زمان‌بندی پروژه‌ها

حمید خدادادی - آرمین افشاریان

زمان بندی در xv6:

```c
void
scheduler(void)
{
  struct proc *p;
  struct cpu *c = mycpu();
  c->proc = 0;

  for(;;){
    // Enable interrupts on this processor.
    sti();

    // Loop over process table looking for process to run.
    acquire(&ptable.lock);
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
      if(p->state != RUNNABLE)
        continue;

      // Switch to chosen process.  It is the process's job
      // to release ptable.lock and then reacquire it
      // before jumping back to us.
      c->proc = p;
      switchuvm(p);
      p->state = RUNNING;

      swtch(&(c->scheduler), p->context);
      switchkvm();

      // Process is done running for now.
      // It should have changed its p->state before coming back.
      c->proc = 0;
    }
    release(&ptable.lock);

  }
}
```

- زمان بندی Round Robin
- تابع scheduler از فایل proc.c

زمان بندی در xv6:

- یک context switch صورت می‌گیرد.
- پوینتر c نشان دهنده پردازنده است.
- پردازه بعدی که با p مشخص شده، زمان‌بندی می‌شود.
- در یک کوانتوم زمانی پردازنده دست پردازه است.

```
swtch(&(c->scheduler), p->context);
```

زمان بندی در xv6:

- اگر کار پردازه تمام شود، از صف خارج می‌شود.
- ولی ممکن است کارش تمام نشود.
- Timer interrupt یک وقفه صادر می‌کند. (trap.c)

```
if(myproc() && myproc()->state == RUNNING &&
    tf->trapno == T_IRQ0+IRQ_TIMER)
  yield();
```

- تابع yield از فایل proc.c صدا زده می‌شود.
- پردازه به ته صف می‌رود و پردازنده به اولین پردازه سر صف تعلق می‌گیرد.

زمان بندی در xv6:

- تابع yield از proc.c

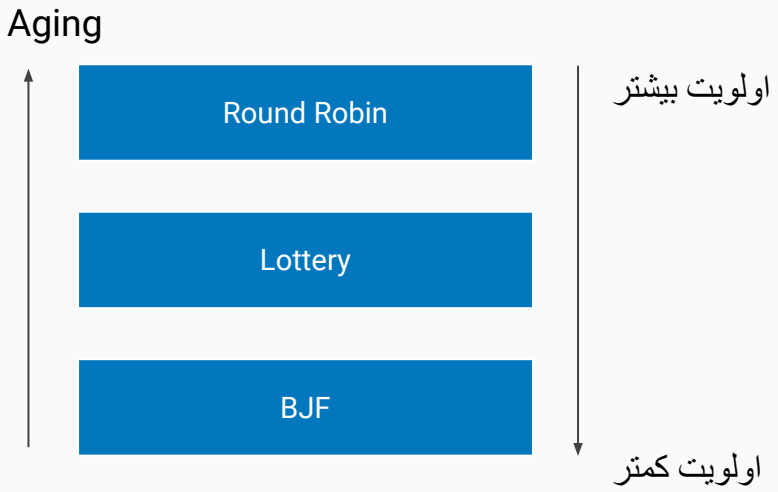
```c
void
yield(void)
{
  acquire(&ptable.lock);
  myproc()->state = RUNNABLE;
  sched();
  release(&ptable.lock);
}
```

زمان‌بندی جدید:

- زمان‌بندی بازخوردی چند سطحی:
- صف اول: Round Robin
- صف دوم: Lottery
- صف سوم: BJF

زمان‌بندی بازخوردی چند سطحی:

- ۳ سطح
- اولویت با صف با شماره کمتر
- اول همه پردازه های صف ۱ بعد ۲ بعد ۳
- فراخوانی سیستمی برای تغییر صف پردازه‌ها



زمان بندی Round Robin:

- یک کوانتوم زمانی در نظر می‌گیریم.
- زمان‌بند پردازه را برای یک بازه حداکثر یک کوانتومی زمان بندی می‌کند.
- اگر کمتر از یک کوانتوم کار پردازه طول بکشد، پردازه پردازنده را رها می‌کند و زمان‌بند پردازه بعدی را از سر صف انتخاب می‌کند.
- اگر بیشتر طول بکشد، یک اینتراپت زمانی صادر می‌شود، پردازه در حال اجرا به **ته صف می رود و زمان بند، پردازه بعدی را از سر صف انتخاب می‌کند**.
- نیازی به پیاده‌سازی این زمان‌بند نیست و می‌توانید از الگوریتم زمان‌بندی نوبت گردشی پیاده‌سازی شده در XV6 استفاده نمایید.

زمان بندی Lottery:

- تخصیص منابع به پردازه ها به صورت تصادفی است.
- احتمال انتخاب شدن بر اساس بلیت های بخت آزمایی مشخص می شود.
- فایل proc.c و proc.h برای نگهداری این اطلاعات باید ویرایش شود.

زمان بندی BJF:

- زمان ورود و تعداد سیکل اجرای هر پردازه برای این الگوریتم لازم است.
- **زمان ورود**: زمان سیستم عامل هنگام ورود را در نظر بگیرید.
- **تعداد سیکل**: در پردازه این ویژگی را نگهداری کنید. (پیش فرض: ۰)
- با هر بار اجرا یک واحد (۰.۱) سیکل را افزایش دهید.
- ضریب معادله با یک فراخوانی سیستمی مقداردهی می‌شود.

rank = (Priority * PriorityRatio) + (ArrivalTime * ArrivalTimeRatio) + (ExecutedCycle * ExecutedCycleRatio)

مکانیزم Aging:

- در هر صفی هر پردازه ای که بیشتر از ۸۰۰۰ سیکل زمان‌بندی نشد یک سطح بالاتر میرود.

  (۳ -> ۲ -> ۱)

- جلوگیری از starvation

فراخوانی های سیستمی:

- تغییر صف پردازه
- مقداردهی بلیت بخت‌آزمایی
- مقداردهی پارامتر BJF در سطح پردازه
- مقداردهی پارامتر BJF در سطح سیستم
- چاپ اطلاعات

برنامه تست:

- برای تست یک برنامه سطح کاربر بنویسید.
- شامل تعدادی پردازه و عملیات پردازشی (به قدر کافی طولانی باشد.)
- اجرا در پس زمینه: foo&

نکات پایانی:

- آدرس مخزن و شناسه آخرین تغییر
- گزارش

ممنون از توجهتون!