Bare Metal Container

National Institute of Advanced Industrial Science and Technology (AIST)

Kuniyasu Suzaki
Contents

• Background of BMC
  – Drawbacks of container, general kernel, and accounting.
• What is BMC?
• Current implementation
• Evaluation
• Extension
  – NVIDIA Docker, Moby, Intel Clear Container, etc.
• Conclusions
Background of BMC 1/3

**Drawback of Container**

- Container technology (Docker) becomes popular.
  - Docker offers an environment to customize an application easily.
  - It looks like to be good for an application, but it is a server centric.
- **It does not allow to change the kernel.**
  - Kernel options passed through /sys are not effective.
- Some applications cannot run on Docker.
  - DPDK on Docker does not work on some machines, because it depends on “igb_uio” and “rte_kni” kernel modules.
    - Some provider offers the kernel which can treat DPDK on Docker, but it is case by case solution. It is not fundamental solution.
Background of BMC 1/3

**Drawback of Container**

• Container technology (Docker) becomes popular.
  – Docker offers an environment to customize an application easily.
  – It looks like to be good for an application, but it is a **server centric**.

• **It does not allow to change the kernel.**
  – Kernel options passed through /sys are not effective.

• Some applications cannot run on Docker.
  – DPDK on Docker does not work on some machines, because it depends on “igb_uio” and “rte_kni” kernel modules.
  – Some provider offers the kernel which can treat DPDK on Docker, but it is case by case solution. It is not fundamental solution.

Container is a jail for a kernel optimizer.
Background of PMC 1/3

HPC users want to optimize the kernel *for their applications*. Kernel is a servant. Container way is not fit for them.

- Docker offers an environment to customize an application easily.
- It looks like to be good for an application, but it is a server centric.
  
  **It does not adapt to the kernel.**
  
  - Kernel options passed through /sys are not effective.

- Some applications cannot run on Docker.
  
  - DPDK on Docker does not work on some machines, because it depends on “igb_uio” and “rte_kni” kernel modules.
    
    - Some provider offers the kernel which can treat DPDK on Docker, but it is case by case solution. It is not a fundamental solution.

Container is a jail for a kernel optimizer.
Background of BMC 2/3

**General kernel leads weak performance**

- Arrakis[OSDI’14] showed that nearly 70% of network latency was spent in the network stack in a Linux kernel.
- Many DB applications (e.g., Oracle, MongoDB) reduce the performance by THP (Transparent Huge Pages) which is enabled on most Linux distributions.

4.11 Disabling Transparent HugePages

Oracle recommends that you disable Transparent HugePages before you start installation.

Transparent Huge Pages (THP) is a Linux memory management system that reduces the overhead of Translation Lookaside Buffer (TLB) lookups on machines with large amounts of memory by using larger memory pages.

However, database workloads often perform poorly with THP, because they tend to have sparse rather than contiguous memory access patterns. You should disable THP on Linux machines to ensure best performance with MongoDB.
Background of BMC 2/3

**General kernel leads weak performance**

- Arrakis[OSDI’14] showed that nearly 70% of network latency was spent in the network stack in a Linux kernel.
- Many DB applications (e.g., Oracle, MongoDB) reduce the performance by THP (Transparent Huge Pages) which is enabled on most Linux distributions.

4.11 Disabling Transparent HugePages

Oracle recommends that you disable Transparent HugePages before you start installation.

It is not fundamental solution. HPC users want to optimize the kernel *for their applications.* Kernel is a servant.

However, database workloads often perform poorly with THP, because they tend to have sparse rather than contiguous memory access patterns. You should disable THP on Linux machines to ensure best performance with MongoDB.
Background of BMC 3/3

**Power consumption for each application**

- Current power measurement is coarse.
  - Power Usage Effectiveness: PUE only shows usage of data-center scale.
  - Current power consumption is theme for vendor and administrators.

