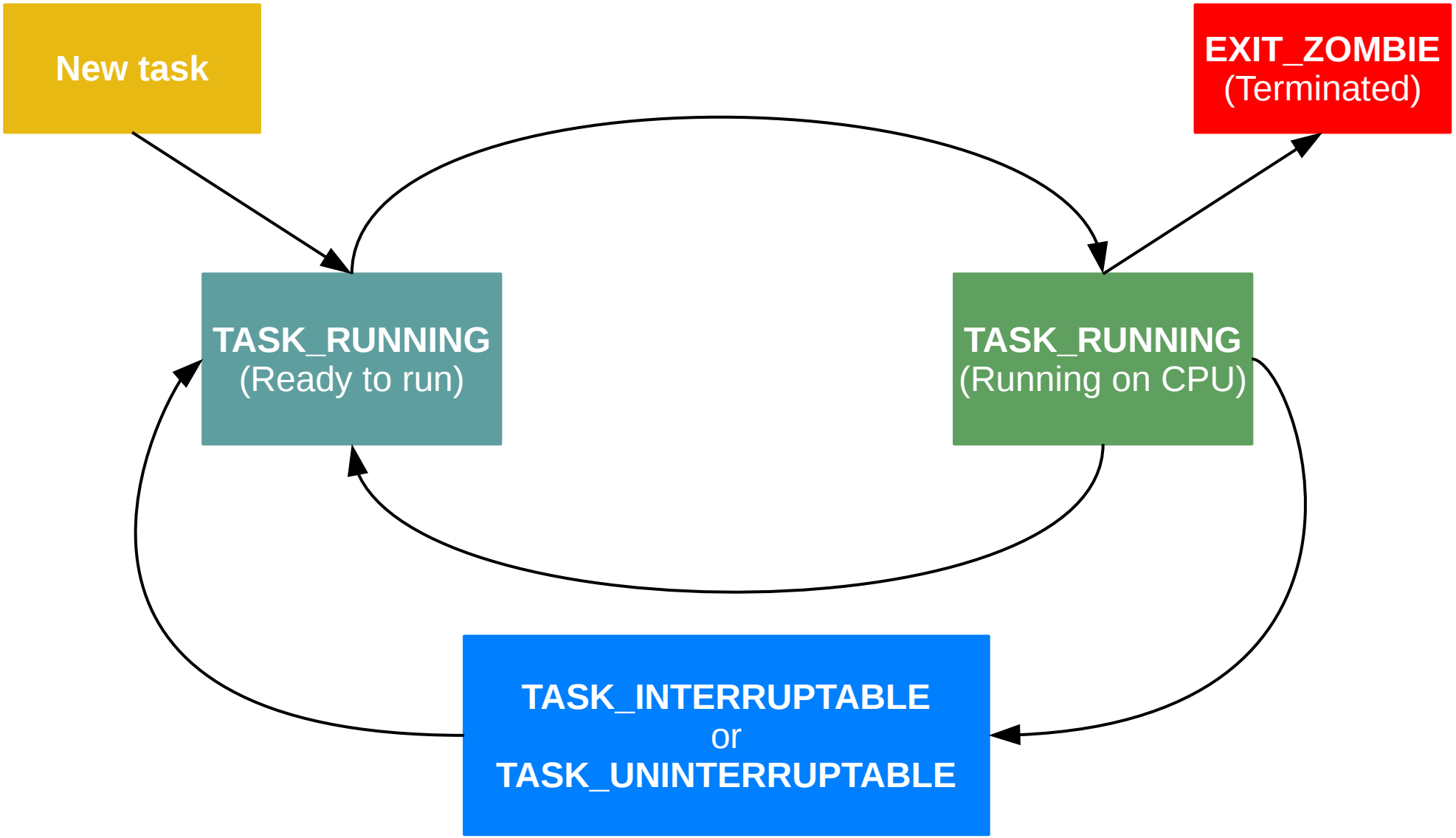
Case Study – Scheduler

Beijing Trace Training 2017

Gary Lin
Software Engineer, SUSE Labs
glin@suse.com

Scheduling

Task State Transition



Task States

```
#define TASK_RUNNING          0   /* R: Running                          */
#define TASK_INTERRUPTIBLE    1   /* S: Interruptible Sleep              */
#define TASK_UNINTERRUPTIBLE  2   /* D: Uninterruptible Sleep            */
                                  /*    (Do not process signals)         */
#define __TASK_STOPPED        4   /* T: Stopped (SIGSTOP, SIGTSTP)       */
#define __TASK_TRACED         8   /* t: Traced by a debugger             */
#define EXIT_ZOMBIE           16  /* Z: Zombie (EXIT STATE)              */
                                  /*    (waits for its parent)           */
#define EXIT_DEAD             32  /* X: Dead (EXT STATE)                 */
                                  /*    (is being removed from the       */
                                  /*     system)                         */
#define TASK_DEAD             64  /* x: Dead                             */
