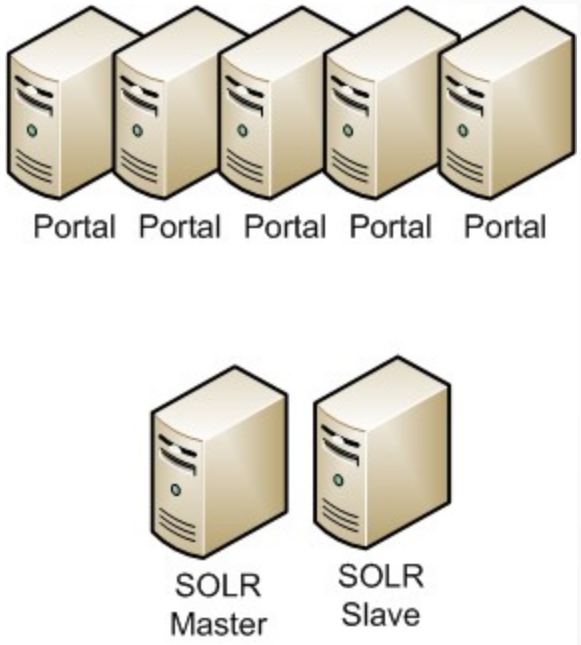
Implementation

- Configure an appropriate shard selector in portal.properties
- Configure list of shard data sources
- Add META-INF/shard-data-source-spring.xml to list of spring configurations

Enterprise Services

- Performance of your enterprise services matter!

- SOLR replaces Liferay's embedded Lucene
  - Enables scaling of search separately from portal
  - Built-in index replication scheme



Portal Tuning

Portal Tuning

- Servlet Filters
- Performance Properties
- Caching

Filters are friends?

- Servlet filter decorates each HTTP request
- Liferay ships with ~30 servlet filters

- Deactivate the filters you do not need:

  AuditFilter, MonitoringFilter, CASFilter, NTLMFilter, NTLMPostFilter, OpenSSOFilter, SharepointFilter

- Deactivate in portal.properties
  - LR 6.0 EE and LR 6.1 introduces improved servlet filter configurations

Filters be gone!

Deactivate in portal-ext.properties:

com.liferay.portal.servlet.filters.sso.ntlm.NtlmFilter=false

com.liferay.portal.servlet.filters.sso.ntlm.NtlmPostFilter=false

com.liferay.portal.servlet.filters.sso.opensso.OpenSSOFilter=false

- Pre-LR 6.1, comment out in web.xml

Performance Properties

- Default configuration in portal.properties is set to optimize performance

- portal-developer.properties makes life easier on developers
  - *theme.css.fast.load=false*
  - *theme.images.fast.load=false*
  - *javascript.fast.load=false*
  - *combo.check.timestamp=true*

- Search portal.properties on “performance” for other tips

Developer Tricks

- Performance configuration can make troubleshooting a production instance more difficult

- Manually add request parameters as a temporary workaround
  - *css_fast_load=0*
  - *images_fast_load=0*
  - *js_fast_load=0*
  - *strip=0*

- Firebug

Caching Overview

Improves application scalability

- Reduce database utilization and latency
- Reduce overhead due to object-relational impedance
- Reduce expensive object creation and excessive garbage collection

Facilitates horizontal vs vertical scaling

- Sun E15K > $1MM per server
- 2 Dual CPU, Quad Core < $10K per server

Caching in Liferay

L1 – “chip level cache”

