OSPM19

CFS wakeup path and ARM big.LITTLE/DynamIQ

Load spreading vs consolidating on Last Level Cache (LLC)

Dietmar Eggemann

20.05.2019

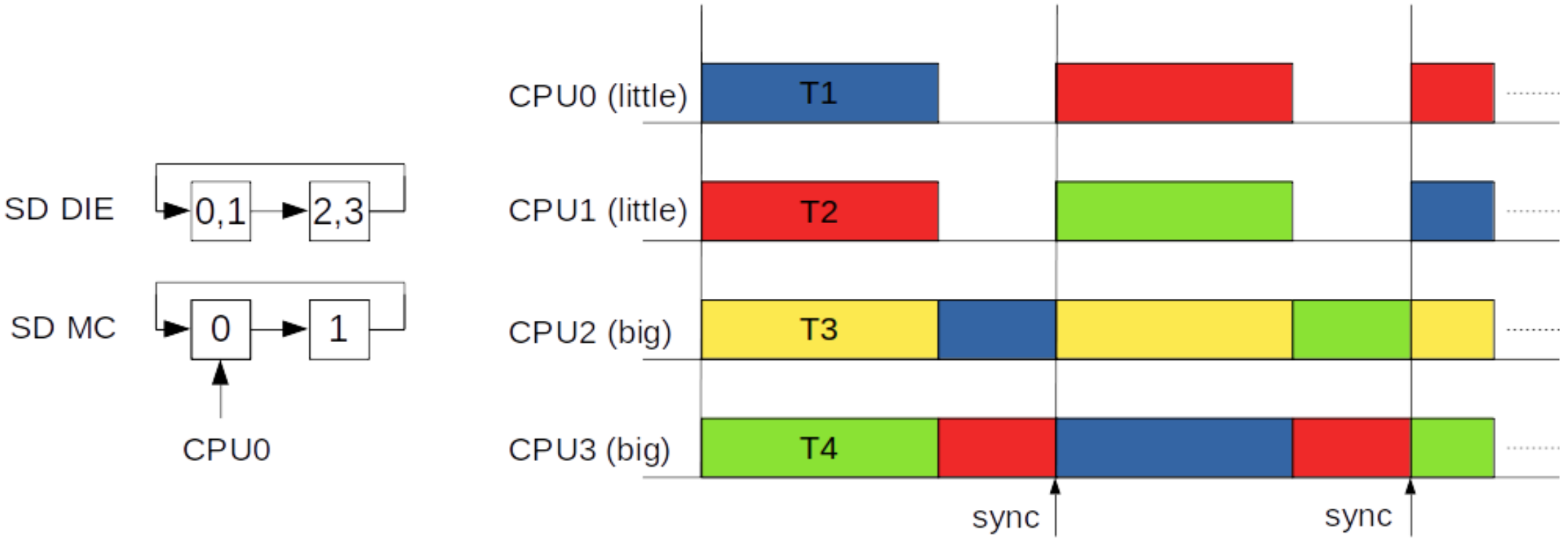
big.LITTLE (asymetric CPU capacity) specific CFS wakeup code

- find_energy_efficient_cpu()
  - EAS is only enabled for asymmetric CPU capacity topologies
  - Only if system is not over-utilized
- wake_cap()
  - Disable WAKE_AFFINE in the case task doesn't fit in the capacity of waking and previous CPU

```
want_affine = !wake_wide(p) && !wake_cap(p, cpu, prev_cpu) && cpumask_test_cpu(cpu, &p->cpus_allowed)
```

Outstanding issue - *1 task per CPU* workload

- big.LITTLE (asymmetric CPU capacity) specific CFS load-balancing code
  - Misfit task
  - Vincent Guittot - sched/fair: fix 1 task per CPU

