Introduction and Improvement of PSI

Chengming Zhou
ByteDance STE Team

CLK 2021

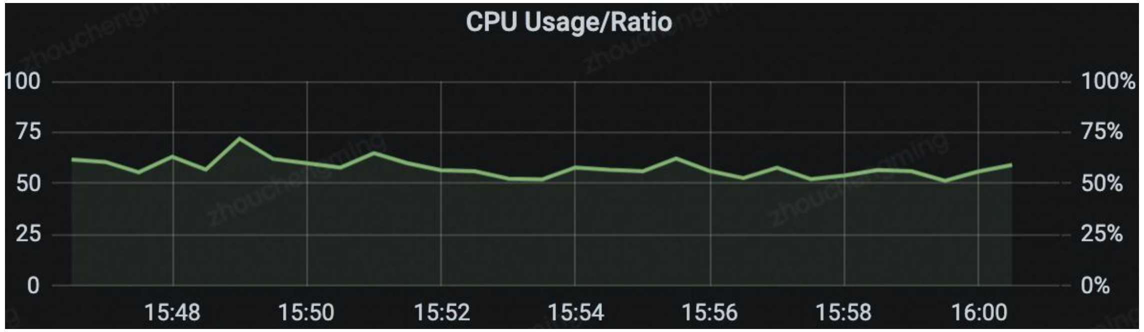
Agenda

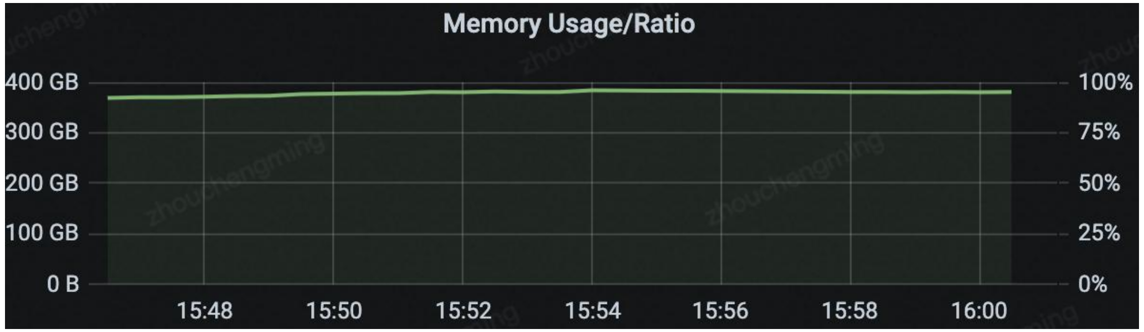
- Background
- PSI introduction
- PSI improvement
- Status & Future Work

