FUJITSU

shaping tomorrow with you

Improvements on Kdump Scalability Issues for Terabyte-Scale Memory System

HATAYAMA, Daisuke
FUJITSU LIMITED.

May 20, 2014
LinuxCon Japan 2014

Agenda

FUJITSU

- Background
- Review kdump structure
- 3 improvements on the scalability issues
  - Use fast compression format
  - Copyless processing with mmap()
  - Break a 1-CPU restriction of kdump capture kernel
- Ongoing work
  - kexec-tools multiple CPUs support

Agenda

FUJITSU

- Background
- Review kdump structure
- 3 improvements on the scalability issues
  - Use fast compression format
  - Copyless processing with mmap()
  - Break a 1-CPU restriction of kdump capture kernel
- Ongoing work
  - kexec-tools multiple CPUs support

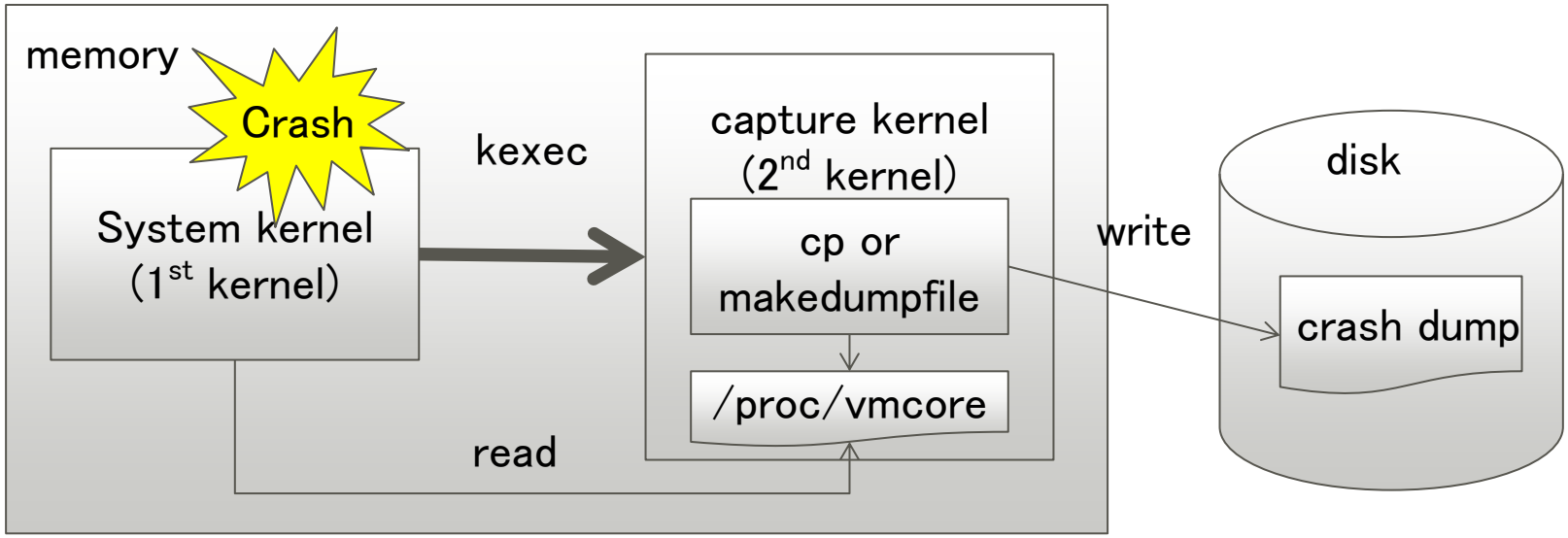
Crash dumping

FUJITSU

- Save system memory in local or remote disk at system crash to record the situation.
  - Engineers can see what happens.
  - Essential for mission critical enterprise use

Kdump

- Linux standard crash dumping feature
  - Since v2.6.13
- Structure of kdump
  - Crashed system kernel boots up capture kernel that saves image of the system kernel via /proc/vmcore



Current Kdump scalability issues

FUJITSU

- Issue
  - Kdump has been too slow to capture terabyte-scale memory system
  - Fujitsu PRIMQUEST 2800E can have 12TB memory
    - Full dump needs 35 hours
    - Even partial dump (only kernel memory) could need 2 hours
      - amount of kernel memory depends on runtime situation

Need optimization in order to complete huge crash dump processing within 1 hour

Terabyte-scale memory system

FUJITSU

- Vendor catalog

Vendor	Model	Memory Size
Fujitsu	PRIMEQUEST 2800E	12TiB
HP	HP ProLiant DL980 G7	4TiB
IBM	System x3950 X6	12.8TiB
NEC	NX7700x	4TiB
SGI	SGI UV 2000	64TiB

