Strange Kernel Performance Changes?

Cache Alignment Matters

Feng Tang
Intel Linux Kernel Team

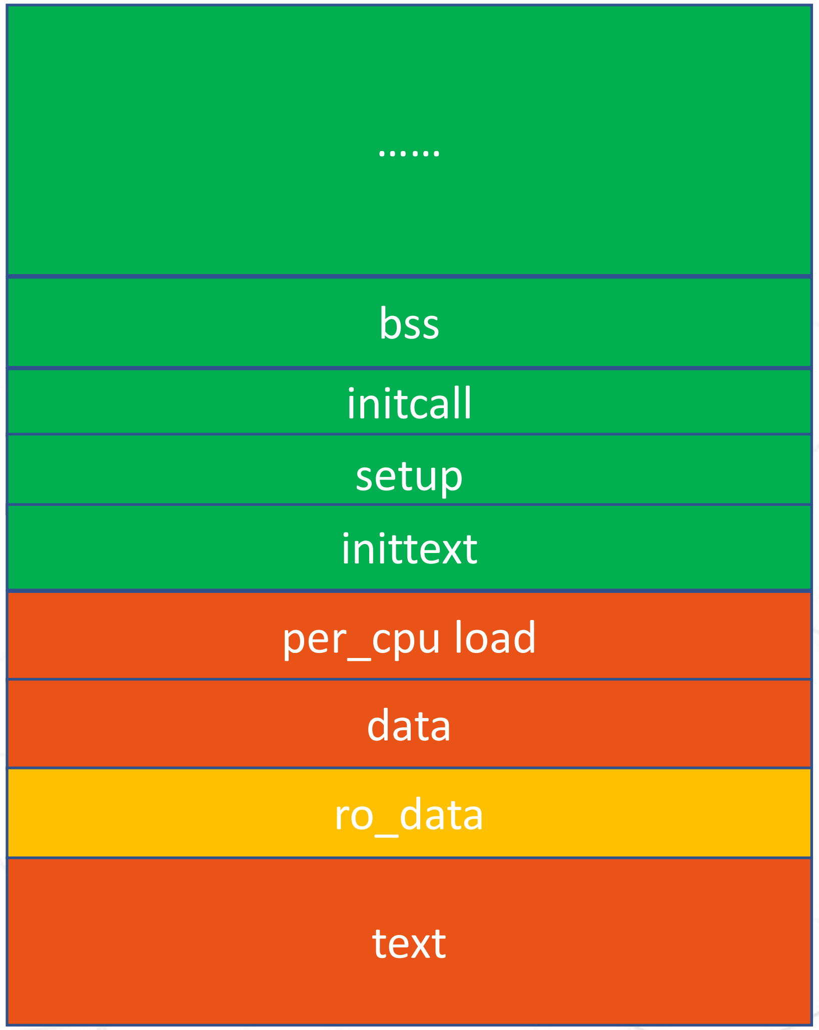
LINUX PLUMBERS CONFERENCE > September 20-24, 2021

Background

- 0Day (kernel test robot) keeps testing kernel performance and reporting regressions and improvements
- Recently, there are many **Strange** cases, which are hard to explain as bisected culprit commits seem to have nothing to do with the benchmark
- Kernel developers including Linus suspected and even challenged the reports: “What?” “Why it matters?”
- Goal: understand and explain them, try to mitigate (make everyone’s life easier)
- Hints and ideas are welcome and appreciated!

LINUX PLUMBERS CONFERENCE September 20-24, 2021

Kernel Sections Layout



```
0000000000000000 D __per_cpu_start
000000000002e000 D __per_cpu_end
ffffffff81000000 T _stext
ffffffff81000000 T _text
ffffffff81e011b7 T _etext
ffffffff82000000 R __start_rodata
ffffffff8249c000 D __end_rodata
ffffffff82600000 D _sdata
ffffffff82876840 D _edata
ffffffff82cc1000 D __init_begin
ffffffff82cc1000 D __per_cpu_load
ffffffff82cef000 T _sinittext
ffffffff82d5663f T _einittext
ffffffff82f03390 T __initcall_start
ffffffff82f03e88 T __initcall_end
ffffffff82f1e000 R __init_end
ffffffff82f2a000 B __bss_start
ffffffff83400000 B __bss_stop
ffffffff8342c000 B _end
```

