Affinity & KOS

Jalal Kawash

Linux affinity sys calls

http://linux.die.net/man/2/sched_getaffinity

int sched_setaffinity(pid_t *pid*, size_t *cpusetsize*, cpu_set_t **mask*);
int sched_getaffinity(pid_t *pid*, size_t *cpusetsize*, cpu_set_t **mask*);

int sched_setaffinity(pid_t *pid*, size_t *cpusetsize*, cpu_set_t **mask*);

- "sets the CPU affinity mask of the process whose ID is *pid* to the value specified by *mask*. If *pid* is zero, then the calling process is used. The argument *cpusetsize* is the length (in bytes) of the data pointed to by *mask*. Normally this argument would be specified as *sizeof(cpu_set_t)*.
- If the process specified by *pid* is not currently running on one of the CPUs specified in *mask*, then that process is migrated to one of the CPUs specified in *mask*. "

int sched_getaffinity(pid_t *pid*, size_t *cpusetsize*, cpu_set_t **mask*);

- "writes the affinity mask of the process whose ID is *pid* into the *cpu_set_t* structure pointed to by *mask*. The *cpusetsize* argument specifies the size (in bytes) of *mask*. If *pid* is zero, then the mask of the calling process is returned.
- **Return Value**
- On success, **sched_setaffinity**() and **sched_getaffinity**() return 0. On error, -1 is returned, and *errno* is set appropriately. "

Errors

Errors

EFAULT

A supplied memory address was invalid.

EINVAL

The affinity bit mask *mask* contains no processors that are currently physically on the system and permitted to th

EINVAL

(**sched_getaffinity**() and, in kernels before 2.6.9, **sched_setaffinity**()) *cpusetsize* is smaller than the size of the

EPERM

(**sched_setaffinity**()) The calling process does not have appropriate privileges. The caller needs an effective us

ESRCH

The process whose ID is *pid* could not be found.

cpu_set_t

- cpu_set_t is already defined as a 64 bit unsigned integer in kos/src/include/kostypes.h.
- KOS assumes 4 cores only
- Only need the least significant 4 bits
- Examples:
- Mask $0^{60}0001$ => process can be only assigned to core 1
- $0^{60}0101$ => process can be only assigned to cores 1 & 3
- $0^{60}1111$ => process can be assigned to any of the 4 cores

Scheduler::preempt() in src/runtime/Scheduler.cc

```
void Scheduler::preempt() {          // IRQs disabled, lock count inflated
#if TESTING_NEVER_MIGRATE
  switchThread(this);
#else /* migration enabled */
  Scheduler* target =  Runtime::getCurrThread()->getAffinity();      Replace for A1
#if TESTING_ALWAYS_MIGRATE
  if (!target) target = partner;
#else /* simple load balancing */
  if (!target) target = (partner->readyCount + 2 < readyCount) ? partner : this;
#endif
  switchThread(target);
#endif
}
```

Assignment 1

- Get the affinity mask
  - mword affinityMask = Runtime::getCurrThread()->getAffinityMask()
- If the affinity mask = 0, there should be no change in policy
  - target = Runtime::getCurrThread()->getAffinity()
  - Same as before; returns a scheduler object
- Else check every bit in affinity mask
  - If bit i is set, get the scheduler for that core
    - Scheduler *sched = Machine::getScheduler(i)
  - For all set bits, assign the process to the core with the smallest ready queue
    - Queue size: sched->readyCount
    - Assignment: target = sched