#define TASK_WAKEKILL         128 /* W: Received fatal signals           */
```

* See include/linux/sched.h.

Scheduling Class

- Abstract the scheduling policy of a scheduler
- "`struct sched_class`" defined in kernel/sched/sched.h

```
enqueue_task()
dequeue_task()
yield_task()
check_preempt_curr()
pick_next_task()
set_curr_task()
task_tick()
...
```

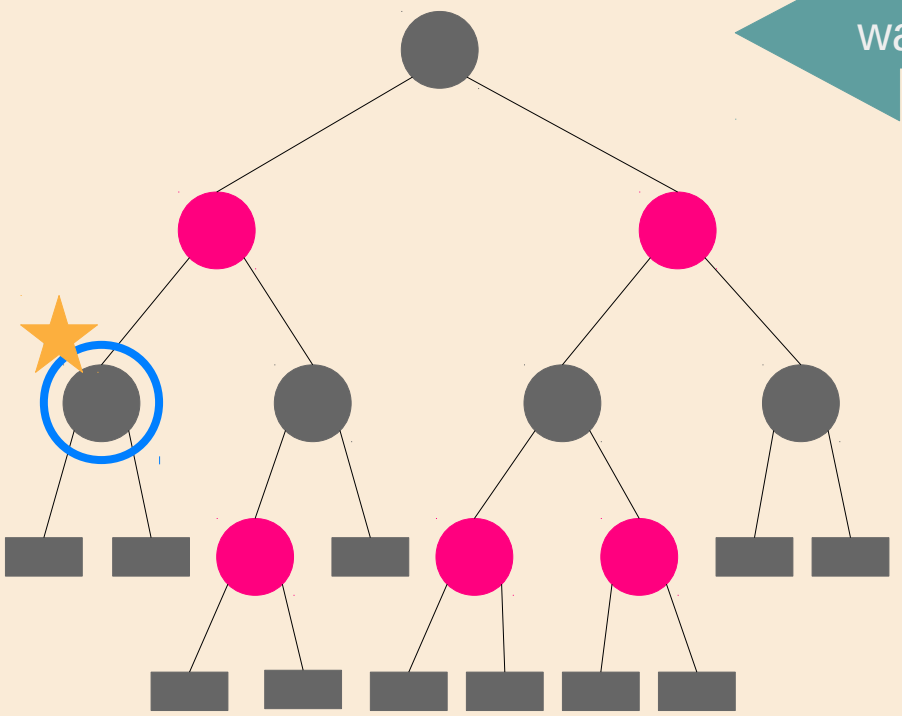
Scheduling Events

- **sched_wakeup_new:** waking up a new task (such as fork)
- **sched_wakeup:** waking up a task
- **sched_switch:** switching to another task
- **sched_migrate_task:** a task being migrated
- **sched_stat_runtime:** accounting runtime (time that the task is executing on a CPU)
- **sched_stat_wait:** accounting wait time (time that the task is runnable but not running)
- **sched_stat_iowait:** accounting iowait time (time that is not runnable due to waiting on IO to complete)
- **sched_stat_sleep:** accounting sleep time (time that the task is not runnable, including iowait)

Complete Fair Scheduler – Concept

CPU Runqueues

Runnable (TASK_RUNNING)