System.map

Kernel Section Layout -II

- Text/Data sections layout

Text	A.text	B.text	C.text	D.text	E.text	F.text	G.text
Data	A.data	B.data	C.data	D.data	E.data	F.data	G.data

- Link order matters (from Makefile)

init	arch/sub-arch	kernel	mm	fs	security	block	drivers	net

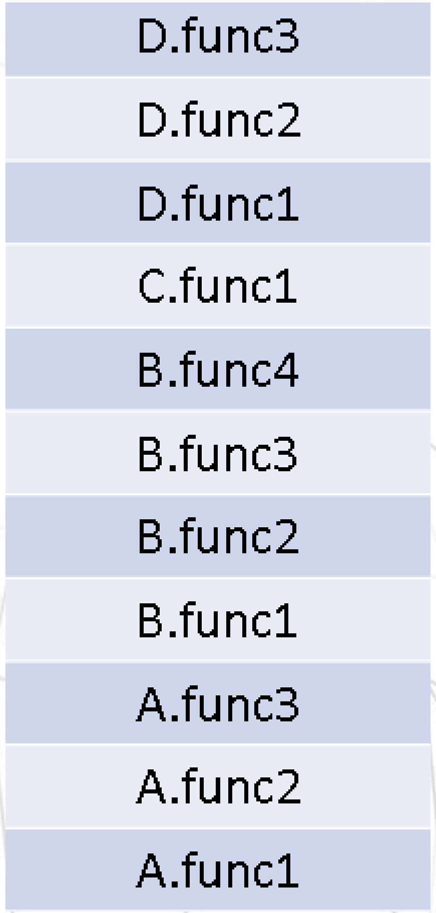
Cache Alignment Matters

Most of them are caused unnoticeably by underlying cache alignment changes:

- Text (function) alignment
- Data alignment (false sharing)
- HW cache prefetchers
    - Adjacent cache lines prefetch (2N, 2N+1)
    - L2 cache prefetcher

Text Alignment

- Kernel functions are all linked together compactly
- One line of code change may cause changes to the whole kernel text/function's alignment
- The earlier a .o get linked, the more parts it can affect
- Can be explained, but hard to be solved
- Kconfig or compiler change can greatly affect the result
- Examples
  - [LKP] Re: [mm] fd4d9c7d0c: stress-ng.switch.ops_per_sec -30.5% regression
  - [mm/hugetlb] c77c0a8ac4: will-it-scale.per_process_ops 15.9% improvement



LINUX PLUMBERS CONFERENCE September 20-24, 2021

Case Study – Text Alignment

- A one-line mm fix patch cause 30.6% regression for stress-ng.switch case
- change in kmem_cache_alloc_bulk()

  c->tid = next_tid(c->tid);

- 16 more bytes in binary for the function

  "49 83 40 08 01   addq $0x1,0x8(%r8)“

- The change is gone with forced function alignment

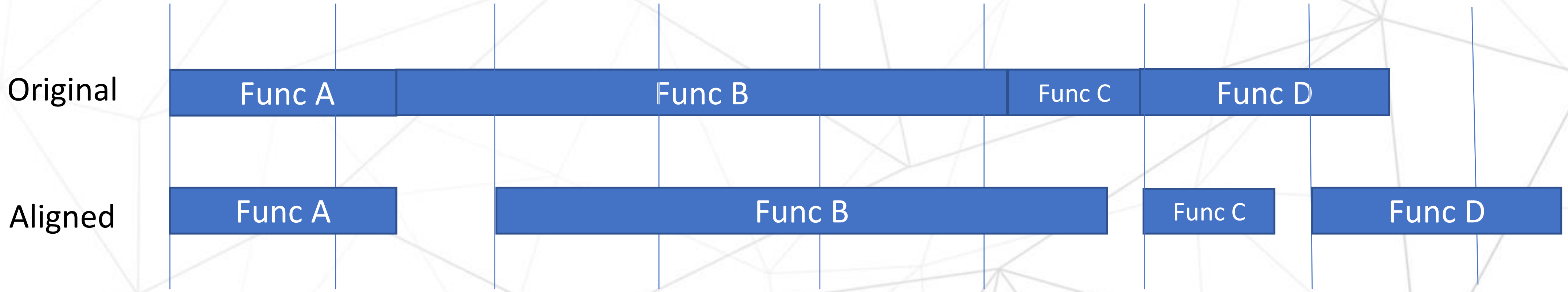
```
old map:
    ffffffff812a1880 T kmem_cache_alloc_bulk
    ffffffff812a1a80 t kmalloc_large_node
    ffffffff812a1b10 t calculate_sizes
    ffffffff812a1eb0 t store_user_store
    ffffffff812a1f20 t poison_store
    ffffffff812a1f90 t red_zone_store
    ffffffff812a2000 t order_store

new map:
    ffffffff812a1880 T kmem_cache_alloc_bulk
    ffffffff812a1a90 t kmalloc_large_node
    ffffffff812a1b20 T __kmalloc_node     ---> relocated
    ffffffff812a1e40 t calculate_sizes
    ffffffff812a21e0 t store_user_store
    ffffffff812a2250 t poison_store
    ffffffff812a22c0 t red_zone_store
    ffffffff812a2330 t order_store
```