- Use cases
  - In-memory database
  - VM consolidation
  - HPC

Agenda

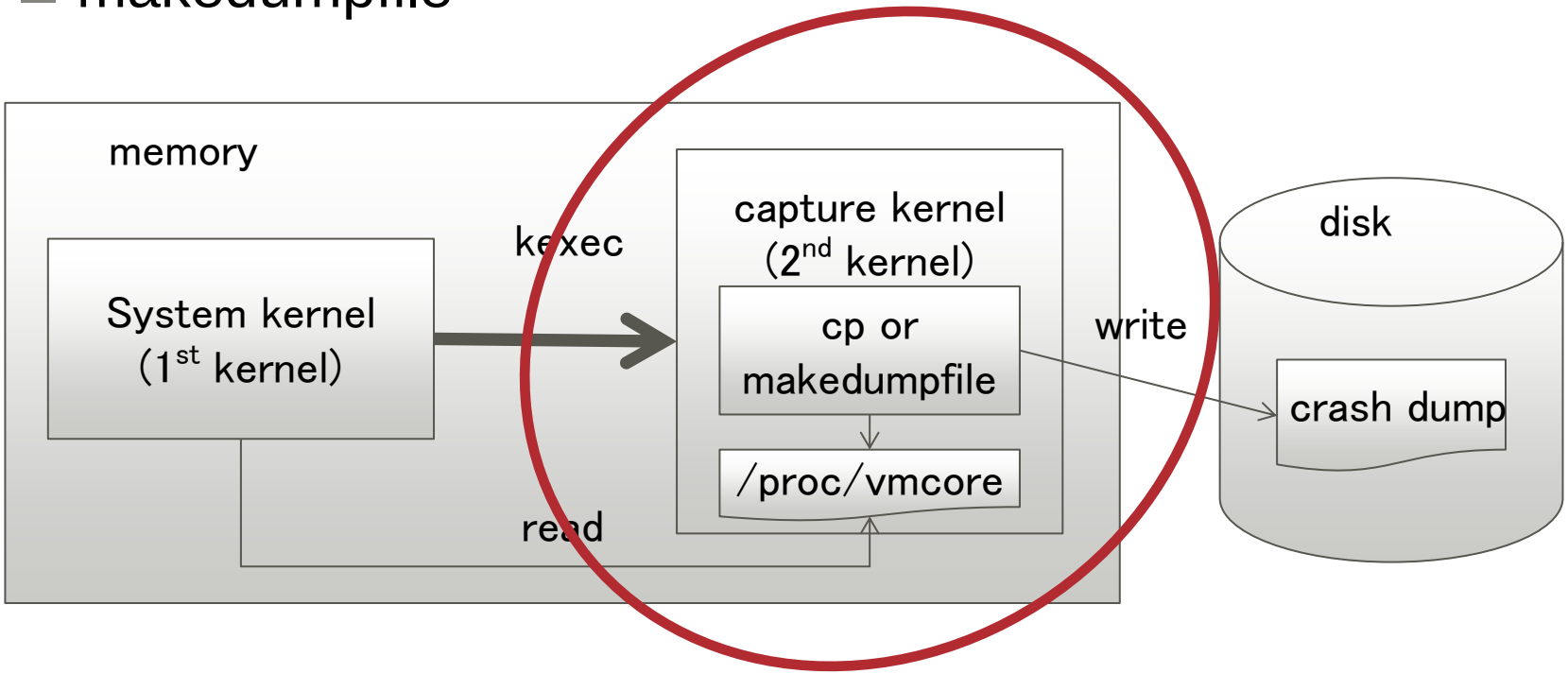
FUJITSU

- Background
- Review kdump structure
- 3 improvements on the scalability issues
  - Use fast compression format
  - Copyless processing with mmap()
  - Break a 1-CPU restriction of kdump capture kernel
- Ongoing work
  - kexec-tools multiple CPUs support

Kdump components for crash dumping

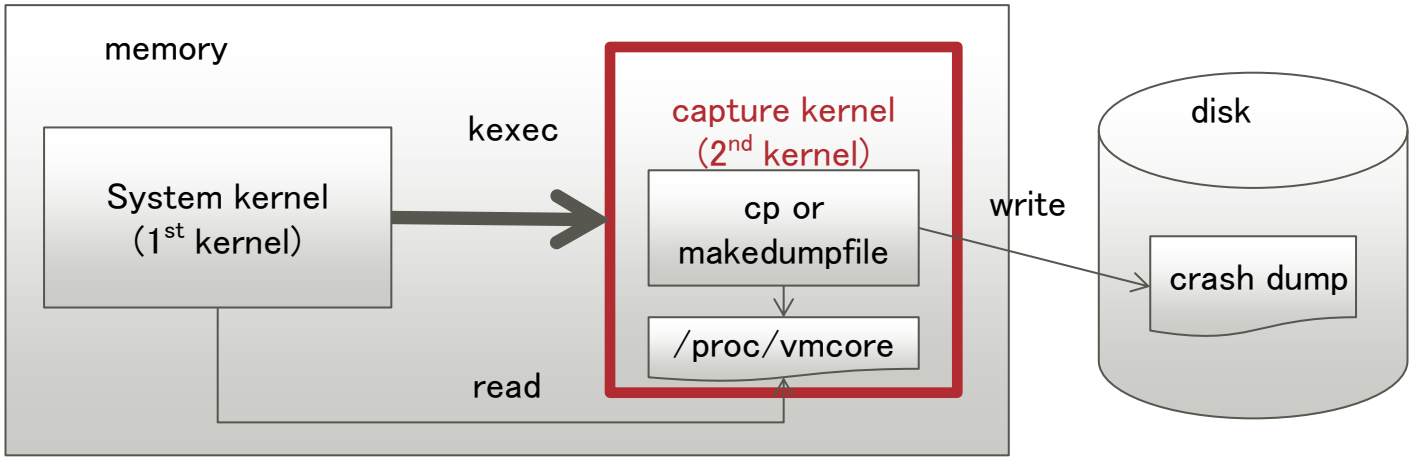
FUJITSU

- 3 components related to crash dumping
  - Capture kernel
  - /proc/vmcore
  - makedumpfile



Capture kernel

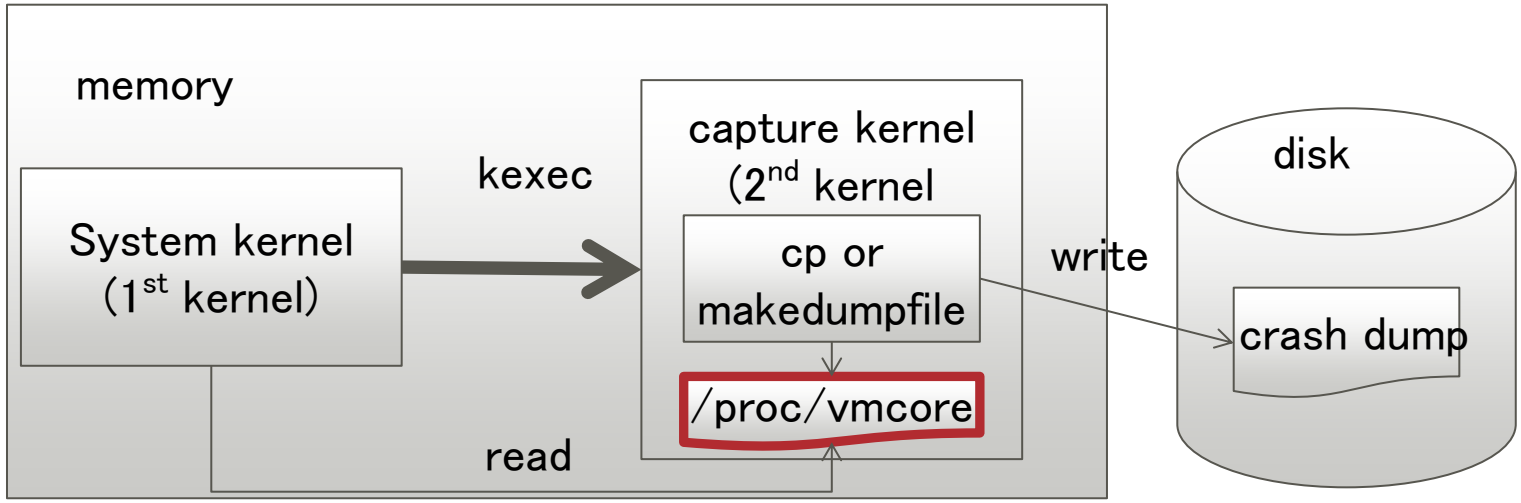
- Running on the memory reserved at system kernel
  - crashkernel=<memory size>
    - 128MiB ~
- Booted from system kernel at crash



