Pivotal

GemFire Jump Start

General Comments

These recommendations provide a good starting point but there is no substitute for testing so ***please*** *…*

- ✓ Performance test early
- ✓ Expect several performance tuning cycles
- ✓ Try to test at a load that is above your desired maximum
- ✓ Always do a long running test (48 hours) to expose “slow motion” gc problems like heap fragmentation
- ✓ Put monitoring and capacity planning in place , provision additional hardware to stay ahead of the load
- ✓ There is relatively long learning curve for operations, especially if persistence is in use bring them into the process early

Hardware Selection / Virtualization

- 1 core per 8-16G of memory (more toward 8 for compute intensive loads)
- Intermittent latency on the intra-cluster network can cause instability.
- Separate intra-cluster traffic from client-server traffic. The ideal is for there to be separate network paths (NICS, switches). Intra-cluster traffic should not wait behind client-server traffic.
- In virtualized environments:
  - virtualization is OK but do not configure and manage it like a consolidation workload
  - do not overcommit
  - do not allow automatic vMotion
  - much more detail here: http://gemfire.docs.pivotal.io/docs-gemfire/latest/managing/monitor_tune/gemfire_performance_on_vsphere.html

Pivotal

OS Settings

- File Descriptor Limits

  GemFire can use hundreds or thousands of sockets. Each one is a file descriptor. Be sure the file descriptor quota is unlimited or very high. See http://gemfire.docs.pivotal.io/docs-gemfire/latest/managing/monitor_tune/socket_communication_have_enough_sockets.html
- Socket Buffer Size Limits

  It's often advantageous to increase the size of the socket buffers GemFire uses. The OS imposes limits which you may need to increase (e.g. rmem_max, wmem_max)

JVM Settings

- Maximum and minimum memory should be the same, use parallel new gc collector and cms old collector, set the young gen to roughly ⅛ of total heap but not less than 1G and not more than 8G
- Set the CMSInitiatingOccupancyFraction to slightly less than the eviction threshold.
- Example of a 32G JVM with a 80% eviction threshold

```
-Xmx32g -Xms32g -Xmn4g -XX:+UseConcMarkSweepGC -XX:+UseParNewGC
-XX:CMSInitiatingOccupancyFraction=75 -XX:+UnlockDiagnosticVMOptions
-XX:ParGCCardsPerStrideChunk=4096
```

- Consider using HugePages, especially if GemFire is the only thing on the box.

```
-XX:+UseLargePages (also need to reserve them at the OS level)
```

GemFire Settings

Set In "gemfire.properties" or pass `--J=-Dgemfire.someprop=someval` to gfhs start

- Change default socket utilization model

  ```
  conserve-sockets=false
  ```

- Set the log level to config and limit the space they will consume

  ```
  log-level=config
  log-file-size-limit=100 (the size of one file, value in MB)
  log-disk-space-limit=1000 (total size of all logs, value in MB)
  ```

- Turn on statistics and limit the space they will consume

  ```
  statistic-sampling-enabled=true
  statistic-archive-file=datanode.gfs (the .gfs suffix is important!)
  archive-file-size-limit=10 (the size of one file, value in MB)
  archive-disk-space-limit=100 (total size of all stats files, value in MB)
  ```

GemFire Settings (cont.)

Set In “gemfire.properties” or pass `--J=-Dgemfire.someprop=someval` to gfhs start

- Set split brain detection based on your desired CAP theorem behavior

```
enable-network-partition-detection=true
```

GemFire Settings - Size Socket Buffers

1. Set the peer to peer socket buffer size in gemfire.properties

```
socket-buffer-size=1048576
```

2. On the “pool” element, set the client socket buffer size. Example using spring-data-gemfire:

```
<gfe:pool id="default-pool" subscription-enabled="false" socket-buffer-size="1048576">
      <gfe:locator host="localhost" port="10000"/>
</gfe:pool>
```

3. Set the server socket buffer to the same value using in cache.xml or using a gfsh argument.

   Example using cache.xml

```
<cache-server port="40404" socket-buffer-size=”1048576”/>
```

- see:
  http://gemfire.docs.pivotal.io/docs-gemfire/latest/managing/monitor_tune/socket_communication_setting_socket_buffer_sizes.html

GemFire Settings - Use PDX Serialization

- It is faster and more compact than java.io Serialization
- It produces much less garbage during queries and can be a big win
- Configured in cache.xml on both client and server.

```
<pdx persistent="true" disk-store-name="default-disk-store" read-serialized="true">
     <pdx-serializer>
          <class-name>com.gemstone.gemfire.pdx.ReflectionBasedAutoSerializer</class-name>
          <parameter name="classes">
               <string>io.pivotal.pde.sample.*</string>
          </parameter>
     </pdx-serializer>
</pdx>
```

- see:

  http://gemfire.docs.pivotal.io/docs-gemfire/latest/developing/data_serialization/auto_serialization.html

GemFire Settings - Redundancy Recovery

```
<region name=sample refid=”PARTITION_REDUNDANT”>
      <region-attributes>
        <partition-attributes recovery-delay=”600000” redundant-copies="1" />
      </region-attributes>
</region>
```

- Understand the recovery delay setting and choose a value accordingly.
- By default, redundancy is not automatically recovered!

Performance Testing and Tuning Guidelines

- Test with a similar read/write mix
- Test at a fixed throughput. As fast as possible tests don't tell you anything.
- Ideal: test at a fixed load slightly higher than peak and with a similar number of concurrent connections
- Primary "knobs" are:
  - GC Behavior
  - Socket Buffer Sizes
- Once the system is stable at the target throughput, run it at that throughput for 48 hours with gc logging turned on.
  - ensure there are no long pauses
  - ensure that heap fragmentation is not increasing

Pivotal

Operations Best Practices

- Set up monitoring - there is usually smoke before there is fire.
  - Monitor all resources: CPU, memory, disk, file descriptors
  - See additional document for JMX monitoring recommendations
- Set up a capacity planning process. Keep it in the envelope!
- Automate. Many outages have been caused by avoidable human error!
- When performing a rolling bounce, always wait for redundancy to be established before stopping a server.
- If disk stores are present, much more care is needed:
  - always use gfsh shutdown to stop the cluster, don’t stop the members one at a time
  - always start servers in parallel

Pivotal®

Transforming How The World Builds Software