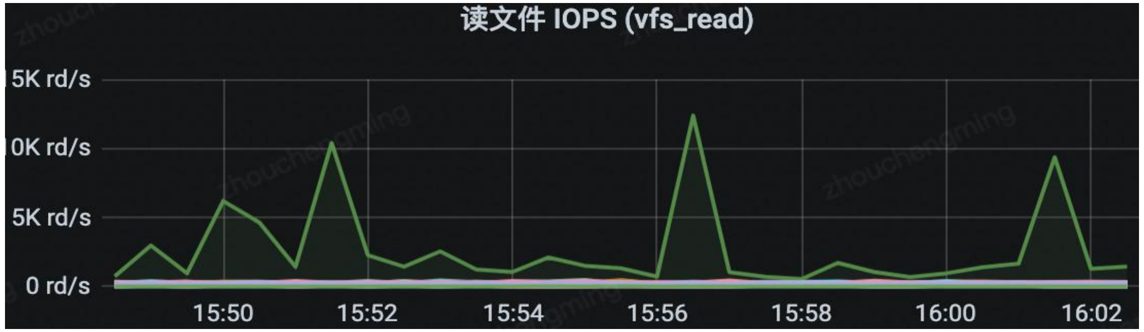
Background

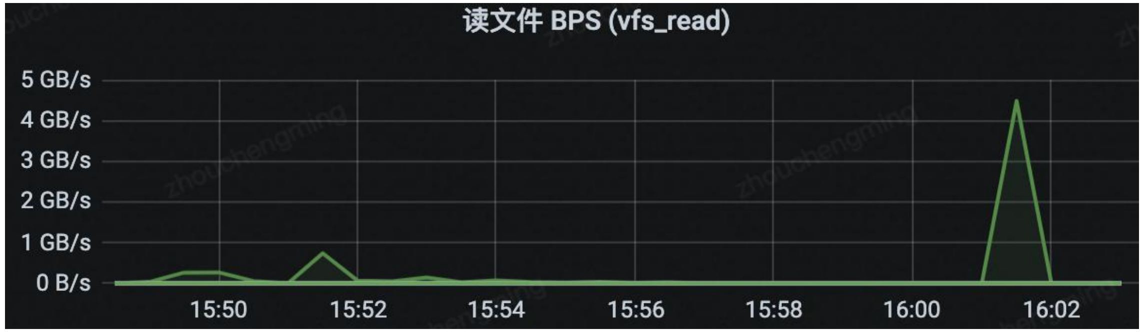
Resource

Maximizing Resource Utilization



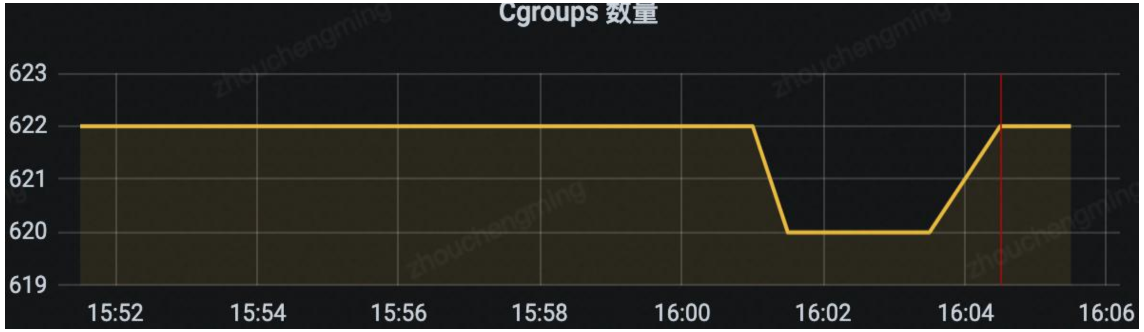


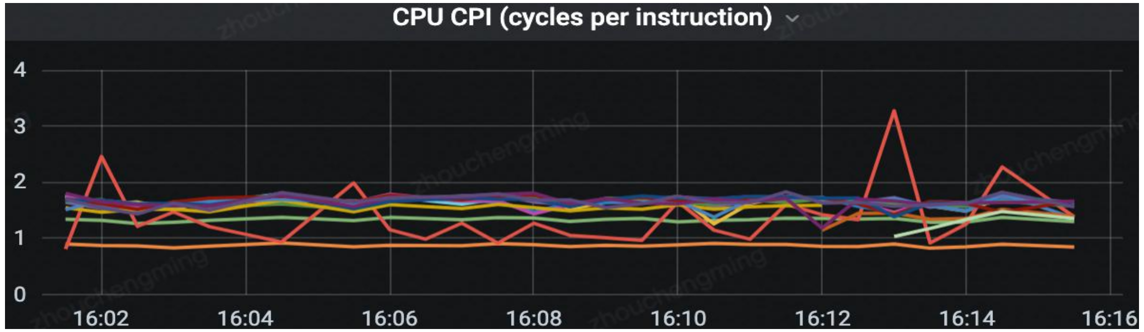


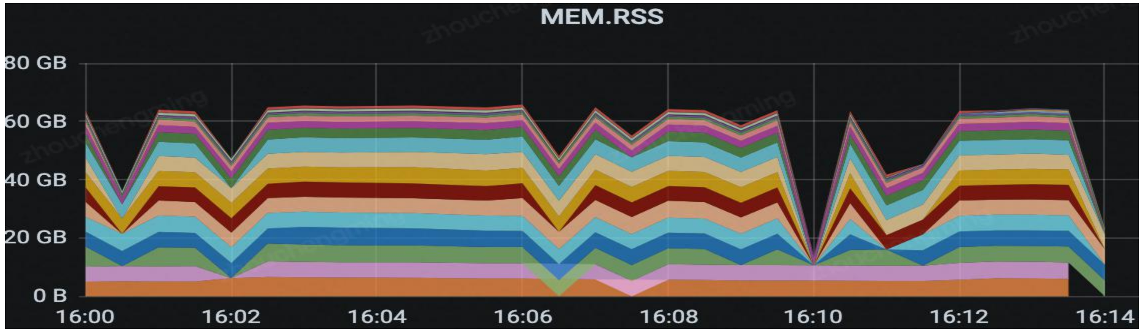


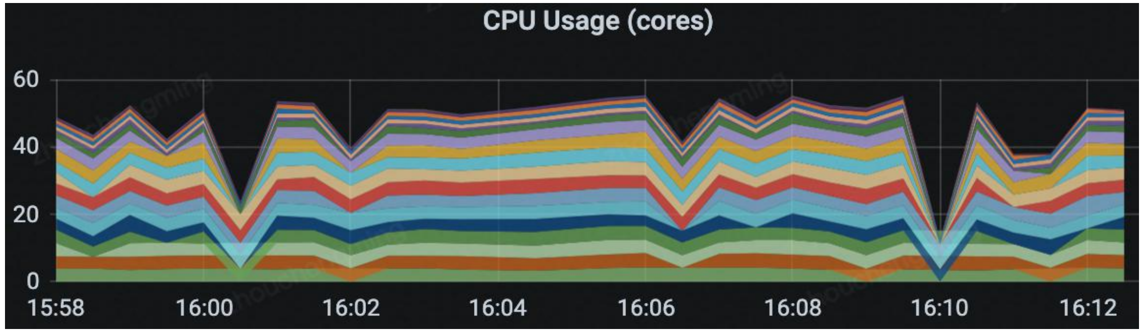
Cgroup

More Workload Control Groups









Pressure

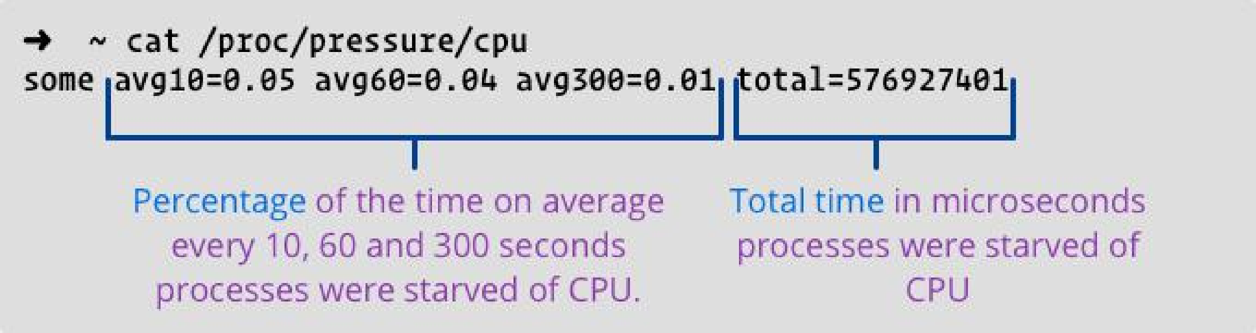
- How much pressure are workload causing
  - Useful for performance debug like latency burst
  - Maximizing resource utilization while maintain performance
  - evict specified workloads to meet the needs of priority workloads
  - kill workloads to spare minimal resource for system usability

- Old ways
  - Load Average
  - Vmpressure

PSI introduction

Interface

- proc interface for system resource pressure
  - for CPU



```
➜  ~ cat /proc/pressure/cpu
some avg10=0.05 avg60=0.04 avg300=0.01 total=576927401
```

Percentage of the time on average every 10, 60 and 300 seconds processes were starved of CPU.