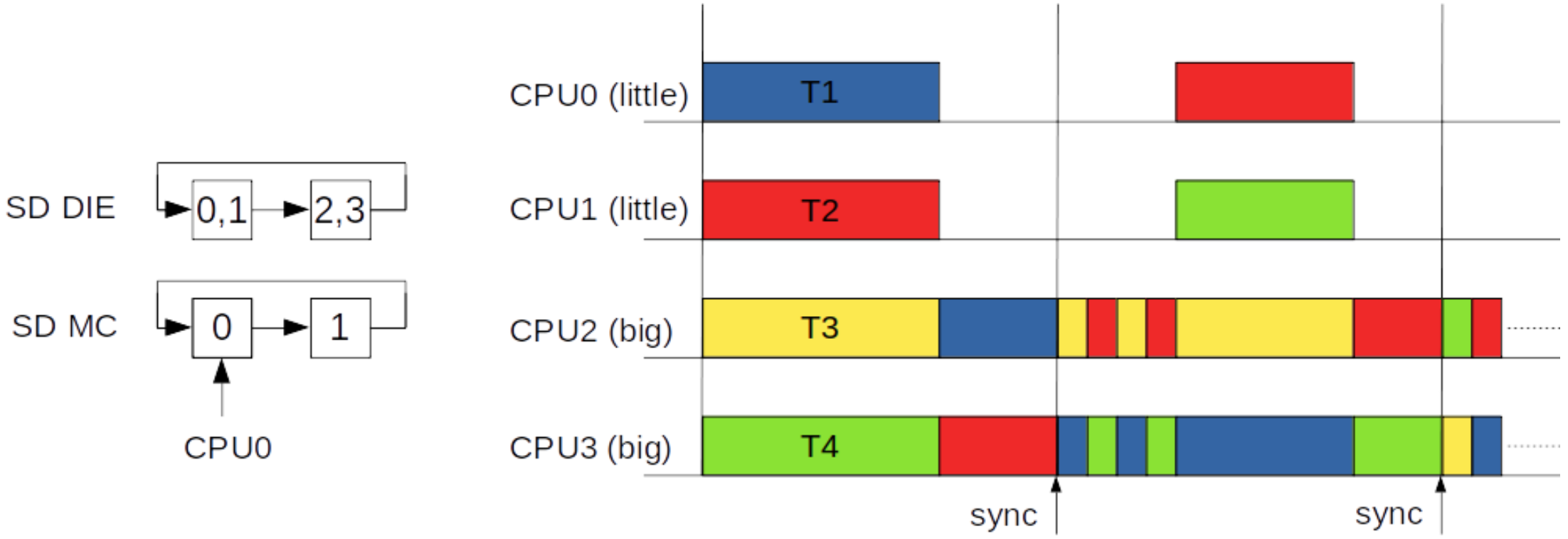


Ideal scheduling on big.LITTLE

arm

Outstanding issue - *1 task per CPU* workload (2)

- Prev_cpu biasing towards big CPUs
- System is over-utilized and wakee/waker wakee_flips and wake_cap() have no effect
- Big and Little CPUs form Sched Domain (SD) DIE level sched groups



Real scheduling on big.LITTLE

arm

Load spreading vs consolidating on Last Level Cache (LLC)

1. Wakee/waker wakee_flips
   - Commit 62470419e993 ("sched: Implement smarter wake-affine logic") (June 2013)
   - Client/server, worker/dispatcher, interrupt source (**M:N waker/wakee** relationship)

with T1->wakee_flips <= T2->wakee_flips

if T1->wakee_flips > llc_size && T2->wakee_flips > (T1->wakee_flips * llc_size) -> spread load

```
AsyncTask #2-9725 [007]  sched_wake_wide:  T1=AsyncTask #2 9725  T2=AsyncTask #2 9729  T1->wakee_flips=11  T2->wakee_flips=22
AsyncTask #2-9725 [007]  sched_wake_wide:  T1=AsyncTask #2 9725  T2=AsyncTask #2 9731  T1->wakee_flips=12  T2->wakee_flips=13
AsyncTask #2-9725 [007]  sched_wake_wide:  T1=AsyncTask #2 9725  T2=AsyncTask #2 9728  T1->wakee_flips=13  T2->wakee_flips=15
AsyncTask #2-9725 [007]  sched_wake_wide:  T2=AsyncTask #2 9725  T1=AsyncTask #2 9727  T2->wakee_flips=14  T1->wakee_flips=13
AsyncTask #2-9725 [007]  sched_wake_wide:  T1=AsyncTask #2 9725  T2=AsyncTask #2 9730  T1->wakee_flips=15  T2->wakee_flips=15
AsyncTask #2-9725 [007]  sched_wake_wide:  T1=AsyncTask #2 9725  T2=AsyncTask #2 9113  T1->wakee_flips=16  T2->wakee_flips=23
AsyncTask #2-9725 [007]  sched_wake_wide:  T2=AsyncTask #2 9725  T1=AsyncTask #2 9726  T2->wakee_flips=17  T1->wakee_flips=13
AsyncTask #2-9725 [007]  sched_wake_wide:  T2=AsyncTask #2 9725  T1=AsyncTask #2 9732  T2->wakee_flips=18  T1->wakee_flips=8
```

Sync of Geekbench Multi-Core testcase (HDR)

2. wake_cap()