- Users have no incentive for low power, even if they make a low power application.
  - Current accounting is based on time consumption.
Background of BMC 3/3

**Power consumption for each application**

- Current power measurement is coarse.
  - Power Usage Effectiveness: PUE only shows usage of data-center scale.
  - Current power consumption is theme for vendor and administrators

- Users have no incentive for low power, even if they make a low power application.
  - Current accounting is based on time consumption.

There is no good method to measure power consumption “for an application”. No accounting which considers power consumption.
What is BMC?

• BMC(Bare-Metal Container) runs a container (Docker) image with a suitable Linux kernel on a remote physical machine.
  – Application on Container can change kernel settings and machine which fit for it. BMC extracts the full performance.
  – On BMC, the power on the machine is almost used for the application.
    • BMC tells the power usage on each machine architecture. Users can know which architecture is good for their application.

BMC offers incentives to customize kernel and select machine architecture
What is BMC?

• BMC (Bare-Metal Container) runs a container (Docker) image with a suitable Linux kernel on a remote physical machine.
  – Application on Container can change kernel settings and machine which fit for it. BMC extracts the full performance.
  – On BMC, the power on the machine is almost used for the application.
    • BMC tells the power usage on each machine architecture. Users can know which architecture is good for their application.

BMC offers incentives to customize kernel and select machine architecture
What is BMC?

• BMC (Bare-Metal Container) runs a container (Docker) image with a suitable Linux kernel on a remote physical machine.
  – Application on Container can change kernel settings and machine which fit for it. BMC extracts the full performance.
  – On BMC, the power on the machine is almost used for the application.
  • BMC tells the power usage on each machine architecture. Users can know which architecture is good for their application.

BMC offers incentives to customize kernel and select machine architecture
What is BMC?

• BMC (Bare-Metal Container) runs a container (Docker) image with a suitable Linux kernel on a remote physical machine.
  – Application on Container can change kernel settings and machine which fit for application to extract the full performance.
  – It means the power on the machine is almost used for an application.
• BMC tells the users which machine architecture is best used for an application. Users can know which machine architecture is good for their application.

BMC offers incentives to customize kernel and select machine architecture.
Comparison

Traditional Style (Ex: container)

- User’s Space: app container app container app container
- Admin’s Space: kernel machine
- Invoke app.
- Power always up

Server Centric Architecture

Pros:
• Multi Tenant
• Quick Response (No Rebooting)
Cons:
• Kernel is not replaced.

Application Centric Architecture

Pros:
• Apps can select a kernel & hardware.
• Apps occupy the machine and extract the performance.
Cons:
• Set up overhead (Rebooting)

BMC

- Select a kernel
- Select a physical machine
- BMC manager
- Remote Machine management (WOL, AMT, IPMI)
- Power frequently up/down

Boots the kernel & app.

Machine
- network bootloader
- machine
- network bootloader
- machine
- network bootloader
- machine

Remote Machine management (WOL, AMT, IPMI)
Procedure to execute BMC command

```
#bmc run "docker-img" "kernel" "initrd" "command"
```

Node-1

---

**BMC Command**

1. `ssh pub-key`
2. HTTPS (apache)
3. iPXE script
4. kernel & initrd
5. NFS mount or download to RAM FS
6. docker image
7. `ssh`
8. Power Off (shutdown command, AMT, IPMI)

---

**BMC Manager**

- Power On (WOL, AMT, IPMI)
- Platform authentication
- Authenticate
- Download iPXE script
- Download kernel & initrd
- NFS mount or download to RAM FS
- Request ssh connection

---

**Docker Hub**

- Docker Image

---

**Docker Hub**

- Kernel & initrd
- ipxe
- Power On (WOL, AMT, IPMI)

---

**BMC Hub**

- Kernel & initrd
- iPXE
- Power On (WOL, AMT, IPMI)

---

**Node-1**

- iPXE
- IP address (bmc-ID)

---

**Procedure to execute BMC command**

1. Connect to BMC using ssh with public key.
2. Use HTTPS to access the BMC.
3. Run iPXE script to power on the system.
4. Authenticate the platform.
5. Download the kernel and initrd.
6. NFS mount or download to RAM FS.
7. Request ssh connection to the BMC.
8. Power off using the shutdown command.
Remote Machine Boot Procedure