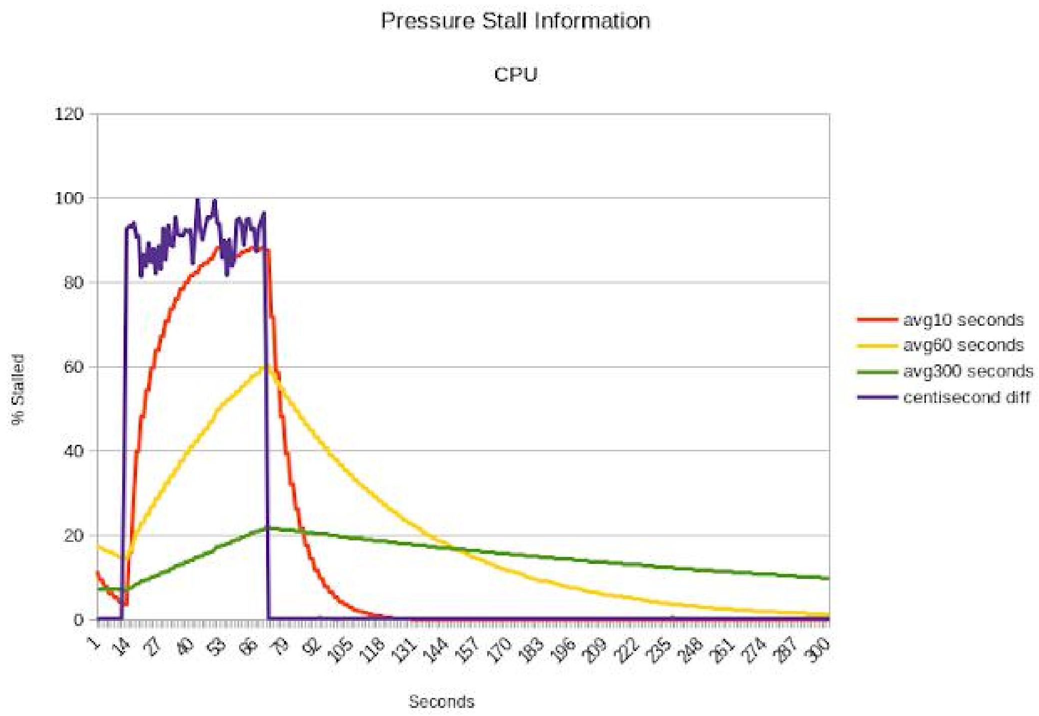
Total time in microseconds processes were starved of CPU

- cgroup interface for cgroup resource pressure
  - for memory and IO

```
some avg10=0.00 avg60=0.00 avg300=0.00 total=0
full avg10=0.00 avg60=0.00 avg300=0.00 total=0
```

Interface

- notification when reaching specified pressure threshold
  - echo “some 100000 1000000” > /proc/pressure/cpu
  - use poll/epoll/select to wait for notification



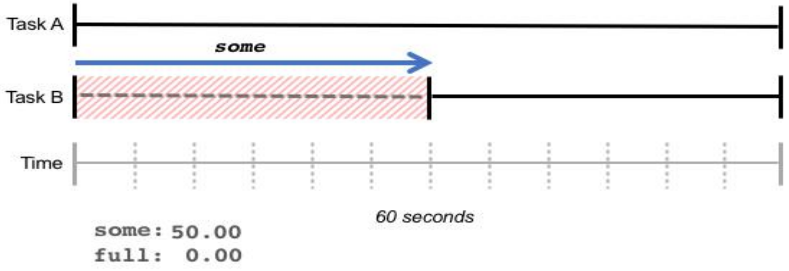
Definition

- some

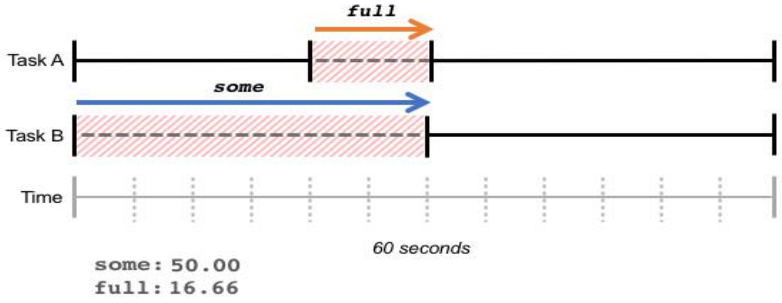
  percentage of time some (one or more) tasks were delayed due to lack of resources

- full

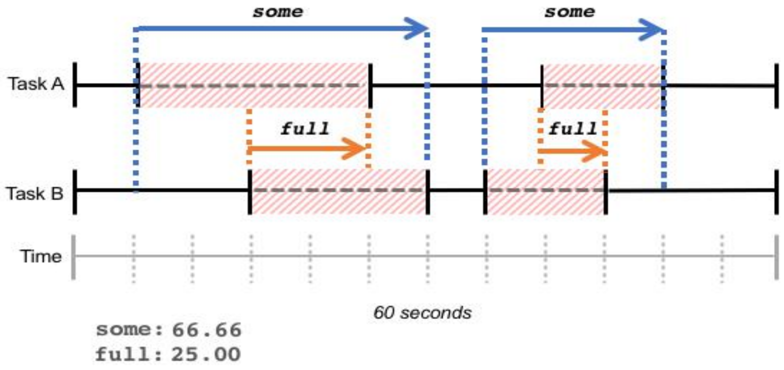
  percentage of time in which all tasks were delayed by lack of resources



```
some: 50.00
full:  0.00
```



```
some: 50.00
full: 16.66
```



```
some: 66.66
full: 25.00
```

