Introducing Cache Pseudo-Locking to reduce memory access latency

Reinette Chatre

About me

Software Engineer at Intel (~12 years)

- Open Source Technology Center (OTC)

Currently

- Enabling Cache Pseudo-Locking in the Linux kernel

Previous Linux kernel work

- Ultra-wideband (UWB) enabling
- Maintainer of Intel Wireless WiFi (iwlwifi) driver

Goal

Introduce Cache Pseudo-Locking* and demonstrate that it can be used to reduce memory access latency in the presence of noisy neighbors.

*might not be supported on all processors

Agenda

- Overview of CPU caches
- Review of Cache Allocation Technology (CAT)
- Introduction to Cache Pseudo-Locking
- How to pseudo-lock memory to cache
- Cache Pseudo-Locking in Linux
- Cache Pseudo-Locking performance
- Current status and Future work

Overview of CPU caches

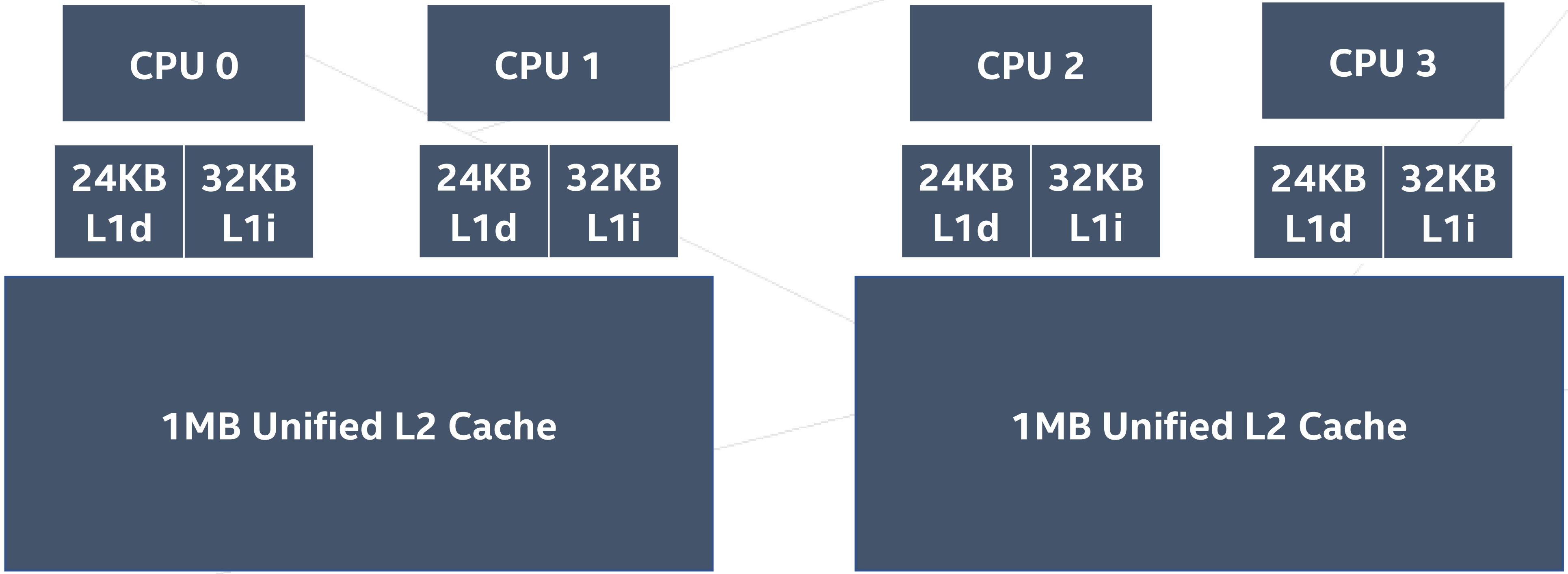
Hardware cache

- Memory has trade-off between size and speed. Fastest memory is small, larger memory is slower.
- Cache memory is smaller than main memory, but closer to CPU to be able to serve data faster than main memory.
- Systems address trade-off with multiple levels of cache.
- Some caches may be specific to data or instructions.
- Cache details available in

  /sys/devices/system/cpu/cpu/cache/index**

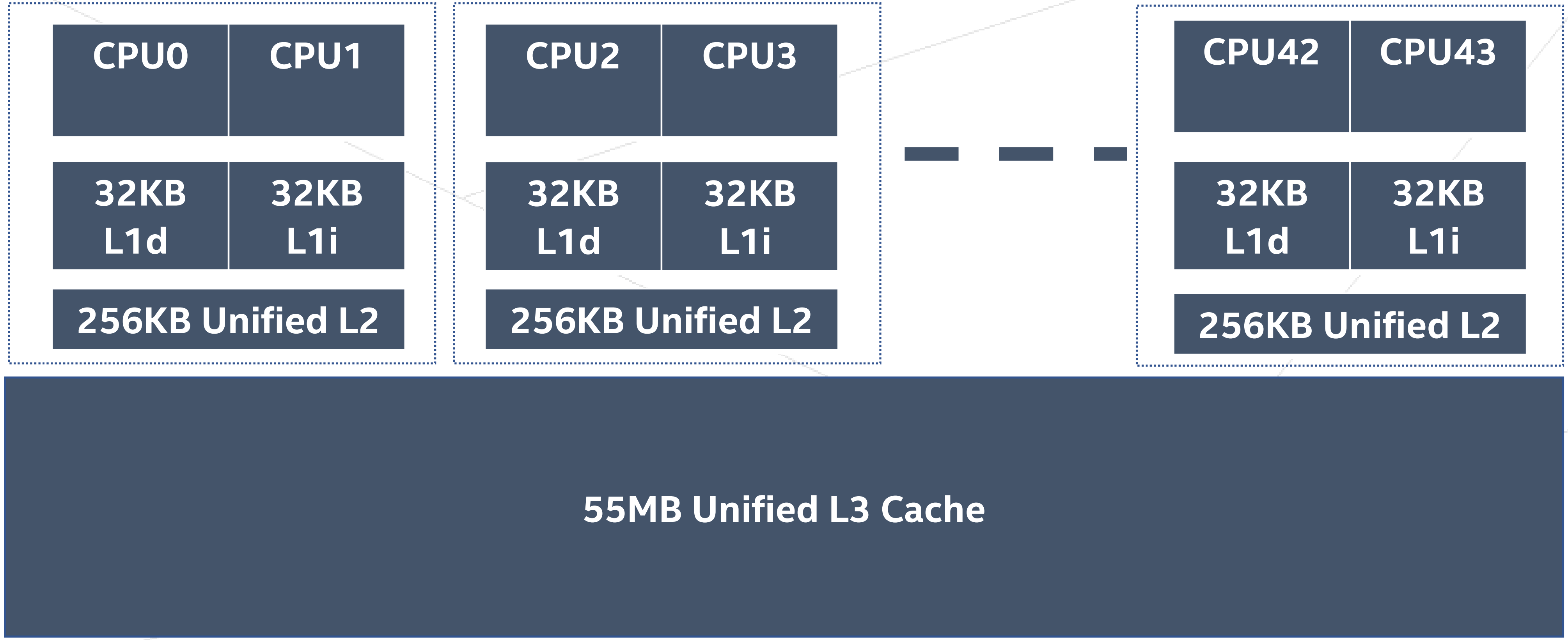
Hardware cache example 1

Intel® Celeron® Processor J3455 (Atom)