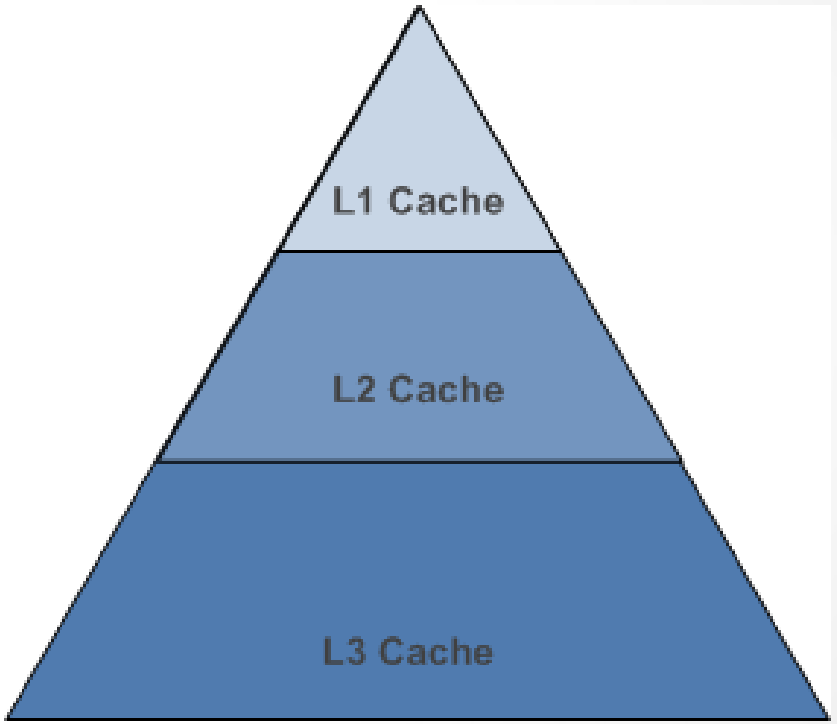
Request scoped cache

L2 - “system memory”

Constrained by heap size

L3 – “swap space”

Equivalent to virtual memory swap space



Liferay L1 Caching

- Improve concurrency by caching to executing thread.
- Prevent repeated calls to remote caches.
- Reduce object cloning within L2 caches
  - Ehcache clones a cached object before providing to cache clients.
- Able to accept a short lived “dirtiness” of data.
  - Permission cache
  - Service value cache

Your own L1 Caching

- All ServiceBuilder generated services can automatically leverage Liferay's L1 cache
- Automatic clearing of all Liferay L1 caches
- Cache via AOP using ThreadLocalCachable method annotation:

```
@ThreadLocalCachable
    public Group getGroup(long groupId)
    throws PortalException, SystemException {

    return groupPersistence.findByPrimaryKey(groupId);

    }
```

L2 and L3 with Ehcache

Advantages

- Cache expiration algorithms
  - LRU (least recently used)
  - Timeout
- Cache coherence resolved via replication algorithms
  - Asynchronous vs. Synchronous Replication
  - Key vs. full object replication
- Can be paired with disk overflow/swapping for larger caches
- Easily add BigMemory to configuration (LR 6.1 EE)

Disadvantages

- Cache size dictated by JVM heap capacity
- Each JVM maintains a copy of the cached data.
- Difficult to control cache size (out of memory error)
- Requires careful tuning of cache element count
- Increased file IO due to swapping.
- Potential degradation with growth of swap file sizes.

Your own L2/L3 cache

- Add your own L2/L3 cache elements using Liferay utilities
  - Determine a unique cacheName
  - Generate a unique objectKey
  - Update the cache elements within your LocalServiceImpl

```
Object value = SingleVMPoolUtil.get(cacheName, objectKey);
Object value = MultiVMPoolUtil.get(cacheName, objectKey);
```