Mitigation (Debug) – Text Alignment

Force all function start address aligned on 64 bytes (merged)

- A black box check which we are not 100% sure
- Kconfig option CONFIG_DEBUG_FORCE_FUNCTION_ALIGN_64B
- Much less report after 0Day enabled it
- Why not default on? – 10% more kernel size, more TLB usage

```
ifdef CONFIG_DEBUG_FORCE_FUNCTION_ALIGN_64B
KBUILD_CFLAGS += -falign-functions=64
endif
```



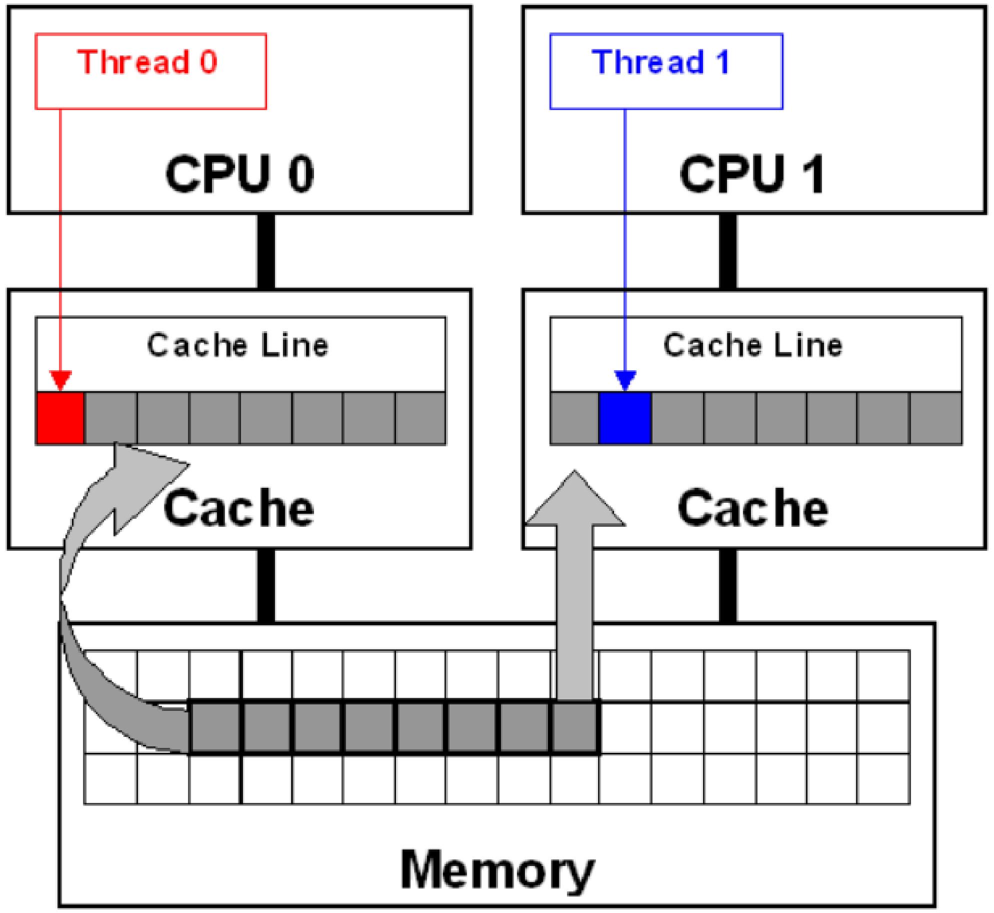
Data Alignment

- Key is the Cache False Sharing
- Data is more complex than text
  - Static Layout
    - .data section
    - specific sections like (percpu)
  - Dynamic Allocation: kmalloc/slab/vmalloc
- Debug Methods
  - perf-c2c
  - pahole
  - add padding

