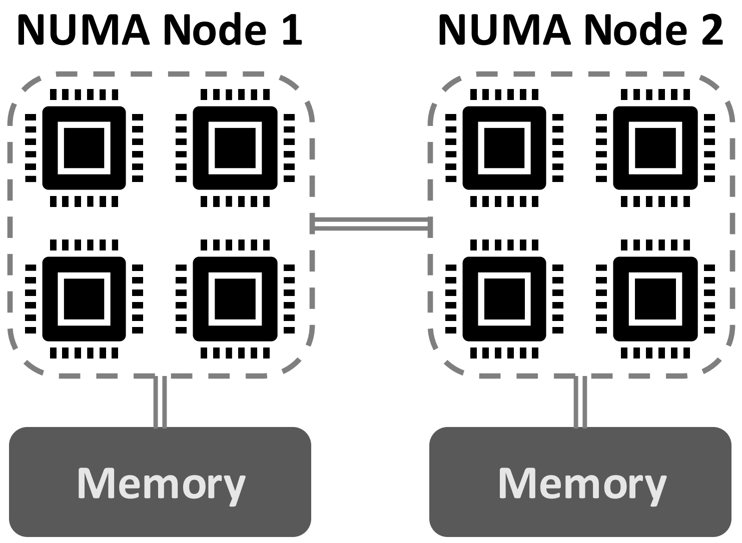
Systems Software &
Architecture Lab.
Seoul National University

2025.10.22

Project #3:
The Road to Balance

CPU Affinity

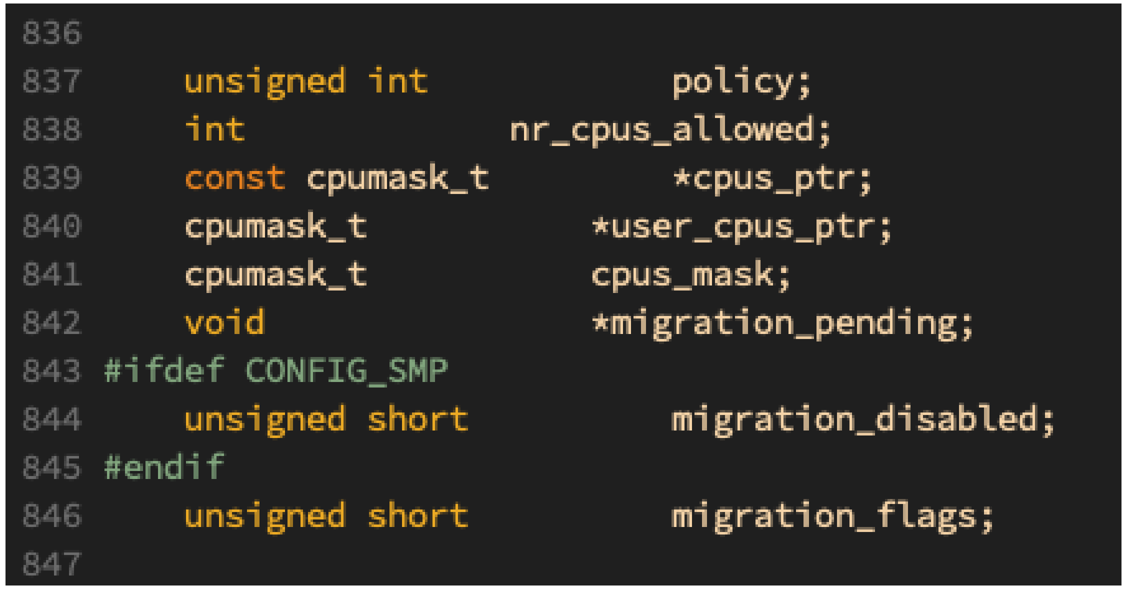
- **What?**
  - **CPU affinity** ensures that a process (or kernel thread) executes only on a specified subset of CPUs.
- **Why?**
  - Cache locality and performance
  - NUMA locality
  - Task isolation
  - Balancing multi-threaded application
- **How?**
  - Let's see…



