Tuning guide for MongoDB on IBM Power Systems

Beth Hoffman
Lilian Romero
Calvin Sze

June 24, 2016

This tuning guide will help you achieve optimal performance when running MongoDB on IBM Power Systems by providing guidelines for system-level, virtualization-level and application-level tuning.

MongoDB, which is a document-oriented NoSQL database, runs well on IBM® Power Systems™ using a default environment. However, you can achieve optimal performance on Power Systems, by tuning several environment settings, which we'll describe in this article.

System-level tuning

- Turbo frequency, disk tuning, CPU binding, memory affinity

Virtualization-level tuning with PowerKVM

- Host tuning: Turbo frequency, disk tuning
- Guest tuning: Memory affinity, data disk pass through, network Peripheral Component Interconnect (PCI) pass through, CPU unpin

Application-level tuning

- Java tuning: Huge pages, heap size, garbage collection threads, compressed reference
- Shard/Cluster tuning

The guidelines in this article were derived from testing the Yahoo! Cloud Serving Benchmark (YCSB) workload and other customer proof of concept (POC) workloads. Every client workload will drive different requirements. Use these recommendations as a guide only; you'll need to adjust your specific tuning requirements based on the actual workload you plan to deploy.

System-level tuning

The following guidelines are for all MongoDB workloads running on Linux on Power except where noted.

Frequency

This section describes how to set frequency to turbo.
In a bare metal environment, the CPU energy governor is controlled by the operating system (OS).

- Use the `cpupower` command to verify and set the frequency as follows:
  
  ```
  cpupower -c 0-159 frequency-info
  cpupower -c 0-159 frequency-set -g performance
  ```

- In an IBM PowerVM™ environment, use the Hardware Management Console (HMC) to set favor performance:
  
  1. Log into ASM (Advanced System Management).
  2. Expand **System Configuration** -> **Power Management**.
  3. Click **Power Mode Setup** -> **Enable Dynamic Power Save (favor performance) mode**.
  4. Click **Continue**.
  5. Click **Save Settings**.

**Simultaneous multithreading (SMT)**

Using SMT you can concurrently run instruction streams of multiple threads on the same core. On IBM POWER8® processor-based servers, SMT8 is the default setting and most workloads will run well with this default. For databases, setting SMT to the SMT8 mode is considered a best practice.

Use the `ppc64_cpu` command to set the SMT mode to 8, 4, or 2 as follows:

- SMT8: `ppc64_cpu --smt=on`
- SMT4: `ppc64_cpu --smt=4`
- SMT2: `ppc64_cpu --smt=2`

**ulimit**

For ulimit settings, use the recommendations in the *MongoDB Administration Guide*.

**Disk tuning**

For internal storage:

- Use just a bunch of disks (JBOD) to format the drives using the `iprconfig` command. Find more details about the iprconfig utility at: https://www.ibm.com/support/knowledgecenter/HW4L4/p8ebk/starting_iprconfig.htm
- Turn on write disk cache using the following setting, where `x` is the disk number:
  
  ```
  sdparm--set=WCE /dev/sdx
  ```
- For production workloads, you may need solid-state drives (SSDs) to meet the input/output operations per second (IOPS) requirement of the workload.

For external storage:

For customer workloads, use Redundant Array of Independent Disks (RAID) protection. Use RAID0 for benchmarks. You should determine the RAID level that best suits your particular environment.

**Memory affinity and CPU binding**

- Memory affinity is very important when using PowerVM. For more information about memory affinity, review "Chapter 3. The IBM POWER Hypervisor" in IBM Redbooks®: *Performance*
Optimization and Tuning Techniques for IBM Power Systems Processors Including IBM POWER8

• Pin running benchmark processes to specific cores.
• To avoid non-uniform memory access (NUMA) affects when creating virtual machines (VMs), confine memory to a single chip or node.
• CPU binding is not recommended in a production environment, however, for benchmark POCs, use the taskset command to do the CPU binding. For example, to run MongoDB on the first socket of a CPU that has 10 cores (80 logical CPUs) use the following command:

```
taskset –c 0-79 numactl –localalloc mongod --config /home/mongoDB/test/mongodb.conf
```

Virtualization-level tuning
When running MongoDB in a virtualized environment, the following guidelines describe how to configure the actual server (host) and the virtual machines (guests).