- Both are not helping with *1 task per CPU* workload issue
  - Waker and wakee change between consecutive sync/wakeups and CPUs involved are big CPUs

arm

Possible solution - wake queue length for wake wide decision

- **pthread_barrier_init**(…, **unsigned int count**);
- count argument … # of threads that must call **pthread_barrier_wait()** before any of them successfully return from the call
- Kernel implementation – FUTEX WAIT/WAKE (wake_q_add{_safe}(), wake_up_q())
- Propagate wake queue length (count) with the task to the CFS select task function and use it in wake_wide() so it can be compared with LLC size
- Other Locking mechanisms using wake queues (e.g. R/W semaphores, POSIX and SYSV message queues)

Possible solution – wake queue length for wake wide decision (2)

```
struct task_struct {
struct task_struct *last_wakee;
+ unsigned int nr_wakee_siblings;

struct wake_q_head {
struct wake_q_node **lastp;
+ int count;

#define DEFINE_WAKE_Q(name) \
- struct wake_q_head name = { WAKE_Q_TAIL, &name.first }
+ struct wake_q_head name = { WAKE_Q_TAIL, &name.first, 0 }

static bool __wake_q_add(struct wake_q_head *head, struct task_struct *task)
head->lastp = &node->next;
+ head->count++;

void wake_up_q(struct wake_q_head *head)
task->wake_q.next = NULL;
+ task->nr_wakee_siblings = head->count;
```

```
static int wake_wide(struct task_struct *p)
int factor = this_cpu_read(sd_llc_size);
+ int ret = 1;
+ if (p->nr_wakee_siblings >= factor)
+     goto out;

if (master < slave)
swap(master, slave);
if (slave < factor || master < slave *
factor)
-     return 0;
-return 1;
+     ret = 0;
+out:
+p->nr_wakee_siblings = 1;
+return ret;
```

Patch from Brendan Jackman (Oct 2017): https://lore.kernel.org/lkml/87efqln7xs.fsf@arm.com

arm

Use case - Benchmarks, e.g. Geekbench

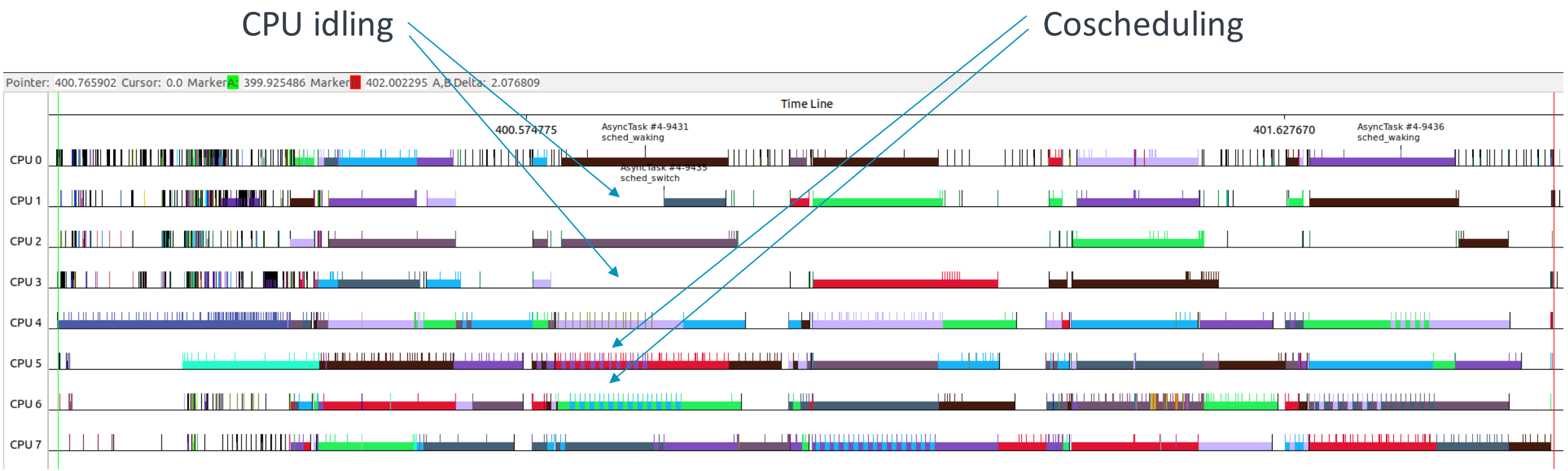
- 25 testcases (Crypto, Integer, Floating Point, Memory) for Single- (SC) & Multi-Core (MC)
- Pixel 3 (Snapdragon 845, [CPU0-3] LITTLE [CPU4-7] big, DynIQ, Phantom SD