CPU Affinity in Linux

- "task_struct" structure
  - nr_cpus_allowed
  - user_cpus_ptr
  - cpus_mask
- System calls
  - `sched_setaffinity()`
  - `sched_getaffinity()`

include/linux/sched.h - struct task_struct



```

	unsigned int			policy;
	int				nr_cpus_allowed;
	const cpumask_t			*cpus_ptr;
	cpumask_t			*user_cpus_ptr;
	cpumask_t			cpus_mask;
	void				*migration_pending;
#ifdef CONFIG_SMP
	unsigned short			migration_disabled;
#endif
	unsigned short			migration_flags;

```

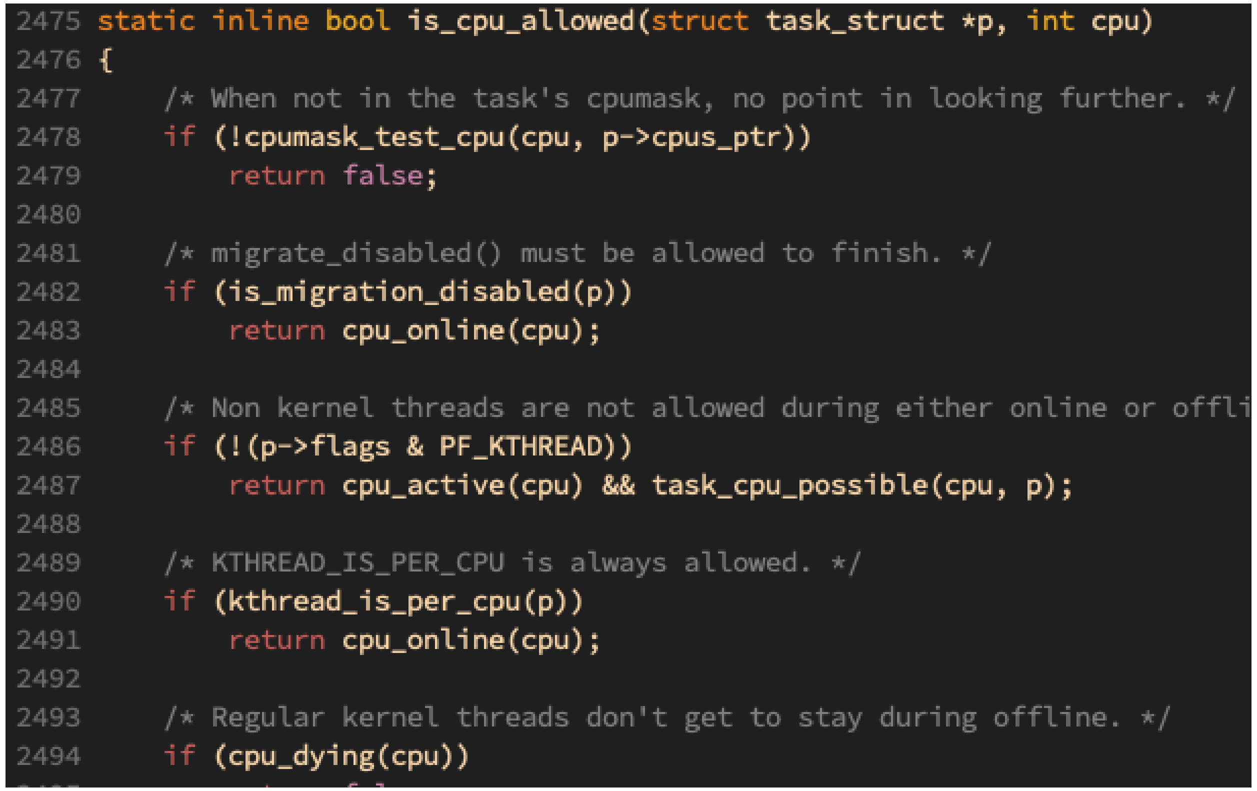
***linux v6.9**

Field	Description	Type
nr_cpus_allowed	Numbers of allowed cpus in cpus_mask	int
cpus_ptr	Effective mask of cpus allowed	cpumask_t
users_cpus_ptr	Mask set by user, set by `sched_setaffinity()`	cpumaks_t

CPU Affinity in Linux

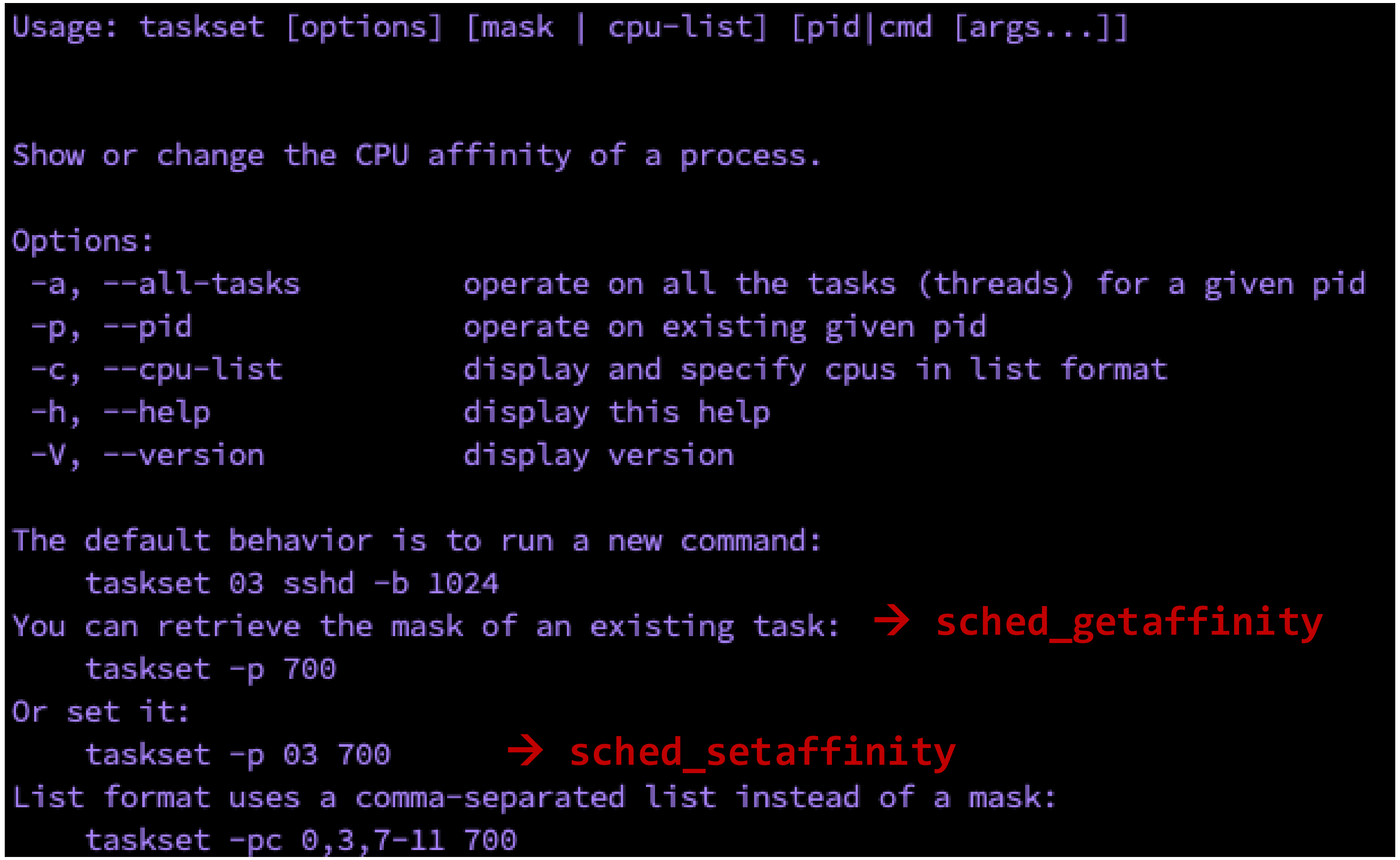
- Check allowed cpus before
  - Scheduling
  - Migrating
  - Load balancing
  - …