Host tuning
- Turbo frequency: follow the recommendations described in the system-level tuning section.
- Disk tuning: follow the recommendations described in the system-level tuning section.

Guest tuning
The following guest tuning guidelines describe how each virtual machine should be configured.

Memory affinity
The following recommendations are for IBM PowerKVM® environments prior to PowerKVM 3.1 unless otherwise noted.

• When using PowerVM, use the dynamic platform optimizer (DPO) tool. For more information, review the IBM Redbooks, Performance Optimization and Tuning Techniques for IBM Power Systems Processors Including IBM POWER8

• POWER8 processor-based servers come with several chips or sockets. Depending on the CPU and memory requirements for a workload, it is a best practice to confine a guest within chip or socket boundaries. The following example shows how to confine a guest on a socket on an IBM Power® System S822L host which has four nodes:

```
numactl -H
available: 4 nodes (0-1,16-17)
node 0 cpus: 0 8 16 24 32
node 0 size: 65536 MB
node 0 free: 63286 MB
node 1 cpus: 40 48 56 64 72
node 1 size: 131072 MB
node 1 free: 131000 MB
node 16 cpus: 80 88 96 104 112
node 16 size: 65536 MB
node 16 free: 64286MB
node 17 cpus: 120 128 136 144 152
node 17 size: 65536 MB
node 17 free: 64286 MB
```

• If a guest requires more than 10 cores and around 100GB of memory, then the guest should fit into two nodes. In the following example libvirt XML definition, nodes 16 and 17 are used to confine the guest.

```
...
<memory unit='KiB'>125566976</memory>
<currentMemory unit='KiB'>125566976</currentMemory>

<vcpu placement='static'>80</vcpu>
<cputune>
    <vcpupin vcpu='0' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='1' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='2' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='3' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='4' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='5' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='6' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='7' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='8' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='9' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='10' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='11' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='12' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='13' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='14' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='15' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='16' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='17' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='18' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='19' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='20' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='21' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='22' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='23' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='24' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='25' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='26' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='27' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='28' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='29' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='30' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='31' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='32' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='33' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='34' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='35' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='36' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='37' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='38' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='39' cpuset='80,88,96,104,112'/>
    <vcpupin vcpu='40' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='41' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='42' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='43' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='44' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='45' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='46' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='47' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='48' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='49' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='50' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='51' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='52' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='53' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='54' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='55' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='56' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='57' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='58' cpuset='120,128,136,144,152'/>
    <vcpupin vcpu='59' cpuset='120,128,136,144,152'/>
• Assign enough huge pages to nodes 16 and 17 on the host when the guest is using huge pages:
  
  ```bash
  echo 3840 >/sys/devices/system/node/node16/hugepages/hugepages-16384kB/nr_hugepages
  echo 3840 >/sys/devices/system/node/node17/hugepages/hugepages-16384kB/nr_hugepages
  ```

• Ensure that the guest is confined in nodes 16 and 17 by using the `pagemapscan-numa` utility as follows:
  
  ```bash
  ./pagemapscan-numa 85538 2
  ```

<table>
<thead>
<tr>
<th>host-node00</th>
<th>host-node01</th>
<th>host-node16</th>
<th>host-node17</th>
<th>not-present</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM-node00</td>
<td>0(000%)</td>
<td>0(000%)</td>
<td>978176(100%)</td>
<td>0(000%)</td>
</tr>
<tr>
<td>VM-node01</td>
<td>0(000%)</td>
<td>0(000%)</td>
<td>978176(100%)</td>
<td>0(000%)</td>
</tr>
</tbody>
</table>

• In PowerKVM 3.1, the libvirt XML definition has a new memory pinning option that does not require the use of huge pages to set up memory affinity on the guest.

• The new `numatune memnode` feature can map the guest node, `cellid`, which matches the cell creation in the section above it to the corresponding host node, `nodeset`. The following code sample shows the first cell creation section being defined:

```xml
<cpu>
  <topology sockets='2' cores='10' threads='8'/>
  <numa>
    <cell id='0' cpus='0-79' memory='5242880' unit='KiB'/>
    <cell id='1' cpus='80-159' memory='5242880' unit='KiB'/>
  </numa>
</cpu>
```