Geekbench SC & MC testcases

arm

Use case - Benchmarks, e.g. Geekbench (2)

- Pixel 3 (android-9.0.0_r0.73 (4.9.124-stable))

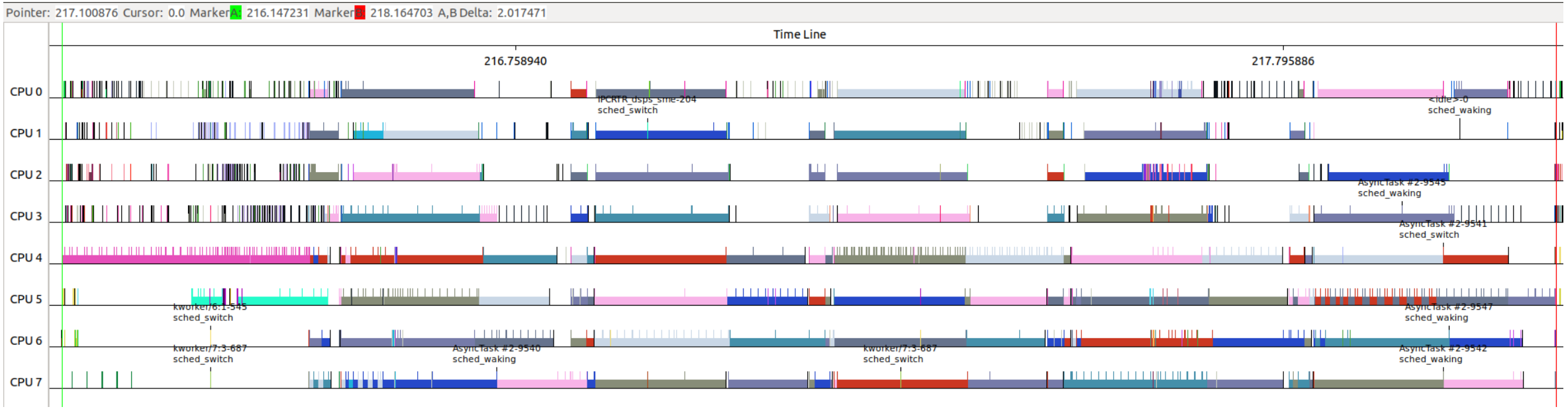


Geekbench MC testcase (Canny, Integer)

arm

Use case - Benchmarks, e.g. Geekbench (3)

- Pixel 3 (android-9.0.0_r0.73 (4.9.124-stable) + patch)

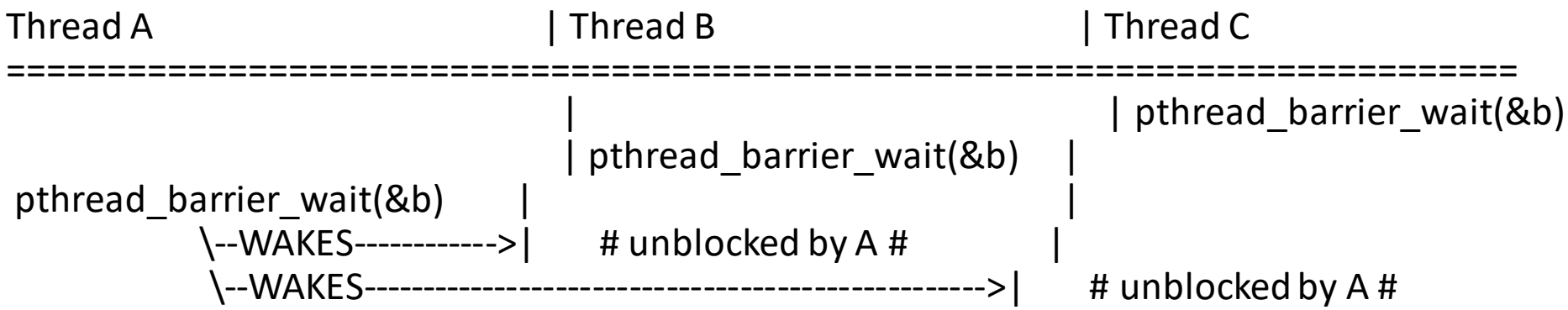


Geekbench MC testcase (Canny, Integer)

arm

Test tool - rt-app barrier event extension

- Current barrier event uses pthread_cond_wait()/pthread_cond_broadcast()
- For pthread_barrier_wait() use the same event + default barrier type in global section ?



pthread_barrier_wait() rt-app barrier event

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה