LinuxCon Japan 2012

FUJITSU

shaping tomorrow with you

Evaluation of CPU cgroup

Jun 7th, 2012

Taku Izumi
Fujitsu Limited

Table of Contents

FUJITSU

Motivation

FUJITSU

- Fujitsu sells cloud computing service for mission critical customers. In that service QoS/accounting is very important
- CPU bandwidth control feature is for cpu QoS
  - CPU resource provisioning according to the service level
- CPU cgroup is now enhanced to support CFS bandwidth control feature
  - was merged with linux-3.2 kernel
  - RHEL6.2 bundles kernel with CFS bandwidth control

FUJITSU

Outline of CPU cgroup

CPU cgroup subsystem

FUJITSU

- CPU cgroup is one of subsystems of cgroups and provides a control interface for scheduler

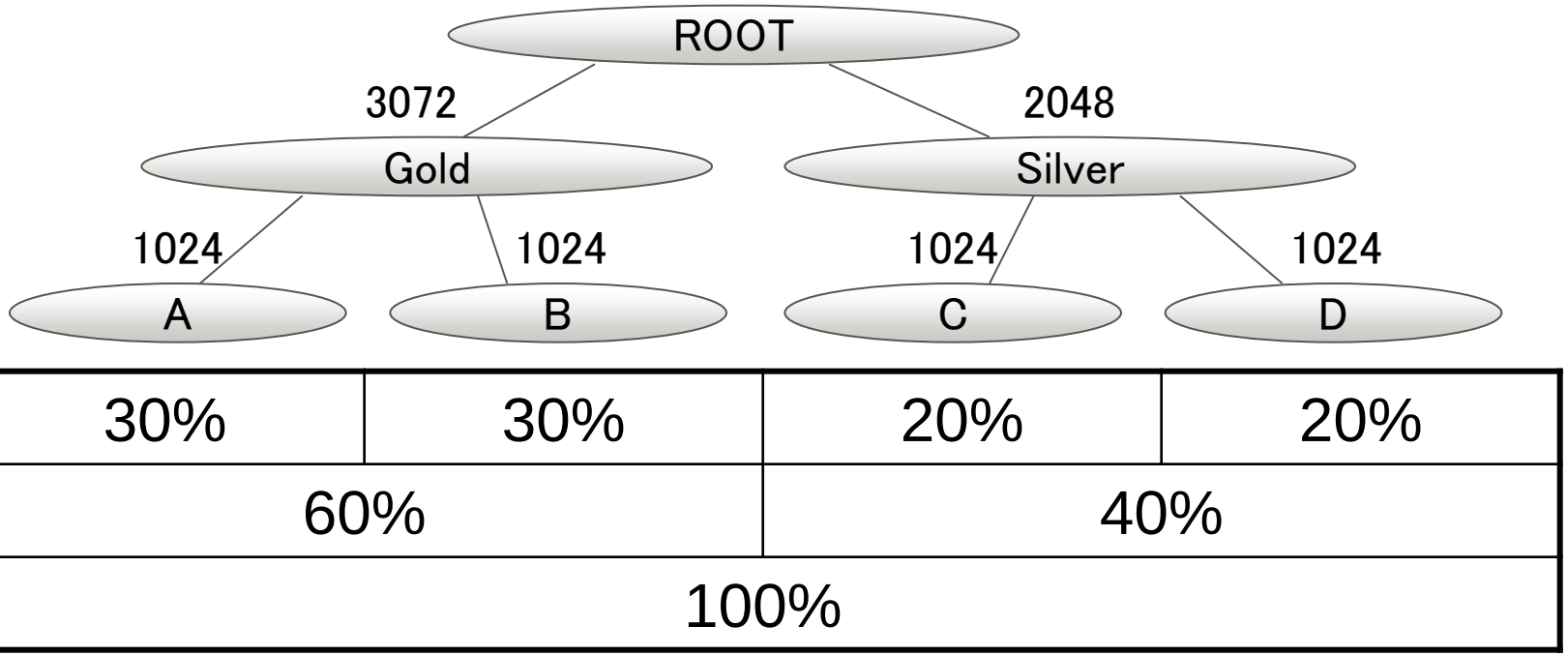
- Facilities
  - "share"
    - provisioning of proportional CPU resource through weight
    - lower-bound provisioning of CPU resource
  - bandwidth control for completely fair scheduler
    - aka CFS bandwidth control
    - provide an upper limit of CPU resource
    - very new feature
  - bandwidth control for real-time scheduler

main topic

“share” facility

FUJITSU

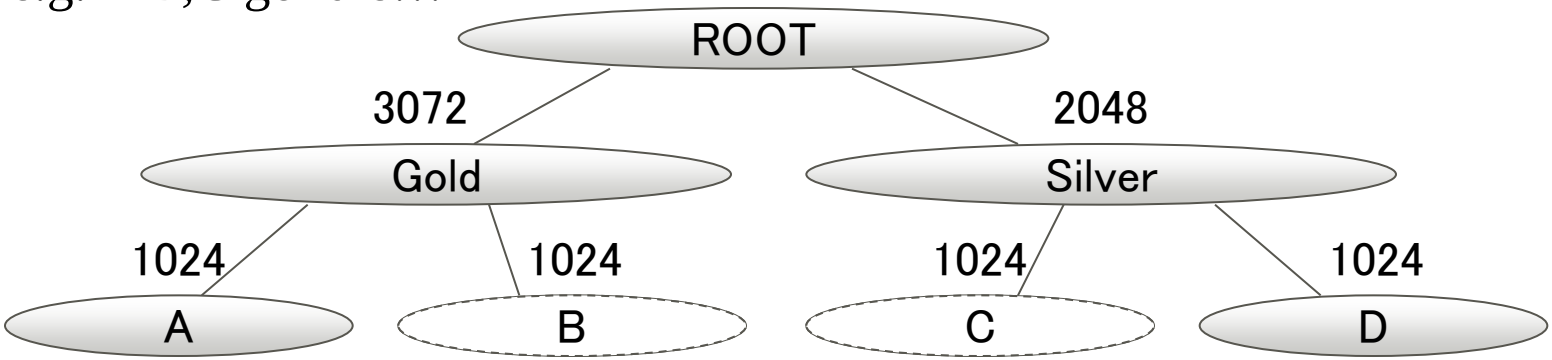
- cpu.shares to specify the weight to provide CPU time
  - e.g. Configure “Gold” group to receive 1.5x the CPU bandwidth that of “Silver” group
    - # echo 3072 > Gold/cpu.shares
    - # echo 2048 > Silver/cpu.shares



Known issue with share facility

- CPU time is shared among groups with runnable tasks
  - The amount of CPU time provided to the group depends on the state of neibouring groups
- It is difficult to estimate performance and not good for selling cpu time in enterprise system

e.g. if B,C go idle…



30%→60%	30%→0%	30%→0%	20%→40%
60%		40%	
100%			

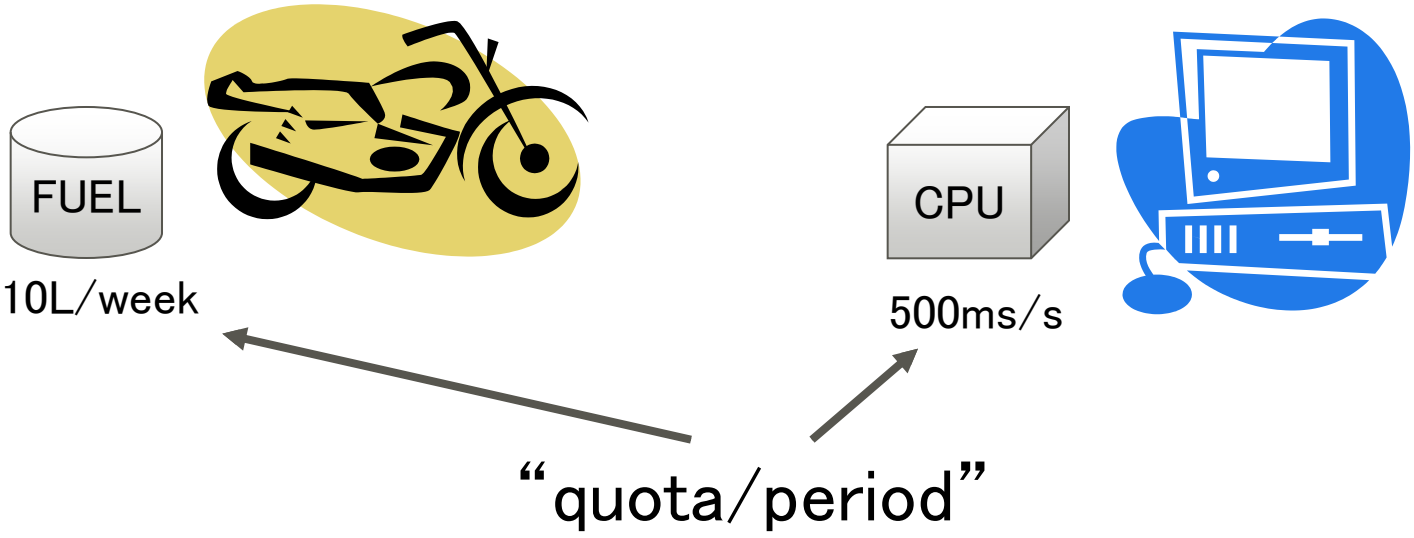
What we ask for is

FUJITSU

- CFS bandwidth control !
  - “share” facility is not suitable for our service
- Our requirement is the feature to limit CPU usage according to service class
  - The service level of low class users must not exceed that of high class users no matter what happens.

CFS Bandwidth control

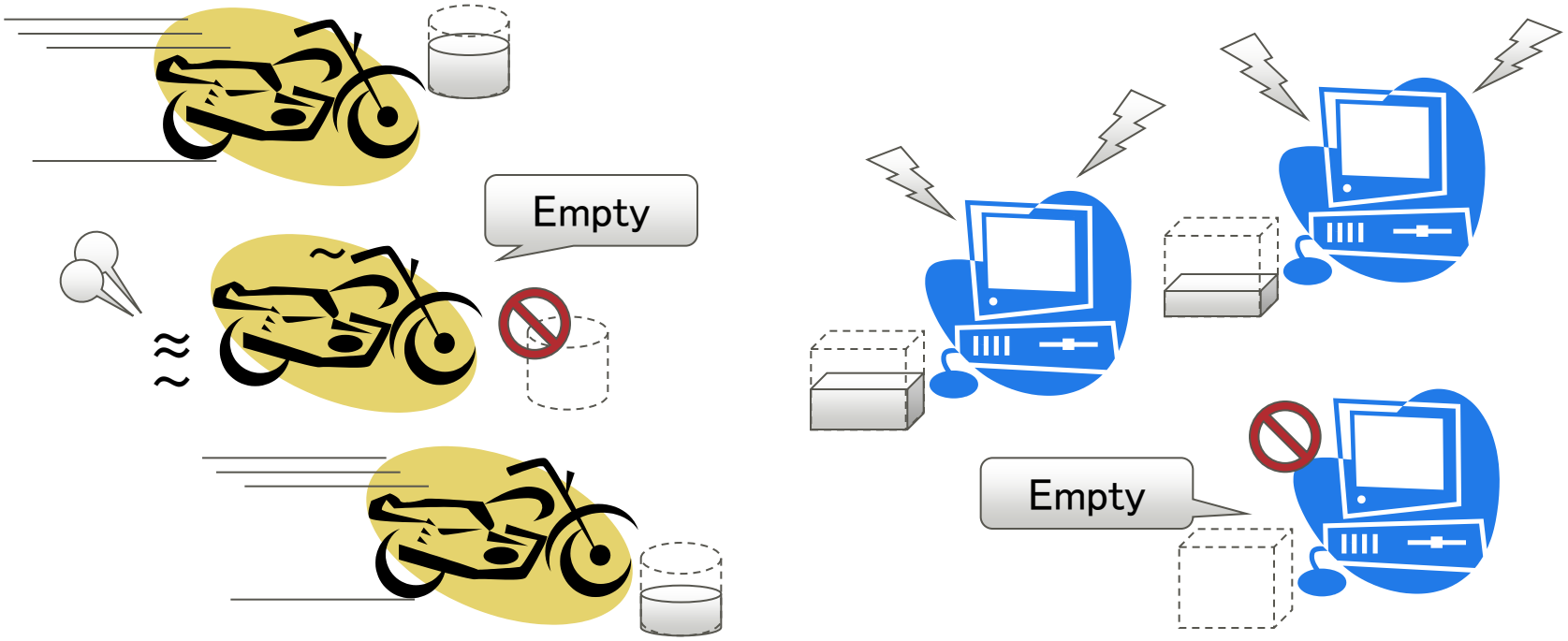
- The bandwidth allowed for a group is specified by quota and period.
- Within each given "period" (microseconds), a group is allowed to consume only up to "quota" microseconds of CPU time.
  - “quota” means maximum run-time in a specified “period”



CFS Bandwidth control (cont.)