```xml
<numatune>
  <memnode cellid='0' mode='strict' nodeset='1'/>
  <memnode cellid='1' mode='strict' nodeset='16'/>
</numatune>
```
**Huge page backing**

You'll only need to configure huge page backing on the guest when using Java™-based MongoDB applications.

- The guest definition in the libvirt XML definition file must have `memoryBacking` specified as shown below:

```xml
<domain type='kvm'>
  <name>myguest</name>
  <uuid>...</uuid>
  <memory>2048</memory>
  <currentMemory>2048</currentMemory>
  <memoryBacking>
    <hugepages/>
  </memoryBacking>
  ...
</domain>
```

- Run the following commands to enable huge pages in the guest:
  ```
  echo 2000 > /proc/sys/vm/nr_hugepages
  hugeadm --create-mounts
  ```

- You also need to allocate huge memory on the host as described later in the Application-level -> Java Tuning section.

**Data disk pass through**

Because virtual SCSI (VSCSI) emulation disks on a guest partition are typically slow, you should set device to `lun` in the libvirt XML definition file as shown below:

```xml
<disk type="block" device="lun">
  <driver name="qemu" type="raw" cache="directsync"/>
  <source dev="/dev/sdb"/>
  <target dev="sdb" bus="scsi"/>
  <address type="drive" controller="0" bus="0" target="3" unit="0"/>
</disk>
```

For more information, see page 98 of the IBM Redbooks, *IBM PowerKVM: Configuration and Use*.

**Network PCI pass through**

Set `type` attribute to `pci` in the libvirt xml definition as in the example below:

```xml
<devices>
  ...
  <hostdev mode='subsystem' type='pci' managed='yes'>
    <source>
      <address domain='0x000' bus='0x15' slot='0x00' function='0x0'/>
    </source>
  </hostdev>
</devices>
```

For more information, see page 23 of the IBM Redbooks, *IBM PowerKVM: Configuration and Use*.

**Application-level tuning**

In addition to system-level and virtualization-level settings, additional tuning is often needed depending on the application you're running. This section describes how to configure some of the more common application-level settings.
Java tuning

The recommendations in this section, only apply to Java-based workloads, including the YCSB.

Huge pages

The following settings should be applied to bare metal and PowerKVM host environments.

- The number of huge pages you should use depends on your particular application requirement. Set it to contain the maximum needed, for example, the total Java virtual machine (JVM) heap of all the Java processes.
- The maximum memory you allocate to huge pages should not exceed 90% of the system memory.
- Use a large number. The following example uses 512 GB:
  ```
  echo 549755813888 > /proc/sys/kernel/shmmax
  echo 549755813888 > /proc/sys/kernel/shmall
  ```
- To set this number permanently at the next system reboot, append it to the `/etc/sysctl.conf` file as shown below:
  ```
  # Shared memory – max segment size:
  kernel.shmmax = 549755813888
  # Total amount of shared memory pages
  kernel.shmall = 549755813888
  ```
- Set the number of reserved huge memory pages, which is the number of reserved pages. The following example reserves 60 GB:
  ```
  echo 3750 > /proc/sys/vm/nr_hugepages
  ```
- To set this number permanently at the next system reboot, append it to the `/etc/sysctl.conf` file as shown below:
  ```
  # Enable kernel to reserve 60GB
  vm.nr_hugepages = 3750
  ```

Use IBM JDK version 7, 8 or later

Java options should be set based on your specific Java workload. For newer versions of YCSB, use the Java defaults; For earlier versions, modify the `ycsb` script file and add the following options:

```
"-Xlp", "-Xms4096m", "-Xmx4096m", "-Xmn3400m", "-Xgcthreads8", "-Xcompressedrefs"
```

Huge pages

A lot of Java workloads run better with huge pages. For YCSB, use huge pages (16 MB), -Xlp. See the Huge pages section above for more information.

Heap size

For YCSB, use 4 GB heap per Java instance.

Garbage collection threads

For YCSB, set garbage collection threads to 8.
Compressed reference

Use the compress reference option to make better use of the available space.

For more information, see page 177 of the IBM Redbooks, Performance Optimization and Tuning Techniques for IBM Power Systems Processors Including IBM POWER8

Shard/cluster tuning

For write/update heavy workloads, align to one socket for best affinity (use maximum of 10 cores per shard). For read/query heavy workloads, align to one socket for best affinity (use maximum of 20 cores per shard).

For more information, go to https://docs.mongodb.com/manual/sharding/