Performance Prediction and Optimization using Linux/Cgroups with IO throttle

Yuzuru Maya, Ph.D.
Hitachi, Ltd., Yokohama Research Laboratory
Agenda

1. Background
2. Outline of Linux/Cgroups
3. System configuration and Requirement
4. Requirement level 1
   (LinuxConJapan2011 Review)
5. Proposed method and Evaluation
6. Conclusion
1. Background

(1) Grow up server consolidation with progress of virtualization technology.
- Guarantee real time and performance at same time.

(2) Expand the field of Linux/KVM
- Control system: Power, Traffic and Water control system
- Information system: Banking system, OLTP*1
- Embedded system

(3) Bandwidth control (cgroups) becomes important and will be more important.
- Optimize Bandwidth control every each application or system.
- Cooperation with access control for resources is necessary.

*1: OLTP: On-Line Transaction Processing
2. Outline of Linux/Cgroups

(1) Bandwidth control
- Bandwidth control of CPU, Memory, Network and Disk.
- Account CPU usage.

(2) Access control
- Access control of device.
- Constrain CPU and memory placement of tasks.
- Name space control.

(3) Group control
- Freeze task.
- Checkpoint and restart.

(4) IO Throttling
- Upper limit of resource.
Parameters of IO Throttling

(1) Upper limit on the number of operations in bytes per second
- blkio.throttle.read_bps_device
  upper limit on the number of read operations in bytes per second.
- blkio.throttle.write_bps_device
  upper limit on the number of write operations in bytes per second.

(2) Upper limit on the number of operations in operations per second
- blkio.throttle.read_iops_device
  upper limit on the number of read operations in operations per second.
- blkio.throttle.write_iops_device
  upper limit on the number of write operations in operations per second.

(3) Reports the number of I/O operations
- blkio.throttle.io_serviced
  reports the number of I/O operations
- blkio.throttle.io_service_bytes
  reports the number of bytes operations
### 3. System configuration and Requirement

<table>
<thead>
<tr>
<th></th>
<th>Control system</th>
<th>Information system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
<td>Process control (Monitoring of systems)</td>
<td>OLTP (Processing for the event)</td>
</tr>
<tr>
<td><strong>System configuration</strong></td>
<td>Periodically</td>
<td>At event occurrence</td>
</tr>
<tr>
<td><strong>Characteristic</strong></td>
<td>Load change is small. (Load prediction is easy.)</td>
<td>Load change is large. (For such as the spark load)</td>
</tr>
<tr>
<td><strong>Requirements:</strong></td>
<td><strong>Level1 blkio.weight</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(LinuxConJapan2011)</td>
<td>Check whether 2 tasks can run at the same time.</td>
</tr>
<tr>
<td></td>
<td><strong>Level2 +IO_throttling</strong></td>
<td>(This presentation: LinuxConEurope2012)</td>
</tr>
<tr>
<td></td>
<td>Take resources away from tasks executing, and assign the other task.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Level3 +Live migration etc</strong></td>
<td>Suitable</td>
</tr>
<tr>
<td></td>
<td>(From viewpoint of Reliability)</td>
<td></td>
</tr>
</tbody>
</table>
IO Performance and Requirements level

Throughput (Load change)

High (Big)

Req. Level 1
Run at same time

Important Task
(Req. Level 1)

Req. Level 2
Take resources away from tasks executing, and create new task.

New Task
(Non-RT task)
(Req. Level 2)

RT Task
(Req. Level 2)

Semi-RT Task
(Req. Level 2)

Non Important Task
(Req. Level 1)

Low (Small)

Latency

Low

High

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
4. Requirement level 1 (LinuxConJapan2011 Review)

[Target system]
(1) Important task
   Require both Throughput and Latency.
(2) Non-important task
   Run in the remaining resources of Important task.

[Performance prediction method based RA*1]
(1) Collect statistic data and normalize IO-weight (blkio.weight/100).
   Total of IO-weight is 10.
(2) Domain division prediction
   Divide IO-weight in the some domains and perform RA every domain to adapt throughput and latency whose characteristic is linear or non-linear.

*1: Regression Analysis
(3) Interactive prediction

Predict IO-Weight corresponding throughput and latency from both the lower side and the upper side.

For example, throughput and latency of 16kB (middle) are predicted by data of 4kB (lower side) and 64kB (upper side).

[Feedback function]

(1) Compare measured data with performance prediction and check whether the difference is within constant value.
(2) Perform the prediction of the weight by a regression analysis again if the difference are more than a constant value.
(3) By this function, we can improve prediction precision to enable to be within an allowable error.
Target we research and develop on Linux/Cgroups

- **AP-A**
  - Priority: High
- **AP-B**
  - Priority: Low

Require IO Throughput and Latency.