FUJITSU

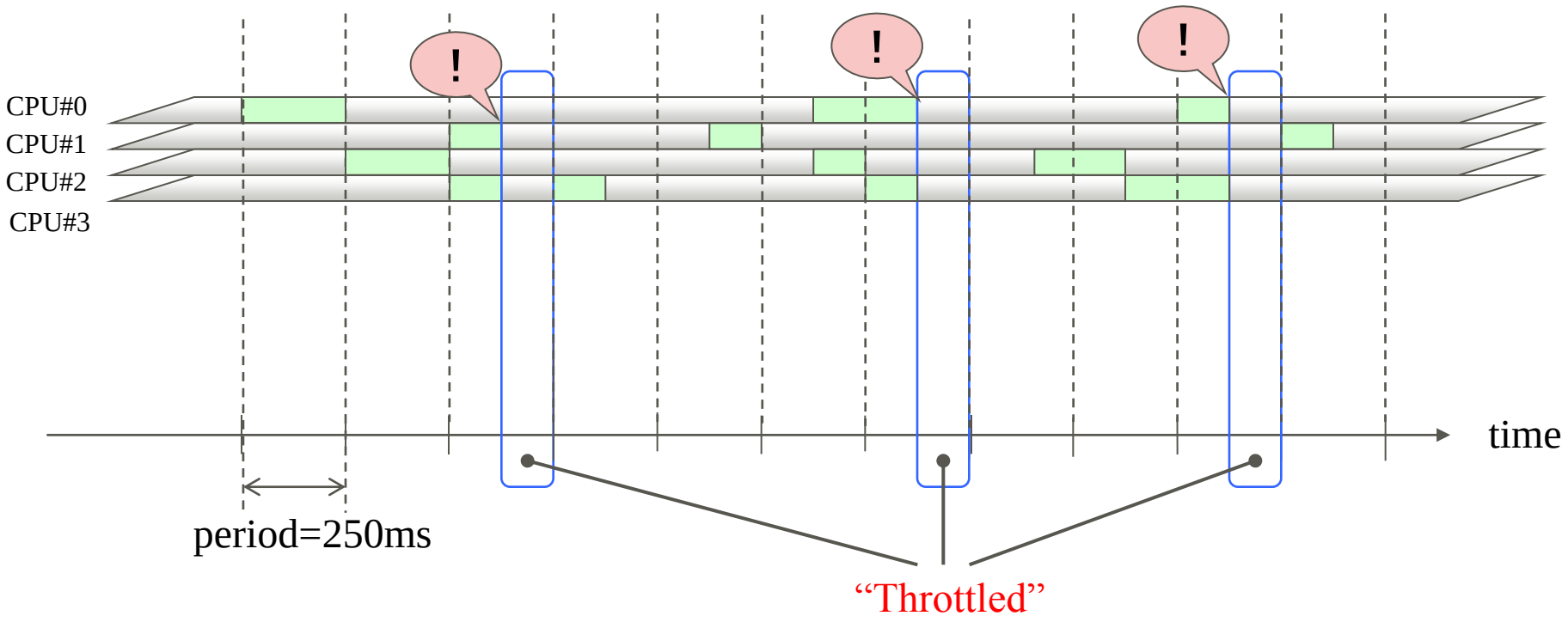
- When a group exhausts its own quota of CPU time per a period, tasks under the cgroup cpu will never scheduled until the next period.



How CFS bandwidth control works

FUJITSU

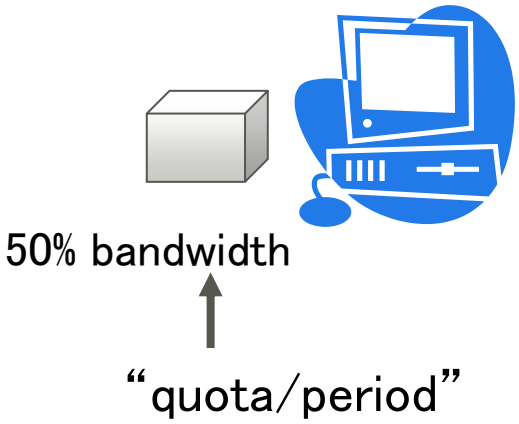
- Example
  - If period is 250ms and quota is also 250ms, the group will get 1 CPU worth of runtime every 250ms.



cgroup interface

FUJITSU

- Quota and period are managed within the cpu subsystem of cgroupfs.
  - cpu.cfs_quota_us:
    - The total available run-time within a period (in microseconds, ~1ms)
    - “-1” (means no restriction) is default
  - cpu.cfs_period_us:
    - The length of a period (in microseconds, 1s~1ms)
    - “100000” (100msec) is default



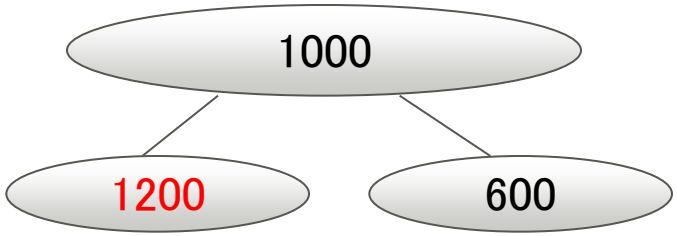

```
# echo 125000 > cpu.cfs_quota_us /* quota = 125ms */
# echo 250000 > cpu.cfs_period_us /* period = 250ms */
```

Hierarchical considerations

FUJITSU

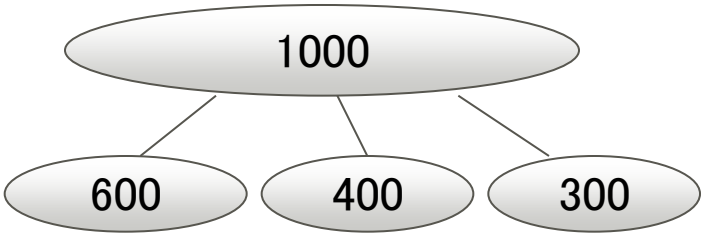
- The interface enforces that a child cgroup's quota/period ratio never be over parent's one
- However, it's allowed that aggregate quota of children is over parent's one for supporting works-conserving semantics

NG



A child's bandwidth cannot exceed the parent's bandwidth

OK

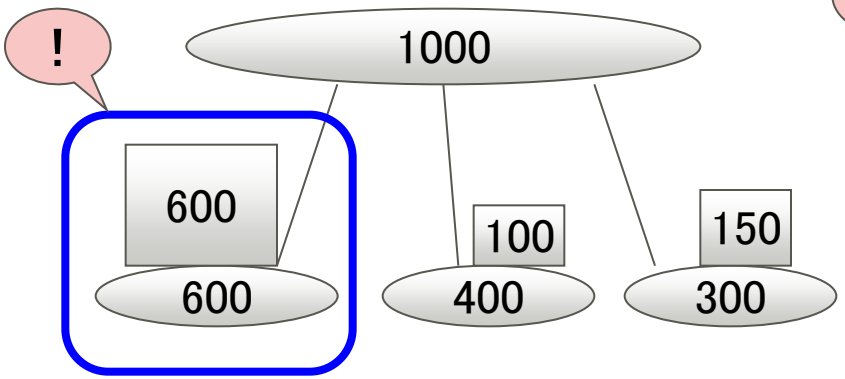


A total of children's bandwidths can exceed the parent's bandwidth

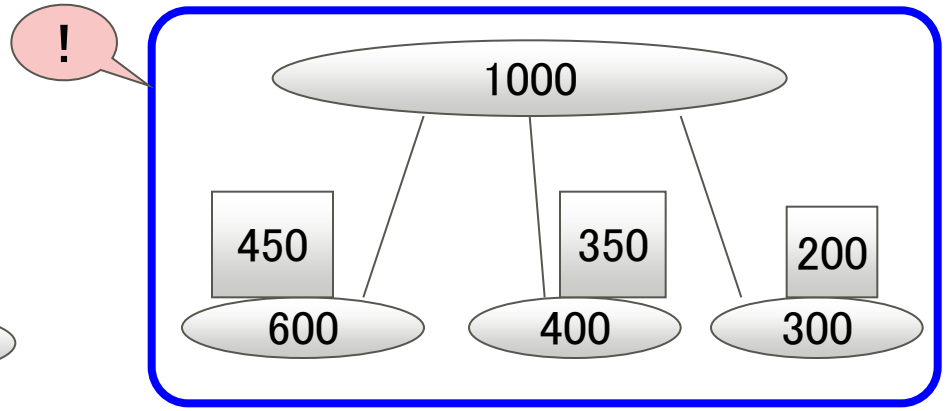
Hierarchical considerations (cont.)

- There are two ways in which a group is throttled:
  a. it fully consumes its own quota within a period
  b. a parent's quota is fully consumed within its period
- In case b) above, even though the child may have runtime remaining, it will not be allowed to run until the parent goes to the next period.

a)



b)



450+350+200 = 1000

FUJITSU

Evaluation of CFS bandwidth control

Experimental setup

FUJITSU

- Hardware
  - Fujitsu PRIMEQUEST 1800 E2

CPU	Intel Xeon E7-8870 (10core / 2.4GHz) * 2
Memory	128GB
NIC	Intel 82576NS

- OS

kernel	3.1.0-rc7-tip CONFIG_CFS_BANDWIDTH=y
qemu-kvm	0.14.0-7
libvirt	0.9.4-1

About benchmarks

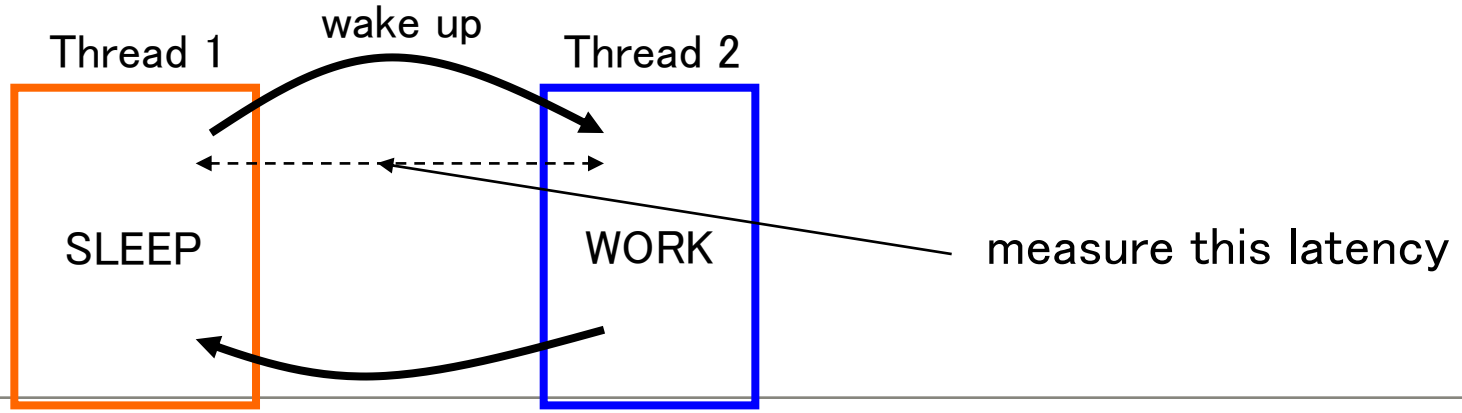
FUJITSU

- Himeno Benchmark M (256x128x128)
  - Benchmark for measuring FLOPS popular in Japan
- Unixbench version 5.1.2
  - Measure performance of UNIX based system
- Hackbench
  - Estimate the time of chat-like operation
    - # hackbench 150 process 500
- SysBench (0.4.12-5) oltp test
  - Benchmark a real database performance
    - # sysbench --test=oltp --num-threads=10 --db-driver=mysql --mysql-table-engine=innodb --oltp-table-size=1000000
    - using mysqld on localhost

About benchmarks (cont.)

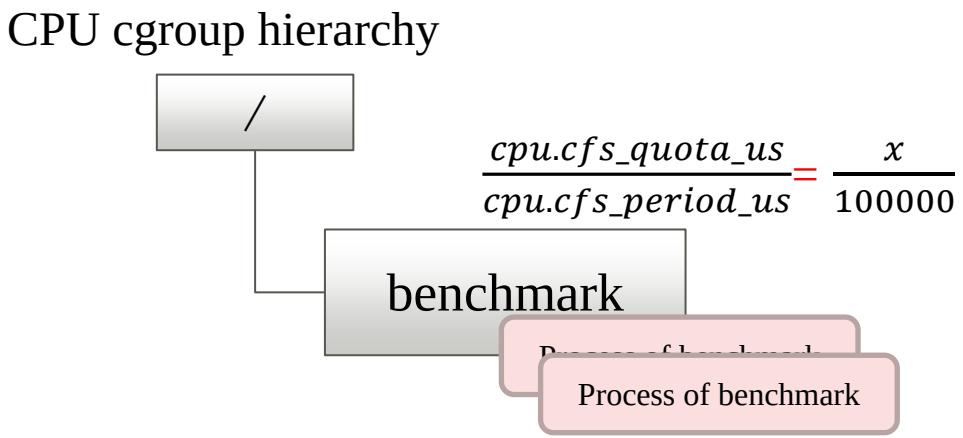
FUJITSU

- Super Pi benchmark ver. 2.0
  - Estimate the time of π calculation up to 1 million (2^{20}) digits
    - # time superpi 20
- Original sleep-work-wakeup benchmark
  - measuring process wake-up latency on not-busy host
  - lots of pairs of threads wake up each other with random sleep and a static job. Because of sleep, CPUs are not fully used.
  - used for emulate some customer's job queuing pipeline latency, waiting event and queuing jobs.