/proc/vmcore

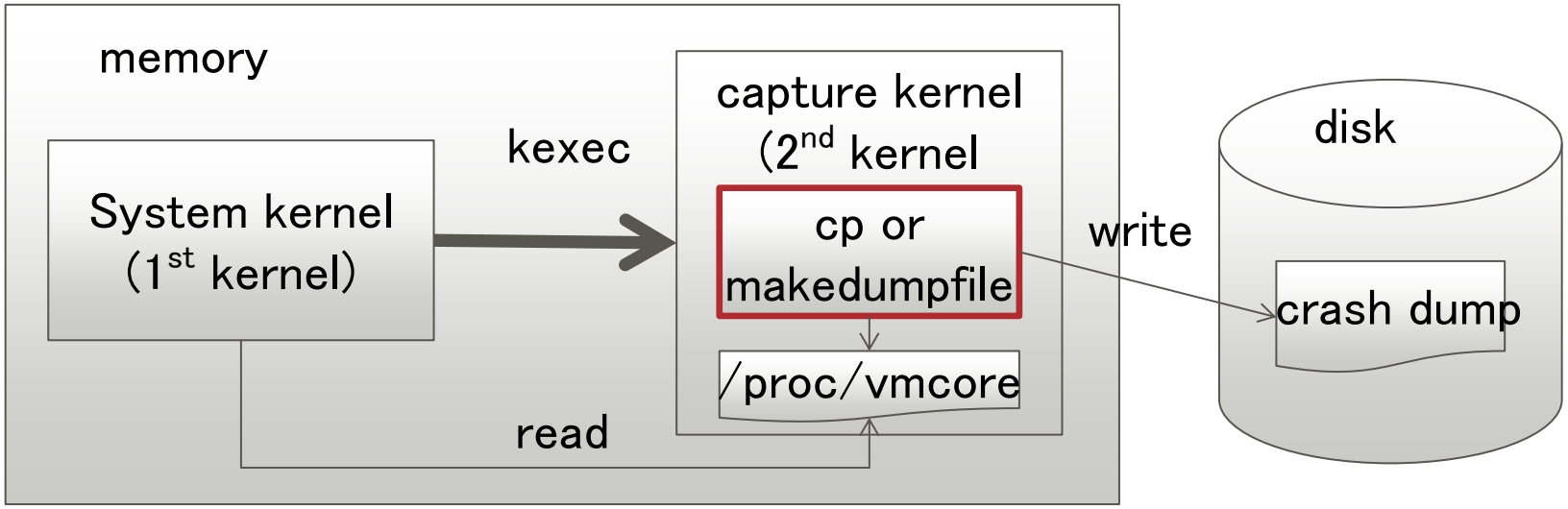
- File interface to access system kernel
  - Exported as ELF
  - To copy vmcore, for example:

```
$ cp /proc/vmcore /var/crash/vmcore
```



makedumpfile

- User-land tool to copy vmcore
  - Compression per 4 KB blocks
  - Dump filtering
    - excludes specified type of memory from vmcore

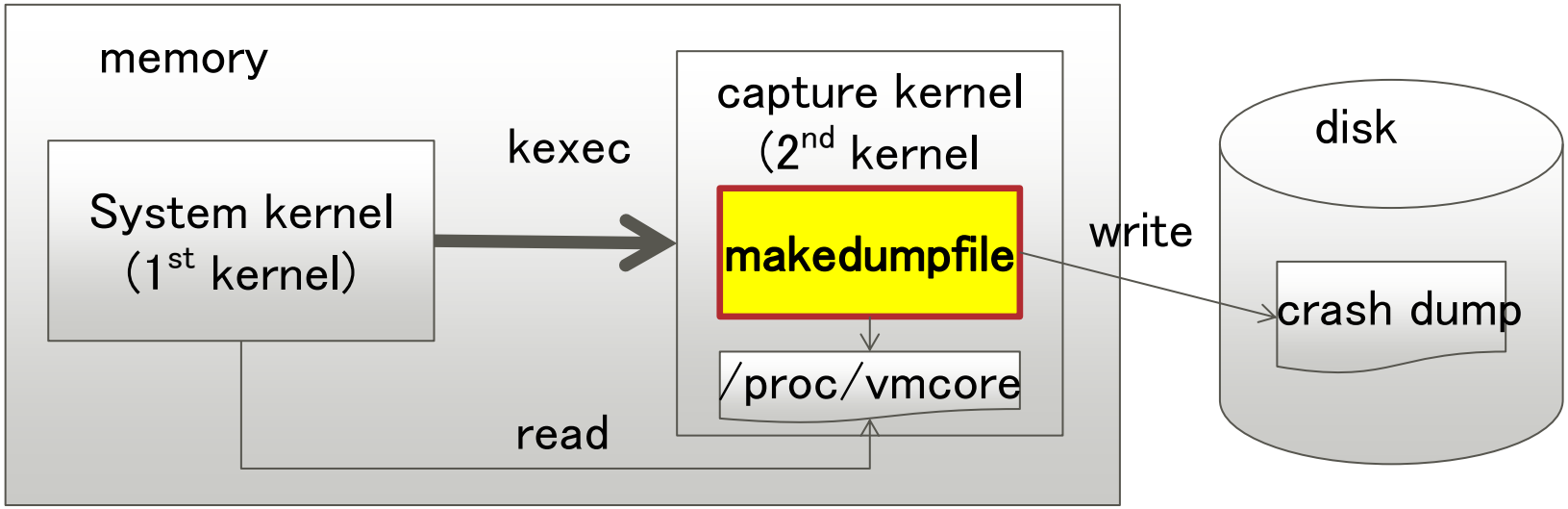


Agenda

FUJITSU

- Background
- Review kdump structure
- 3 improvements on the scalability issues
  - Use fast compression format
  - Copyless processing with mmap()
  - Break a 1-CPU restriction of kdump capture kernel
- Ongoing work
  - kexec-tools multiple CPUs support

Compression



zlib is slow

- Kdump supports zlib.
- zlib is a format used by gzip command
- makedumpfile uses zlib by -c option

```
$ makedumpfile -c /proc/vmcore /mnt/vmcore
```