PSI improvement

Implementation

- Tracking task status
  - TSK_RUNNING
  - TSK_ONCPU
  - TSK_MEMSTALL
  - TSK_IOWAIT

- Accumulate task count per-cgroup per-CPU per-status

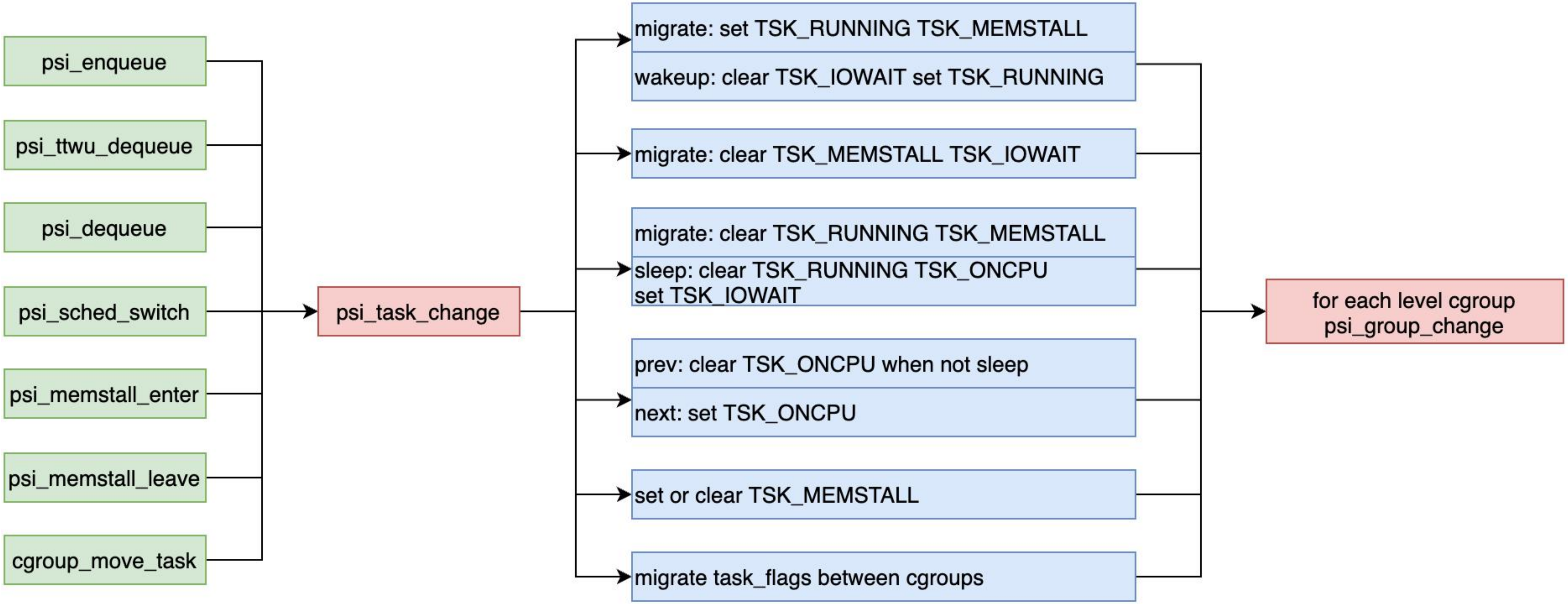
<table>
<tr><td>- PSI_IO_SOME</td><td>tasks[NR_IOWAIT] > 0</td></tr>
<tr><td>- PSI_IO_FULL</td><td>tasks[NR_IOWAIT] > 0 && task[NR_RUNNING] = 0</td></tr>
<tr><td>- PSI_MEM_SOME</td><td>tasks[NR_MEMSTALL] > 0</td></tr>
<tr><td>- PSI_MEM_FULL</td><td>tasks[NR_MEMSTALL] > 0 && tasks[NR_RUNNING] = 0</td></tr>
<tr><td>- PSI_CPU_SOME</td><td>tasks[NR_RUNNING] > tasks[NR_ONCPU]</td></tr>
<tr><td>- PSI_NONIDLE</td><td>tasks[NR_RUNNING] || tasks[NR_MEMSTALL] || tasks[NR_IOWAIT]</td></tr>
</table>