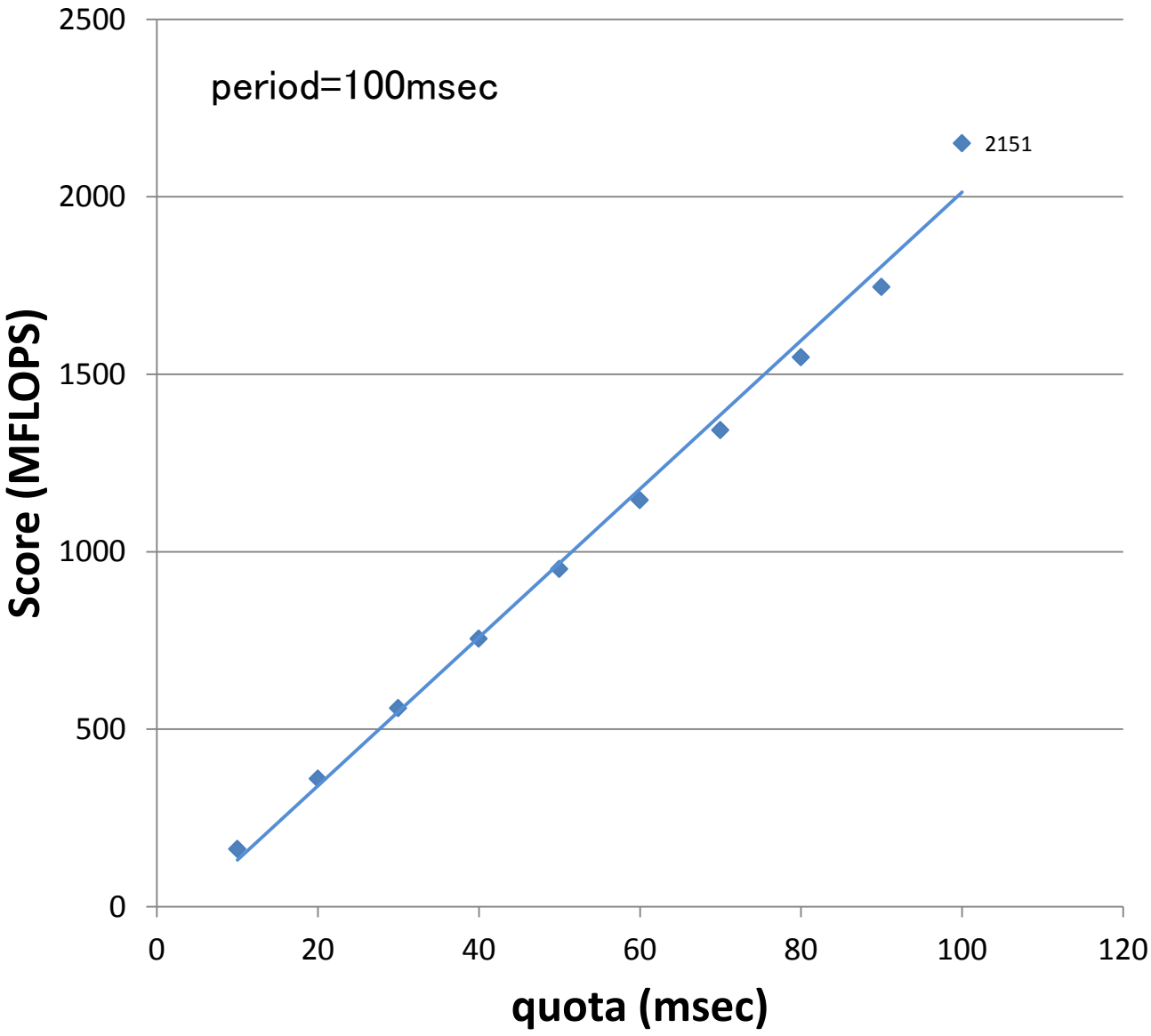


“quota” test

- When changing “quota”, how does the score of benchmark change?
- Procedure
  - Create the CPU cgroup named “benchmark”
  - Specify cpu.cfs_quota_us (cpu.cfs_period_us is left the default)
  - Run the benchmark under “benchmark” cgroup
- Benchmarks
  - Himeno Benchmark
  - Unixbench
  - Hackbench
  - SysBench oltp test
  - Sleep-work-wakeup benchmark