  - compression per 4KiB blocks
- The problem is that zlib is too slow for crash dump
  - 20 ~ 30 MB/sec
  - 14.6 hours/TB

Fast compression support

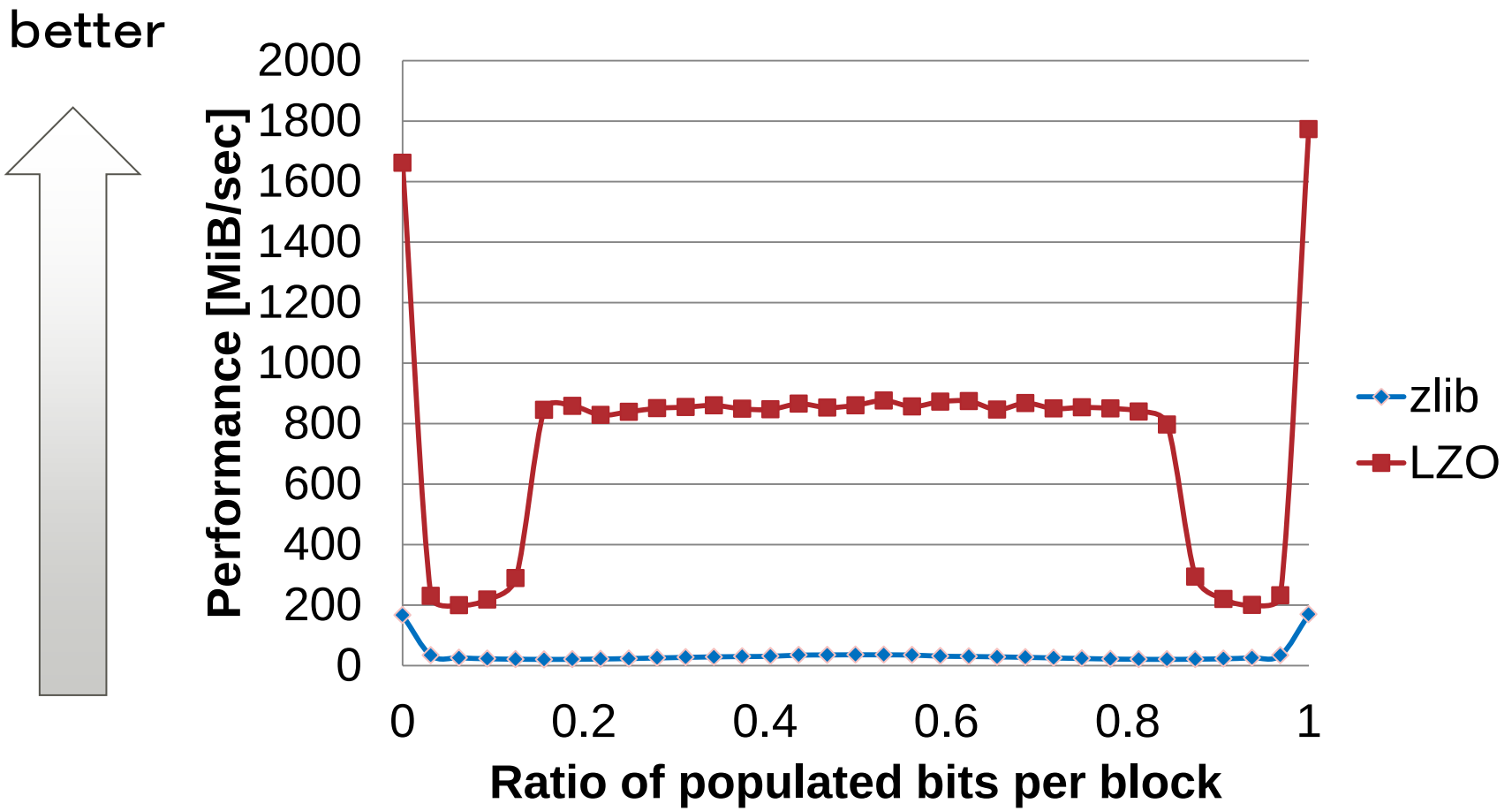
FUJITSU

- LZO
  - Fast compression
    - Almost 800 MB/sec
    - 21.8 min/TB
    - Trade-off
      - compression ratio is slightly worse than zlib
- Supported by makedumpfile since v1.4.4
  - specify -l option

```
$ makedumpfile -l /proc/vmcore vmcore
```

Compression speed w/ dirtiness of data.

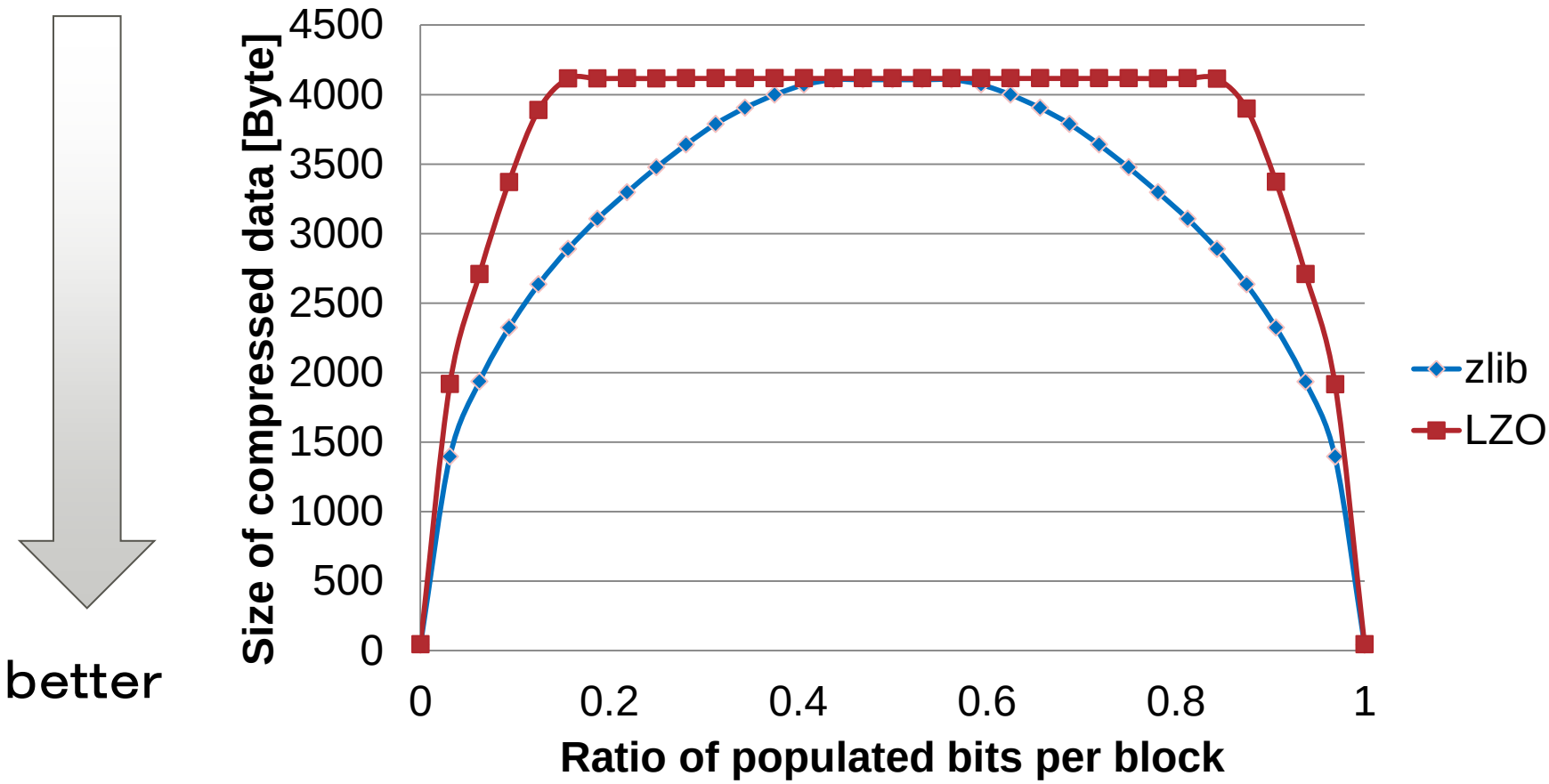
FUJITSU



LZO is better !

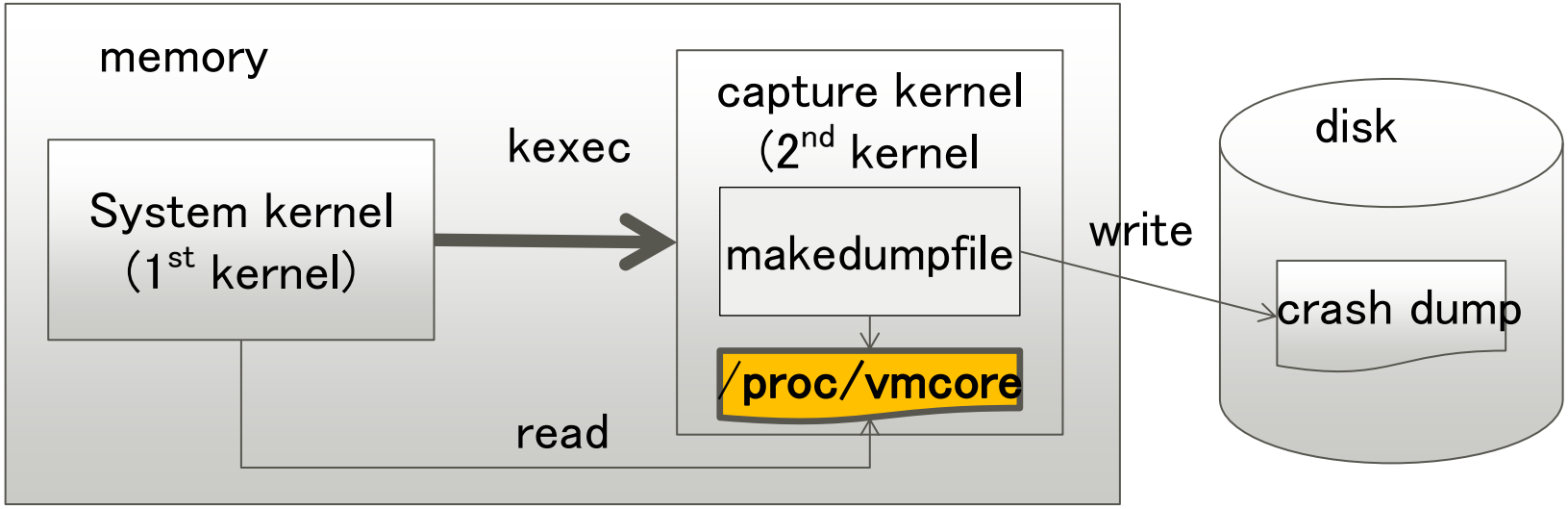
Compression ratio w/ dirtiness of data.

FUJITSU



There will be trade-off !

/proc/vmcore
copyless processing



Reading /proc/vmcore is slow

FUJITSU

- Total size of copy between kernel-land and user-land exceeds 1 TiB
- But…copying itself is too slow.
  - To access physical pages, ioremap() is performed separately in each page
    - even if multiple contiguous pages are requested
  - Many vmalloc data structure updates
  - Many TLB flush

We added a new method “mmap”

mmap() on /proc/vmcore

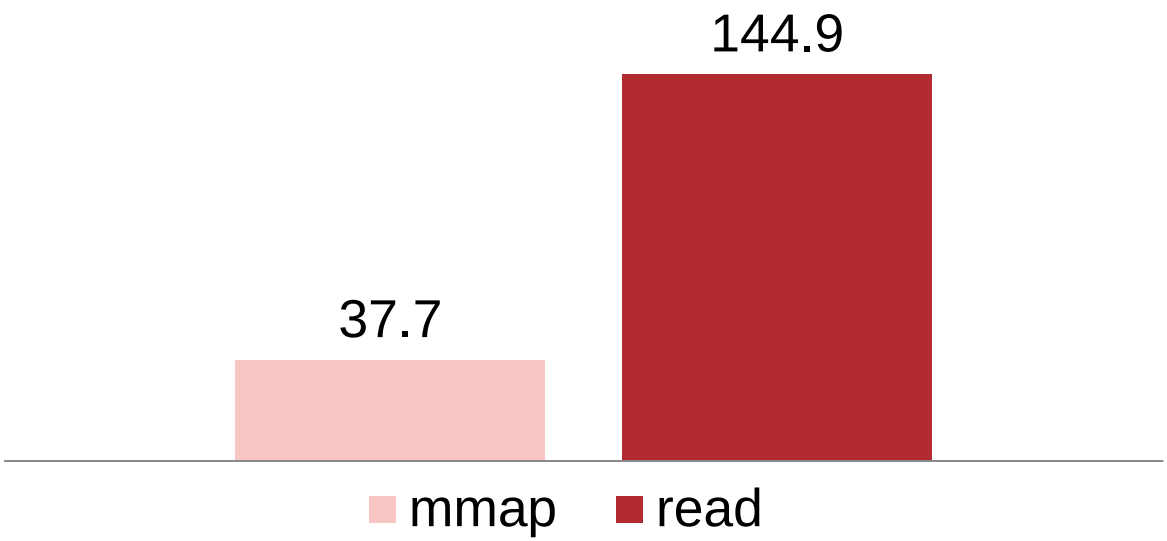
FUJITSU

- No buffer copy between kernel-land and user-land (zero copy)
- Use remap_pfn_range()
  - lightweight since it handles simpler data structure
  - requested size of memory mapping
- Development status
  - Linux kernel since v3.10
  - makedumpfile since v1.5.3

Benchmark

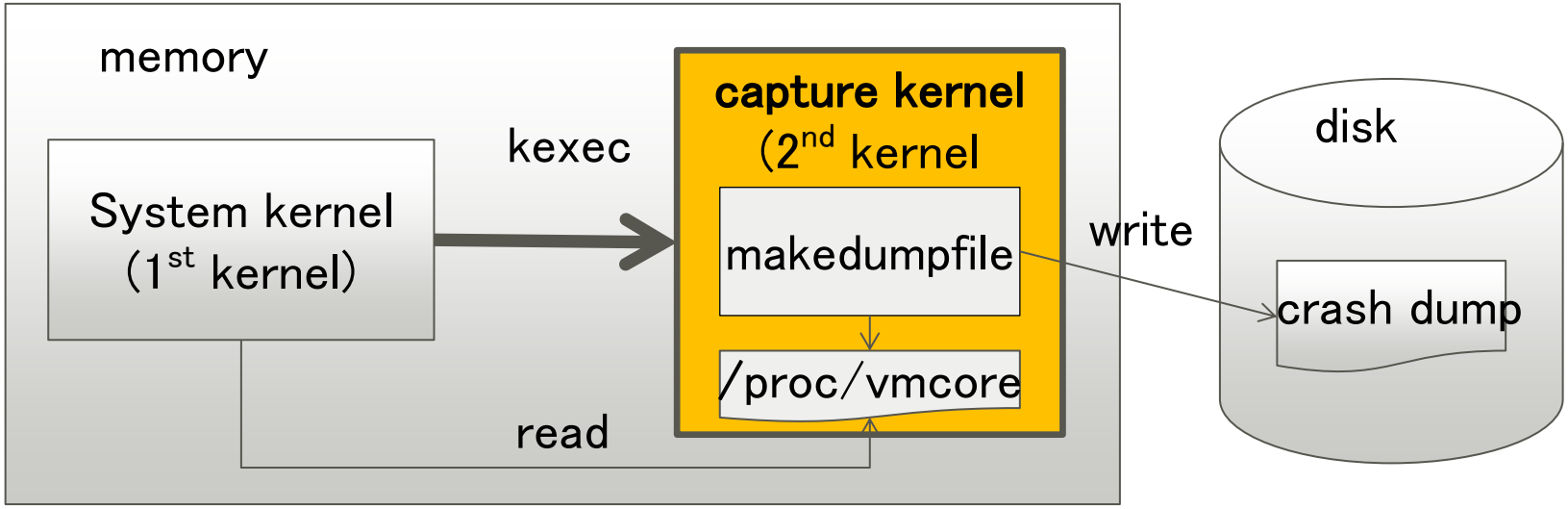
FUJITSU

- Seconds to read /proc/vmcore of 60GB



- **mmap() is 4 times faster!**

SMP capture kernel.



x86 1-CPU restriction on capture kernel

FUJITSU

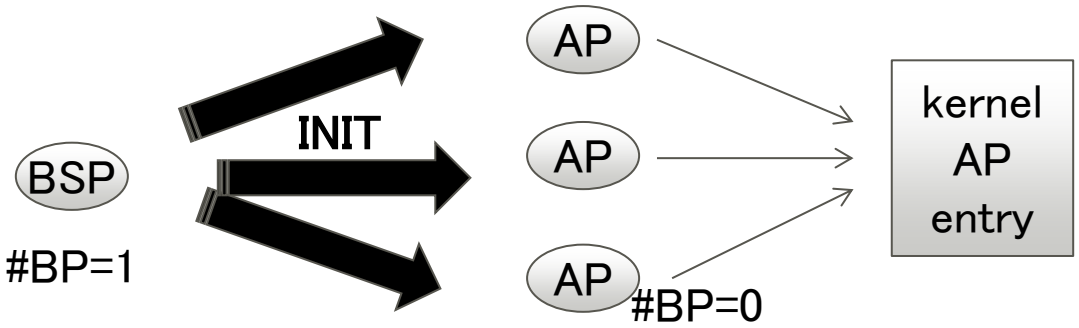
- On capture kernel, only 1-CPU is available even if multiple CPUs are available on system kernel
- `nr_cpus=1` or `maxcpus=1` is specified in kernel parameter of capture kernel

This restriction has been a bottleneck of kdump speed.
SMP should be available for capture kernel.

But ... the problem is in MP Initialization Protocol.

MP Initialization Protocol

- N-CPUs consist of 1-BSP and (N-1)-APs
- BSP(boot strap processor)
  - IA32_APIC_BASE MSR #BP is set
  - Jump to BIOS's init code at receiving INIT.
- AP(application processor)
  - halting until INIT from BSP receives



System Kernel
CPU#0 (boot cpu) == BSP

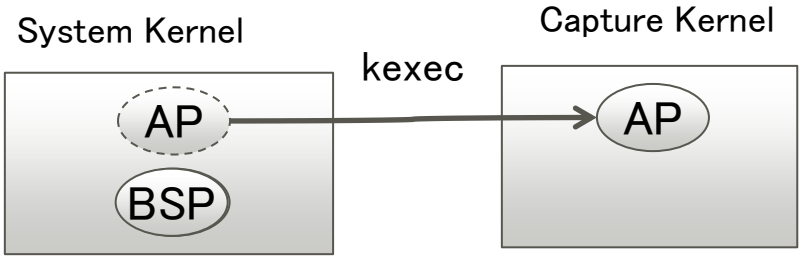
Capture Kernel
CPU#0 (boot cpu) != BSP

MP init protocol issue in capture kernel

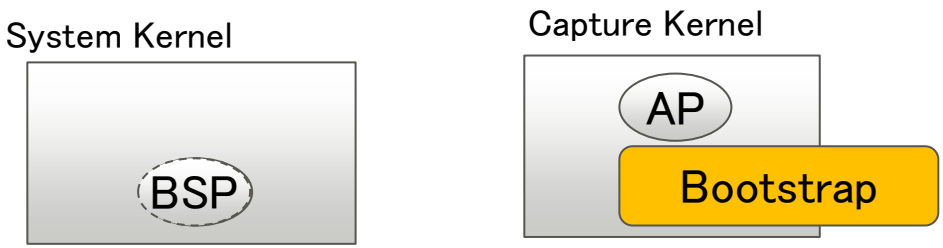
FUJITSU

- If BSP receives INIT IPI, kdump fails.