LINUX PLUMBERS CONFERENCE September 20-24, 2021

Cache False Sharing

- Data loaded from memory to cache on cacheline granularity
- Multiple CPUs access data in one cache line
  - all read → Fine
  - one write → Bad
- Try to separate them in hot data structure

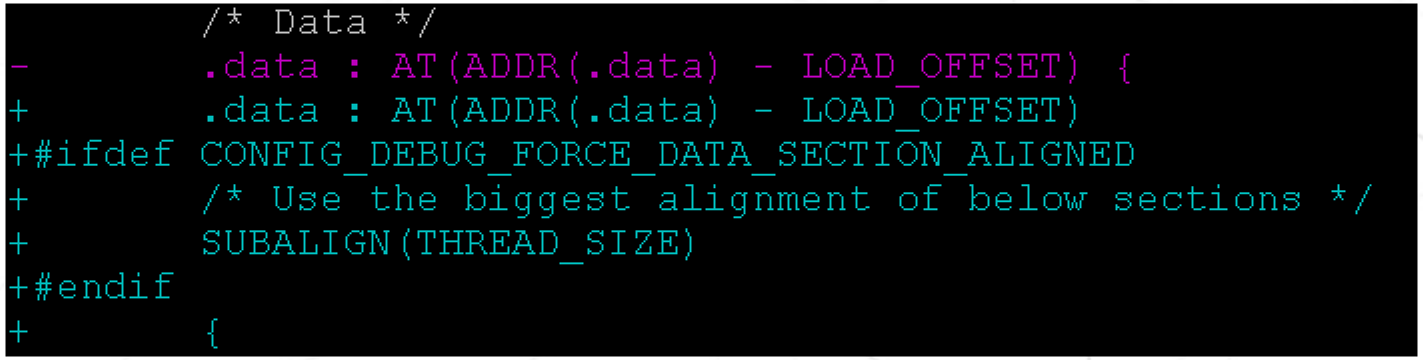


https://software.intel.com/content/www/us/en/develop/articles/avoiding-and-identifying-false-sharing-among-threads.html

LINUX PLUMBERS CONFERENCE September 20-24, 2021

Mitigation (Debug) – Data Alignment

- Force data sections of every .o file aligned (patch posted)
  - Change in linker script vmlinux.lds.S
  - Debug only due to huge size increase



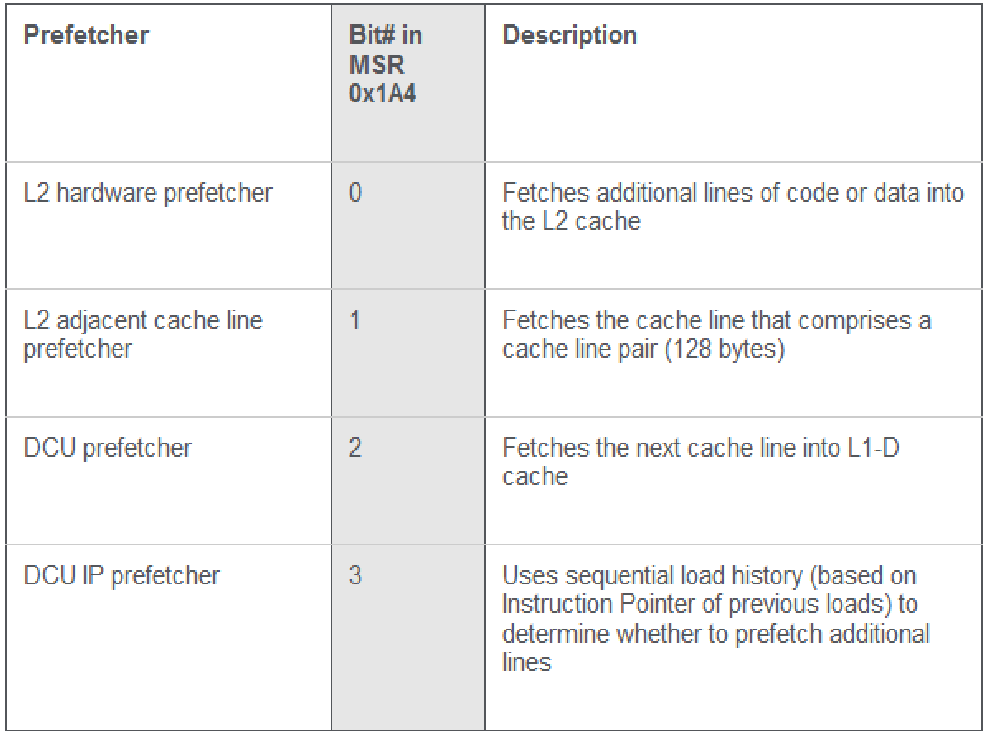
```
        /* Data */
-       .data : AT(ADDR(.data) - LOAD_OFFSET) {
+       .data : AT(ADDR(.data) - LOAD_OFFSET)
+#ifdef CONFIG_DEBUG_FORCE_DATA_SECTION_ALIGNED
+       /* Use the biggest alignment of below sections */
+       SUBALIGN(THREAD_SIZE)
+#endif
+       {
```