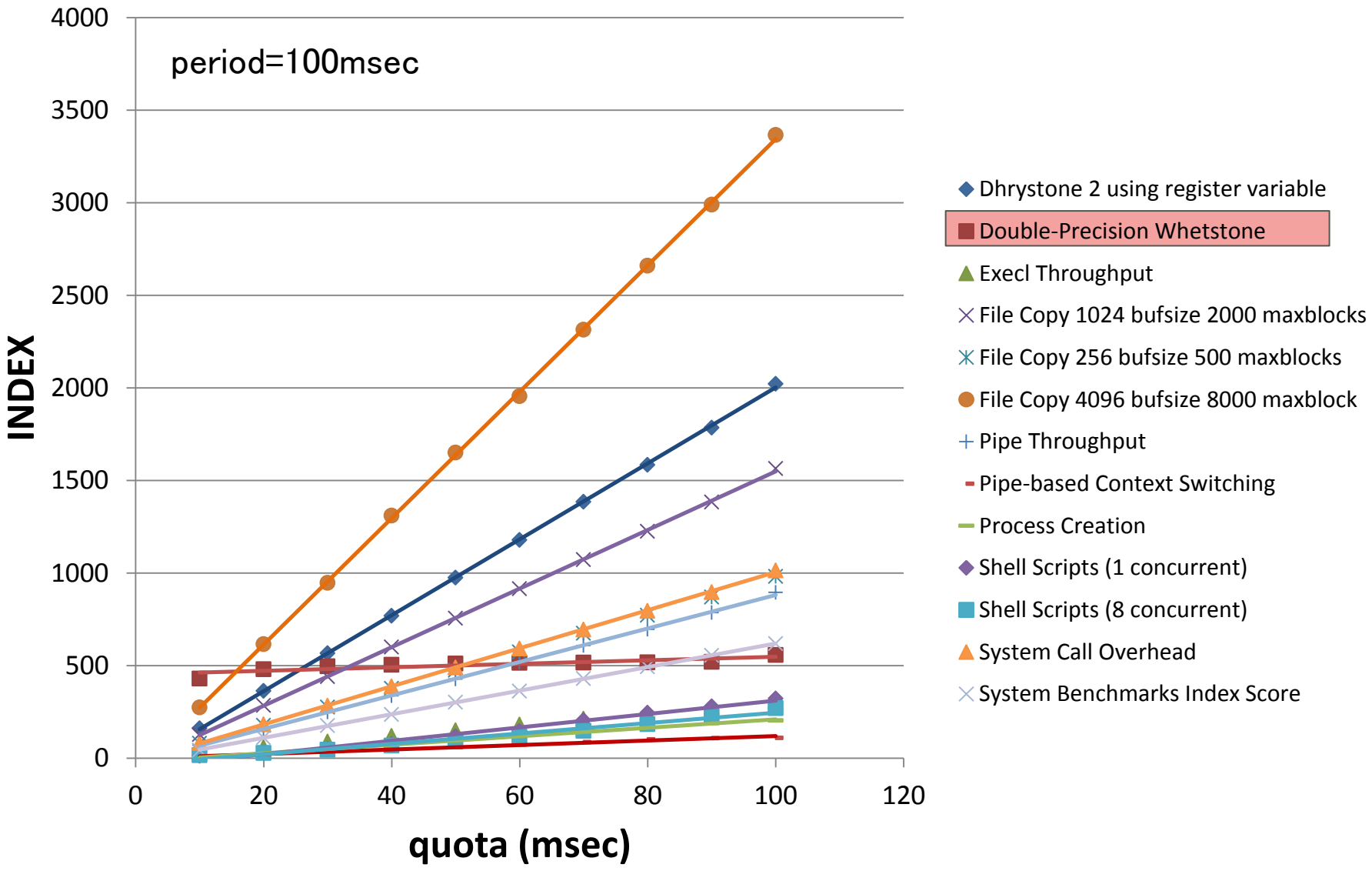


Himeno Benchmark result

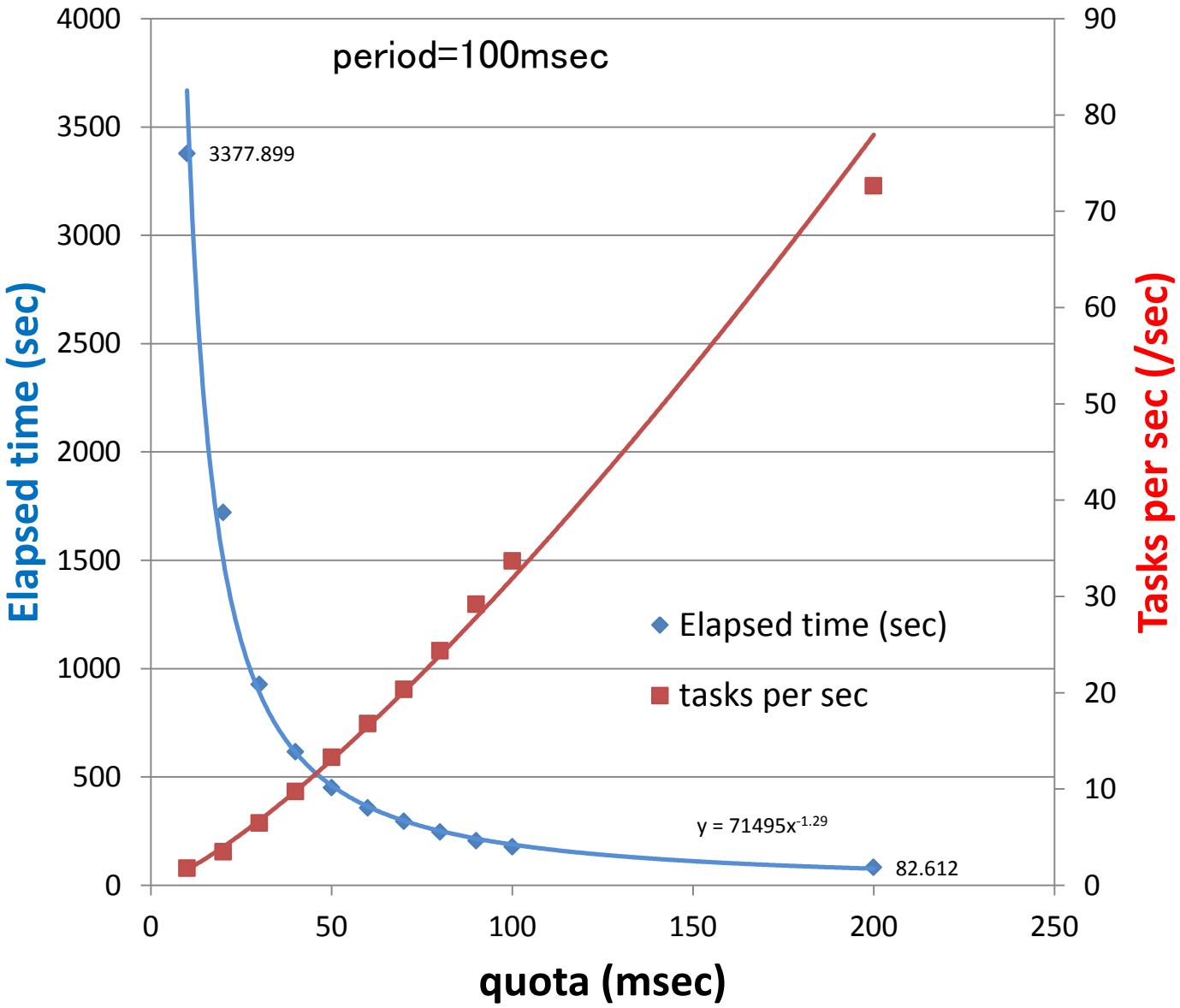


Score when unlimited
2148.3 MFLOPS

Unixbench result

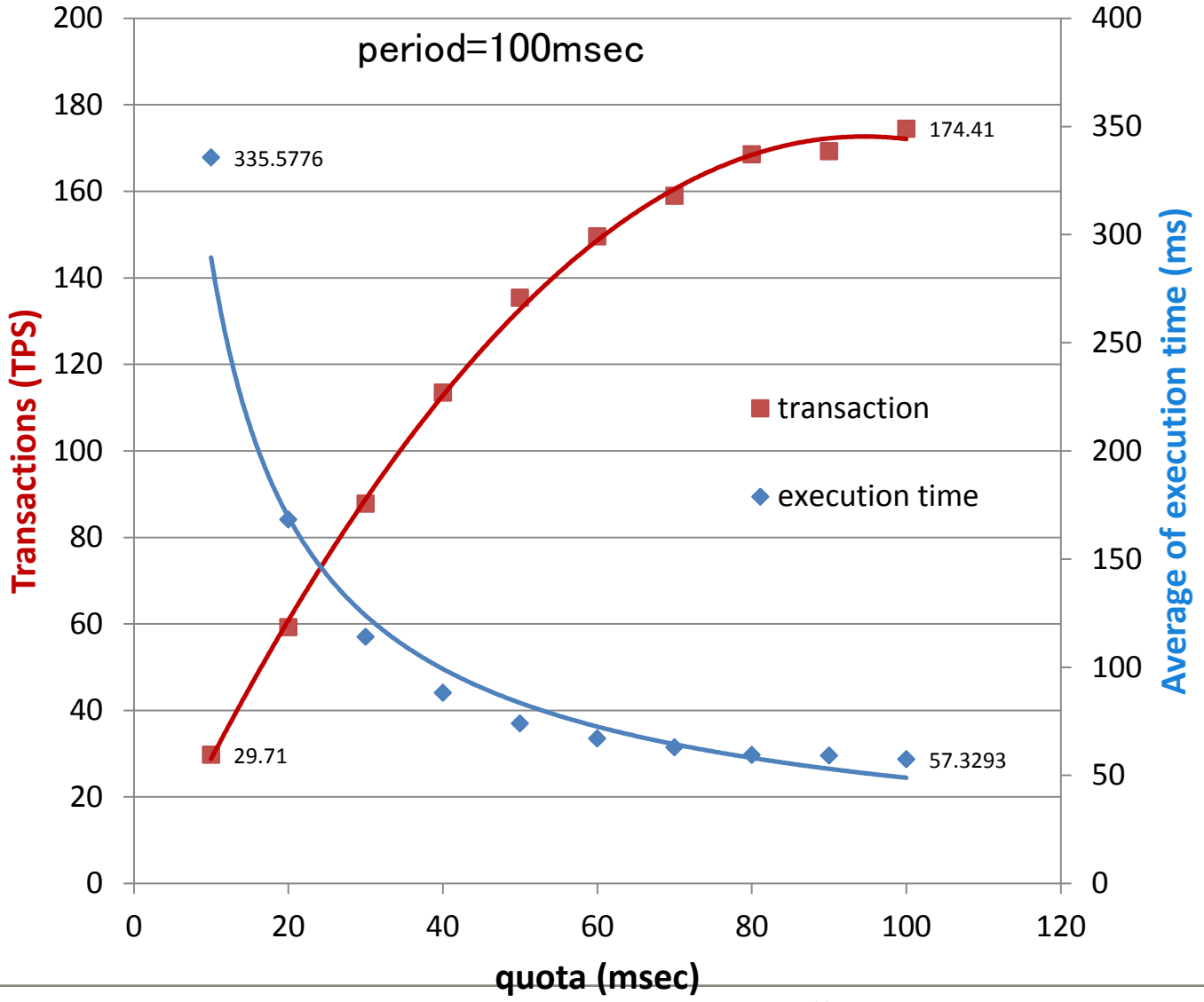


Hackbench result



Score when unlimited
3.578 sec

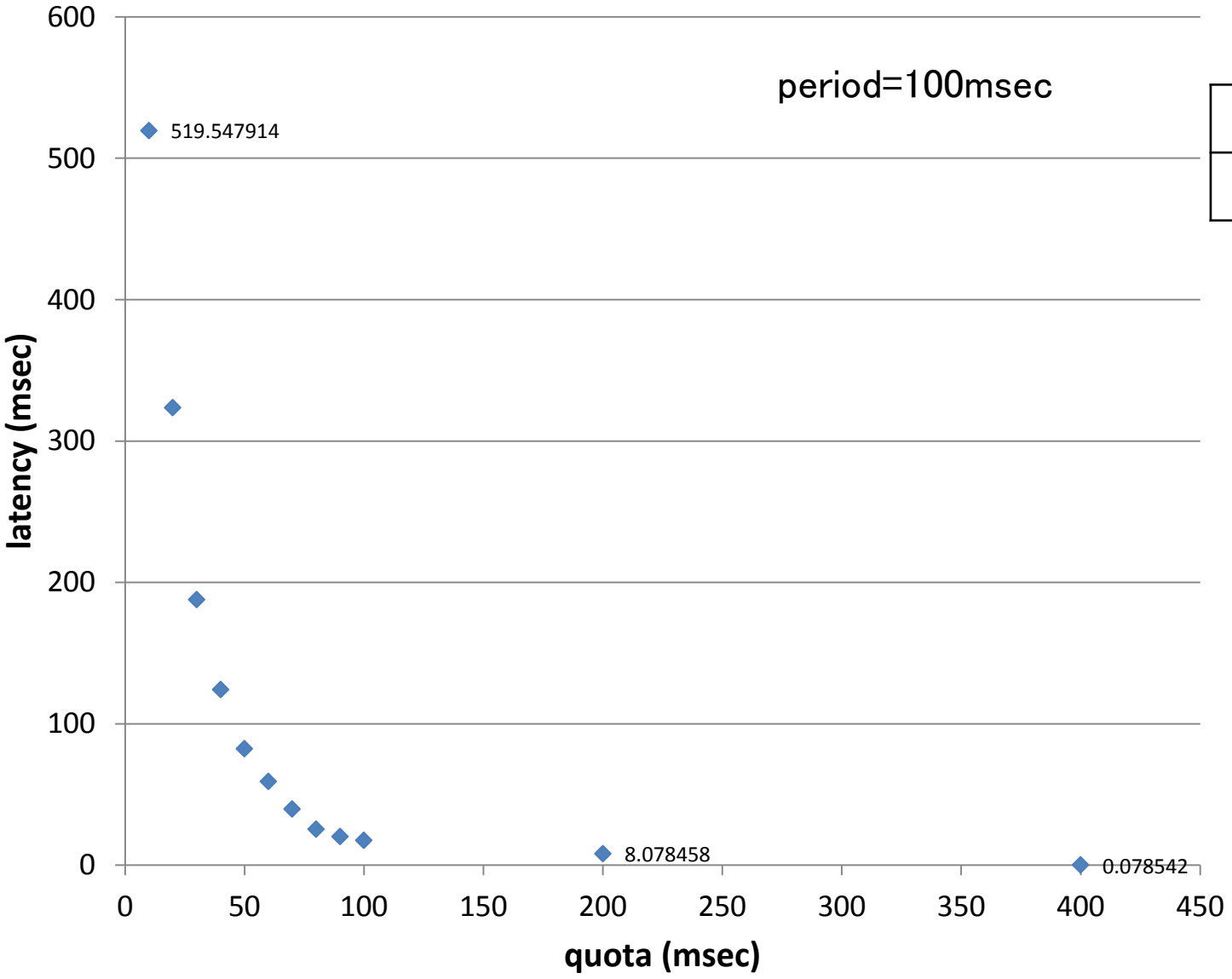
SysBench oltp test result



Score when unlimited
54.61 sec 183.03 TPS

Sleep-work-wakeup benchmark result

FUJITSU



Score when unlimited
0.0697 msec

FUJITSU

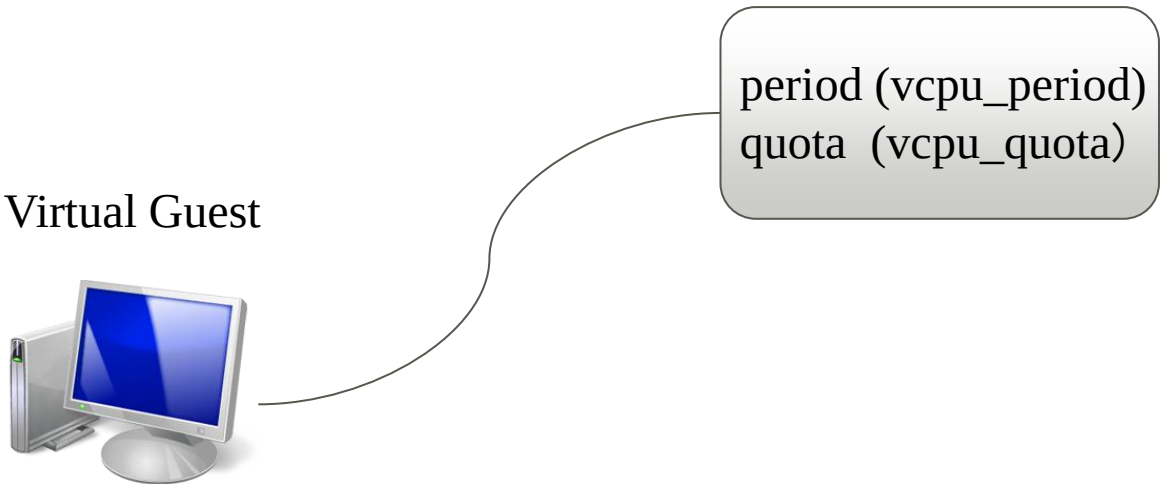
CPU bandwidth control of KVM guests

CPU bandwidth control of KVM guests

FUJITSU

- Virtual CPU bandwidth control of KVM guests can be achieved by using CFS bandwidth control
  - libvirt has already supported this feature

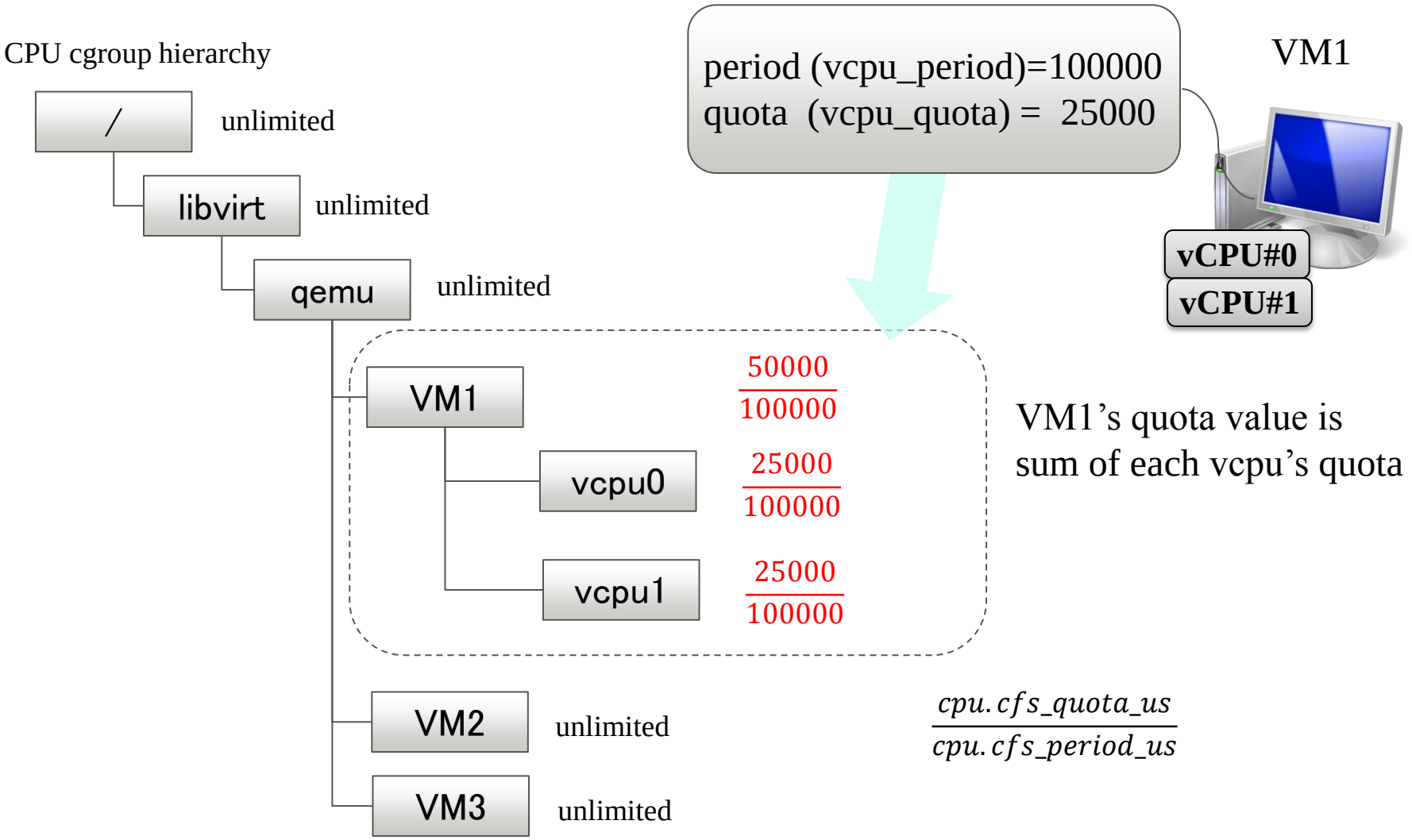
- Specify “vcpu_period” and “vcpu_quota” parameters against each guests
  - libvirt converts them to cpu.cfs_period_us and cpu.cfs_quota_us



CPU Bandwidth control of KVM guests (cont.)

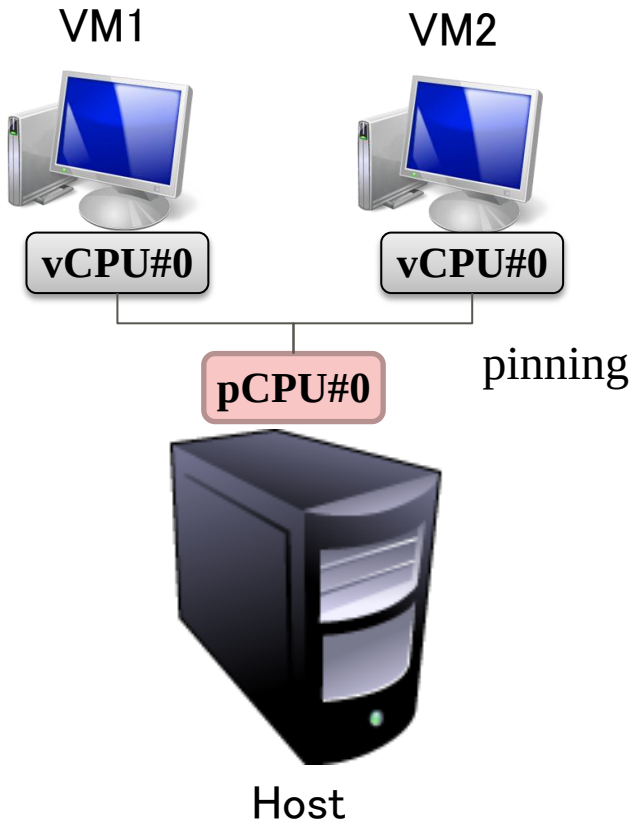
FUJITSU

- vcpu_period, vcpu_quota



Comparison with “share” facility

- Preconditions
  - 2 virtual guests (1 vcpu each)
  - Each vcpu is pinned to the same physical CPU
- Run the super-pi benchmark at VM1 in the following case:
  - A) Without CPU bandwidth control
    i. VM2 is shut-off
    ii. VM2 is running but idle
    iii. VM2 is running and busy
  - B) With CPU bandwidth control
    i. VM2 is shut-off
    ii. VM2 is running but idle
    iii. VM2 is running and busy



Comparison with “share” facility (cont.)

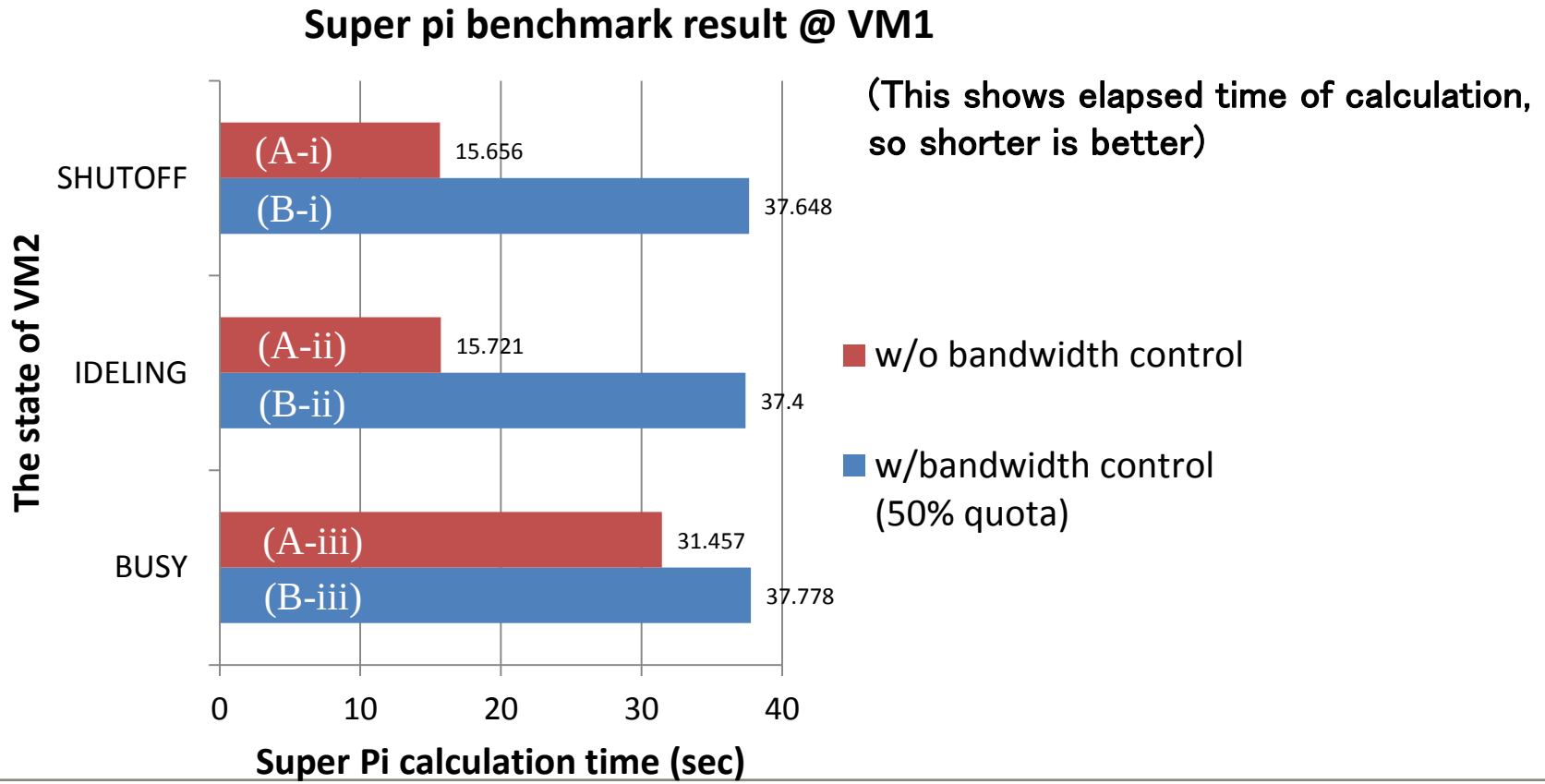
FUJITSU

A) Without CPU bandwidth control on guests <“share”>

- CPU time is allocated proportionally according to the “share” value

B) With CPU bandwidth control

- CPU time is limited according to each quota value



“quota” test on KVM guest

FUJITSU

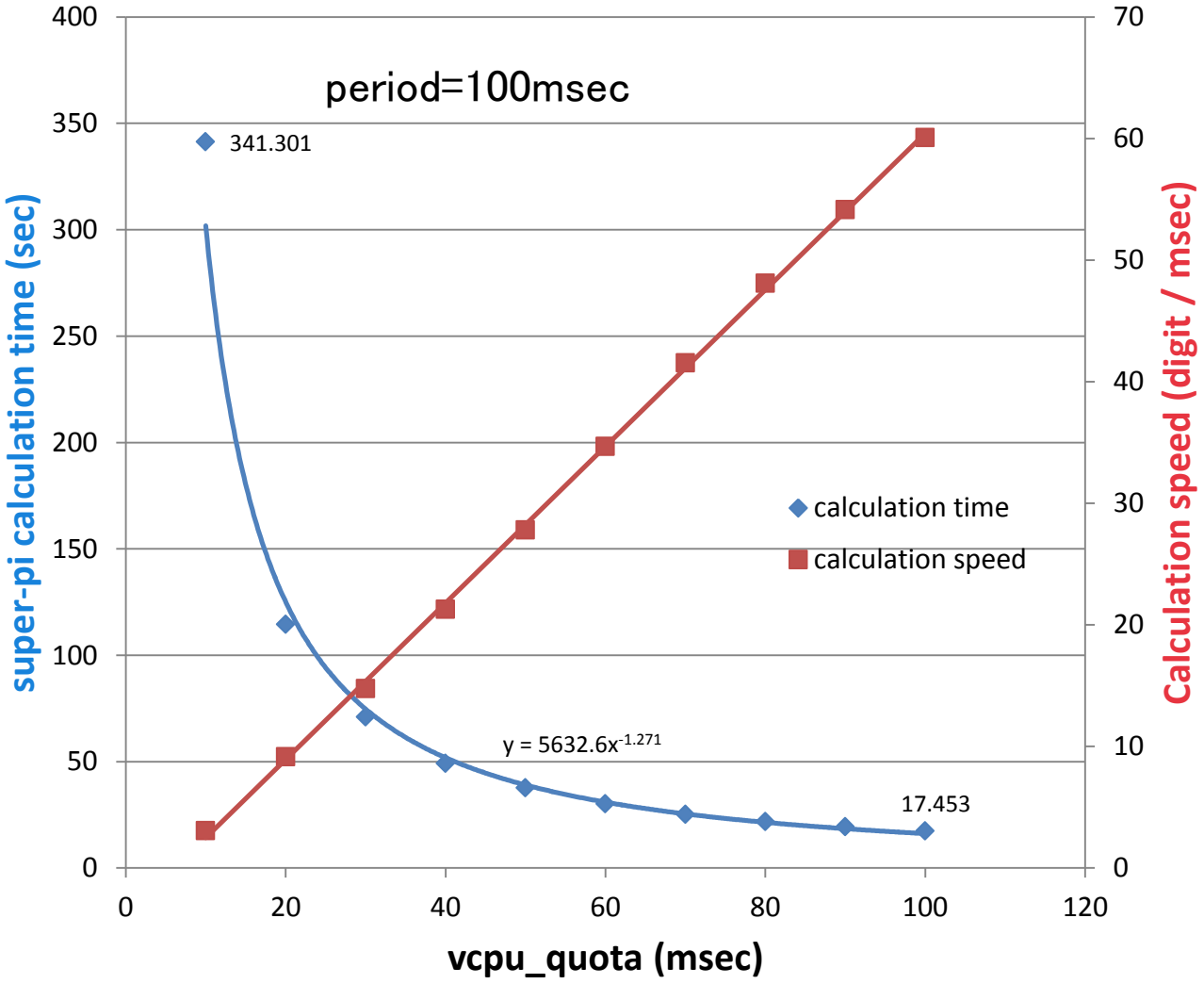
- Preconditions
  - KVM guest

CPU	1 vcpu
Memory	2GB
OS	RHEL6.1 x86_64

- Procedure
  - Specify vcpu_quota (vcpu_quota is left the default) of guests
  - Run the benchmark on the guest

- Benchmark
  - Super Pi benchmark (1 million digits)
  - SysBench oltp test
  - Sleep-work-wakeup benchmark