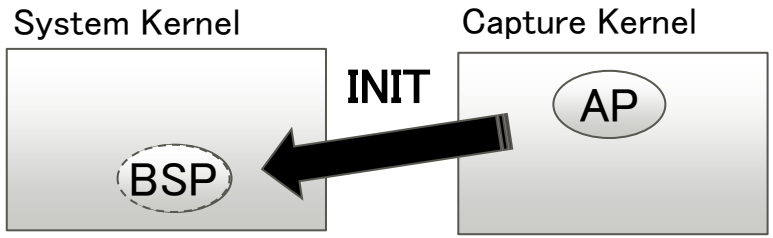
1. Crash happens on a AP. AP boots into capture kernel by kexec. BSP halts.



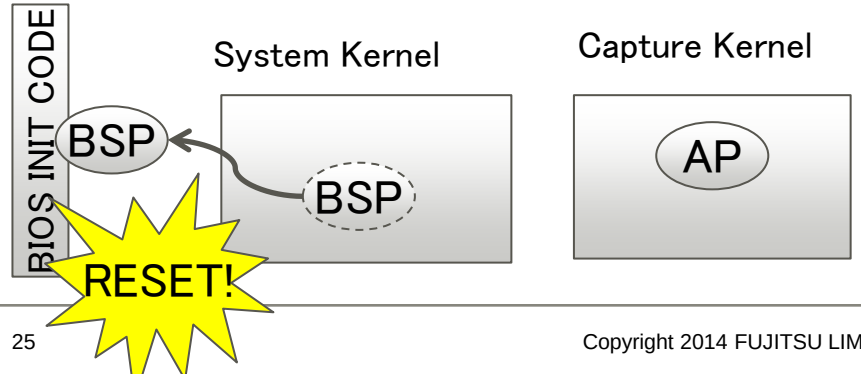
2. System has stopped here. The AP is CPU#0 in capture kernel.



3. The CPU#0 in capture kernel tries to initiate other halting CPUs by INIT, but ･･･the CPU may be BSP**(#BP is set).**



4. Sending INIT to CPU with #BP makes system hangs, get reset or powered off.



Fixing boot issue of capture kernel

FUJITSU

- There has been 4 ideas.

1. Always boot caputure kernel on BSP
2. Clear #BP before kexec
3. Use NMI instead of INIT
4. Avoid using original BSP in the capture kernel

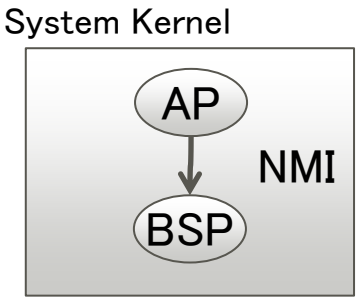
Finally, #4 has been acknowledged.

1: Always boot caputure kernel on BSP

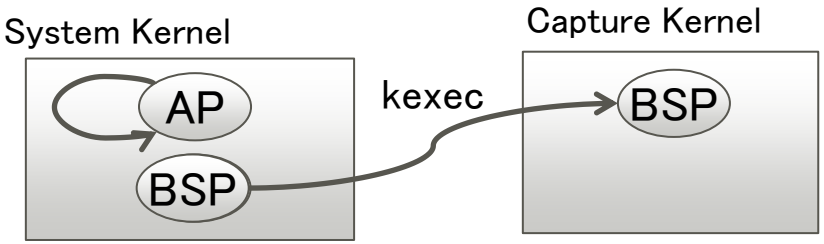
FUJITSU

- kdump stops the non-crashing CPUs by IPI NMI
- Switch to BSP in the IPI NMI processing

1. Crashing AP switches to BSP by NMI



2. The crashing AP halts. Switched BSP boots capture kerenl by kexec



- Nacked. This affects kdump reliability.
  - No guarantee that IPI NMI works well at crash
    - e.g. memory corruption on IRQ vectors
      Some of non-crashing CPUs could be broken

2: Clear #BP before kexec

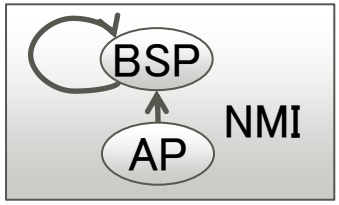
FUJITSU

- Only difference of BSP and AP is whether #BP in BSP's IA32_APIC_BASE MSR is set or not
- How about clearing #BP of BSP?
  - Then, all CPUs are APs.
- Nacked. There's some system assuming the initial mapping of BSP and AP throughout system running
  - HP machine hangs during shutdown process. (http://lists.infradead.org/pipermail/kexec/2013-August/009420.html)
  - Thanks to contribution by Jingbai Mar

3: Use NMI instead of INIT

FUJITSU

- The technique used by CPU0 hot-plugging
  - hot-add / hot-remove CPU0
  - Use NMI to let CPU0 go out from halting state



- Nacked. This cannot be applied to kdump case.
  - BSP could be any buggy state at crash.
  - NMI is signaled from capture kernel to BSP halting in system kernel.
    - BSP needs to load capture kernel's IDT in system kernel

4: Avoid using original BSP in the capture kernel

FUJITSU

- Lose 1 CPU but always work! => Acked.
- We learned
  - No method to reset BSP state except for INIT
    - (MultiProcessor Specification Version 1.4 May 1997)
- Losing 1CPU is no problem in the real world
  - Typically a lot of CPUs on terabyte-scale system
- New **`disable_cpu_apicid`** kernel parameter
  - Introduced at v3.14
  - Specify initial APIC id of CPU#0
  - kexec-tool adds this for capture kernel automatically.