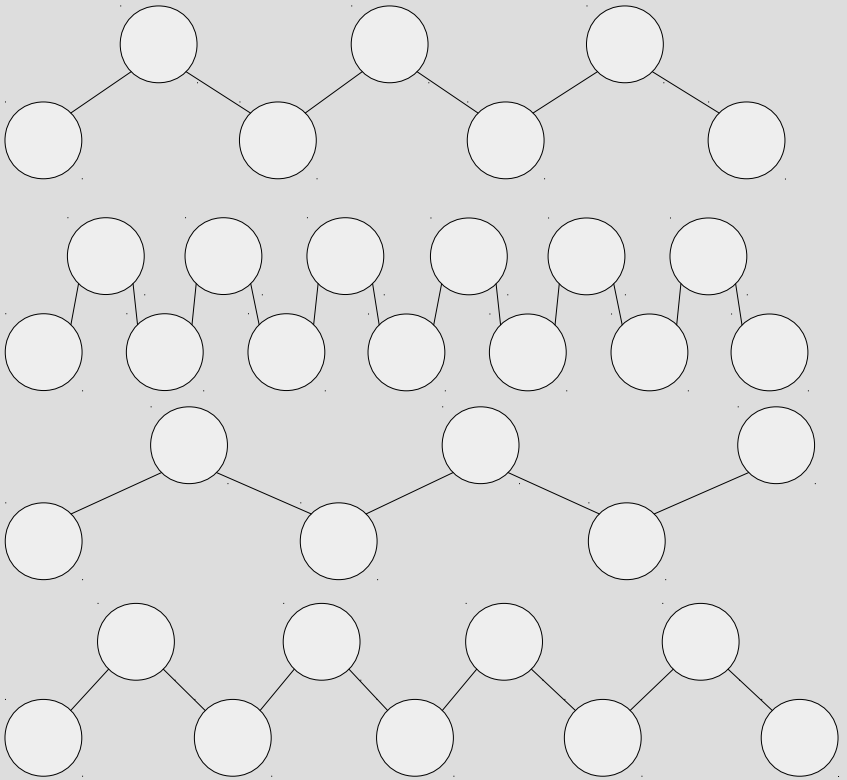
virtual runtime

small large

wake up

Not Runnable Tasks

Wait Queue, Wake Queue, softirq, Work Queue, etc.



Complete Fair Scheduler – Virtual Runtime

“In practice, the virtual runtime of a task is its runtime normalized to the total number of running tasks.”

– sched-design-CFS.txt

perf sched

perf sched

- A tool to trace/measure scheduler properties (latencies)
- Enable schedstats to get more information

```
# echo 1 > /proc/sys/kernel/sched_schedstats
```

perf sched Commands

“`perf sched` **`record`**” records the scheduling events into perf.data.

```
# perf sched record -- sleep 10
# perf sched record command
```

“`perf sched` **`latency`**” reports the per task scheduling latencies.

“`perf sched` **`map`**” prints a textual context-switching outline of workload.

“`perf sched` **`replay`**” simulates the workload.