- Best implemented after profiling indicates bottlenecks

Caching Configuration

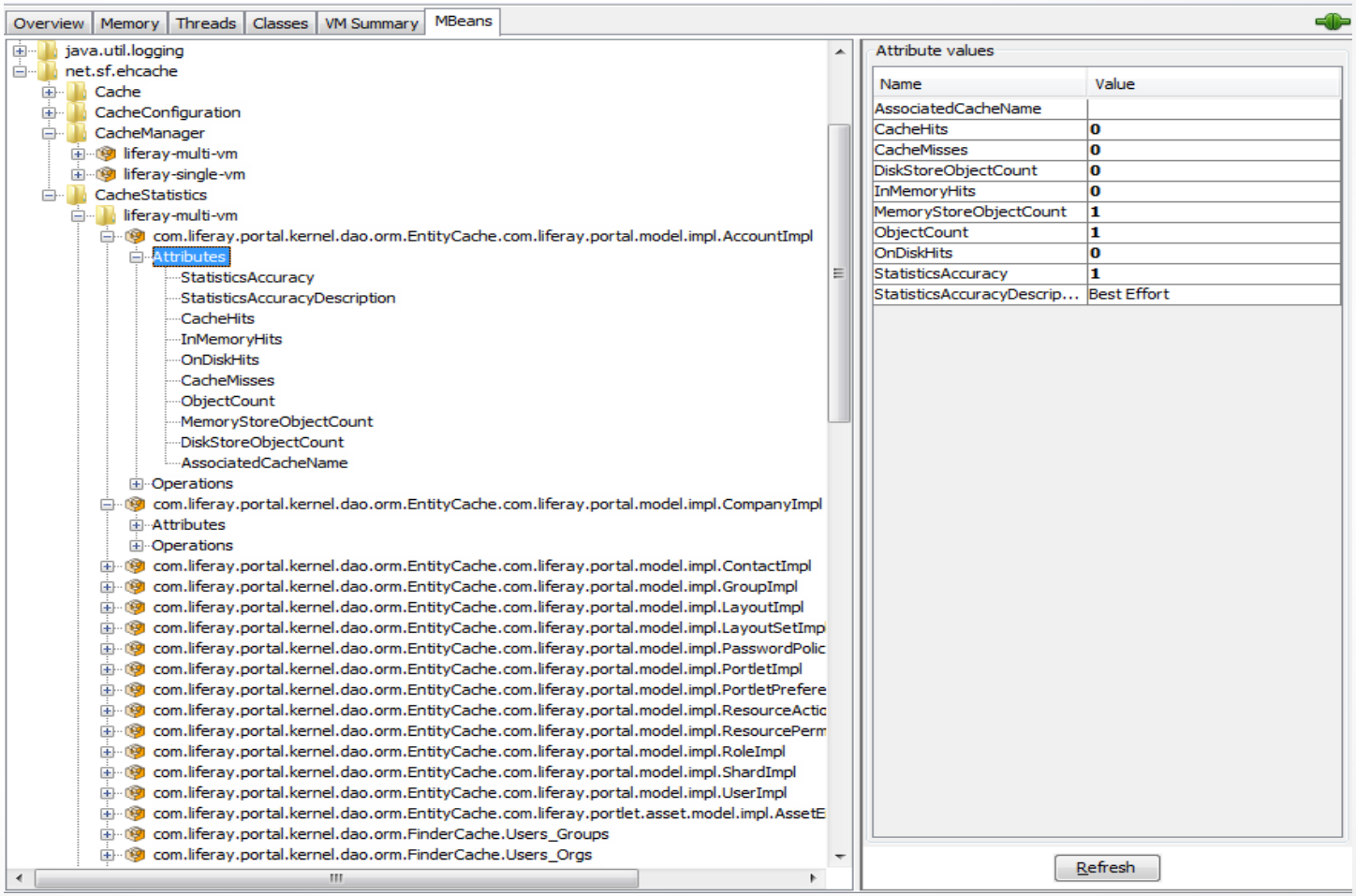
Configure cache sizes and time to live

```
<cache
    name="com.liferay.portal.model.impl.UserImpl"
    maxElementsInMemory="10000"
    eternal="false"
    timeToIdleSeconds="600"
    overflowToDisk="false"
    />
```

Configure disk overflows

```
<cache
    name="com.liferay.portal.model.impl.UserImpl"
    maxElementsInMemory="10000"
    eternal="false"
    timeToIdleSeconds="600"
    overflowToDisk="true"
    maxElementsOnDisk="10000000"
    diskPersistent="false"
    diskExpirationThreadIntervalSeconds="120"
    />
```