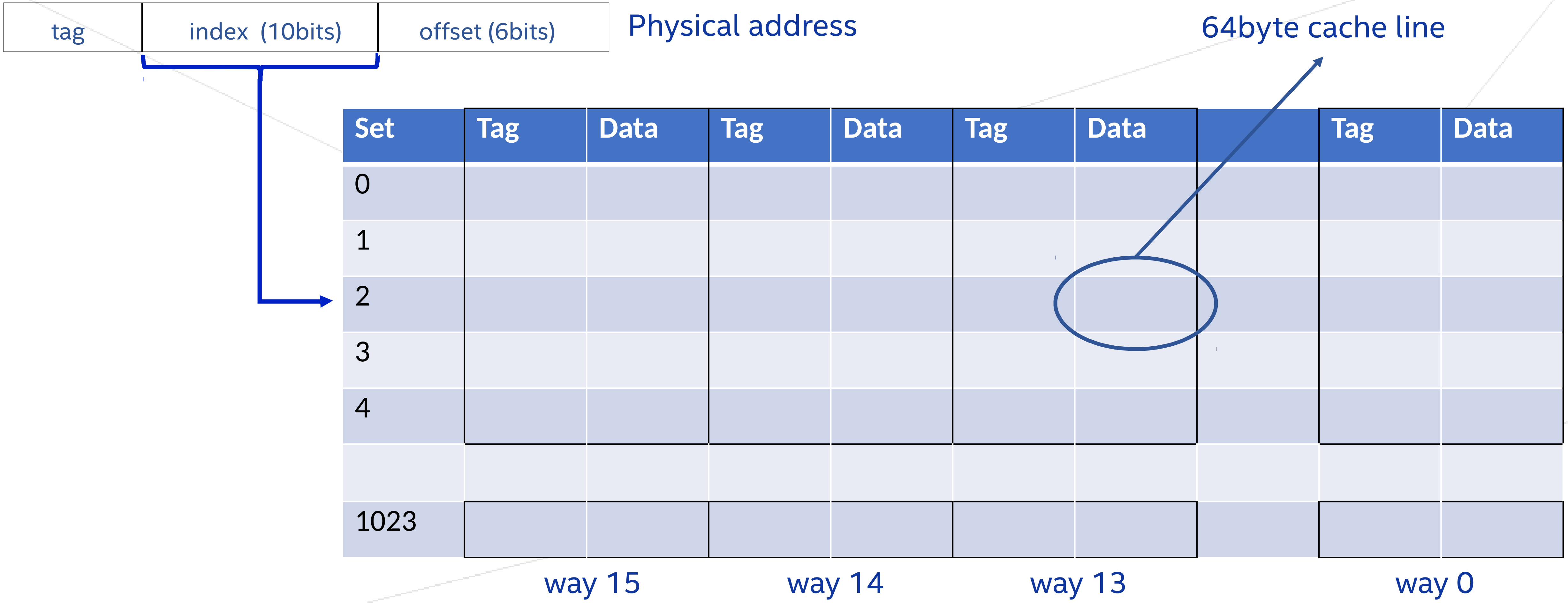


Hardware cache example 2

Intel® Xeon® Processor E5 v4 Family



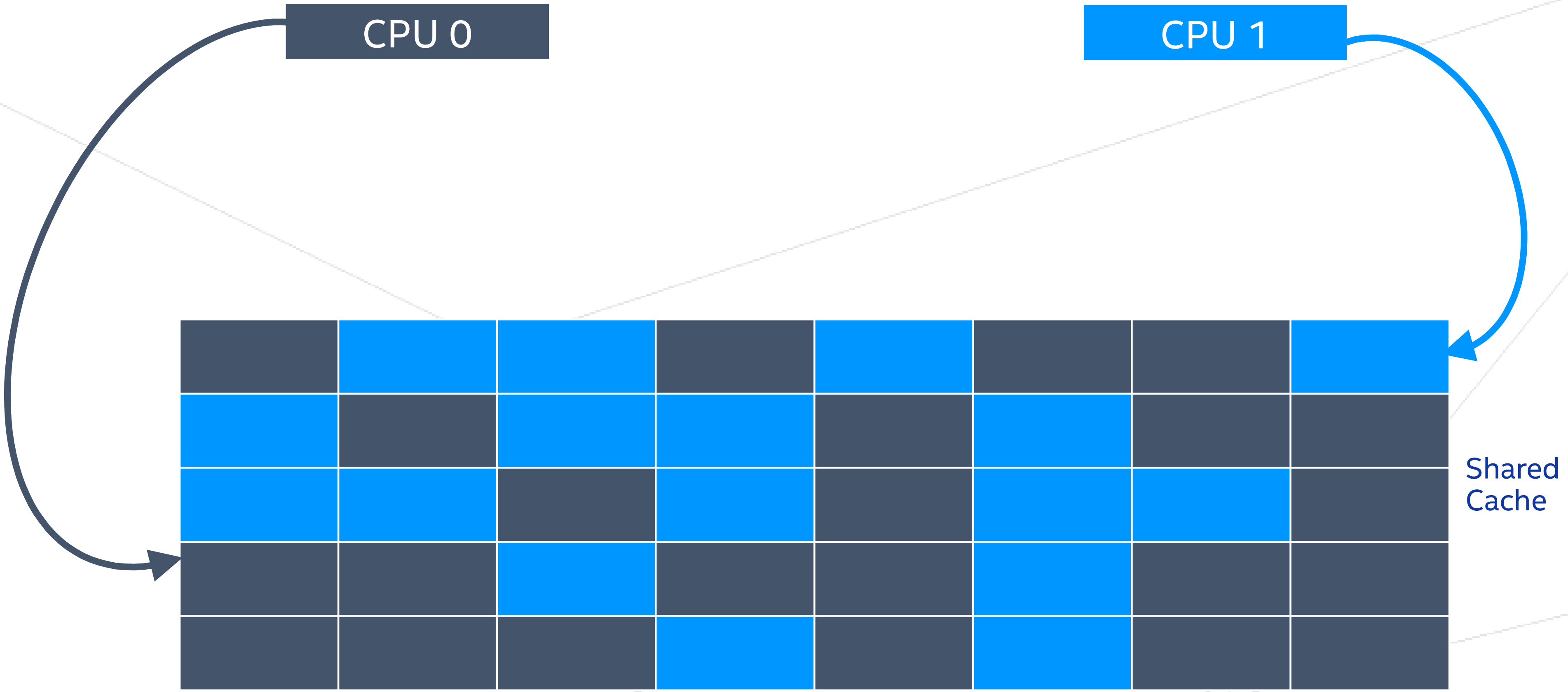
Mapping a physical address to the cache*



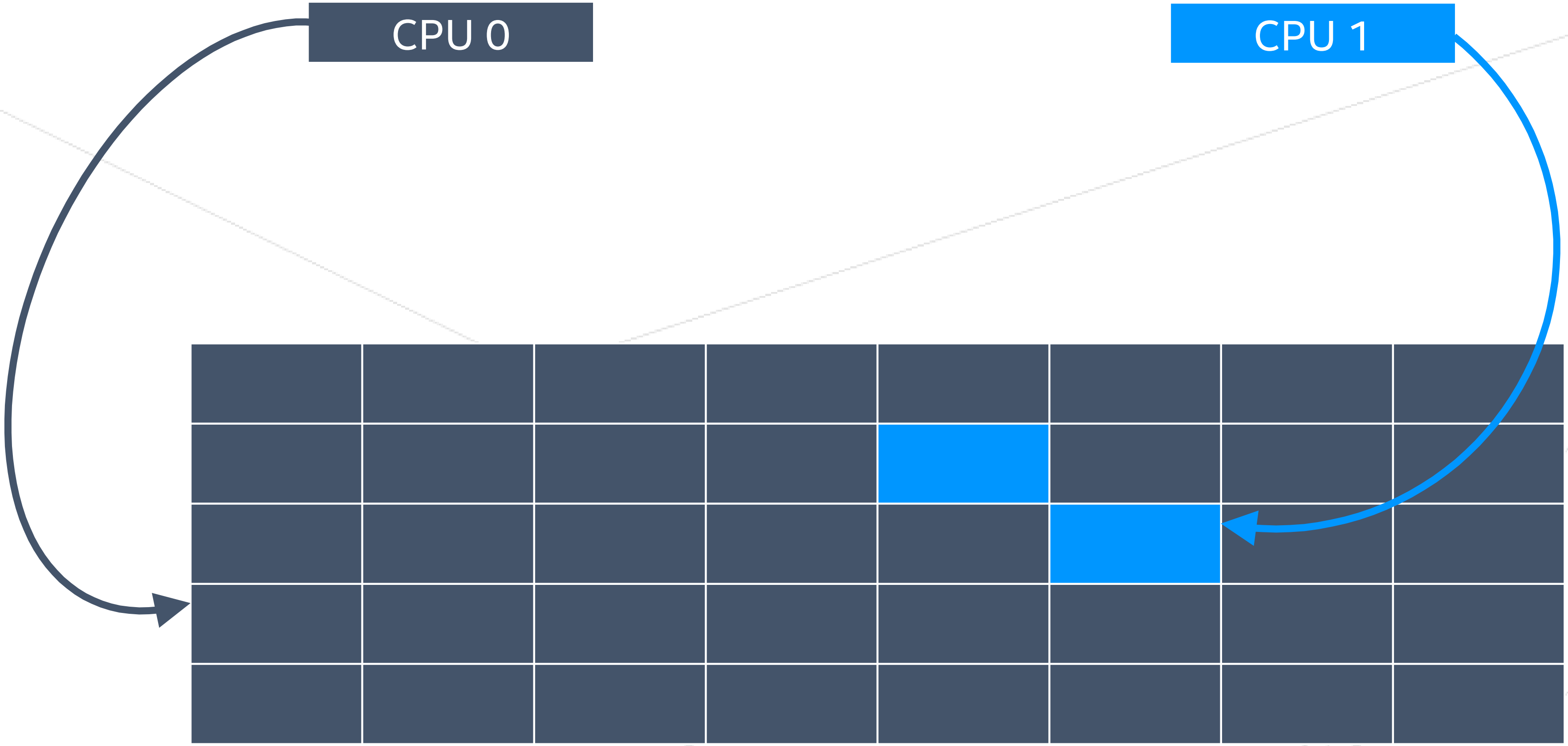
* general example only, not tied to any particular product

Review of Cache Allocation Technology (CAT)

Multiprocessor systems share resources

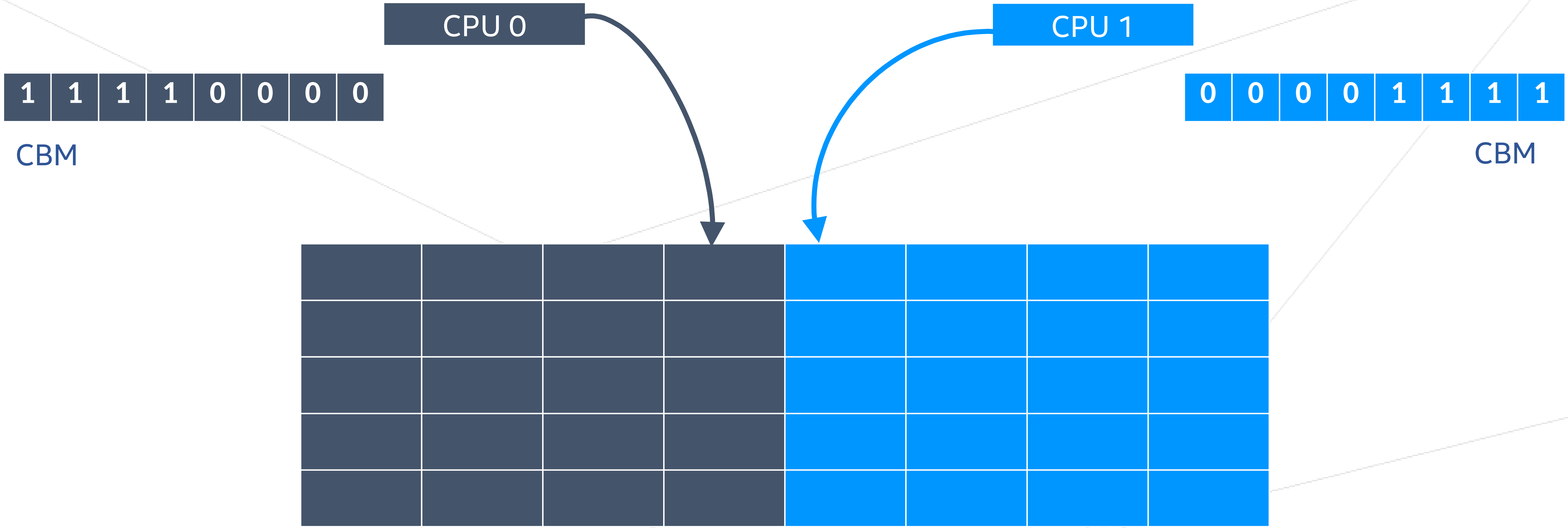


Shared resources and interference



- Tasks may make heavy use of shared resources at varied intervals.
- Low priority task(s) on one CPU could affect high priority task(s) on neighboring CPU(s), also referred to as "Noisy neighbors".

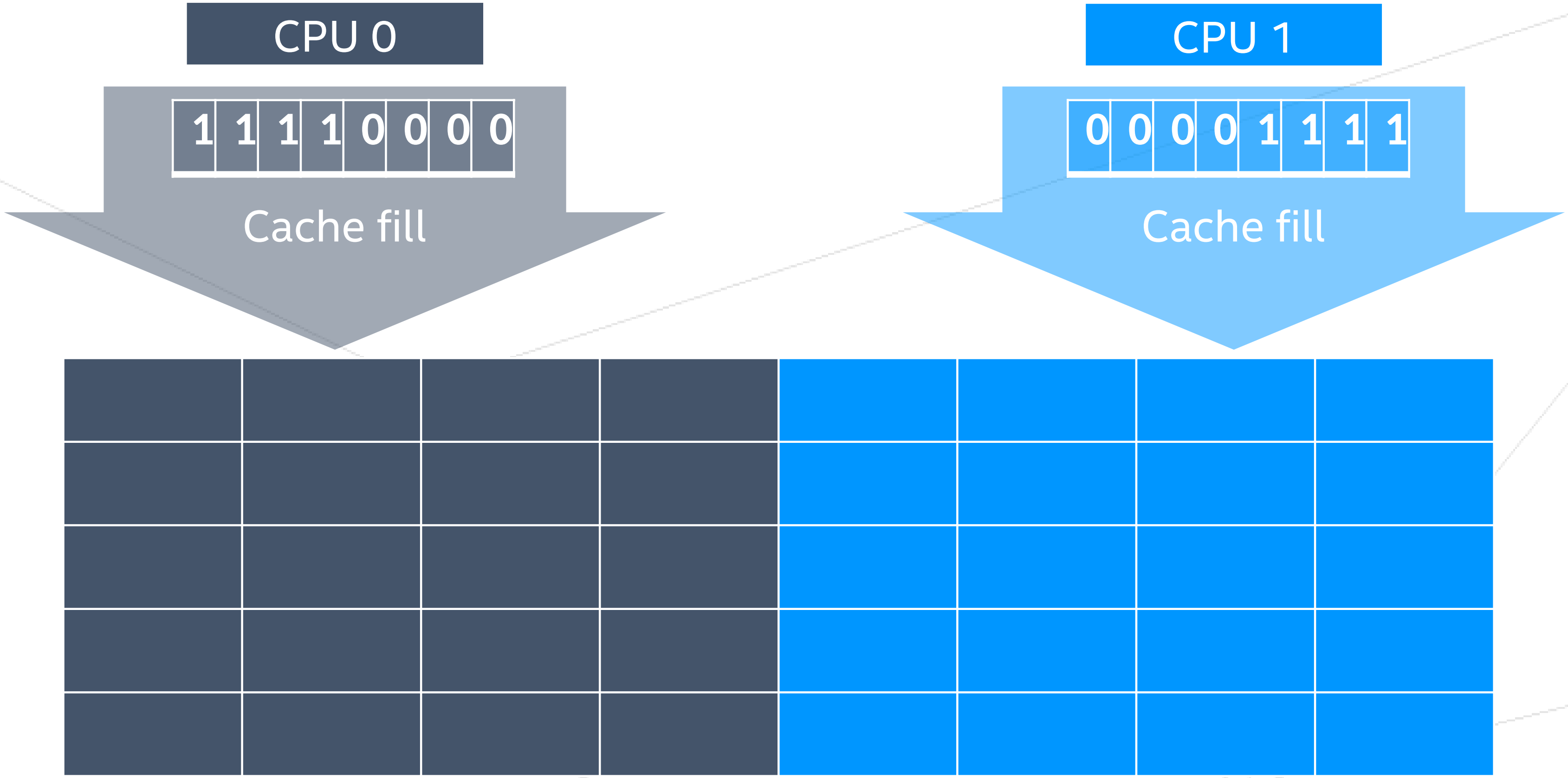
Cache Allocation Technology (CAT)



- CAT restores cache fairness by using a capacity bitmask (CBM) to specify the amount of cache space into which a CPU or task can fill.