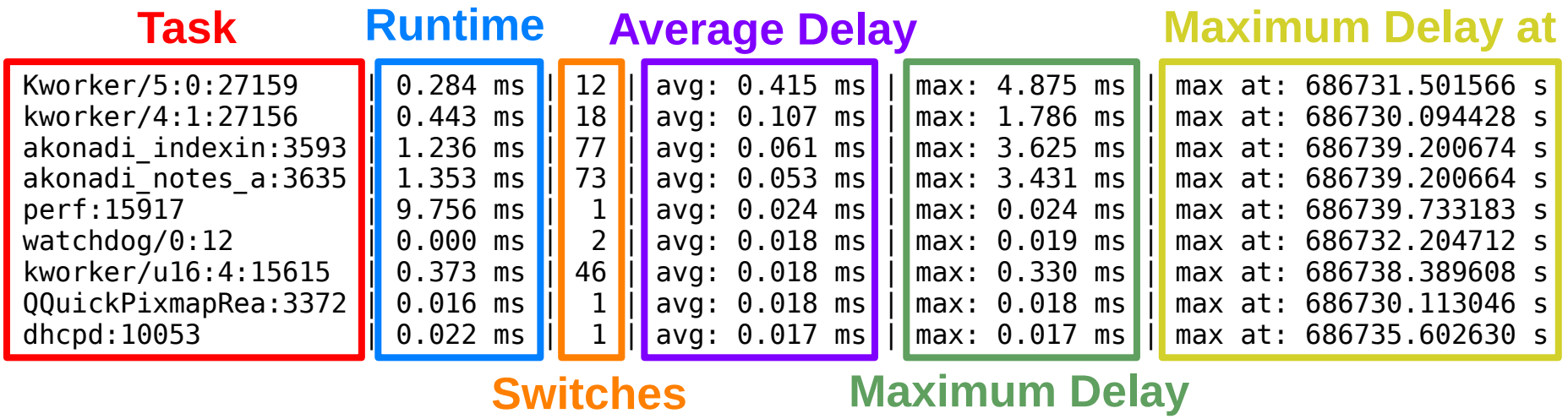
“`perf sched` **`script`**” shows a detailed trace of the workload.

“`perf sched` **`timehist`**” provides an analysis of scheduling events. (since 4.10)

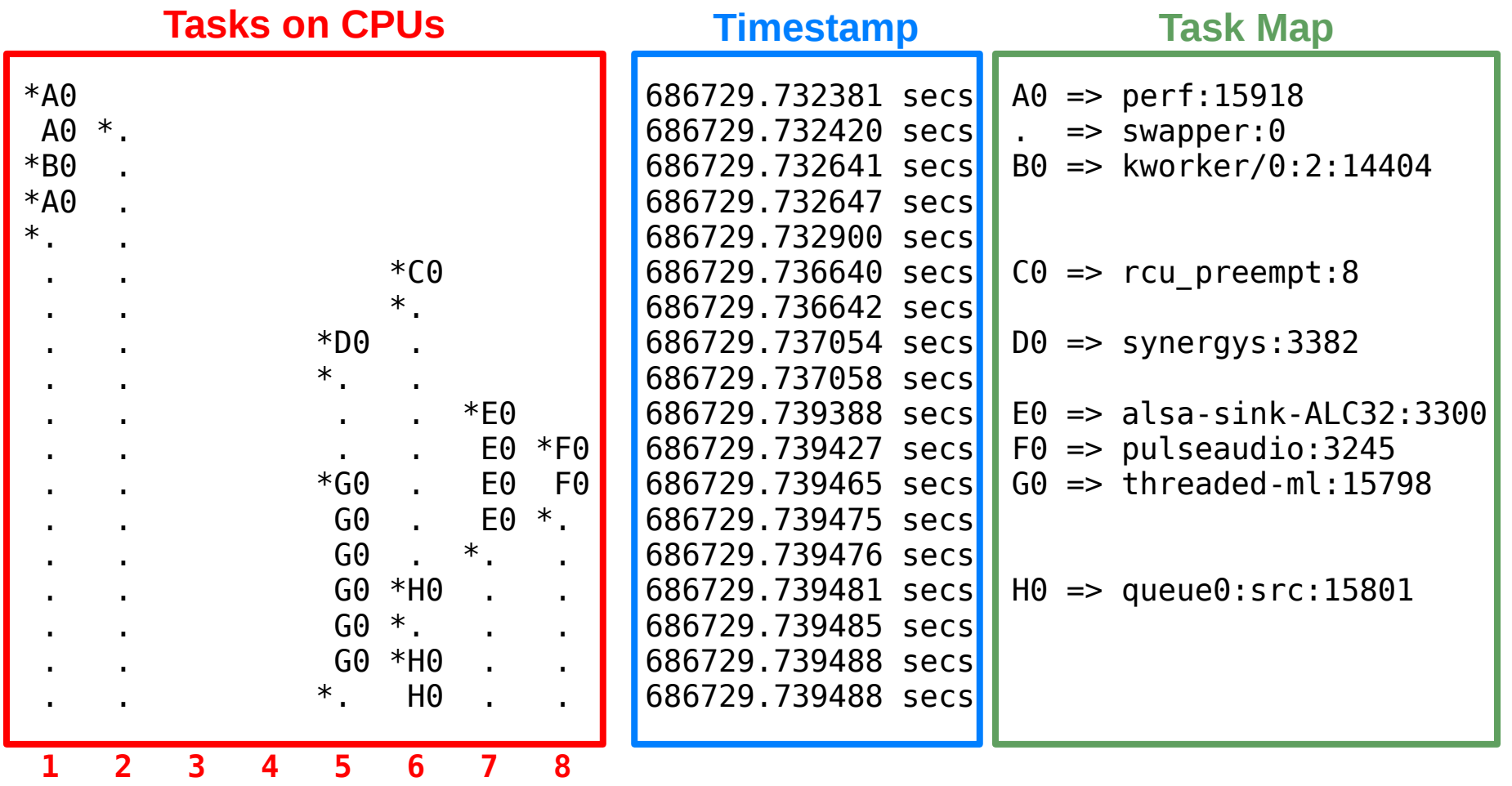
perf sched latency

- Latency:

```
time(sched_switch) - time(sched_wakeup)
time(sched_switch) - time(sched_wakeup_new)
```



perf sched map



* the CPU that had the event
. idle CPU

perf sched replay

```
# perf sched replay

run measurement overhead: 89 nsecs
sleep measurement overhead: 52308 nsecs
the run test took 1000008 nsecs
the sleep test took 1068112 nsecs
nr_run_events:        43620
nr_sleep_events:      44438
nr_wakeup_events:     22784
target-less wakeups:  4
task      0 (            swapper:         0), nr_events: 44251
task      1 (            swapper:         1), nr_events: 1
task      2 (            swapper:         2), nr_events: 1
task      3 (           kthreadd:         6), nr_events: 1
task      4 (           kthreadd:         7), nr_events: 5
task      5 (           kthreadd:         8), nr_events: 569
task      6 (           kthreadd:         9), nr_events: 1
task      7 (           kthreadd:        10), nr_events: 1
task      8 (           kthreadd:        11), nr_events: 1
```

perf sched script

Task	PID	CPU	Timestamp	Event	Event Arguments
swapper	0	[005]	686729.744866:	sched:sched_stat_sleep:	comm=queue1:src pid=158
swapper	0	[005]	686729.744867:	sched:sched_wakeup:	queue1:src:15802 [120]
swapper	0	[005]	686729.744868:	sched:sched_stat_wait:	comm=queue1:src pid=158
swapper	0	[005]	686729.744869:	sched:sched_switch:	swapper/5:0 [120] R ==>
queue1:src	15802	[005]	686729.744896:	sched:sched_stat_sleep:	comm=amarok pid=3352 de
queue1:src	15802	[005]	686729.744896:	sched:sched_wakeup:	amarok:3352 [120] succe
swapper	0	[002]	686729.744898:	sched:sched_stat_wait:	comm=amarok pid=3352 de
swapper	0	[002]	686729.744899:	sched:sched_switch:	swapper/2:0 [120] R ==>
queue1:src	15802	[005]	686729.744932:	sched:sched_stat_runtime:	comm=queue1:src pid=158
queue1:src	15802	[005]	686729.744946:	sched:sched_stat_sleep:	comm=audioPipe:src pid=
queue1:src	15802	[005]	686729.744946:	sched:sched_wakeup:	audioPipe:src:15805 [12
amarok	3352	[002]	686729.744946:	sched:sched_stat_runtime:	comm=amarok pid=3352 ru
swapper	0	[007]	686729.744947:	sched:sched_stat_wait:	comm=audioPipe:src pid=
amarok	3352	[002]	686729.744947:	sched:sched_switch:	amarok:3352 [120] S ==>
swapper	0	[007]	686729.744947:	sched:sched_switch:	swapper/7:0 [120] R ==>
queue1:src	15802	[005]	686729.744952:	sched:sched_stat_runtime:	comm=queue1:src pid=158
queue1:src	15802	[005]	686729.744953:	sched:sched_switch:	queue1:src:15802 [120]

perf sched timehist

```
# perf sched timehist -MVw
```

```
         time    cpu 012345678  task name          wait time sch delay  run time
                                 [tid/pid]          (msec)    (msec)    (msec)
------------- ------ ---------  ------------------ --------- --------- ---------
686729.732377 [0001]            perf[15917]                                        awakened: perf[15918]
686729.732381 [0000] i          <idle>             0.000     0.000     0.000
686729.732420 [0001]  s         perf[15917]        0.000     0.000     0.000
686729.732638 [0000]            perf[15918]                                        awakened: kworker/0:2
686729.732641 [0000] s          perf[15918]        0.000     0.004     0.260
686729.732647 [0000] s          kworker/0:2[14404] 0.000     0.003     0.005
686729.732900 [0000] s          sleep[15918]       0.005     0.000     0.253
686729.736636 [0005]            swapper                                            awakened: rcu_preempt
686729.736640 [0005]      i     <idle>             0.000     0.000     0.000
686729.736642 [0005]      s     rcu_preempt[8]     0.000     0.003     0.002
686729.737051 [0004]            swapper                                            awakened: synergys[33
686729.737054 [0004]     i      <idle>             0.000     0.000     0.000
686729.737058 [0004]     s      synergys[3382]     0.000     0.002     0.004
686729.739385 [0006]            swapper                                            awakened: alsa-sink-A
686729.739388 [0006]       i    <idle>             0.000     0.000     0.000
```