Parallel compression benchmark

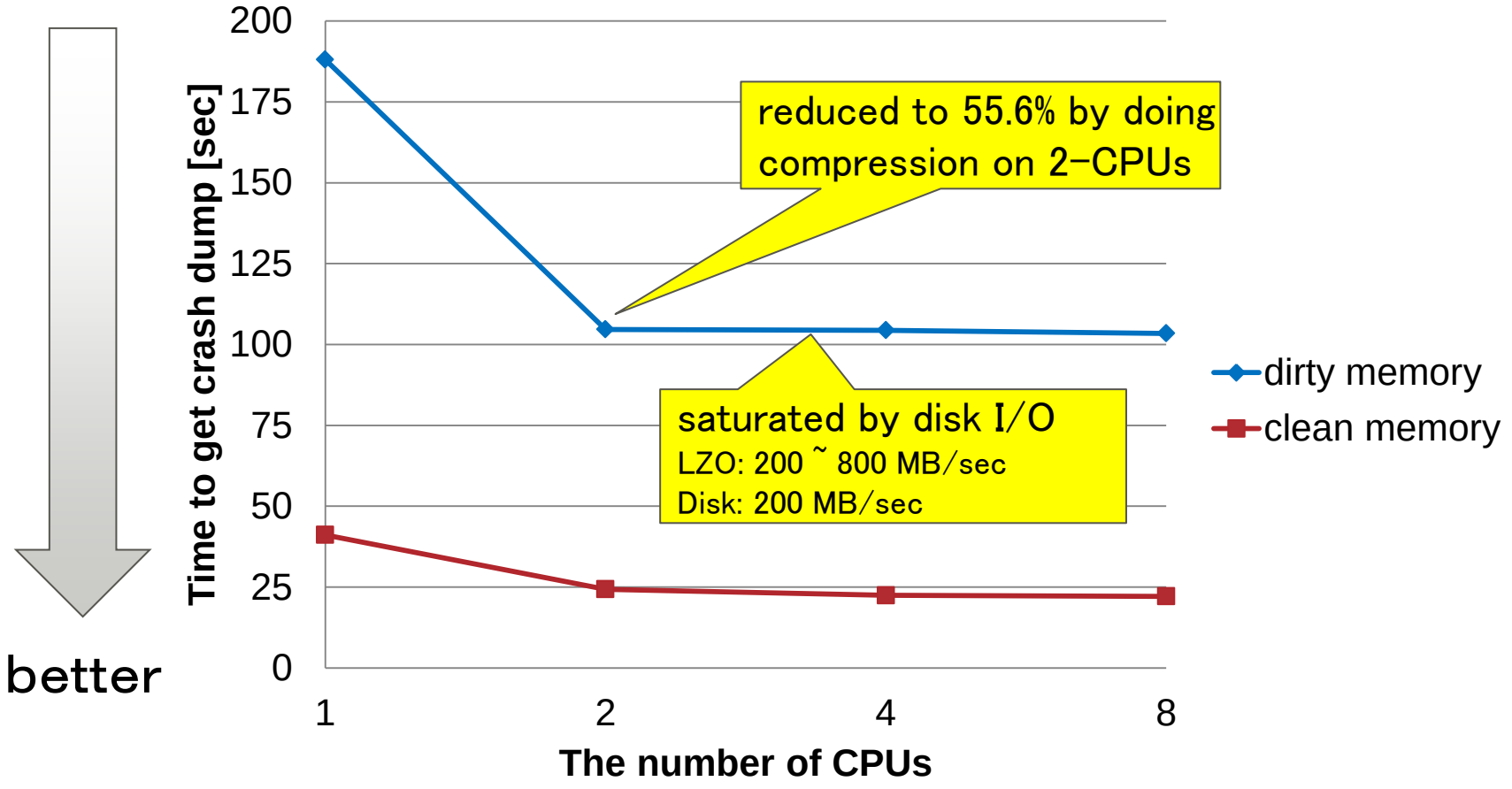
FUJITSU

- Use --split option for multi-processing

```
makedumpfile --split /proc/vmcore vmcore-0 vmcore-1 ...
```

  - Parallel compression
  - No parallel I/O because only a single disk is used
- Environment
  - PRIMQUEST 2800E
  - Memory: 64 GB
  - CPU: Intel(R) Xeon(R) CPU E7-8890 v2@2.80GHz
  - Disk: Performance 200 MB/s

Parallel LZO compression speed



- Please find the best number of CPUs by benchmark!

More performance?

FUJITSU

- If you want to get more performance, you have to optimize I/O work:
  - Use faster disks
  - Use multiple disks

Agenda

FUJITSU

- Background
- Review kdump structure
- 3 improvements on the scalability issues
  - Use fast compression format
  - Copyless processing with mmap()
  - Break a 1-CPU restriction of kdump capture kernel
- Ongoing work
  - kexec-tools multiple CPUs support

kexec-tools multiple CPUs support

FUJITSU

- kexec-tools
  - kexec/kdump configuration utility on fedora
- Users need to specify # of CPUs manually.
  - Default configuration is 1-CPU
    - 1-CPU is most reliable
    - 1-CPU is enough for most systems in performance
  - Increasing CPU needs memory consumption
    - Kernel data structures increase depending on the number of CPUs
    - Capture kernel should be as small as possible for system kernel
- Need more explanation in kexec docs.

SMP Configuration Example

FUJITSU

To use 4-CPUs for capture kernel and makedumpfile:

1. Append nr_cpus kernel parameter in /etc/sysconfig/kdump

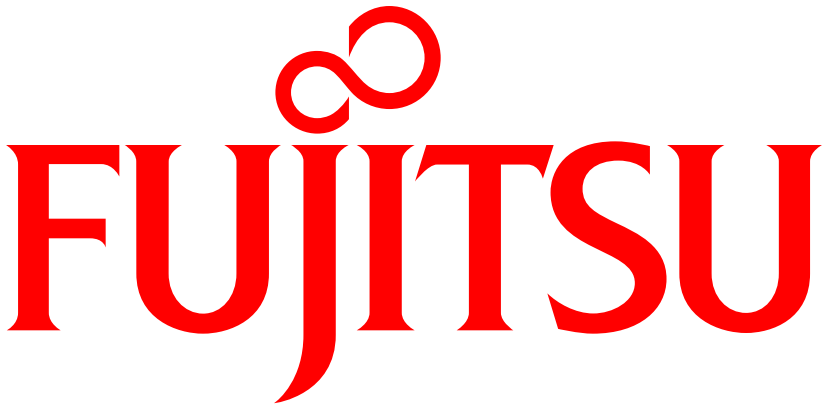
```
KDUMP_COMMANLINE_APPEND="... nr_cpus=4 ..."
```

2. Specify a command in core_collector directive in /etc/kdump.conf

```
core_collector makedumpfile --num_threads 4
```

   - This multi-threading feature is ***under development***.

3. Restart kdump service



shaping tomorrow with you