Improvement

- Accumulate task count per-cgroup per-CPU per-status
  - PSI_IO_SOME tasks[NR_IOWAIT] > 0
  - PSI_IO_FULL tasks[NR_IOWAIT] > 0 && task[NR_RUNNING] = 0
  - PSI_MEM_SOME tasks[NR_MEMSTALL] > 0
  - PSI_MEM_FULL tasks[NR_MEMSTALL] > 0 && tasks[NR_RUNNING] = 0
  - PSI_CPU_SOME tasks[NR_RUNNING] > tasks[NR_ONCPU]
  - PSI_CPU_FULL tasks[NR_RUNNING] > 0 && tasks[NR_ONCPU] = 0
  - PSI_NONIDLE tasks[NR_RUNNING] || tasks[NR_MEMSTALL] || tasks[NR_IOWAIT]

- Meaningful at the cgroup level
  - means all non-idle tasks in a cgroup are delayed on the CPU resource
  - throttled by the CPU bandwidth control
  - CPU used by other cgroups

Tracking



psi_enqueue

psi_ttwu_dequeue

psi_dequeue

psi_sched_switch

psi_memstall_enter

psi_memstall_leave

cgroup_move_task

psi_task_change

migrate: set TSK_RUNNING TSK_MEMSTALL

wakeup: clear TSK_IOWAIT set TSK_RUNNING

migrate: clear TSK_MEMSTALL TSK_IOWAIT