kernel/sched/core.c



```
static inline bool is_cpu_allowed(struct task_struct *p, int cpu)
{
    /* When not in the task's cpumask, no point in looking further. */
    if (!cpumask_test_cpu(cpu, p->cpus_ptr))
        return false;

    /* migrate_disabled() must be allowed to finish. */
    if (is_migration_disabled(p))
        return cpu_online(cpu);

    /* Non kernel threads are not allowed during either online or offli
    if (!(p->flags & PF_KTHREAD))
        return cpu_active(cpu) && task_cpu_possible(cpu, p);

    /* KTHREAD_IS_PER_CPU is always allowed. */
    if (kthread_is_per_cpu(p))
        return cpu_online(cpu);

    /* Regular kernel threads don't get to stay during offline. */
    if (cpu_dying(cpu))
```

Taskset in Linux

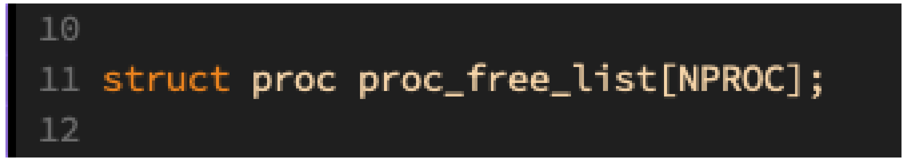


```
Usage: taskset [options] [mask | cpu-list] [pid|cmd [args...]]


Show or change the CPU affinity of a process.

Options:
 -a, --all-tasks         operate on all the tasks (threads) for a given pid
 -p, --pid               operate on existing given pid
 -c, --cpu-list          display and specify cpus in list format
 -h, --help              display this help
 -V, --version           display version

The default behavior is to run a new command:
    taskset 03 sshd -b 1024
You can retrieve the mask of an existing task:  → sched_getaffinity
    taskset -p 700
Or set it:
    taskset -p 03 700      → sched_setaffinity
List format uses a comma-separated list instead of a mask:
    taskset -pc 0,3,7-11 700
```