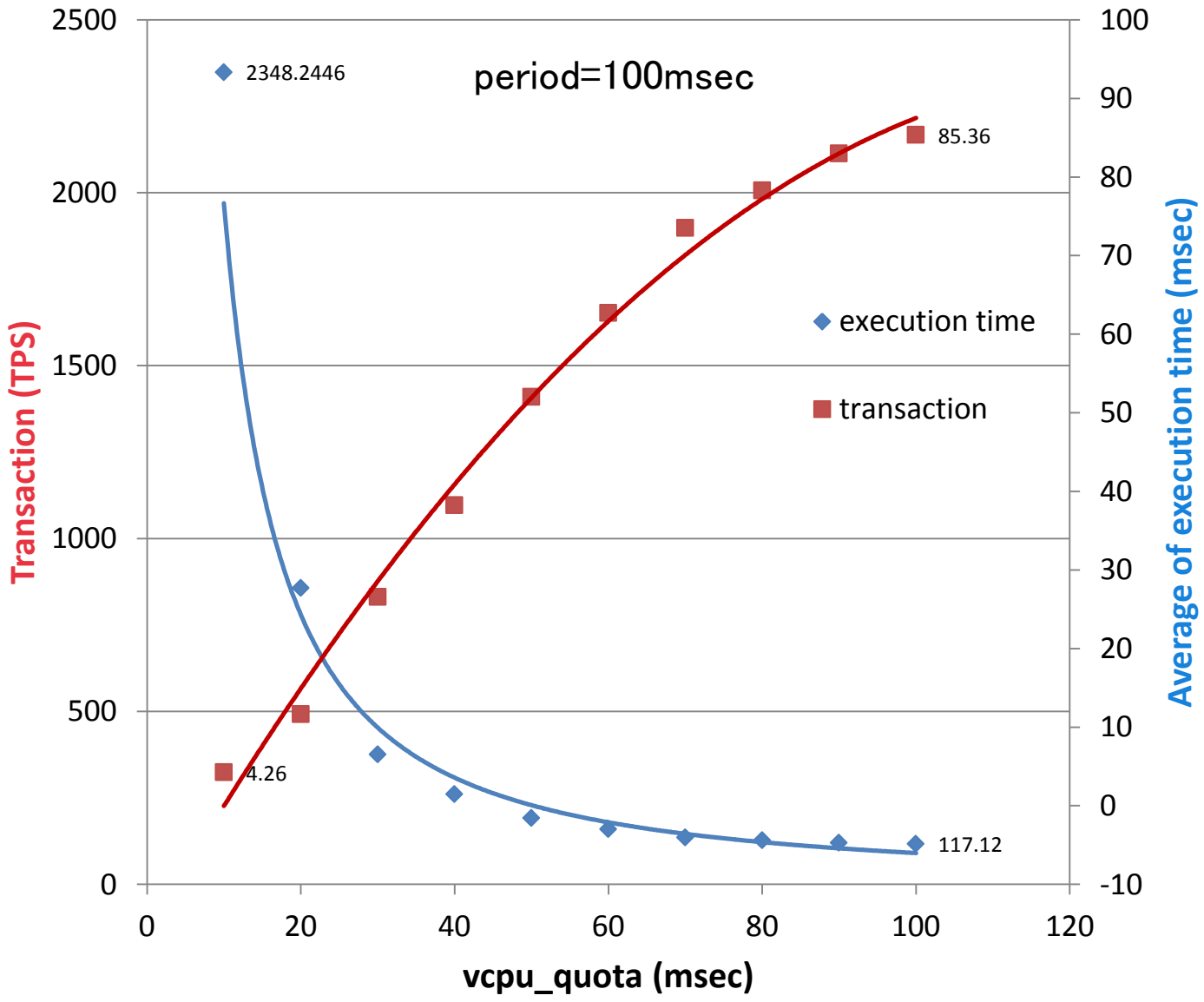
Super Pi benchmark result



Score when unlimited
15.722 sec

SysBench oltp test result

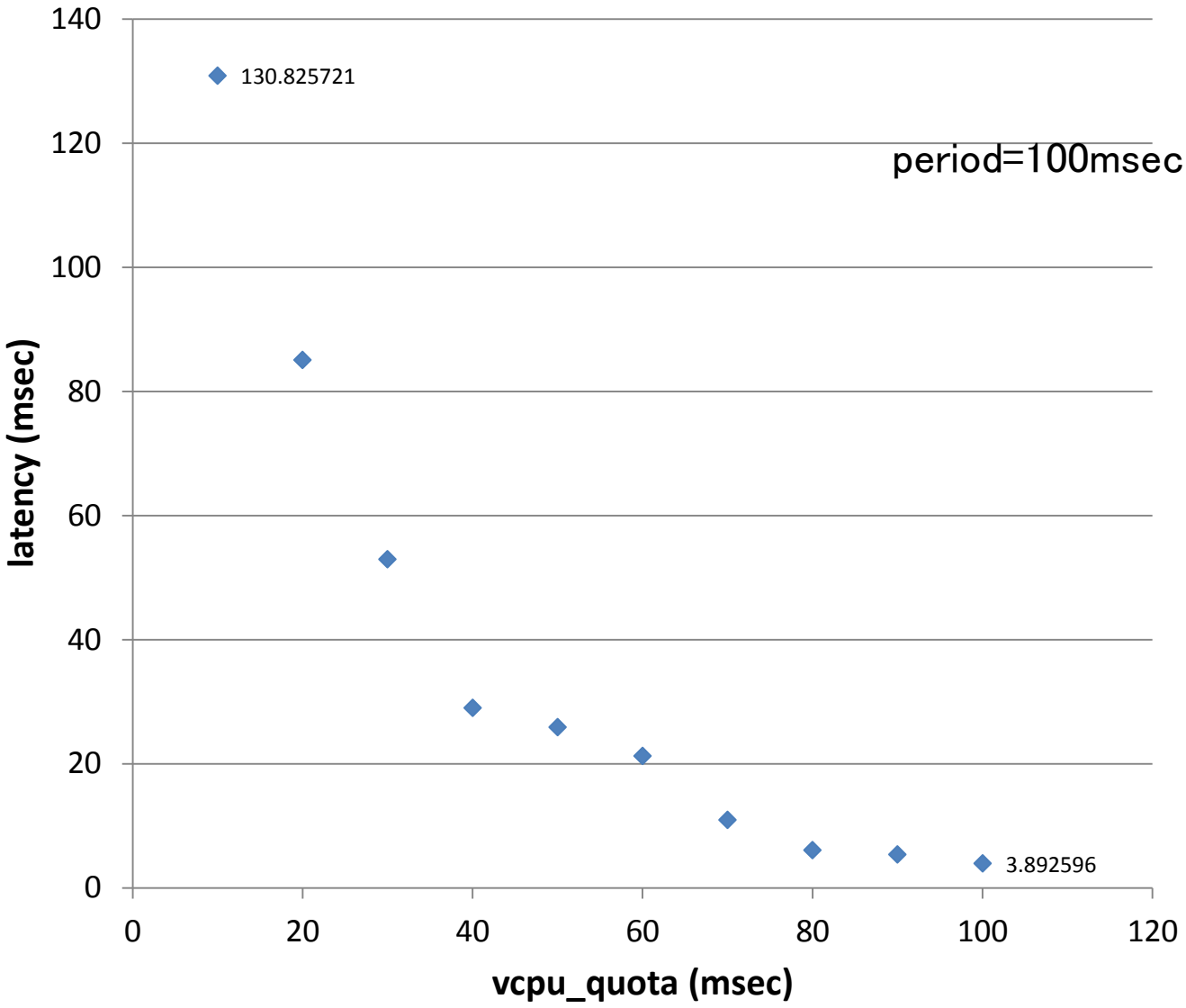
FUJITSU



Score when unlimited
94.57 msec 105.71 TPS

Sleep-work-wakeup benchmark result

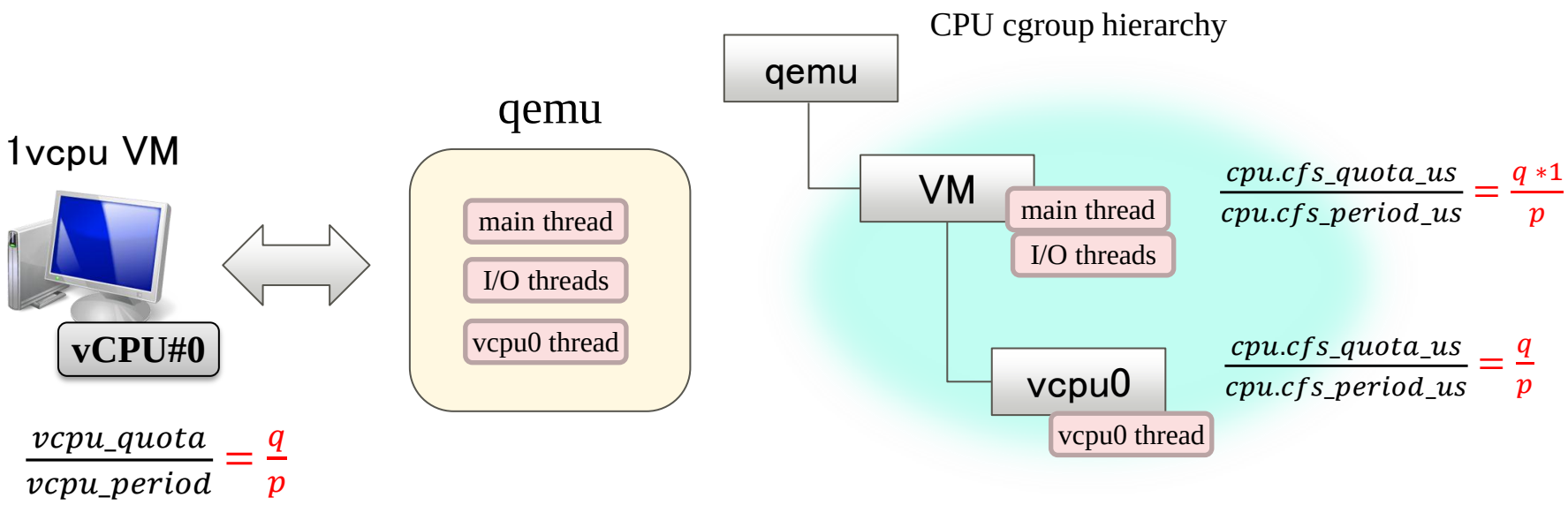
FUJITSU



Score when unlimited
2.611 sec

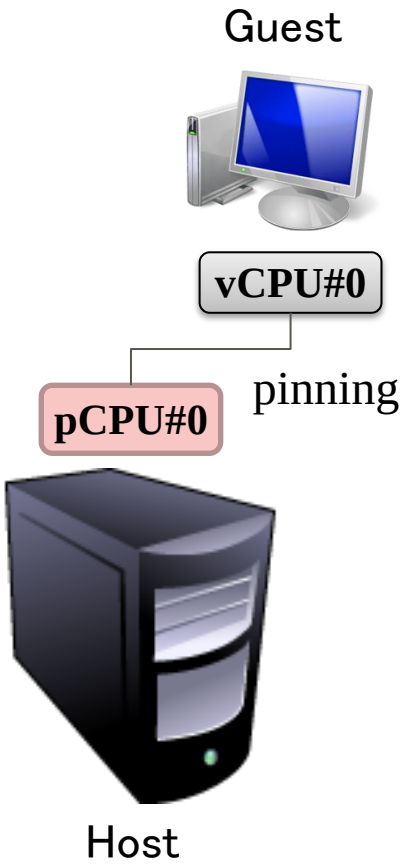
Consideration of vcpu_quota

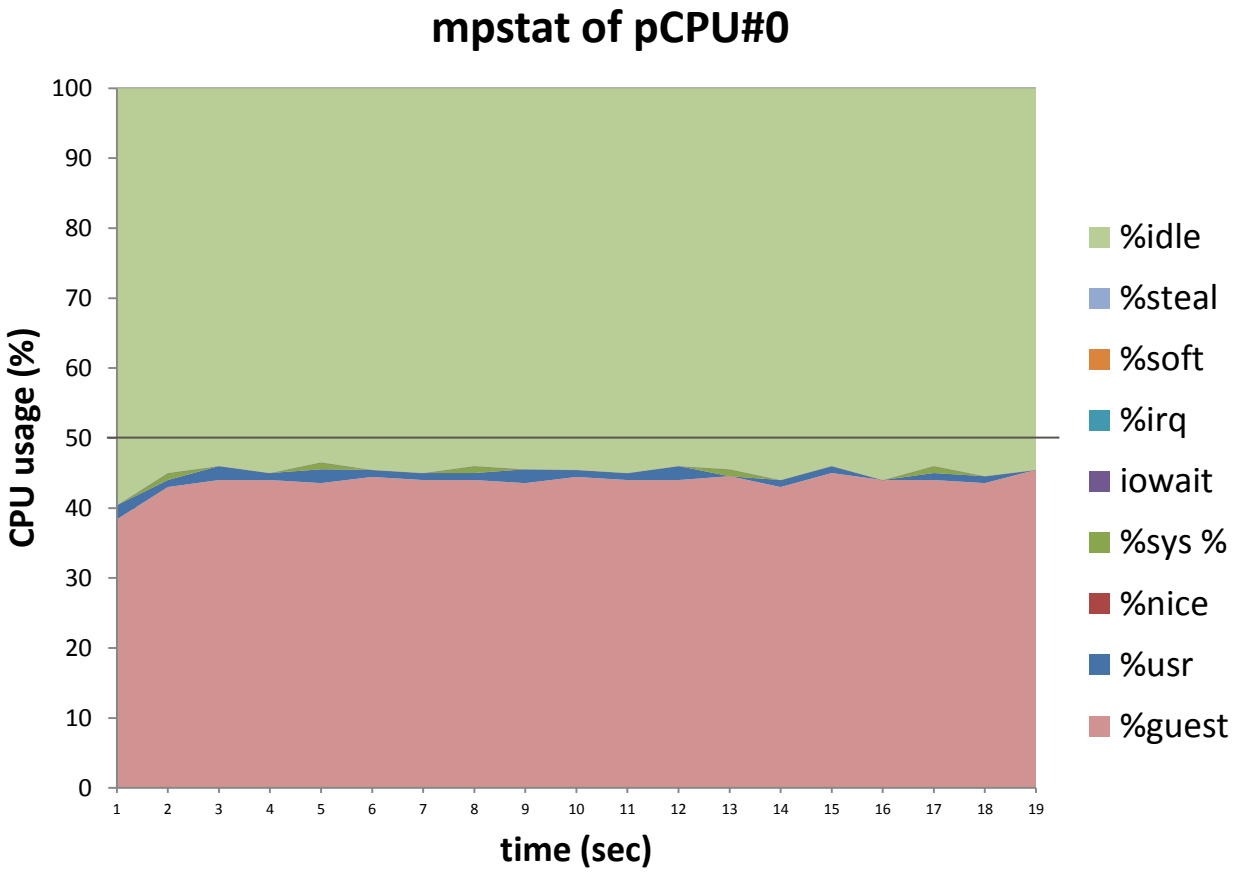
- When vcpu_quota is specified, not only vcpu but virtual machine is throttled
- Even if the CPU time of vcpu is less than quota value, it may be throttled
  - In case of heavy I/O



Consideration of vcpu_quota (cont.)