**Statistic data**

**CPU/Memory Control**

**IO Control**

**Resource Control/Access Control**

**Monitor**

**Performance Prediction**

**Compare Data**
required data and measured.

**Feedback**

(1) Dev target 1
Develop Cgroups itself.

(2) Dev target 2
Develop using Cgroups

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
Summary of Requirement level 1

(1) Performance prediction
- Throughput
  Almost proportional and easy to predict and calculate.
- Latency
  Detect important point which turns worse suddenly and is not proportional.

(2) Set IO-Weight
- Detect IO-Weight meeting throughput and latency at the same time and can run them using this IO-Weight.
5. Proposed method and Evaluation

[Purpose]
Create other task and run even if we sacrifice throughput and latency temporarily.

[Target system]
(1) RT task
- Require latency mainly.
(2) Semi-RT task
- Require latency weekly.
(3) Non-RT task
- Create task at maintenance and so on.
- Need throughput.
[Requirements]

(1) At operation
   Realize both throughput and latency that user appointed.
   But they are not always so severe.

(2) At emergency e.g. maintenance
   “How to collect resources from other VM at emergency” is an important theme. Because in virtualized system, the version-up of OS and applications is needed without affecting other VMs.

   It is important that we assign resources and can use them effectively. Therefore, resource management (bandwidth control) becomes the required function.

VM : Virtual machine
[Proposed method]

(1) Cooperation IO-Weight and IO-Throttling parameters.
   - There is Range that does not depend on IO-Weight in latency.
   - More effective bandwidth control by Upper limit setting of IO-Throttling.

(2) Calculate performance by Formula roughly.

(3) Compare measured data with predicted data in detail.

(4) Redistribute some resources
   - Decide IO-Weight and IO-Throttling satisfying Throughput and Latency by resources redistribution.

(5) Feedback
   - Evaluate it again until an error is within the allowable range.

error : difference of measured value and predicted value.
Evaluation from Formula

[Purpose]
Roughly evaluate performance (Latency, Throughput) from the calculating Formula as follows.

\[
\text{Perf (Latency, Throughput)} = \min (\text{Upper Limit} \times \frac{x}{10}, y) \\
\text{of Total}
\]

\[
x: \text{IO-Weight (1–9)}
\]
\[
y: \text{IO-Throttling (Upper limit of IO bandwidth)}
\]

\[
T_p = \min \{ T_p (\text{io-th}), T_p (\text{io-wt}) \}
\]
\[
L_t = \min \{ L_t (\text{io-th}), L_t (\text{io-wt}) \}
\]

Perf : Performance
Evaluation by Formula

(1) Effect by IO-Throttling

Throughput $T_{p(i0-th)}$ vs. IO-Weight
- Resources to hand to new task
- Maximum IO band

Latency $L_{(io-th)}$ vs. IO-Weight
- Resources to hand to new task
- Maximum IO band

(2) Effect by some division of resources

Throughput $T_{p(i0-wt)}$ vs. IO-Weight
- Resources to hand to new task

Latency $L_{(io-wt)}$ vs. IO-Weight
- Resources to hand to new task

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
Evaluation by Measurement

[Purpose]
I evaluate it by the measurement and make the relations with IO-weight and IO-throttling clear.

[Outline]
(1) blkio.weight
Bandwidth of IO, range of weights is from 100 to 1000.
IO-Weight is blkio.weight/100 (from 1 to 10) to simplify.
(2) blkio.throttle.read_bps_device
Upper limit of IO bandwidth.
(3) Construct Measurement environment.

```
mount -t cgroup none /cgroup/ -o blkio
mkdir -p /cgroup/test1 /cgroup/test2
echo 900 > /cgroup/test1/bklio.weight
echo 100 > /cgroup/test2/bklio.weight
echo "8:16 1048576" > /cgroup/test1/bklio.throttle.read_bps_device
echo "8:16 1048576" > /cgroup/test1/bklio.throttle.write_bps_device
./fio-test.sh
```
# Measurement Environment

## [Measurement Machine]

<table>
<thead>
<tr>
<th>#</th>
<th>Item</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Machine</td>
<td>Dell Optiplex 960</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU: Intel Q9550 2.8GHz, Memory: 8 GB</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>Fedora 16</td>
</tr>
<tr>
<td>3</td>
<td>Benchmark</td>
<td>FIO (Flexible IO) version 2.0.7</td>
</tr>
</tbody>
</table>