Monitoring Cache Statistics



Configurable Replication Techniques

Default Ehcache Replication

- Easy to configure
- Multicast discovery with RMI Replication
- Difficulty in scaling beyond 8 cluster members.
- 1 replication thread per cached object
  200+ cached entities = 200+ replication threads

Liferay Portal EE Replication

- Replication requests assigned to queues based on priority.
- Thread pools perform replication.
- ClusterLink for event replication

L2 and L3 via Data Grids

Advantages

Each partition contains unique cached elements

- Coherence no longer a concern

Unlimited cache sizes

- Expand total cache by adding another shard

Cache fault tolerance

Disadvantages

Generating collision-safe keys consumes CPU

- MD5 hash key

Slower than in-memory caches

Cache performance impacted by network performance.

Available Implementations

Terracotta

- Highly scalable, commercial open source solution.
- Supports both partitioned and replicated modes
- Rich set of monitoring tools to manage cache performance.
- Partitioned cache: 1 cache per entity

Memcached

- Popular open source solution used by Facebook, Google, etc.
- Max 2MB cached object size
- Use multiple languages to access cache
- “Roll your own” tools and strategies
- Cache is 1 large cache, no segments per object

Runtime Tuning

- JVM Parameters
- Monitoring
- Assessment

Not Just Max and Min

Java VM – beyond -Xms and -Xmx

Do not rely upon “automatic GC tuning.”

- Carefully tune your young and old generation

Garbage collector algorithm choice critical

- Generational vs parallel vs CMS vs G1

Perform detailed heap tuning: young generation, survivor spaces, etc

Number of threads/CPUs dedicated to GC execution

JVM vendor does matter!