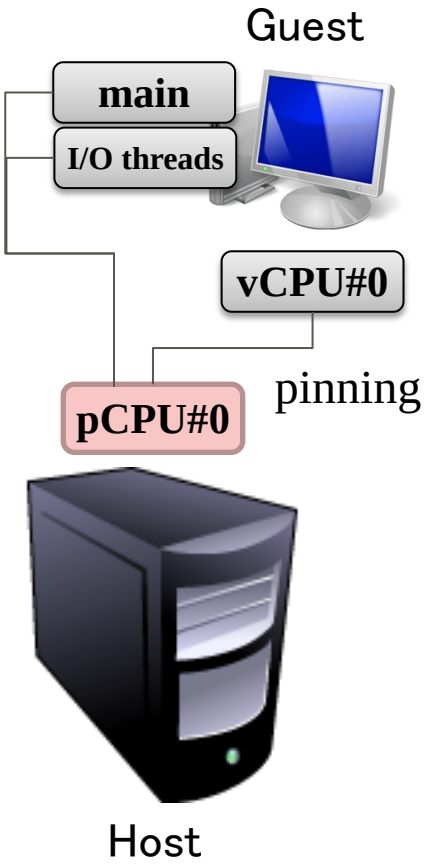
- Even if vcpu_quota is set as 50% of vcpu_period and vcpu0 is placed at full load on guest, CPU usage rate of vcpu0 doesn't reach 50%

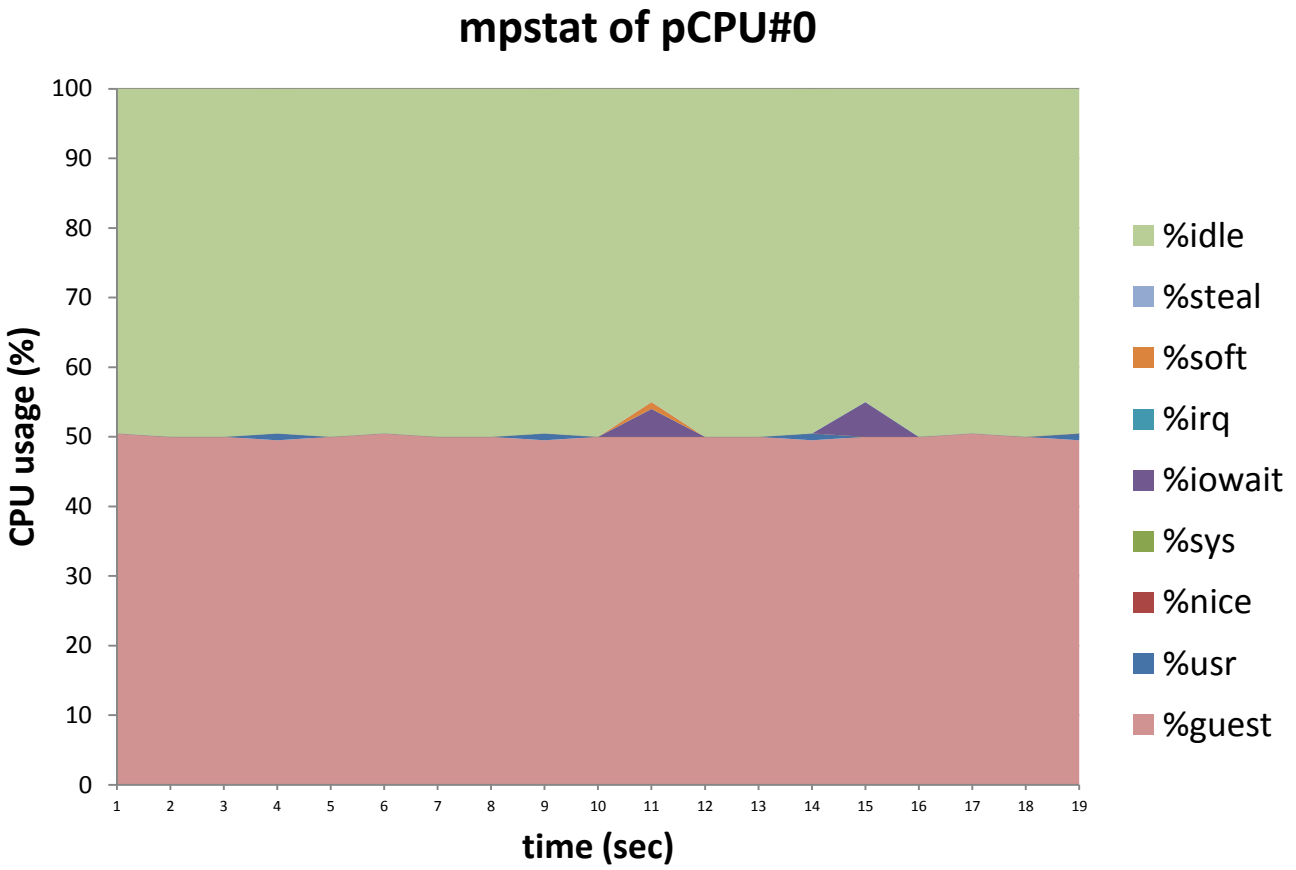




Consideration of vcpu_quota (cont.)

- CPU usage rate of qemu and qemu I/O threads is combined, the sum total will be 50%





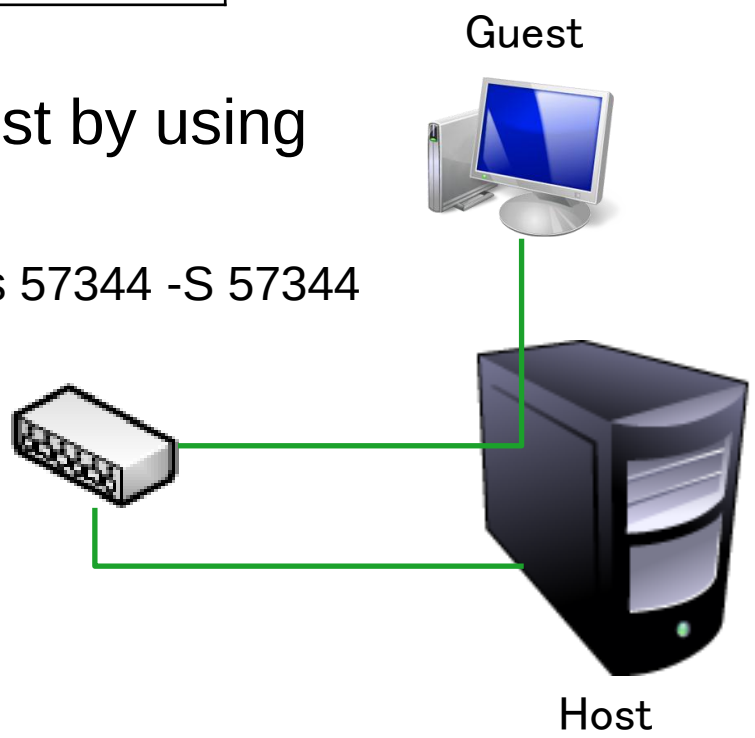
Impact investigation on network I/O

FUJITSU

- Preconditions
  - KVM guest

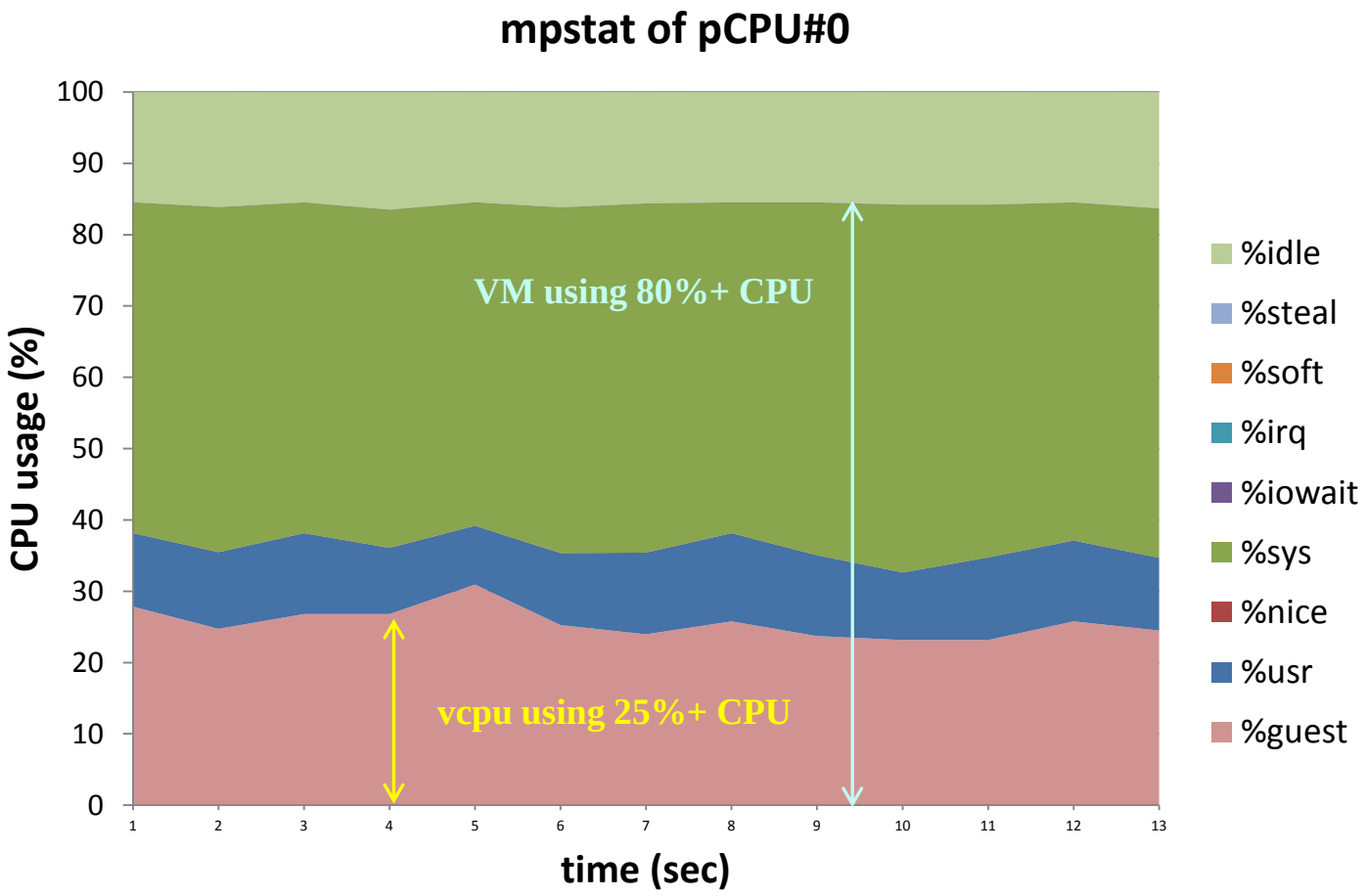
CPU	1vcpu (pinned to pCPU#0)
Memory	2GB
OS	RHEL 6.1 x86_64
NIC	vhost_net, using macvtap

- Place a network load on KVM guest by using netperf between guest and host
  - # netperf –t TCP_STREAM -- -m 32768 -s 57344 -S 57344



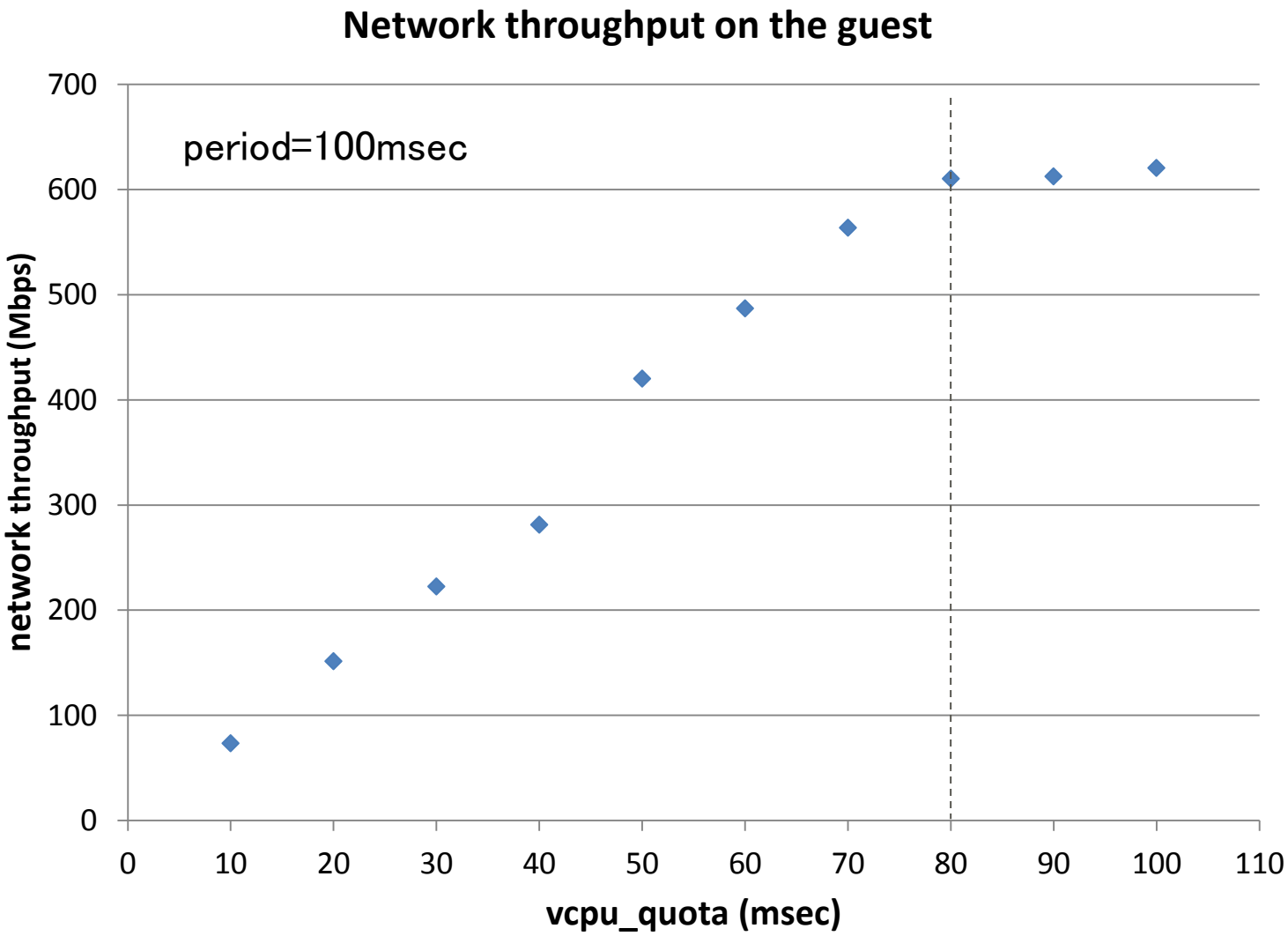
Impact investigation on network I/O (cont.)