CPU Visual

Wakeup Events

i idle time
s scheduler event

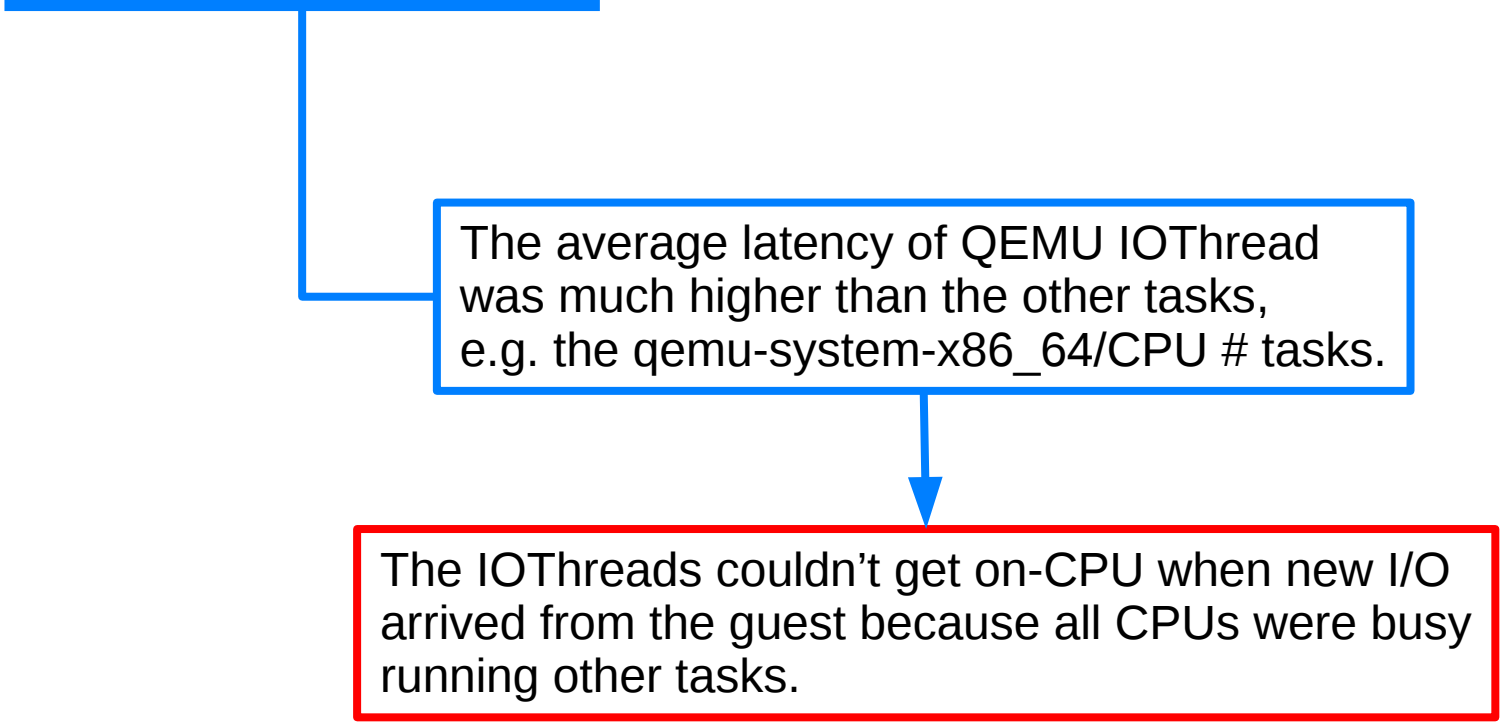
Cases from Matt Fleming

Case 1: Unexpected Long Scheduler Latency

“While tuning KVM for running SAP HANA I had to check that the Qemu IOthreads (tasks used for the Qemu guest's asynchronous I/O) were experiencing delays.”

Profile the tasks

```
# echo 1 > /proc/sys/kernel/sched_schedstats
# perf sched record -a -- ./myapp --my-args param1
# perf sched latency
```



Solution

Leave some physical CPUs unused by the guest and pin the IOThreads to those spare CPUs.

Case 2: High Hackbench Durations

“An upstream patch* was occasionally causing high hackbench durations.”

* https://lkml.kernel.org/r/20170517105350.hk5m4h4jb6dfr65a@hirez.programming.kicks-ass.net

Profile The System

- Record the results

```
# perf sched record -- ./hackbench -pipe 1 20000
# mv perf.data perf.data.fast (fast run)
# mv perf.data perf.data.slow (slow run)
```

- Inspect the events with '**perf sched script**'
- Find out the tasks with non-zero scheduling latency

```
# perf sched script | \
  grep -E ".*sched_stat_wait.*delay=[1-9]"
```

```
sched:sched_stat_wait: comm=gnome-terminal- pid=8245 delay=2711 [ns]
```

Result

```
hackbench 10706 [000]  5123.493992: sched:sched_stat_wait:
                                      comm=hackbench pid=10693
                                      delay=1307 [ns]
...
hackbench 10706 [000]  5123.493993: sched:sched_migrate_task:
                                      comm=hackbench pid=10693
                                      prio=120 orig_cpu=13
                                      dest_cpu=1
...
hackbench 10734 [019]  5123.494014: sched:sched_stat_wait:
                                      comm=hackbench pid=10693
                                      delay=10842 [ns]
...
hackbench 10734 [019]  5123.494017: sched:sched_migrate_task:
                                      comm=hackbench pid=10693
                                      prio=120 orig_cpu=1
                                      dest_cpu=14
```

High scheduling latency before migration!

(CPU 1 was in NUMA node 0 but CPUs 13 and 14 are in NUMA node 1.)

Case 3: Tracking Task Placement During fork()

“While helping to test patch(*) in SLE12-SP3, I had to look at the CPU placement of new tasks because I wanted to ensure they were spread evenly across all NUMA nodes.”

(*) patches.suse/sched-core-Use-load_avg-for-selecting-idlest-group.patch

Check the CPU Placement

```
$ perf sched script | grep wakeup_new
hackbench 10679 [001] 5123.141278: sched:sched_wakeup_new:
                                   hackbench:10680 [120]
                                   success=1 CPU:000
hackbench 10679 [001] 5123.141321: sched:sched_wakeup_new:
                                   hackbench:10681 [120]
                                   success=1 CPU:000
hackbench 10679 [001] 5123.141361: sched:sched_wakeup_new:
                                   hackbench:10682 [120]
                                   success=1 CPU:012
hackbench 10679 [001] 5123.141396: sched:sched_wakeup_new:
                                   hackbench:10683 [120]
                                   success=1 CPU:012
```

Question?


SUSE®
We adapt. You succeed.