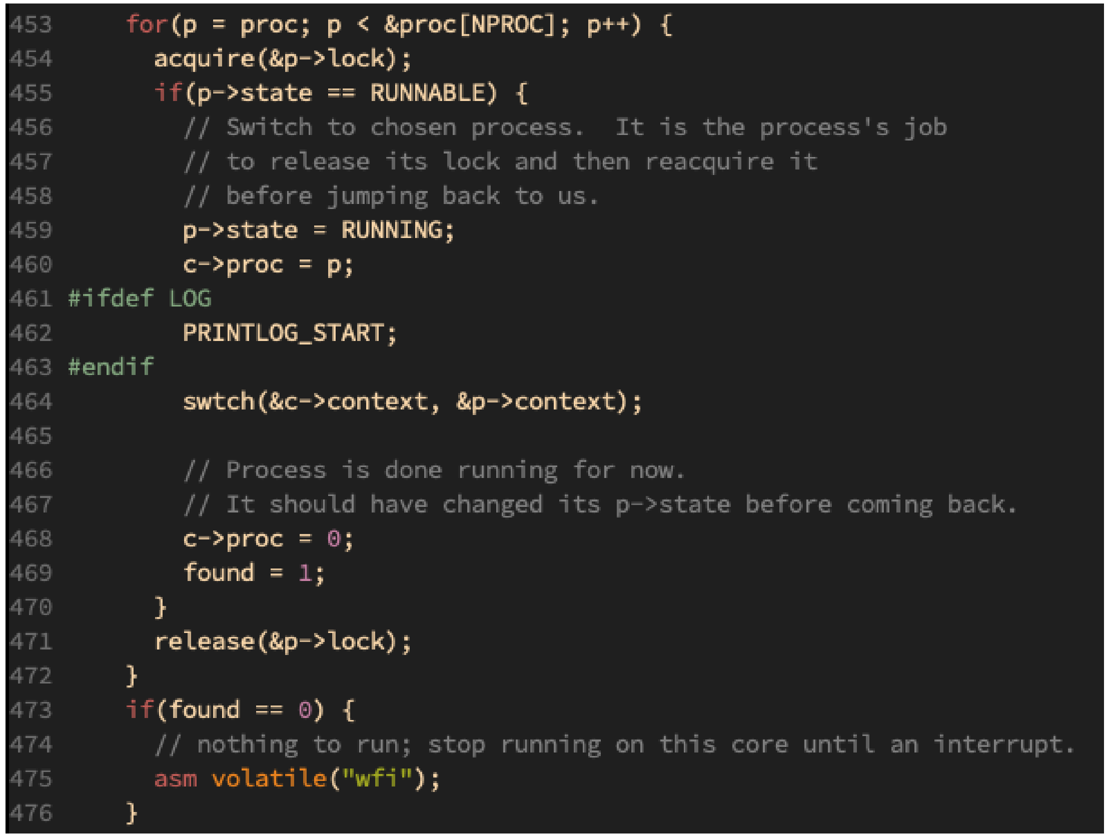
Xv6 Scheduler

- Round-Robin
  - Scan for RUNNABLE proc
  - If found, set RUNNING, switch context
- Single global queue
  - Lock before checking each proc
  - Not scalable!

kernel/proc.c



```

struct proc proc_free_list[NPROC];

```



```
    for(p = proc; p < &proc[NPROC]; p++) {
      acquire(&p->lock);
      if(p->state == RUNNABLE) {
        // Switch to chosen process.  It is the process's job
        // to release its lock and then reacquire it
        // before jumping back to us.
        p->state = RUNNING;
        c->proc = p;
#ifdef LOG
        PRINTLOG_START;
#endif
        swtch(&c->context, &p->context);

        // Process is done running for now.
        // It should have changed its p->state before coming back.
        c->proc = 0;
        found = 1;
      }
      release(&p->lock);
    }
    if(found == 0) {
      // nothing to run; stop running on this core until an interrupt.
      asm volatile("wfi");
    }
```

Part 1. CPU affinity with the pin() system call (10 points)

- `int pin(int mask)` system call
  - Sets or queries CPU affinity of the calling process
  - System call number assigned to 30 (no modifications required)

Mode	Mask Value	Description
SET	mask > 0	The process may run only on RISC-V harts whose bits are set in mask
UNPIN	mask == 0	Allow the process to run on all online harts
QUERY	mask == -1	Returns the current CPU affinity mask of the calling process

Part 2. Per-core run queues & process migration (30 points)

- Implement Per hart run-queue
  - Expand the existing single global run queue of xv6
  - Single queues maximum capacity is set the QPROC (kernel/param.h)
  - Round-robin policy inside its local queue

- Implement process migration
  - Implement a method to move tasks from one queue to another
  - Child process inherits parent's affinity mask
  - After pin(), if the current hart is no longer permitted, the process must be migrated

Part 3. Load Balancing Policy

- Design and implement your own load balancing policy!
- Minimize the completion time of a given batch of CPU-intensive processes
- 10ms penalty whenever it runs on a different hart, so avoid needless migrations
  - Do not modify the skeleton code parts that refers to this
- Things you might consider…
  - Periodic rebalancing
  - Idle-time stealing
  - …

BONUS

- Using multiple test cases, completion time will be ranked
- 20% bonus to the top 10
- 10% bonus to the top 11 ~ 20

Additional Notes

- You must pull the latest version of skeleton code. (Date: **10/23)**
  - An additional “mask” field to track the allowed harts to run the process

- Every details about your code should be in the document, and all the descriptions done in thedocument should be implemented in your code.

Due date

- Due
  - 11:59 PM, November 6 (Thursday)

- Submission
  - Run the make submit command to generate a tarball named xv6-pa3-{STUDENTID}.tar.gz in the xv6-riscv-snu directory
  - Upload the compressed file to the submission server
  - The total number of submissions for this project will be limited to 30
  - Only the version marked FINAL will be considered for the project score

Thank you!