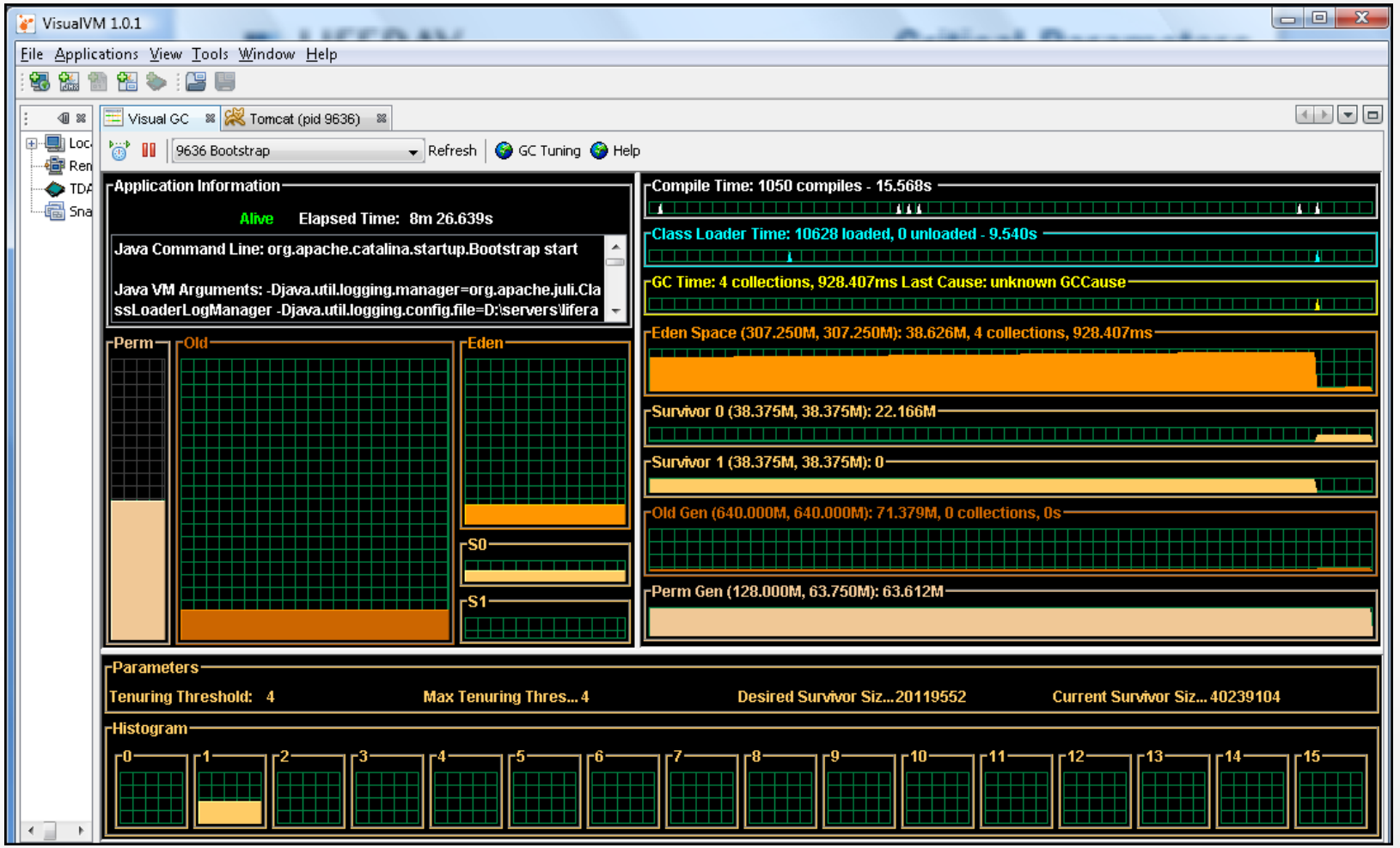
- IBM vs JRockit vs Sun

-server -XX:NewSize=700m -XX:MaxNewSize=700m -Xms2048m -Xmx2048m -XX:MaxPermSize=128m -XX:SurvivorRatio=20 –XX:TargetSurvivorRatio=90 –XX:MaxTenuringThreshold=15 –XX:ParallelGCThread=8

Common JVM Parameters

- **NewSize, MaxNewSize:** The initial size and the maximum size of the New or Young Generation.
- **+UseParNewGC:** Causes garbage collection to happen in parallel, using multiple CPUs. This decreases garbage collection overhead and increases application throughput.
- **+UseConcMarkSweepGC:** Use the Concurrent Mark-Sweep Garbage Collector. This uses shorter garbage collection pauses, and is good for applications that have a relatively large set of long-lived data, and that run on machines with two or more processors, such as web servers.
- **+CMSParallelRemarkEnabled:** For the CMS GC, enables the garbage collector to use multiple threads during the CMS remark phase. This decreases the pauses during this phase.
- **ServivorRatio:** Controls the size of the two survivor spaces. It's a ratio between the survivor space size and Eden. The default is 25. There's not much bang for the buck here, but it may need to be adjusted.
- **ParallelGCThreads:** The number of threads to use for parallel garbage collection. Should be equal to the number of CPU cores in your server.

Monitoring - VisualVM



Monitoring – Lambda Probe

λproβe Version 1.7b running on **node10**, UP for 8 days 23 hours 44 minutes

JVM memory usage

Applications | Data Sources | Deployment | Logs | Threads | Cluster | **System Information** | Status | Connector stats | Quick check

Advise Finalization Advise GC

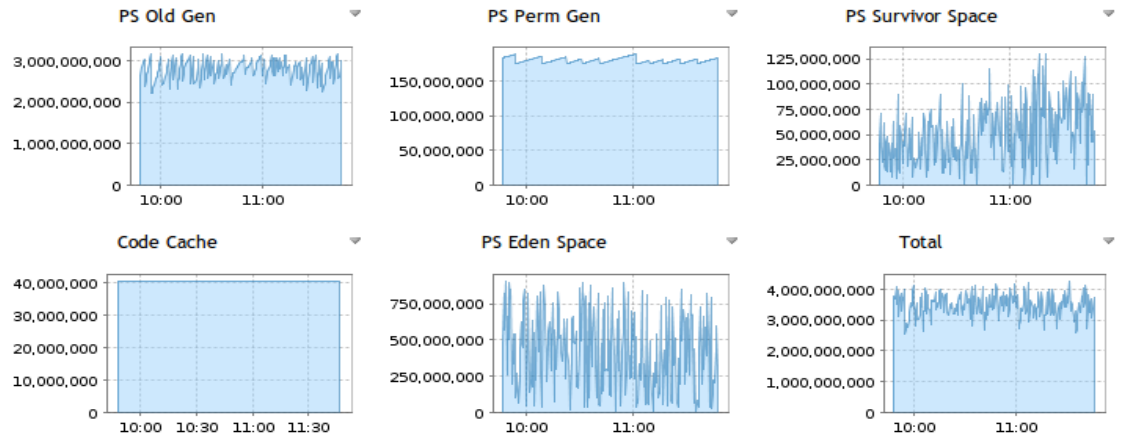
CURRENT MEMORY USAGE

NAME	USAGE SCORE	PLOT	USED	COMMITTED	MAXIMUM	INITIAL	GROUP
PS Old Gen			2.86Gb	3.00Gb	3.00Gb	3.00Gb	HEAP
PS Perm Gen			175.14Mb	178.00Mb	256.00Mb	22.00Mb	NON_HEAP
PS Survivor Space			72.67Mb	128.00Mb	128.00Mb	128.00Mb	HEAP
Code Cache			38.70Mb	39.00Mb	48.00Mb	2.44Mb	NON_HEAP
PS Eden Space			271.25Mb	768.00Mb	768.00Mb	768.00Mb	HEAP
Total			3.41Gb	4.09Gb	4.17Gb	3.90Gb	TOTAL

SYSTEM INFORMATION

- Overview
- Memory utilization
- System properties
- OS information
- Wrapper control

MEMORY USAGE HISTORY



| Applications | Data Sources | Deployment | Logs | Threads | Cluster | System Information | Status | Connector stats | Quick check |

JVM and the OS

Monitor CPU and virtual memory utilization

- vmstat shows CPU and memory utilization
- mpstat shows the performance of each core/thread of a CPU

Monitor network and disk IO

- iostat shows disk utilization and IO performance
- ifstat shows network interface performance

CPU Monitors

mpstat

```
Average:      CPU    %user   %nice    %sys %iowait    %irq   %soft  %steal   %idle    intr/s
Average:      all    73.92    0.00    1.62    0.00    0.09    1.09    0.00   23.27  15775.70
Average:        0    72.75    0.00    1.23    0.00    0.00    0.29    0.00   25.73   1000.15
Average:        1    71.01    0.00    1.69    0.01    0.10    1.38    0.00   25.80   1043.51
Average:        2    77.98    0.00    1.52    0.00    0.11    1.39    0.00   18.99    984.69
Average:        3    70.38    0.00    1.75    0.00    0.08    1.02    0.00   26.78    745.29
Average:        4    78.69    0.00    1.52    0.00    0.13    1.47    0.00   18.19   1046.35
Average:        5    70.59    0.00    1.73    0.02    0.10    1.34    0.00   26.21   1013.86
Average:        6    78.63    0.00    1.55    0.00    0.11    1.42    0.00   18.29    997.11
Average:        7    66.28    0.00    1.92    0.00    0.00    0.24    0.00   31.56      0.00
Average:        8    73.14    0.00    1.57    0.00    0.00    0.37    0.00   24.92      0.00
Average:        9    74.30    0.00    1.67    0.01    0.09    1.25    0.00   22.68    923.20
Average:       10    79.58    0.00    1.48    0.00    0.10    1.29    0.00   17.54    883.71
Average:       11    69.64    0.00    1.71    0.00    0.00    1.19    0.00   27.46   1455.46
Average:       12    79.14    0.00    1.50    0.00    0.09    1.22    0.00   18.06    818.89
Average:       13    73.74    0.00    1.71    0.01    0.09    1.18    0.00   23.27    861.36
Average:       14    78.49    0.00    1.52    0.00    0.08    1.19    0.00   18.72    784.57
Average:       15    68.33    0.00    1.85    0.00    0.42    1.25    0.00   28.14   3217.55
```

vmstat

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free    buff  cache   si   so    bi    bo   in   cs us sy id wa st
10  0 517784 140276 445580 6618964    0    0     0  1016 16299 18695 77  3 21  0  0
16  0 517784 128996 445916 6624724    0    0     0  1122 16401 18877 77  3 20  0  0
17  0 517784 128256 446280 6619660    0    0     0  1176 16429 18646 77  3 20  0  0
21  0 517784 121668 446608 6619732    0    0     0  1087 16349 18391 76  3 21  0  0
11  0 517784 116576 446928 6620072    0    0     0  1140 16489 18399 78  3 19  0  0
29  0 517784 110564 447360 6620596    0    0     0  1213 16466 18471 77  3 20  0  0
27  0 517784 104024 447676 6620704    0    0     0  1212 16467 18097 78  3 19  0  0
12  0 517784  98304 448096 6620884    0    0     0  1255 16489 18118 78  3 19  0  0
33  0 517784  92360 448476 6621268    0    0     0  1168 16325 18348 77  3 20  0  0
```

IO Monitors

ifstat

```
fstat 120 30
       eth0                eth2
 KB/s in  KB/s out    KB/s in  KB/s out
1031.08    905.03     268.41    6784.79
1631.43   1452.57     427.60   10720.72
2182.75   1979.44     580.26   14590.15
2553.86   2341.64     689.64   17621.64
2557.83   2364.12     709.60   17979.73
2497.14   2322.48     706.79   17912.81
2407.74   2262.54     703.76   17843.63
2418.52   2284.49     716.69   18130.09
2382.43   2255.08     710.48   17999.31
2354.54   2240.10     714.85   18052.40
2360.24   2240.56     712.49   18038.24
2360.93   2243.86     711.22   17983.79
2352.89   2238.60     713.41   18120.09
2336.80   2228.08     714.30   18088.35
2343.26   2228.38     713.19   18060.29
2346.96   2234.16     714.80   18141.19
2363.37   2247.24     715.54   18169.56
2379.06   2253.64     713.11   18162.15
2342.97   2229.20     710.60   18032.76
2392.82   2268.89     716.72   18241.21
```

iostat

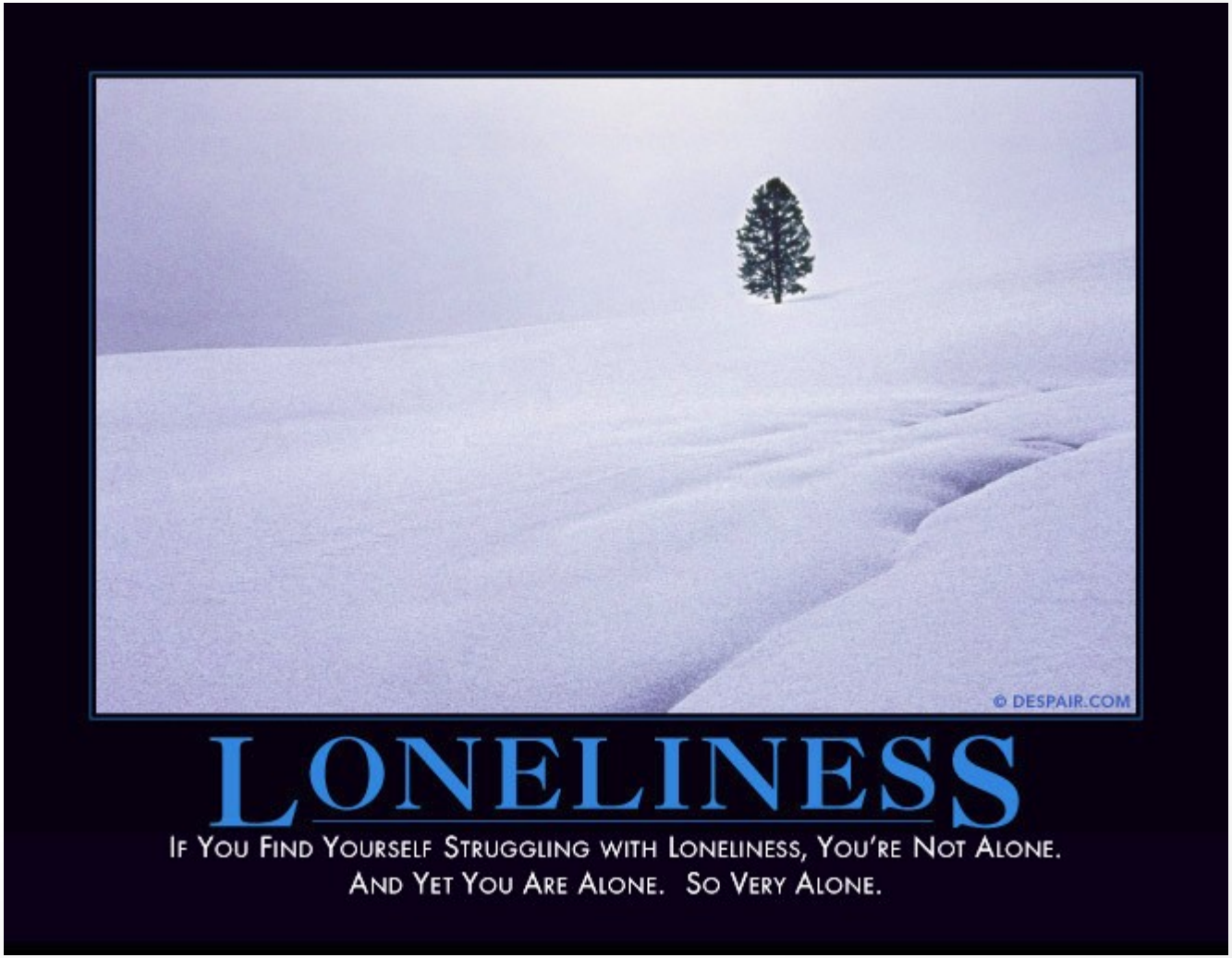
```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          64.57    0.00    2.55    0.00    0.00   32.87

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda              43.25         0.00      1562.13          0     187456
sda1              0.00         0.00         0.00          0          0
sda2             43.25         0.00      1562.13          0     187456
dm-0            195.27         0.00      1562.13          0     187456
dm-1              0.00         0.00         0.00          0          0

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          76.53    0.00    2.86    0.00    0.00   20.61

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda              50.33         0.00      2048.00          0     245760
sda1              0.00         0.00         0.00          0          0
sda2             50.33         0.00      2048.00          0     245760
dm-0            256.00         0.00      2048.00          0     245760
dm-1              0.00         0.00         0.00          0          0
```

despair.com


© DESPAIR.COM
LONELINESS
IF YOU FIND YOURSELF STRUGGLING WITH LONELINESS, YOU'RE NOT ALONE.
AND YET YOU ARE ALONE. SO VERY ALONE.

Assessing performance issues

1. When/where?
   - Which pages or portlets?
   - Under what conditions?
   - What's new in the system?
2. What's happening in the environment?
   - Memory?
   - Network?
   - Database?
3. Where to turn?
   1. Another pair of eyes - team lead, system admin, DBA.
   2. liferay.com – whitepapers, forums, wiki
   3. Liferay Service Partners and Liferay Global Services

Questions?