- CPU usage of pCPU#0 during netperf (w/o CPU bandwidth control of the guest)



Impact investigation on network I/O (cont.)

FUJITSU



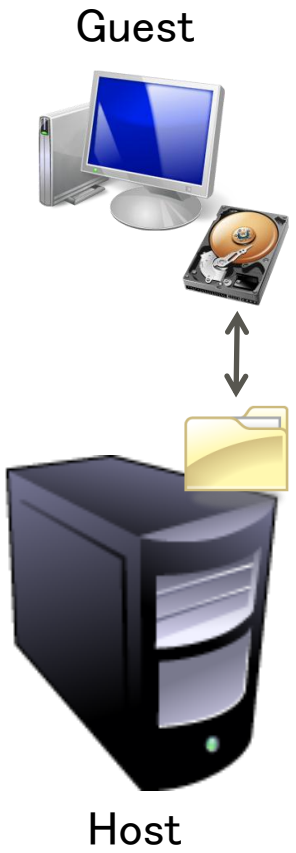
Impact investigation on disk I/O

FUJITSU

- Preconditions
  - KVM guest

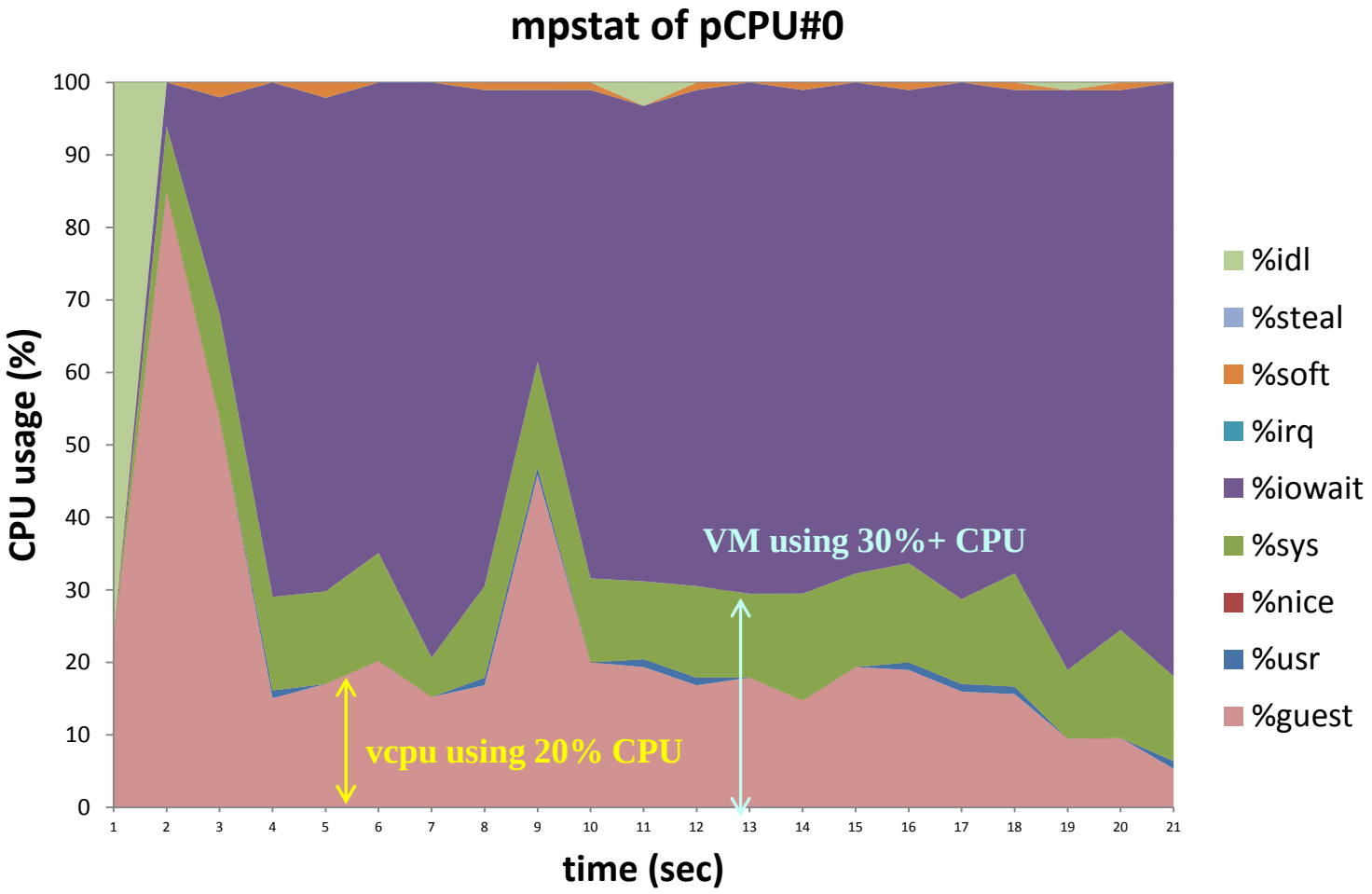
CPU	1vcpu (pinned to pCPU#0)
Memory	4GB
OS	RHEL 6.1 x86_64
HDD	File based virtual HDD, 16GB

- Place a disk load on KVM guest by using dd command
  - # dd if=/dev/zero of=testfile bs=1024M count=2

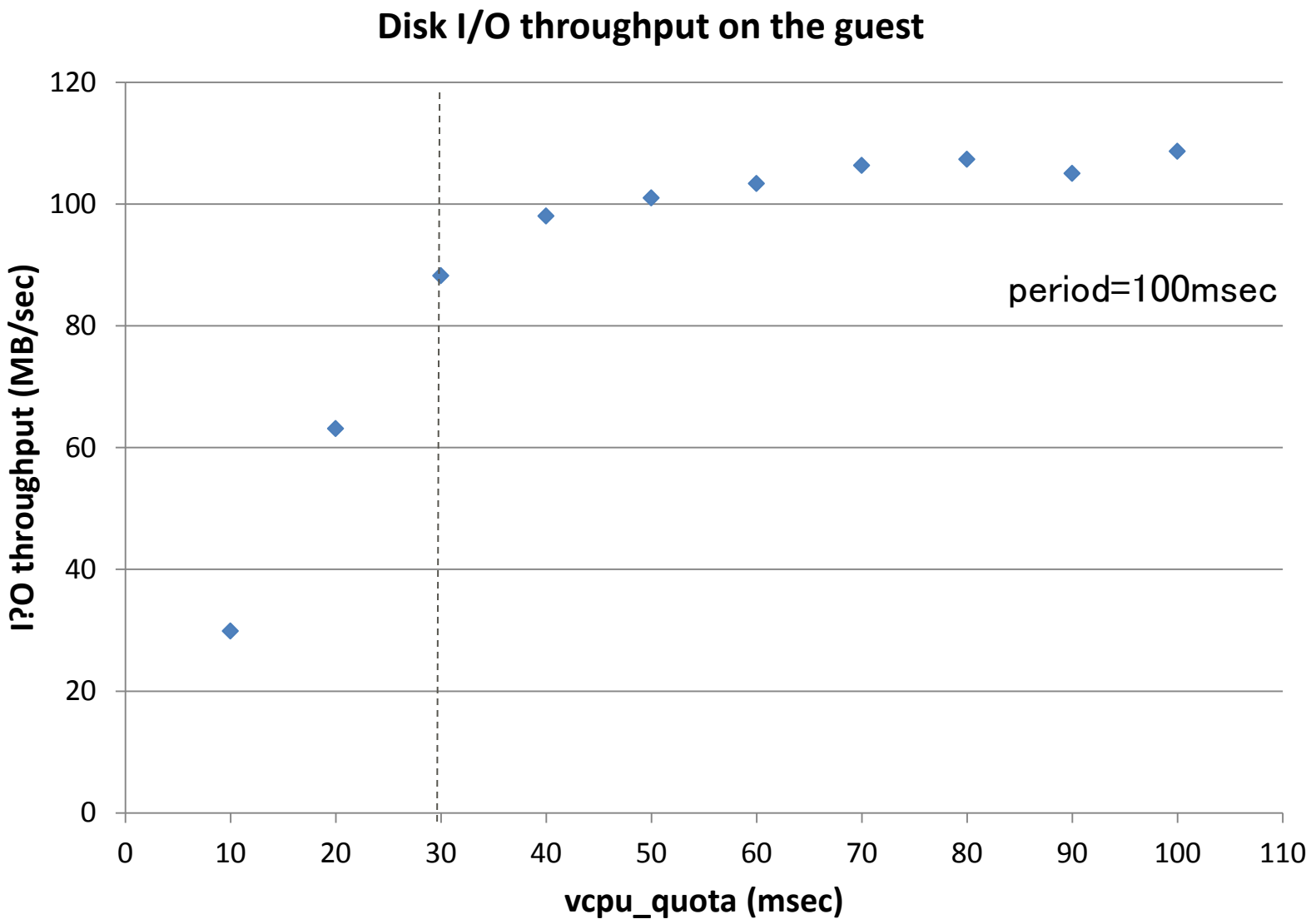


Impact investigation on disk I/O (cont.)

- CPU usage of pCPU#0 during dd command (w/o CPU bandwidth control of the guest)



Impact investigation on disk I/O (cont.)



Comparison with other hypervisors

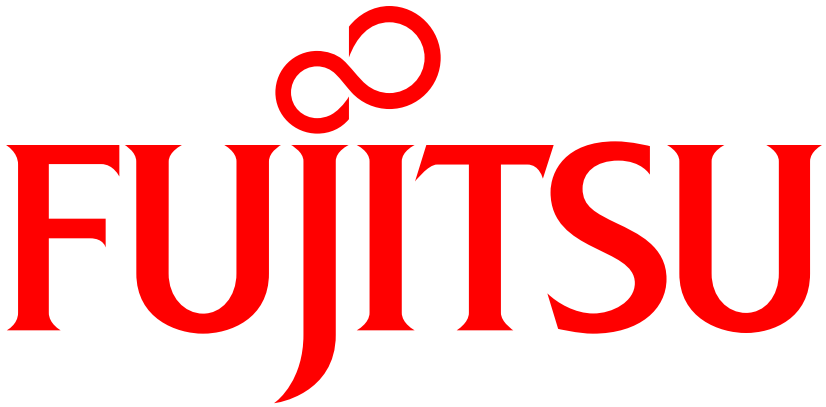
FUJITSU

- vCPU bandwidth control of other hypervisors (Xen, VMWare)
  - [virtual CPU usage] <= [limit value]
    - Limit virtual CPU usage
    - Do NOT limit hypervisor CPU usage
- vCPU bandwidth control of KVM
  - [virtual CPU usage] + [hypervisor CPU usage] <= [limit value]
    - Limit total of virtual CPU usage and hypervisor CPU usage
- Both methods have Pros. and Cons.
  - We Fujitsu are now enhancing libvirt to support “virtual CPU usage only” limiting on KVM

Summary

FUJITSU

- Outline of CPU cgroup
  - CFS bandwidth control provides the upper limit of CPU resource
- Evaluation of CFS bandwidth control
  - The effect of bandwidth control depends on workload:
    - tasks which use CPU for full time are affected directly, while I/O bounded tasks are a little insensitive
- CPU bandwidth control of KVM guests
  - useful when building KVM based cloud system
  - necessary to take impact on guests' I/Os into account



shaping tomorrow with you