- Per-CPU data - Add debug allocation macros to force all percpu-data address aligned
- Kmalloc/slab - Force alignment (slab has parameter)

LINUX PLUMBERS CONFERENCE September 20-24, 2021

HW Cache Prefetcher

- Most platforms have them ON by default as being helpful generally
- Transparent to SW programmer
- Accuracy affects bus BW hugely
- May vary on different generations as the algorithm evolves
- Consider them if SW debugging can’t help
- Real cases related to the first 2 types

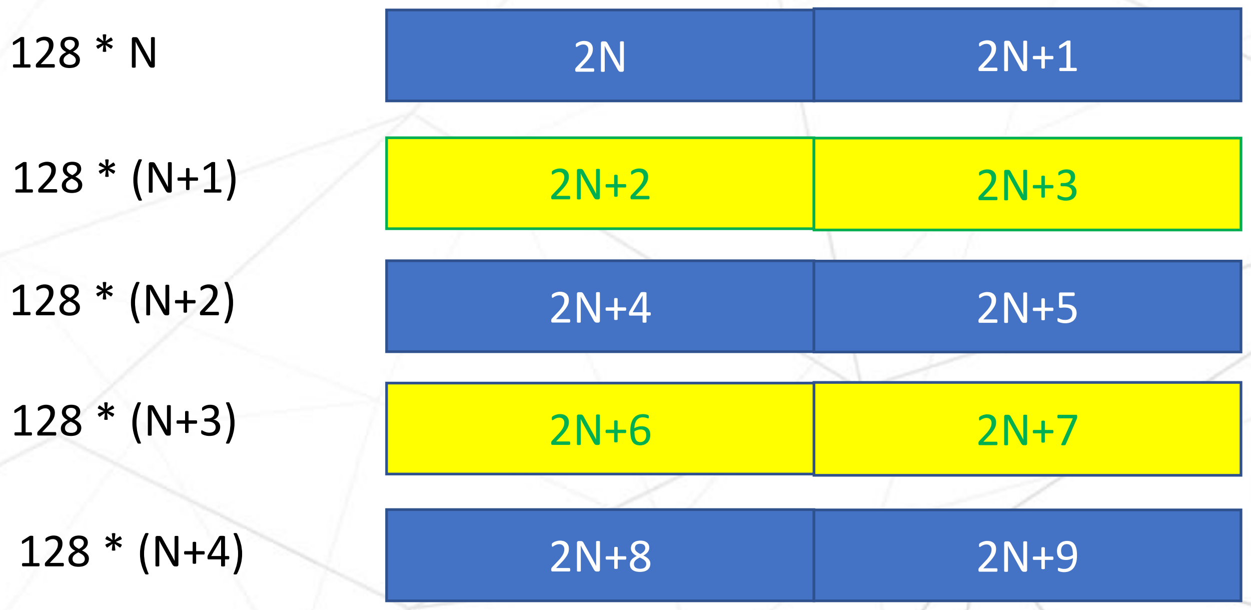


Prefetcher	Bit# in MSR 0x1A4	Description
L2 hardware prefetcher	0	Fetches additional lines of code or data into the L2 cache
L2 adjacent cache line prefetcher	1	Fetches the cache line that comprises a cache line pair (128 bytes)
DCU prefetcher	2	Fetches the next cache line into L1-D cache
DCU IP prefetcher	3	Uses sequential load history (based on Instruction Pointer of previous loads) to determine whether to prefetch additional lines

https://software.intel.com/content/www/us/en/develop/articles/disclosure-of-hw-prefetcher-control-on-some-intel-processors.html