1. Power-on a node machine with Remote Machine Management (WOL, Intel AMT, IPMI)

2. Network Boot Loader (iPXE)
   - Get kernel and initrd from a HTTP/HTTPS server.

3. The downloaded initrd mounts a Docker image.
   - NFS mode
   - RAM FS mode

4. Boot procedure in a Docker image
   - Fortunately, Docker image keeps boot procedure.

5. SSH is connected from BMC command
   - Run an application.
# Remote Machine Management

<table>
<thead>
<tr>
<th>Protocol</th>
<th>WOL</th>
<th>Intel AMT</th>
<th>IPMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic Packet (MAC address)</td>
<td>HTTPS (IP address)</td>
<td>RMPC (IP address)</td>
<td></td>
</tr>
<tr>
<td>Power-On</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Power-Off</td>
<td>×</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Security</td>
<td>×</td>
<td>Password</td>
<td>Password</td>
</tr>
<tr>
<td>Comment</td>
<td>Most PCs have WOL.</td>
<td>High level Intel machine</td>
<td>Server Machine (Slow BIOS)</td>
</tr>
</tbody>
</table>
Network Boot Loader

- PXE is the most famous, but it is limited for LAN, because it depends on “magic packet” of Layer 2.

- BMC uses iPXE which download “kernel” and “initrd” from HTTP/HTTPS.
  - iPXE is custimzed by its scripting language. BMC uses it.

- The iPXE downloads kernel and initrd.

```
#!/ipxe
ifopen net0
set net0/ip 192.168.0.101
set net0/netmask 255.255.255.0
set net0/gateway 192.168.0.1
set dns 192.168.0.1
:loop
chain http://192.168.0.200/cgi-bin/baremetal.ipxe || goto waiting
exit
:waiting
sleep 1
goto loop
```
How to boot OS (Linux)

• The downloaded “initrd” is customized to mount an Docker image. It offers 2 mount methods.
  – NFS mode
    • Download necessary data only and fast boot, but it needs to download data to run applications after boot.
  – RAMFS mode
    • Download full Docker image and slow boot, but application runs fast after boot.

• Boot procedure in the Docker image.
  – An Docker image keeps boot procedure for each application because each application package designed to include them.
  – BMC utilizes these boot procedures to run daemons, such as the SSH, because an application in the Docker image is executed by remote procedure calls from BMC manager.
Power Consumption

• Each node has power meter “WattChecker”.
• WattChecker measures power consumption from the power-on caused by WOL, AMT, or IPMI.
• BMC manager keeps the log of the power consumption.
• The log is used for power accounting.

• It is coarse, but it shows affinity between application and architecture.
Current Implementation

• Current BMC Manager is implemented with shell script.
  – 4,500 LOC.

• Power consumed on each node is measured by WattChecker.

• We have tried several machines as BMC nodes.
  – From Atom to Xeon.
  – Application can select machine considering power consumption.
## Spec of Test Machines

<table>
<thead>
<tr>
<th></th>
<th>Remote machine management</th>
<th>CPU, Core/thread, Clock (Burst time), Power</th>
<th>Logical performance GFLOPS (Burst time)</th>
<th>Issue date</th>
<th>Memory</th>
<th>NIC (queue)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Power</td>
<td>WOL</td>
<td>Celeron (N3050), 2/2, 1.6 (2.16)GHz, 8W</td>
<td>6.4 (8.6)</td>
<td>2015</td>
<td>8GB</td>
<td>RealTek r8169 (1)</td>
</tr>
<tr>
<td>Intel NUC 5CPYH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NotePC</td>
<td>Intel AMT</td>
<td>i7 (3520M) 2/4, 2.9 (3.6)GHz, 35W</td>
<td>46.4 (57.6)</td>
<td>2012</td>
<td>16GB</td>
<td>Intel e1000 (1)</td>
</tr>
<tr>
<td>Lenovo ThinkPAD T430s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DesktopPC</td>
<td>Intel AMT</td>
<td>Core 2Quad (Q9400) 4/4, 2.66GHz, 95W</td>
<td>42.656</td>
<td>2008</td>
<td>16GB</td>
<td>Intel e1000 (1)</td>
</tr>
<tr>
<td>Dell Optiplex 960</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td>IPMI</td>
<td>Xeon (X5650) 6/12, 2.66 (3.06)GHz, 95W</td>
<td>63.984 (73.44)</td>
<td>2010</td>
<td>8GB</td>
<td>Broadcom NeXtreme II (8)</td>
</tr>
<tr>
<td>Dell PowerEdge T410</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Boot performance (overhead)