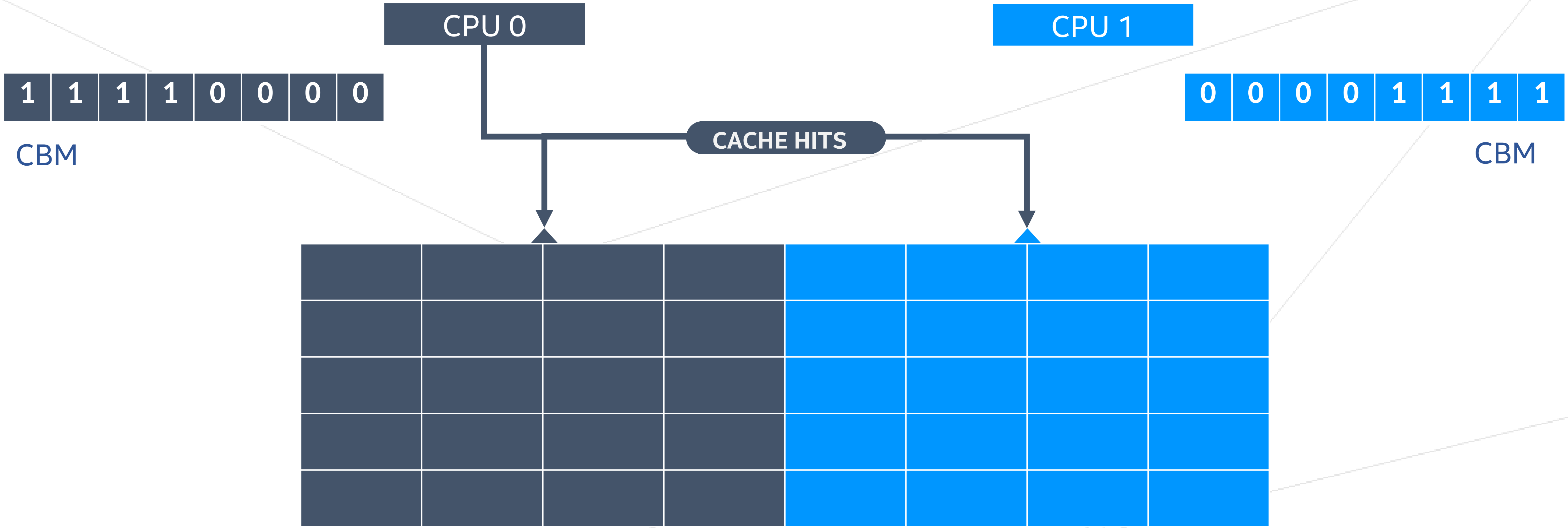
Introduction to Cache Pseudo-Locking

Cache **miss** can only fill into allocated region



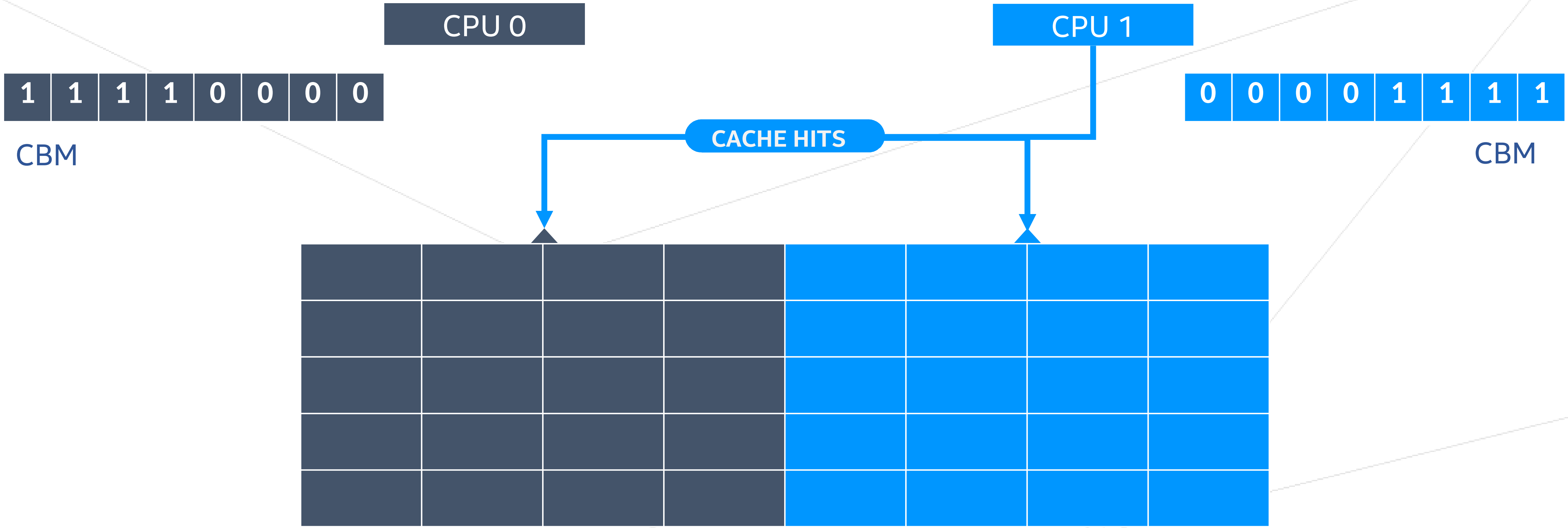
- CAT restores cache fairness by restricting **cache-fill** to allocated cache region.

Cache **hits** still serviced from entire cache



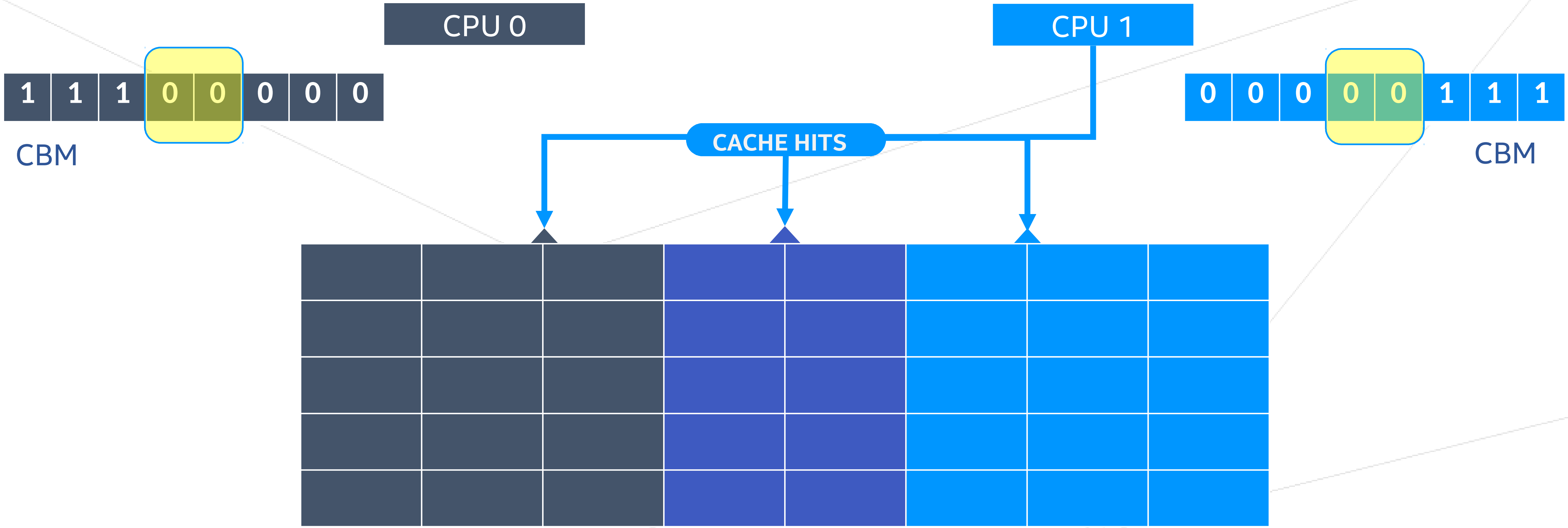
- CPU can still read and modify data outside allocated region on cache hit.

Cache **hits** still serviced from entire cache



- CPU can still read and modify data outside allocated region on cache hit.

Cache Pseudo-Locking

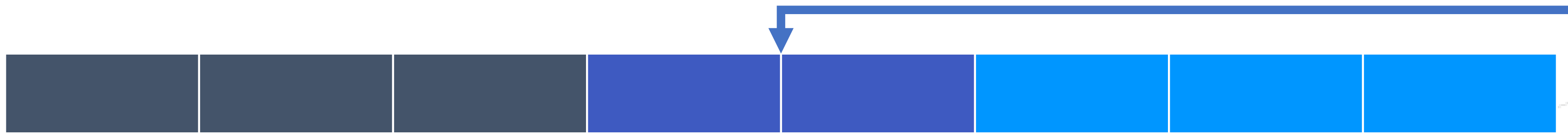


- Preload memory into region of cache and then orphan it (no fill possible).
  - Cache region only serves cache hits to the "pseudo-locked" memory.

How to pseudo-lock memory to cache

Pseudo-lock physical memory to cache

Cache



1. Ensure cache region not in any active CBM.

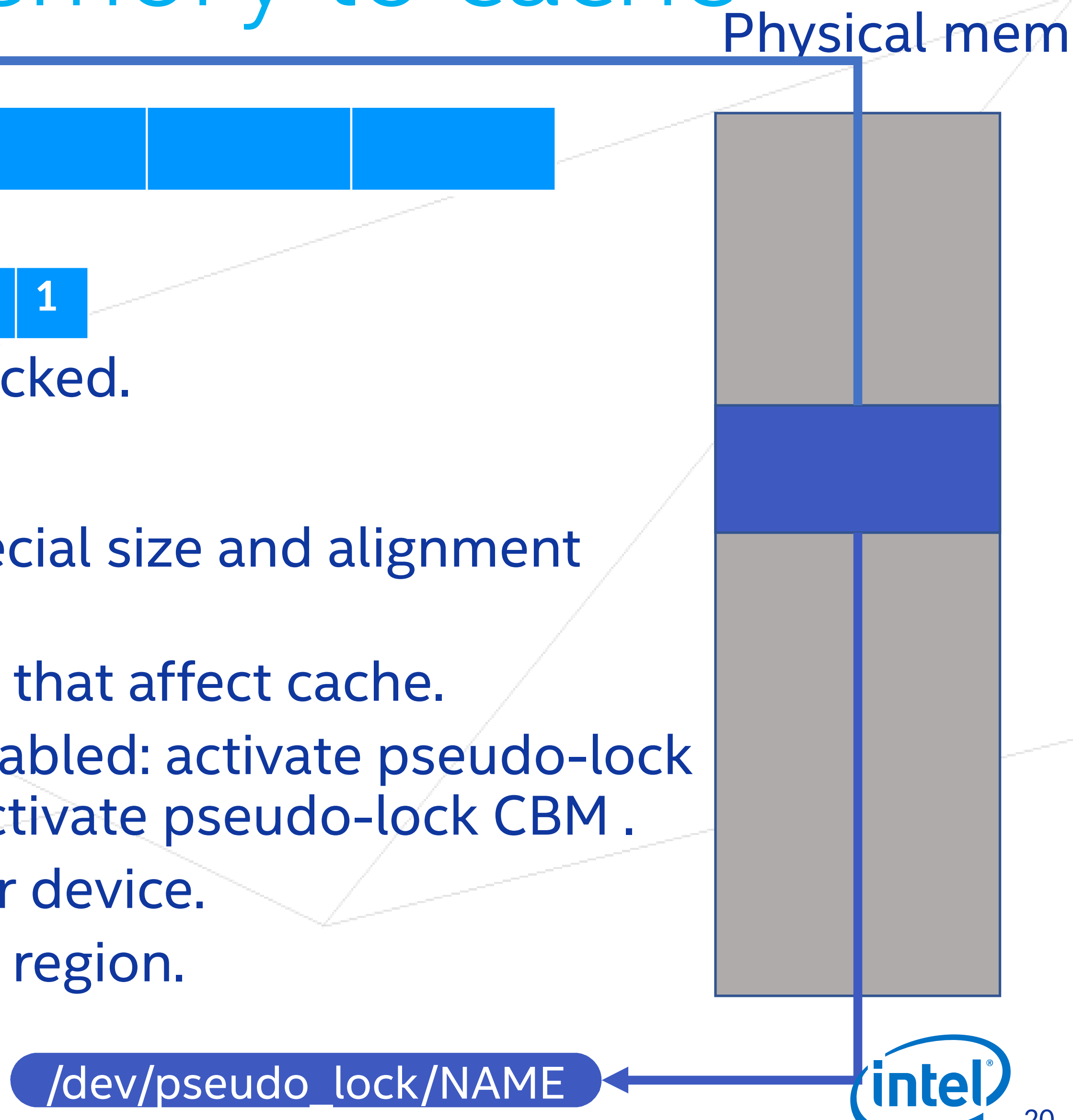
1	1	1	0	0	0	0	0

0	0	0	0	0	1	1	1

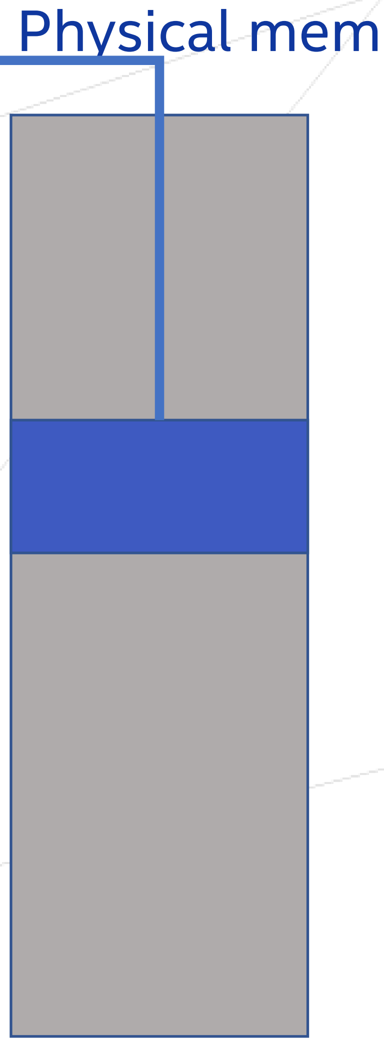
2. Specify CBM of cache region to be pseudo-locked.

0	0	0	1	1	0	0	0

3. Contiguous region of physical memory of special size and alignment allocated and cleared.
4. Prevent system from entering deeper C-state that affect cache.
5. Kernel thread: clear cache, with interrupts disabled: activate pseudo-lock CBM, read physical memory into cache, de-activate pseudo-lock CBM .
6. Pseudo-locked memory exposed as character device.
7. No CBM allowed to overlap with pseudo-lock region.



How to read memory into cache …



CBM specifies which region of cache can be filled into

0	0	0	1	1	0	0	0

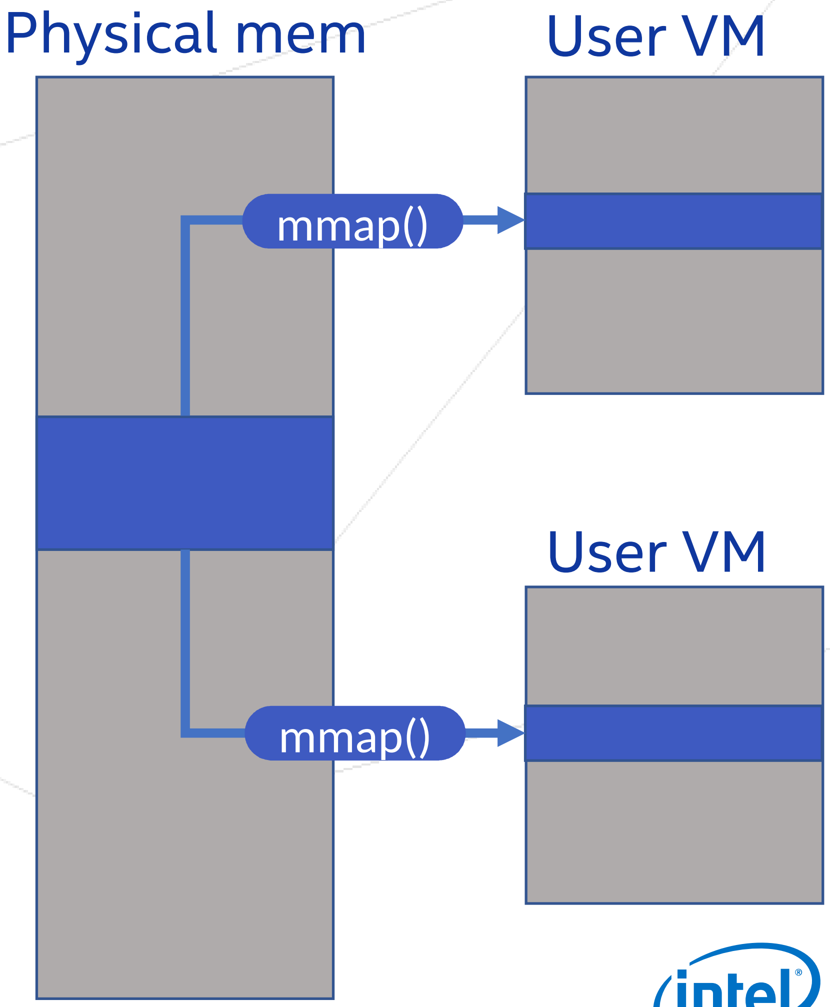
1. Ensure variables describing physical memory are in registers and/or L1 cache.
2. Memory traversed using kernel logical addresses. Consider the page walker as it populates the paging structure caches and Translation Lookaside Buffer (TLB).
   - Loop over data twice: first loop at stride of PAGE_SIZE, to populate paging structure caches; second loop at stride of cache line size.
3. Disable hardware prefetchers.
4. Add barriers to prevent speculative execution of loop used to traverse the memory.

Map pseudo-locked memory to user space

- User space maps (mmap()) pages of pseudo-locked memory into its own address space.
  fd = open("/dev/pseudo_lock/NAME", ...);
  ptr = mmap(..., fd, ...);

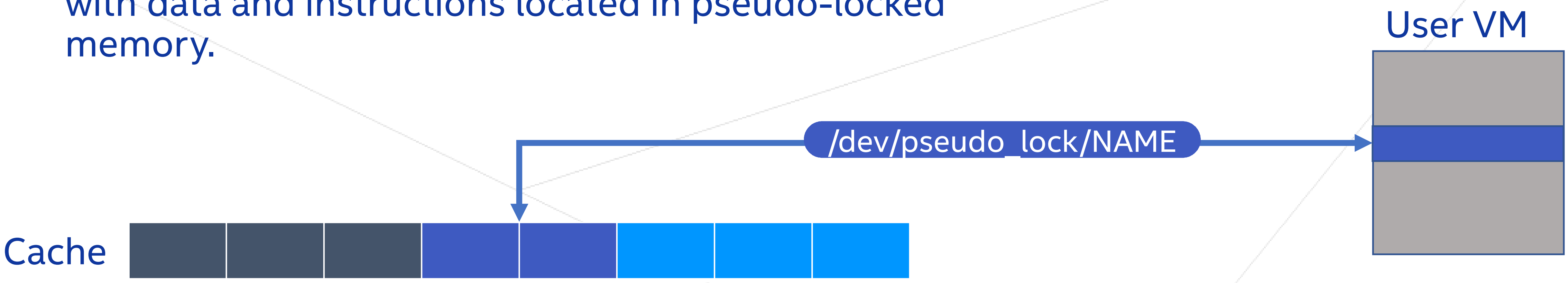
- Pseudo-locked memory can be mapped by multiple tasks.

- Pseudo-locked cache region in unified cache so user space could copy critical data and/or instructions to pseudo-locked memory.



Low latency memory in user space

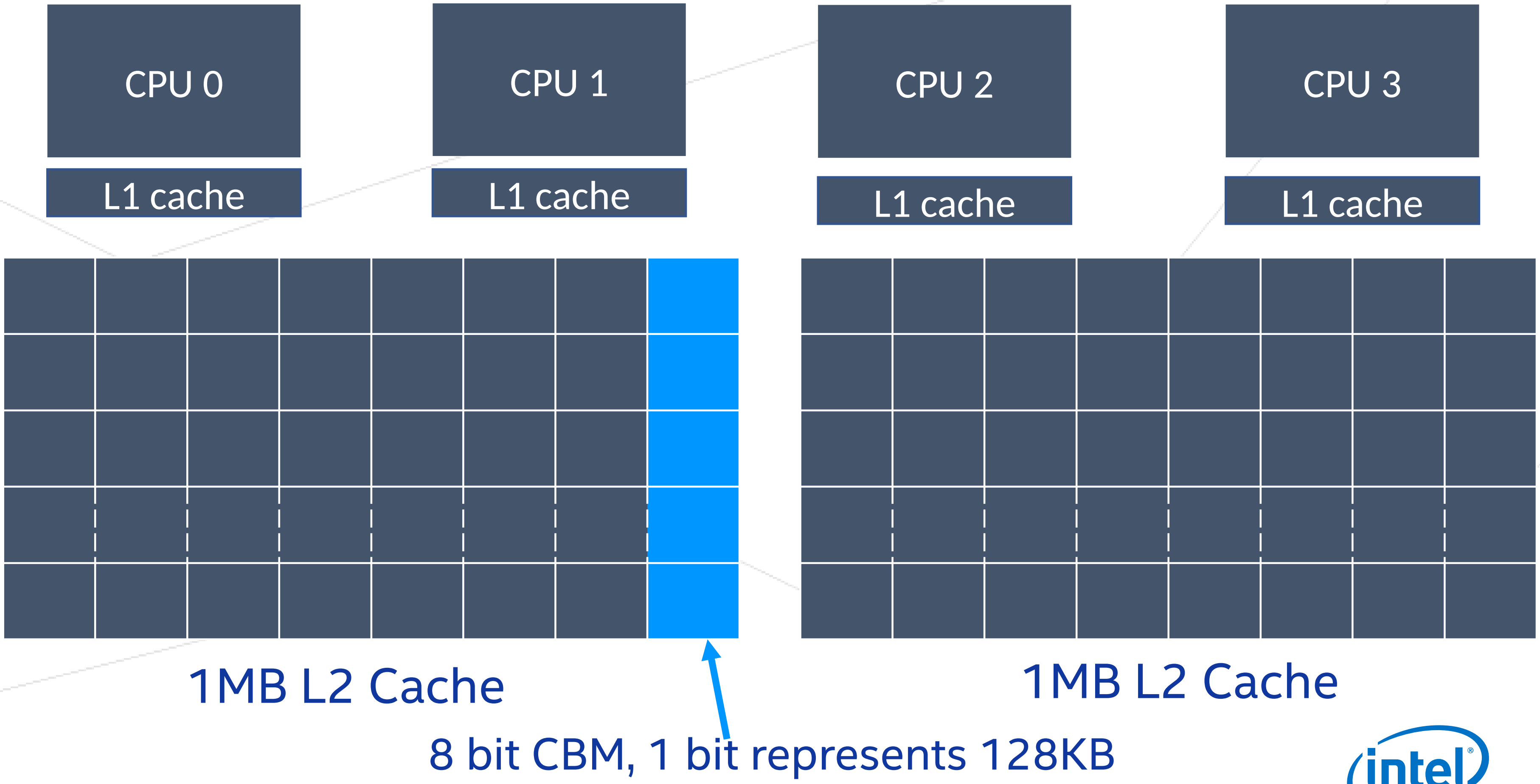
- User space obtains cache access latency interacting with data and instructions located in pseudo-locked memory.



Cache Pseudo-Locking in Linux

Test system: Intel® Celeron® Processor J3455 (Atom)

Intel® NUC NUC6CAYS



Cache Allocation Technology (CAT) interface

- Platform needs to support CAT – look for cat_l[23] in /proc/cpuinfo
- Kernel compiled with CONFIG_INTEL_RDT=y
- New resctrl filesystem introduced as part of CAT enabling

```
# mount -t resctrl resctrl /sys/fs/resctrl
# grep -r . /sys/fs/resctrl/info/*
/sys/fs/resctrl/info/last_cmd_status:ok
/sys/fs/resctrl/info/L2/min_cbm_bits:1
/sys/fs/resctrl/info/L2/shareable_bits:0
/sys/fs/resctrl/info/L2/num_closids:4
/sys/fs/resctrl/info/L2/bit_usage:0=SSSSSSSS;1=SSSSSSSS
/sys/fs/resctrl/info/L2/cbm_mask:ff
```

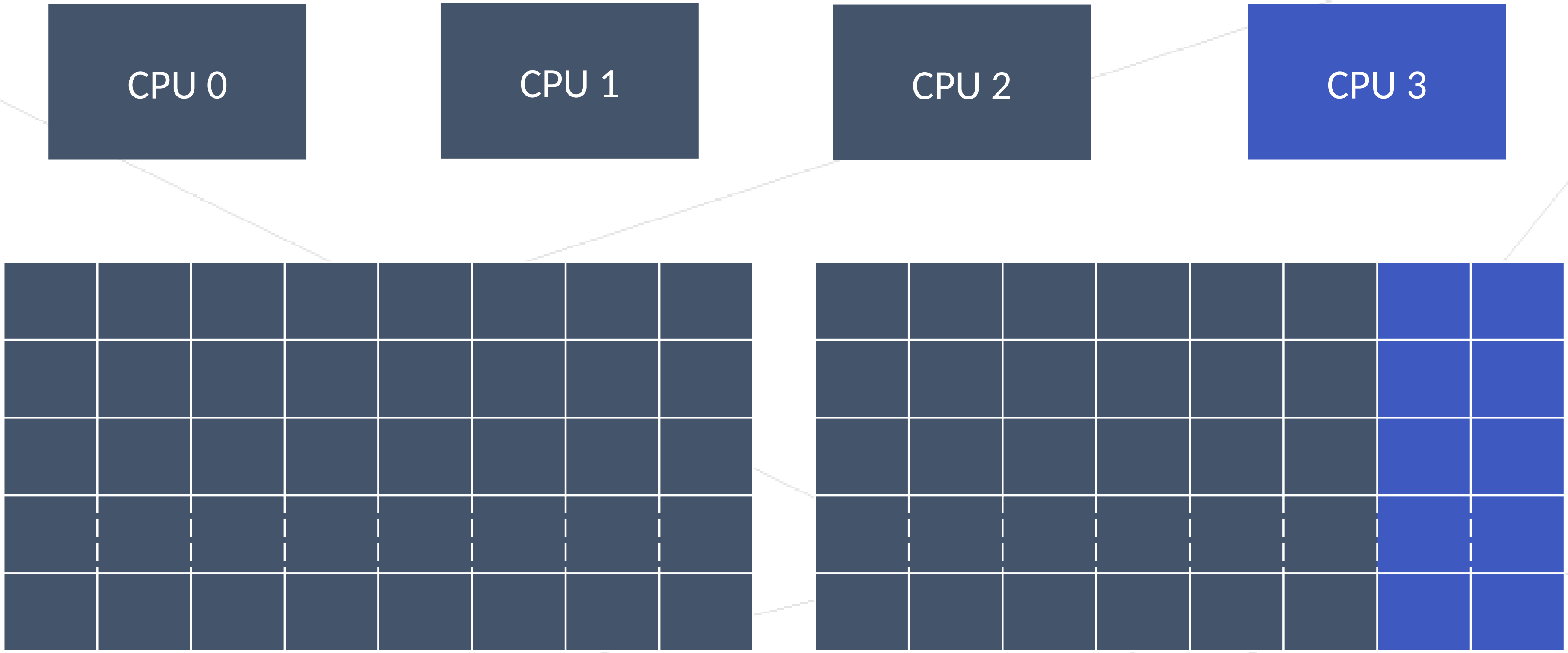
CAT Interface (continued)

- By default all CPUs and tasks run with default CBM set to fill to entire cache.

```
# grep -r . /sys/fs/resctrl/* | grep -v info
/sys/fs/resctrl/cpus:f
/sys/fs/resctrl/cpus_list:0-3
/sys/fs/resctrl/mode:shareable
/sys/fs/resctrl/schemata:L2:0=ff;1=ff
/sys/fs/resctrl/size:L2:0=1048576;1=1048576
/sys/fs/resctrl/tasks:1
/sys/fs/resctrl/tasks:2
/sys/fs/resctrl/tasks:3
[SNIP]
```

CBM

Example: Pseudo-lock 256KB memory to cache



- High priority task needing low latency pseudo-locked memory to run on CPU3.
- Task profiling or monitoring reveals memory requirements – may include data and instructions.

Pseudo-lock physical memory to cache

Cache

Physical mem

1. Ensure cache region not in any active CBM.

   1 1 1 0 0 0 0 0   0 0 0 0 0 1 1 1

2. Specify CBM of cache region to be pseudo-locked.

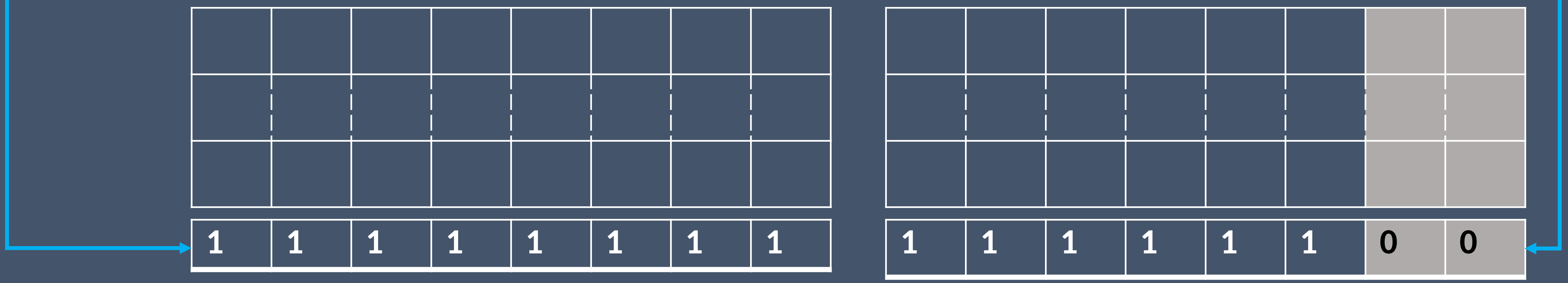
   0 0 0 1 1 0 0 0

3. Contiguous region of physical memory of special size and alignment allocated and cleared.
4. Prevent system from entering deeper C-state that affect cache.
5. Kernel thread: clear cache, disable interrupts, activate pseudo-lock CBM, read physical memory into cache, de-activate pseudo-lock CBM .
6. Pseudo-locked memory exposed as character device.
7. No CBM allowed to overlap with pseudo-lock region.

/dev/pseudo_lock/NAME

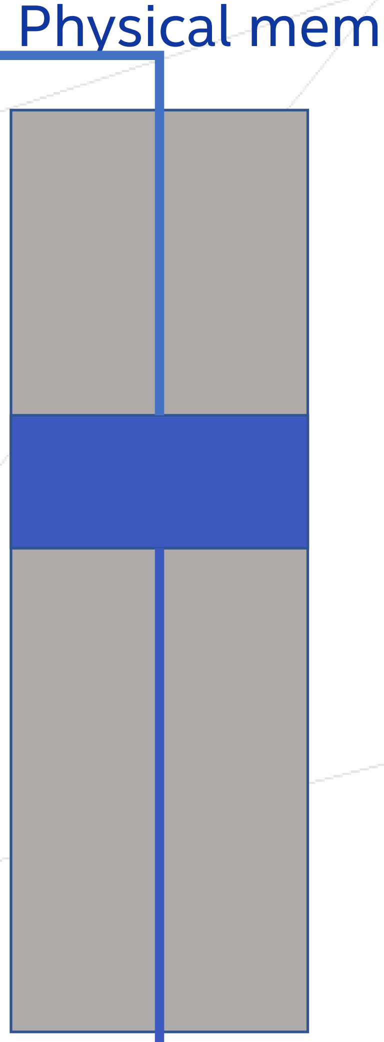
Step1: Ensure cache region not in any CBM

```
# echo 'L2:1=0xfc' > /sys/fs/resctrl/schemata
# cat /sys/fs/resctrl/schemata
L2:0=ff;1=fc
# cat /sys/fs/resctrl/size
L2:0=1048576;1=786432
# cat /sys/fs/resctrl/info/L2/bit_usage
0=SSSSSSSS;1=SSSSSS00
```



Pseudo-lock physical memory to cache

Cache



1. Ensure cache r… …tive CBM.

1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1

2. Specify CBM of cache region to be pseudo-locked.

0	0	0	1	1	0	0	0

3. Contig… …us region of physical memory of …pecial size and alignment allocated and cleared.
4. Prevent system from entering deeper C-state that affect cache.
5. Kernel thread: clear cache, disable interrupts, activate pseudo-lock CBM, read physical memory into cache, de-activate pseudo-lock CBM .
6. Pseudo-locked memory exposed as character device.
7. No CBM allowed to overlap with pseudo-lock region.

/dev/pseudo_lock/NAME

Step2 to Step 7: Specify CBM to pseudo-lock

```
# mkdir /sys/fs/resctrl/p1
```

```
# grep . /sys/fs/resctrl/p1/*
/sys/fs/resctrl/p1/cpus:0
/sys/fs/resctrl/p1/mode:shareable
/sys/fs/resctrl/p1/schemata:L2:0=ff;1=ff
/sys/fs/resctrl/p1/size:L2:0=1048576;1=1048576
```

```
# echo pseudo-locksetup > /sys/fs/resctrl/p1/mode
```

```
# grep -s . /sys/fs/resctrl/p1/*
/sys/fs/resctrl/p1/mode:pseudo-locksetup
/sys/fs/resctrl/p1/schemata:L2:uninitialized
/sys/fs/resctrl/p1/size:L2:0=0;1=0
```

Step2 to Step 7: Specify CBM to pseudo-lock

```
# echo 'L2:1=0x3' > /sys/fs/resctrl/p1/schemata
```

```
# grep -s . /sys/fs/resctrl/p1/*
/sys/fs/resctrl/p1/cpus:c
/sys/fs/resctrl/p1/cpus_list:2-3
/sys/fs/resctrl/p1/mode:pseudo-locked
/sys/fs/resctrl/p1/schemata:L2:1=3
/sys/fs/resctrl/p1/size:L2:1=262144
# grep . /sys/fs/resctrl/info/L2/bit_usage
0=SSSSSSSS;1=SSSSSSPP
```

```
# ls -l /dev/pseudo_lock/p1
crw------- 1 root root 243, 0 Aug  2 06:02 /dev/pseudo_lock/p1
```