Adjacent Cache Lines Prefetch

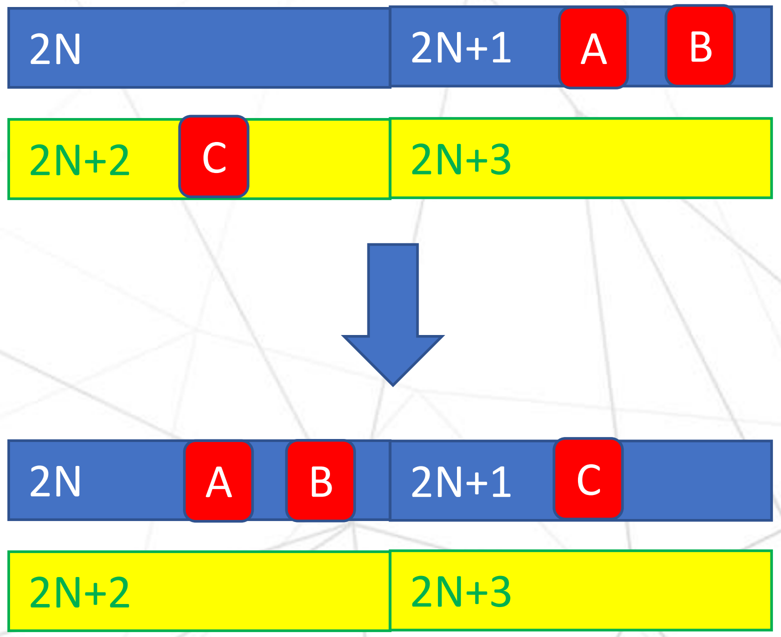
- When one cache line is accessed and fetched, its adjacent cache line will be fetched too
- 64B cache line extended into 128B ‘fat cacheline’
- Can not be detected by tools like perf-c2c



LINUX PLUMBERS CONFERENCE September 20-24, 2021

Case Study – HW Prefetcher

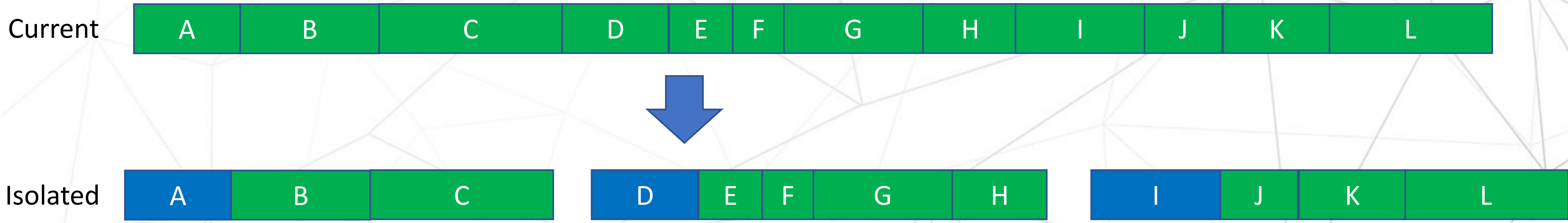
- Patch removing a ‘struct page_counter’ from ‘struct mem_cgroup’, causes -22.7% regression for will-it-scale/page_fault2
- Commit does have relation with the test case, looks to be alignment related
- 3 hot members(A, B, C) sit in 2 adjacent cache lines which were not in one 128B trunk, but were pulled into one by the commit.
- “False sharing” of 2 cachelines
- Solution – sperate them into different 128B trunks



LINUX PLUMBERS CONFERENCE September 20-24, 2021

Mitigation – Selective Isolation

- Goal: Make kernel performance more stable (Less surprise)
- Chose N(10~20) .o files, add 64/128B alignment to one function and one data of them (modules A/D/I below)
- Divide kernel into N independent capsules like capsules in a big ship - one capsule changed/broken won't affect others
- Rule: select more in critical and early modules
- It won't hurt, with minimal increase of kernel size



Todos

- Upstream the mitigation and debug patches
- Extend perf-c2c tool to cover adjacent cache line prefetch
- Explore more about HW prefetcher
- Check cases which are still not explained

Q&A

Thank You!