- They are BMC’s overhead.
- The performance improved by optimization must surpass the overhead.

### NFS mode

<table>
<thead>
<tr>
<th>Processor</th>
<th>Time (s)</th>
<th>Power (J)</th>
<th>Traffic (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celeron</td>
<td>35.4</td>
<td>242</td>
<td>49.5</td>
</tr>
<tr>
<td>Core2 Quad</td>
<td>28.1</td>
<td>1,773</td>
<td>49.3</td>
</tr>
<tr>
<td>i7</td>
<td>20.9</td>
<td>481</td>
<td>49.1</td>
</tr>
<tr>
<td>Xeon</td>
<td>92.6</td>
<td>9,932</td>
<td>49.8</td>
</tr>
</tbody>
</table>

### RAMFS mode

<table>
<thead>
<tr>
<th>Processor</th>
<th>Time (s)</th>
<th>Power (J)</th>
<th>Traffic (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celeron</td>
<td>55.6</td>
<td>402</td>
<td>92.9</td>
</tr>
<tr>
<td>Core2 Quad</td>
<td>40.0</td>
<td>2,493</td>
<td>92.8</td>
</tr>
<tr>
<td>i7</td>
<td>34.3</td>
<td>775</td>
<td>92.8</td>
</tr>
<tr>
<td>Xeon</td>
<td>102.7</td>
<td>11,015</td>
<td>92.6</td>
</tr>
</tbody>
</table>

OO

46 times
Tested Application and Optimization

<table>
<thead>
<tr>
<th>Application</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Multiplication with OpenBlas</td>
<td>Hyper Threading off</td>
</tr>
<tr>
<td>Redis benchmark</td>
<td>Transparent Huge Pages off</td>
</tr>
<tr>
<td>Apache benchmark</td>
<td>Receive Flow Steering off</td>
</tr>
</tbody>
</table>

- This presentation shows the result of Matrix multiplication with/without Hyper Threading.
  - The experiment measured the time for 10 times of matrix multiplications on OpenBlas optimized for each machine.
## Performance Difference

10 times of matrix multiplications $[12800:12800]$ on OpenBlas optimized for each machine.

<table>
<thead>
<tr>
<th></th>
<th>Time (s)</th>
<th>Power (j)</th>
<th>GFLOPS</th>
<th>Power/(GFLOPS*time)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Celeron</strong></td>
<td>12,783.8</td>
<td>125,084</td>
<td>2.99 (34.7%)</td>
<td>3.27</td>
</tr>
<tr>
<td><strong>Core2Quad</strong></td>
<td>1,060.2</td>
<td>140,346</td>
<td>39.8 (93.3%)</td>
<td>3.32</td>
</tr>
<tr>
<td><strong>i7 HTT-on</strong></td>
<td>961.4</td>
<td>55,315</td>
<td>43.8 (76.0%)</td>
<td>1.31</td>
</tr>
<tr>
<td><strong>i7 HTT-off</strong></td>
<td>827.1</td>
<td>45,364</td>
<td>50.9 (88.4%)</td>
<td>1.08</td>
</tr>
<tr>
<td><strong>Xeon HTT-on</strong></td>
<td>945.6</td>
<td>211,908</td>
<td>44.6 (60.7%)</td>
<td>5.02</td>
</tr>
<tr>
<td><strong>Xeon HTT-off</strong></td>
<td>698.9</td>
<td>151,760</td>
<td>60.5 (82.4%)</td>
<td>3.59</td>
</tr>
</tbody>
</table>

- The results show **no hyper threading is better**.
- Xeon shows the best performance, but i7 shows cost effective.