Putting it together

```
root@intel-corei7-64:~# cat /proc/1644/maps
00400000-00401000 r-xp 00000000 b3:02 835661                    /home/root/tests/user_example
00600000-00601000 r--p 00000000 b3:02 835661                    /home/root/tests/user_example
00601000-00602000 rw-p 00001000 b3:02 835661                    /home/root/tests/user_example
7faefc3c0000-7faefc555000 r-xp 00000000 b3:02 1566788           /lib/libc-2.25.so
7faefc555000-7faefc754000 ---p 00195000 b3:02 1566788           /lib/libc-2.25.so
7faefc754000-7faefc758000 r--p 00194000 b3:02 1566788           /lib/libc-2.25.so
7faefc758000-7faefc75a000 rw-p 00198000 b3:02 1566788           /lib/libc-2.25.so
7faefc75a000-7faefc75e000 rw-p 00000000 00:00 0
7faefc75e000-7faefc782000 r-xp 00000000 b3:02 1566402           /lib/ld-2.25.so
7faefc974000-7faefc977000 rw-p 00000000 00:00 0
7faefc97b000-7faefc97f000 rw-s 00000000 00:06 57418             /dev/pseudo_lock/p1
7faefc97f000-7faefc981000 rw-p 00000000 00:00 0
7faefc981000-7faefc982000 r--p 00023000 b3:02 1566402           /lib/ld-2.25.so
7faefc982000-7faefc984000 rw-p 00024000 b3:02 1566402           /lib/ld-2.25.so
7ffe40a4b000-7ffe40a6c000 rw-p 00000000 00:00 0                 [stack]
7ffe40b90000-7ffe40b93000 r--p 00000000 00:00 0                 [vvar]
7ffe40b93000-7ffe40b95000 r-xp 00000000 00:00 0                 [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0         [vsyscall]
```

Cache Pseudo-Locking performance

Testing interface

There is no instruction to query if provided physical address is present in cache.

Platforms have hardware performance monitoring mechanisms. Fine grained control possible in kernel (interrupts and hardware prefetchers disabled).

- MEM_LOAD_UOPS_RETIRED. L2_HIT
- MEM_LOAD_UOPS_RETIRED. L2_MISS

New debugfs directory for each pseudo-locked region.

- /sys/kernel/debug/resctrl/NAME

debugfs file *pseudo_lock_measure* triggers measurement, data captured in tracepoints.

- Count cache hits and misses while reading at cache line granularity from pseudo-locked memory. Tracepoints: *pseudo_lock_l2* and *pseudo_lock_l3.*

Test if memory is in the cache

```
# :> /sys/kernel/debug/tracing/trace
# echo 1 > /sys/kernel/debug/tracing/events/resctrl/pseudo_lock_l2/enable
# echo 2 > /sys/kernel/debug/resctrl/p1/pseudo_lock_measure
# echo 0 > /sys/kernel/debug/tracing/events/resctrl/pseudo_lock_l2/enable
# cat /sys/kernel/debug/tracing/trace
# tracer: nop
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
 pseudo_lock_mea-6992  [002] ....  6339.033465: pseudo_lock_l2: hits=4096 miss=0
```

256KB/64bytes =
4096 cache lines

User space memory access latency

Goal: Compare latency of reading pseudo-locked memory region to latency of reading malloc() (with mlockall()) region of same size.

Measurements taken using system's Time-stamp Counter (TSC) – also referred to as *cycles.*

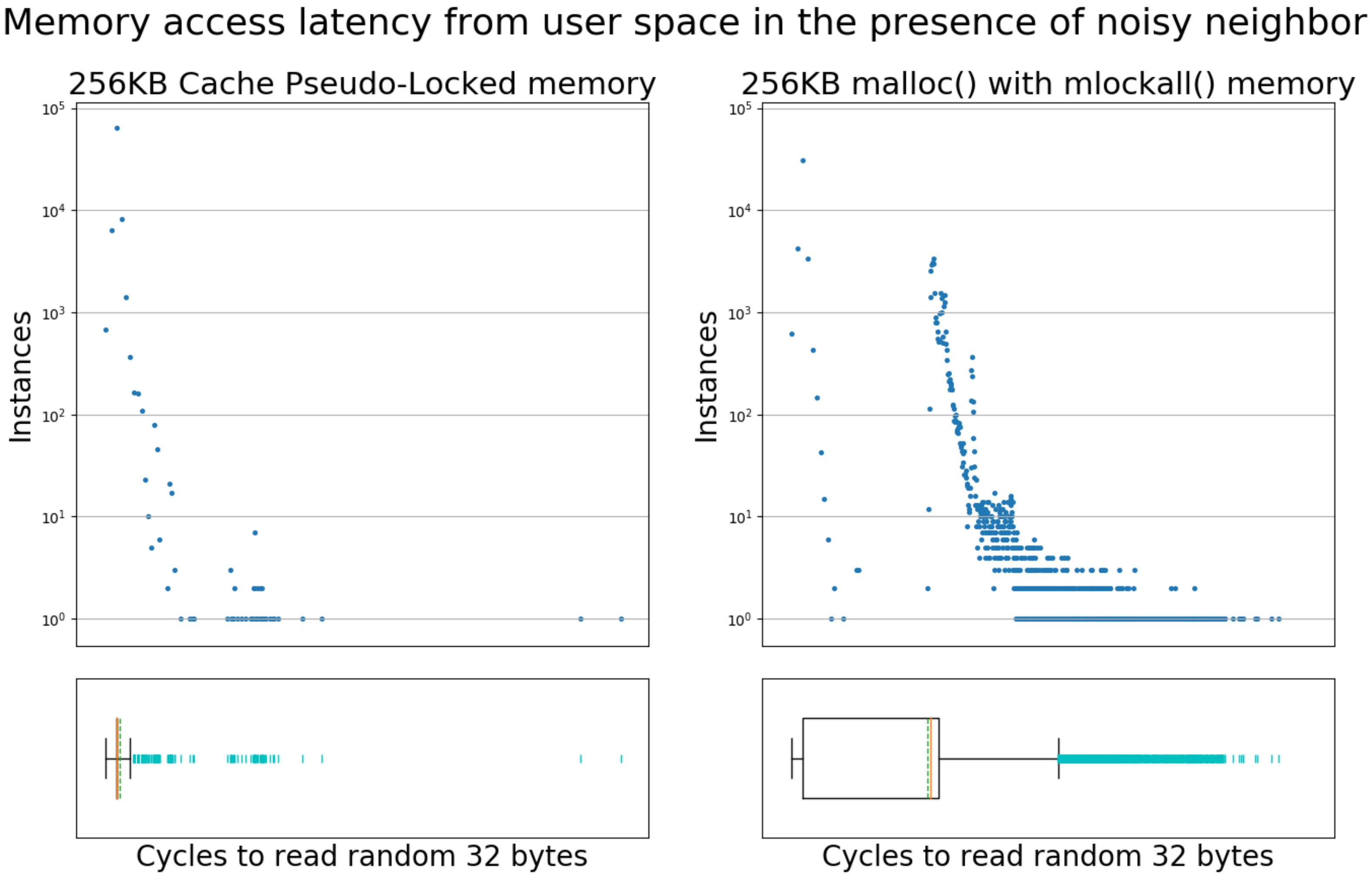
Non Real-Time kernel with no optimizations to reduce jitter.

The test

- One measurement = number of cycles to read random 32 bytes from memory region
- One test iteration = (mem_size / 32) measurements, sleep for 2 seconds
- 10 test iterations
- With noisy neighbor:

```
stress-ng -C 10 --taskset 2 --cache-level 2 –aggressive –t 0
```

User space latency results



- Significantly less variability in latency experienced by task using pseudo-locked memory.
- Median Cache Pseudo-Locked memory access latency is ~7 times lower than median malloc() memory access latency. (Q3 ~8 times lower, 99th percentile ~38 times lower).

DISCLAIMER: Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance measured by those tests.
Any difference in system hardware or software design or configuration may affect actual performance

Current status and Future work

Current Status and Future work

Current Status

- CAT supported since Linux v4.10.
- Cache Pseudo-Locking support will be in Linux v4.19.

Future work

- Restore of Cache Pseudo-Locked regions on detect of WBINVD.
- Use CLFLUSH/CLFLUSHOPT as cache clearing instruction instead of WBINVD.
- Research the potential of including page tables into pseudo-locked region.
- Simpler techniques to relocate instructions to pseudo-locked memory.

More information

- CAT and Cache Pseudo-Locking forms part of Intel® Resource Director Technology framework:

	Cache	Memory Bandwidth
Monitoring	Cache Monitoring Technology (CMT)	Memory Bandwidth Monitoring (MBM)
Allocation	Cache Allocation Technology (CAT)	Memory Bandwidth Allocation (MBA)

- https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html
- Linux support of RDT documented in kernel source *Documentation/x86/intel_rdt_ui.txt*

Questions?
Thank you!

reinette.chatre@intel.com