## [FIO condition]

<table>
<thead>
<tr>
<th>#</th>
<th>Item</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Measurement item</td>
<td>IO Throughput (IO issue per second)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&amp; Latency (IO Complete clat)</td>
</tr>
<tr>
<td>2</td>
<td>Access type</td>
<td>Random read/write</td>
</tr>
<tr>
<td>3</td>
<td>Measure time</td>
<td>360 sec</td>
</tr>
<tr>
<td>4</td>
<td>IO type</td>
<td>Asynchronous I/O</td>
</tr>
<tr>
<td>5</td>
<td>I/O type</td>
<td>4kB</td>
</tr>
</tbody>
</table>
Measurement item and Result

queryA: (g=0): rw=randrw, bs=4K-4K/4K-4K, ioengine=libaio, iodepth=50
Starting 1 process
queryA: (groupid=0, jobs=1): err= 0: pid=6562
read : io=602MB, bw=1,710KB/s, iops=53, runt=360548msec
  slat (usec): min=6, max=331K, avg=260.46, stdev=6901.12
  clat (usec): min=21, max=2,183K, avg=474887.53, stdev=433054.37
bw (KB/s) : min= 0, max= 4187, per=103.88%, avg=1776.39, stdev=847.50
write: io=590MB, bw=1,677KB/s, iops=52, runt=360548msec
  slat (usec): min=6, max=1,192K, avg=459.21, stdev=12281.17
  clat (usec): min=262, max=2,200K, avg=488998.71, stdev=431350.99
bw (KB/s) : min= 0, max= 5396, per=96.80%, avg=1622.29, stdev=947.72
cpu : usr=0.04%, sys=0.23%, ctx=36654, majf=0, minf=41
submit : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=99.6%, >64=0.0%
complete : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, >64=0.0%
issued r/w: total=19268/18893, short=0/0
lat (usec): 50=0.01%, 100=0.01%, 250=0.02%, 500=0.65%, 750=0.09%
lat (usec): 1000=0.12%
lat (msec): 2=0.15%, 4=0.24%, 10=1.78%, 20=3.31%, 50=7.80%
lat (msec): 100=8.29%, 250=16.79%, 500=22.52%, 750=15.86%, 1000=9.92%
lat (msec): 2000=12.33%, >2000=0.12%

Run status group 0 (all jobs):
  READ: io=602MB, aggrb=1,710KB/s, minb=1,751KB/s, maxb=1,751KB/s, minr=360548msec, maxr=360548msec
  WRITE: io=590MB, aggrb=1,676KB/s, minb=1,717KB/s, maxb=1,717KB/s, minr=360548msec, maxr=360548msec
Disk stats (read/write):
  dm:0: ios=31502/34760, merge=0/0, ticks=15140973/16651011, in_queue=31824277, util=100.00%, aggrios=0/0, aggrmerge=0/0,
aggrticks=0/0, aggrin_queue=0, aggrutil=0.00%
  sdb: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=nan%

Measurement item
- Throughput
  (IO issue per sec)
- Latency (clat)
  clat: complete latency
Measurement result of Latency

□: Semi-RTtask, No BW

Note BW: Bandwidth
Measurement result of Throughput

△: Semi-RTtask, No BW

Note BW: Bandwidth
Resources redistribution algorithm

Evaluation of RT Task

Evaluation of New Non-RT Task

1. Lower IO-Throttling and IO-Weight level if possible (sacrifice of RT task).
2. RT task carries resources to new Non-RT Task.
3. Calculate throughput from assigned resources, and the non-RT task checks whether requirements are satisfied.
4. It is repeated until these are satisfied by a feedback function.
Evaluation result (1)

(1) Set the most suitable IO-Wt and IO-Th.
(2) By feedback, I evaluate whether new created task AP–C can be run or not.

<table>
<thead>
<tr>
<th>#case</th>
<th>AP</th>
<th>Before new Non-RT task</th>
<th>After new Non-RT task</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirement</td>
<td>Parameter</td>
<td>Requirement</td>
<td>Parameter</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lt (msec)</td>
<td>Tp (times)</td>
<td>IO-Wt (MB/sec)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>2000</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>AP–A</td>
<td>(RT Task)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>2,000</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>AP–B</td>
<td>(Semi-RT Task)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>1,000</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AP–C</td>
<td>(New Non-RT Task)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>1,500</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Lt: Latency  IO–Wt: Latency
Tp: Throughput IO–Th: Throttling

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
# Evaluation result (2)

<table>
<thead>
<tr>
<th>#case</th>
<th>AP</th>
<th>Before new Non-RT task</th>
<th>After new Non-RT task</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Requirement</td>
<td>Parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt (msec)</td>
<td>Tp (times)</td>
<td>IO-Wt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>1,000</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>AP-A</td>
<td>250</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>(RT Task)</td>
<td>250</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>AP-B</td>
<td>1,400</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(Semi-RT)</td>
<td>1,400</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>AP-C</td>
<td>-</td>
<td>-</td>
<td>1,400</td>
</tr>
<tr>
<td></td>
<td>(New Non-RT)</td>
<td>-</td>
<td>-</td>
<td>1,400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>requirement</td>
<td>Parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt (msec)</td>
<td>Tp (times)</td>
<td>IO-Th</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,400</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,400</td>
<td>1,500</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>AP-A</td>
<td>200</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>(RT Task)</td>
<td>200</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>AP-B</td>
<td>1,400</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(Semi-RT)</td>
<td>1,400</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>AP-C</td>
<td>-</td>
<td>-</td>
<td>1,400</td>
</tr>
<tr>
<td></td>
<td>(New Non-RT)</td>
<td>-</td>
<td>-</td>
<td>1,400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>requirement</td>
<td>Parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt (msec)</td>
<td>Tp (times)</td>
<td>IO-Th</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>700</td>
<td>1,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>1,000</td>
</tr>
<tr>
<td>4</td>
<td>AP-A</td>
<td>150</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>(RT Task)</td>
<td>150</td>
<td>1,000</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>AP-B</td>
<td>700</td>
<td>1,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(Semi-RT)</td>
<td>700</td>
<td>1,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>AP-C</td>
<td>-</td>
<td>-</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>(New Non-RT)</td>
<td>-</td>
<td>-</td>
<td>1,000</td>
</tr>
</tbody>
</table>

*1 sac : sacrifice

Resource shortage after creating new task.
Case 1

Measurement result of Latency

<table>
<thead>
<tr>
<th>RTtask</th>
<th>Semi-RTtask</th>
<th>No BW</th>
<th>BW (100MB/sec)</th>
<th>BW (50MB/sec)</th>
<th>BW (25MB/sec)</th>
<th>BW (10MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Note BW: Bandwidth

Copyrigh (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
Measurement result of Throughput

Case 1

△: Semi-RTtask, No BW

Note BW: Bandwidth
Case 2

Measurement result of Latency

- □: Semi-RTtask, No BW
- ●: RTtask: BW (100MB/sec)
- ■: RTtask: BW (50MB/sec)
- ◆: RTtask: BW (25MB/sec)
- ▲: RTtask: BW (10MB/sec)

Note BW: Bandwidth

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
Case 2

Measurement result of Throughput

△: Semi-RTtask, No BW

RTtask
- □: No BW
- ●: BW (100MB/ sec)
- ○: BW (50MB/ sec)
- □: BW (25MB/ sec)
- ●: BW (10MB/ sec)

New task

Semi-RTtask

Note BW: Bandwidth

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
Case 3

Measurement result of Latency

- □: Semi-RTtask, No BW
- ○: RTtask
- □: No BW
- ▲: BW (100MB/sec)
- ▼: BW (50MB/sec)
- ◇: BW (25MB/sec)
- ■: BW (10MB/sec)

Note BW: Bandwidth
Case 3

Measurement result of Throughput

△: Semi-RTtask, No BW

Note BW: Bandwidth
Case 4

Measurement result of Latency

Note BW: Bandwidth

Copyright (c) 2012 Hitachi Ltd., Yokohama Research Lab. All right reserved
Case 4

Measurement result of Throughput

<table>
<thead>
<tr>
<th>RTtask</th>
<th>Semi-RTtask</th>
</tr>
</thead>
<tbody>
<tr>
<td>No BW</td>
<td>No BW</td>
</tr>
<tr>
<td>BW (100MB/sec)</td>
<td>BW (50MB/sec)</td>
</tr>
<tr>
<td>BW (25MB/sec)</td>
<td>BW (10MB/sec)</td>
</tr>
</tbody>
</table>

Note BW: Bandwidth
6. Conclusion

6.1 Summary
(1) Expand the coverage of bandwidth-control.
   - Show 3 requirement level for resources redistribution.
(2) Evaluate IO-Throttling with IO-Weight.
   - Calculate performance by formula.
   - Predict by the actual measurement.
(3) Propose new resources redistribution algorithm.

6.2 Future Work
(1) Evaluate KVM based system in VM environment.
   - this presentation is for APs.
(2) Predict performance in online and set.
   - CPU and memory etc.
(3) Cooperate resource management and access control.
Gracias por escuchar.

Thank you for your attention.