(\[\text{Celeron (Atom Core) is not cost effective CPU.}\])
Performance improvement which compensates the boot overhead

<table>
<thead>
<tr>
<th></th>
<th>Boot overhead</th>
<th>Improvement at [6400:6400]</th>
<th>Improvement at [12800:12800]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time (sec)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i7</td>
<td>35.4</td>
<td>15.9</td>
<td>134.3</td>
</tr>
<tr>
<td>Xeon</td>
<td>108.0</td>
<td>29.8</td>
<td>246.7</td>
</tr>
<tr>
<td><strong>Power (joule)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i7</td>
<td>1,805.3</td>
<td>1,150</td>
<td>9,951</td>
</tr>
<tr>
<td>Xeon</td>
<td>11,274.5</td>
<td>6,792</td>
<td>60,148</td>
</tr>
</tbody>
</table>

- Overheads caused by booting were compensated before [12800:12800].
BMC Extension

- NVIDIA-Docker
- Moby
- Intel Clear Containers
- OSes for Container
Extension for “NVIDIA-Docker”

• NVIDIA-Docker runs CUDA applications.
  – It manages CUDA and driver of NVIDIA GPU.
    • A suitable CUDA is added Docker image automatically.
    • So, users do not need to install CUDA.
  – Users don’t need to care about CUDA version.

• BMC is now customizing to add CUDA which matches to NVIDIA-driver.
  – The target is TensorFlow on NVIDIA-Docker.
    • https://hub.docker.com/r/tensorflow/
Moby is a framework to assemble container systems. Moby allows to run "redis" container without Docker!

- Why do we need "containerd" and "LinuxKit"?
- Redis should run on native Linux kernel?

https://www.slideshare.net/Docker/dockercon-2017-general-session-day-1-solomon-hykes-75362520
Moby

- Moby is a framework to assemble container systems.
- Moby allows to run “redis” container without Dock!
  - Why do we need “containerd” and “LinuxKit”?
  - Redis should run on native Linux kernel?

https://www.slideshare.net/Docker/dockercon-2017-general-session-day-1-solomon-hykes-75362520
Intel Clear Container

- Intel Clear Container is a counter-thesis of container. It encourages to use virtualization (Intel VT).
  - Intel insists that Intel VT and Linux’s boot (< 200ms!) are fast. Why don’t you use virtualization? Virtualization offers stronger isolation.

- BMC’s offers counter-thesis against Intel Clear Container.
  - If Linux’s boot is fast, why don’t you use native Linux?
  - BMC offers more flexibility of kernel customization.
Current BMC target

• Many Docker OSes are proposed.
  – CoreOS
  – RancherOS
  – Snappy Ubuntu Core
  – RedHat Project Atomic
  – Mesosphere DC/OS
  – VMware Photon

• BMC can use these Linux kernels.
  – BMC only requires kernel and container image.
Related works

• Triton [Joyent’s product]
  – Triton = Docker + SmartOS.
    • In order to optimize, user needs to customize SmarOS.

• LinuxBIOS/BProc Cluster[HPCS’02]
  – Testbed for kernel test. It is not so easy to implement because it requires to replace BIOS.

• SLURM[ICDCN’14]
  – Measure power consumption for an application. It depends on function to measure power (Intel RAPL: Running Average Power Limit, or CRAY machine).
Conclusions

• BMC (Bare-Metal Container) runs a container (Docker) image with a suitable Linux kernel on a remote physical machine.

• The overhead of BMC was compensated by the improved performance of applications.


• Docker Image for BMC manager: [https://hub.docker.com/r/baremetalcontainer/](https://hub.docker.com/r/baremetalcontainer/)

• Source Code: [https://github.com/baremetalcontainer](https://github.com/baremetalcontainer)