migrate: clear TSK_RUNNING TSK_MEMSTALL

sleep: clear TSK_RUNNING TSK_ONCPU
set TSK_IOWAIT

prev: clear TSK_ONCPU when not sleep

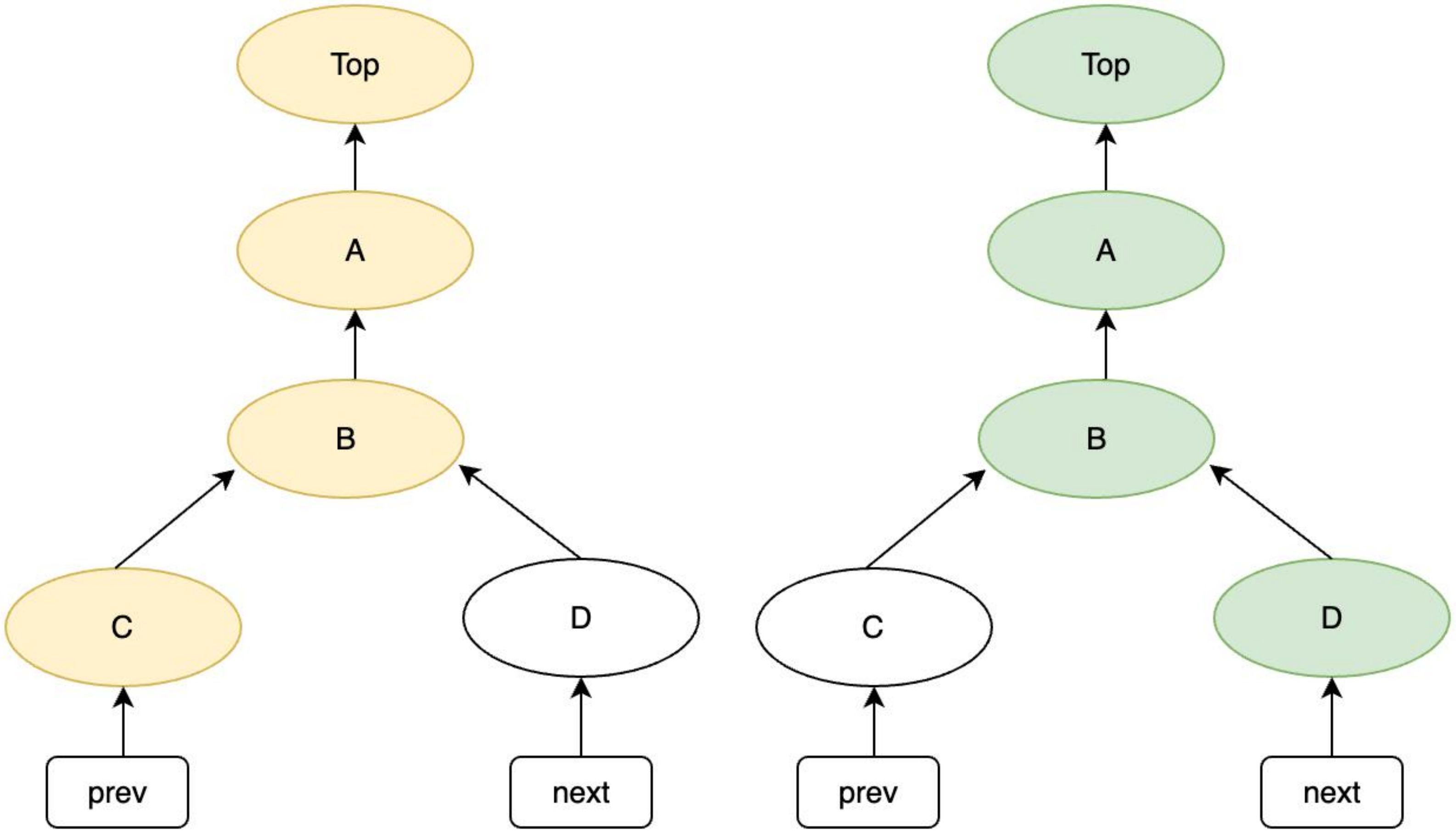
next: set TSK_ONCPU

set or clear TSK_MEMSTALL

migrate task_flags between cgroups

for each level cgroup
psi_group_change

Improvement

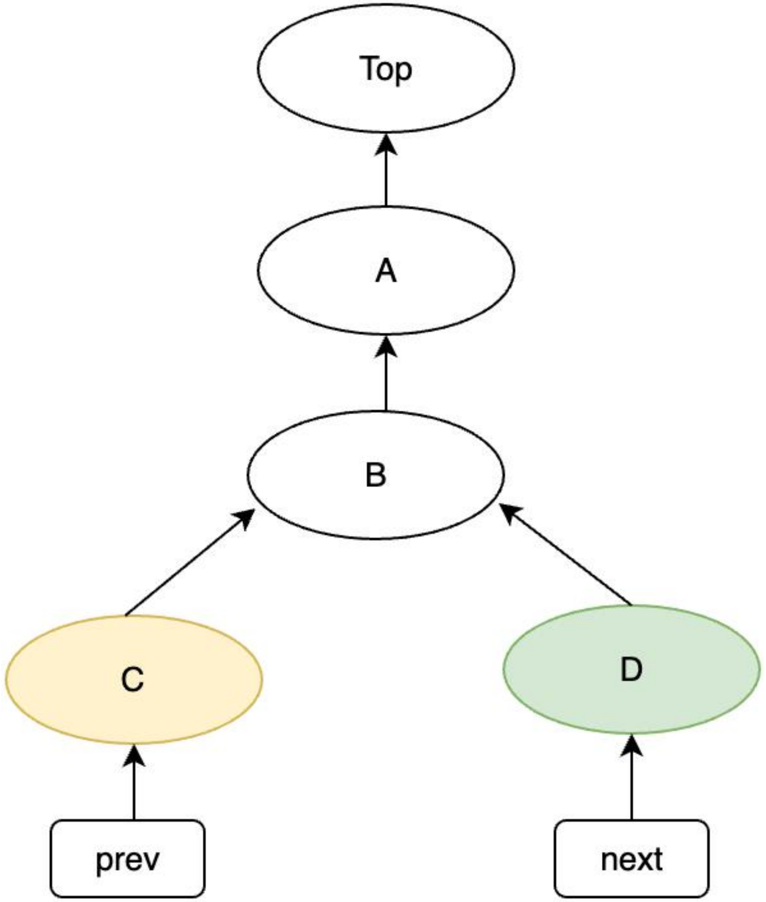


Improvement

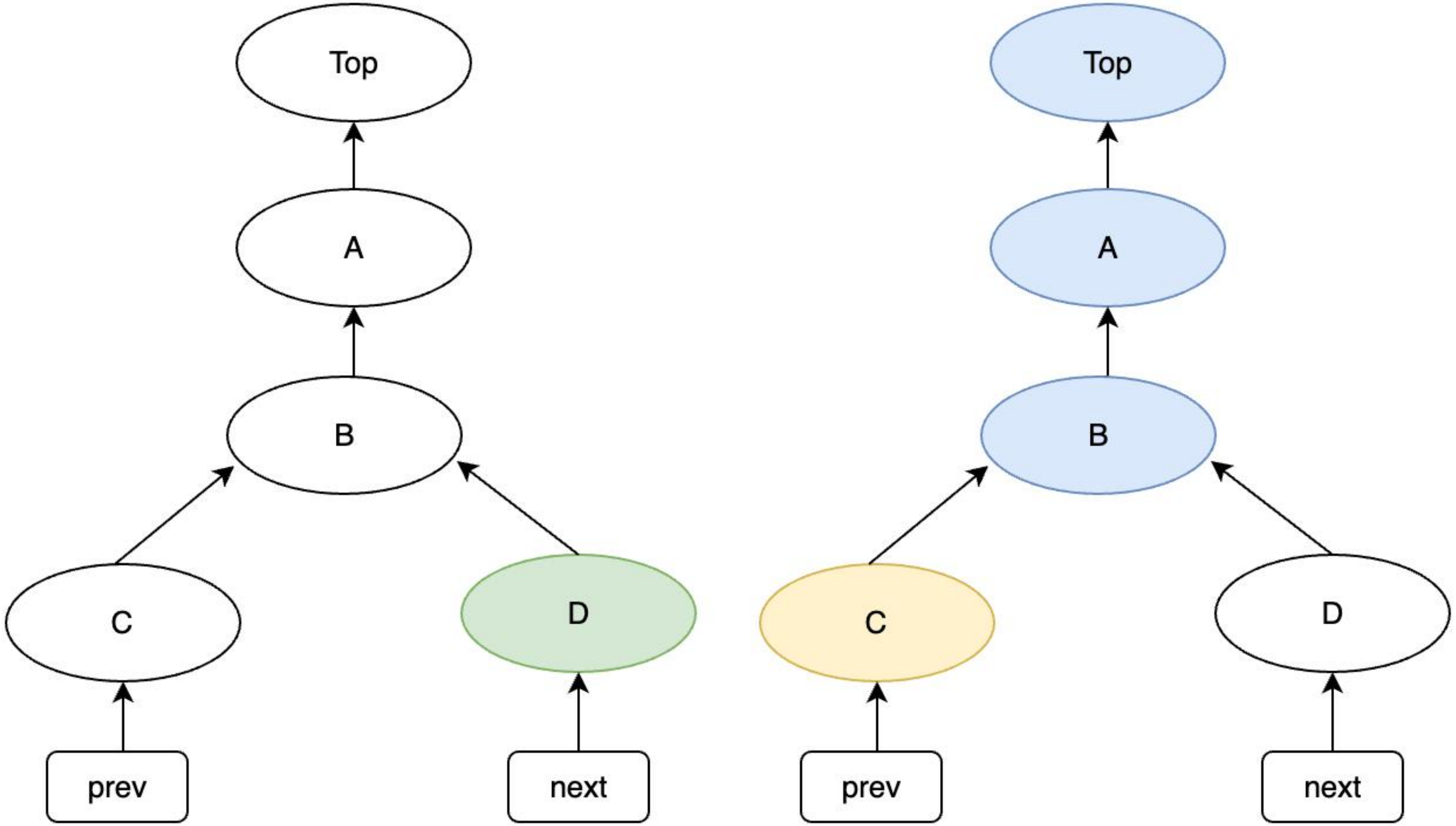
- task switch in deep cgroup levels
  - clear prev TSK_ONCPU until Top
  - set next TSK_ONCPU until Top

- psi_task_switch preempt case
  - clear prev TSK_ONCPU until B
  - set next TSK_ONCPU until B

- psi_task_switch sleep case
  - delay psi_dequeue to psi_task_switch
  - avoid ONCPU changes on common ancestors



Improvement



Improvement

- sleep before:

```
- psi_dequeue()
  while ((group = iterate_groups(prev)))  # all ancestors
      psi_group_change(prev, .clear=TSK_RUNNING|TSK_ONCPU)
- psi_task_switch()
  while ((group = iterate_groups(next)))  # all ancestors
      psi_group_change(next, .set=TSK_ONCPU)
```

- sleep after:

```
- psi_dequeue()
  nop
- psi_task_switch()
  while ((group = iterate_groups(next)))  # until (prev & next)
      psi_group_change(next, .set=TSK_ONCPU)
  while ((group = iterate_groups(prev)))  # all ancestors
      psi_group_change(prev, .clear=common?TSK_RUNNING:TSK_RUNNING|TSK_ONCPU)
```

Gain

The CPU overhead of PSI is reduced by about 10% for our workload.

Status & Future Work

Status & Future Work

Status

- Improvement patchset has been merged into Linux kernel.
  - https://lore.kernel.org/lkml/20210303034659.91735-1-zhouchengming@bytedance.com/

Future Work

- More task status and more PSI metrics
- Monitor and notification for cgroup adaptive tuning
- More performance improvements in task and cgroup tracking

THANKS.

ByteDance 字节跳动