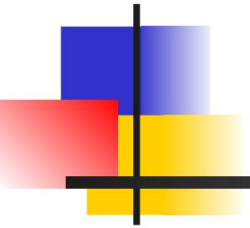
操作系统原理

操作系统原理

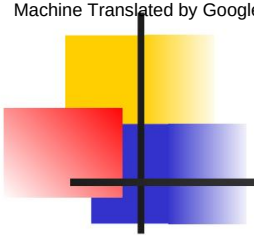


进程/线程调度

李旭东

leexudong@nankai.edu.cn南

开大学



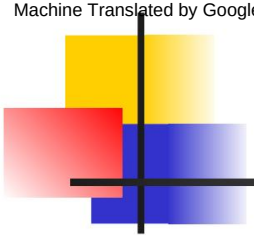
目标

调度器

过程行为

调度模式调度准则调度

算法线程调度



多道程序设计

调度器

调度算法

调度器

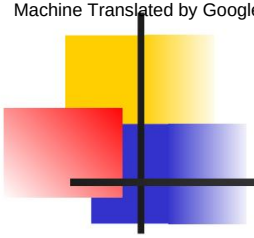
短期调度程序

中央处理器

中期调度程序

内存长期调

度程序作业

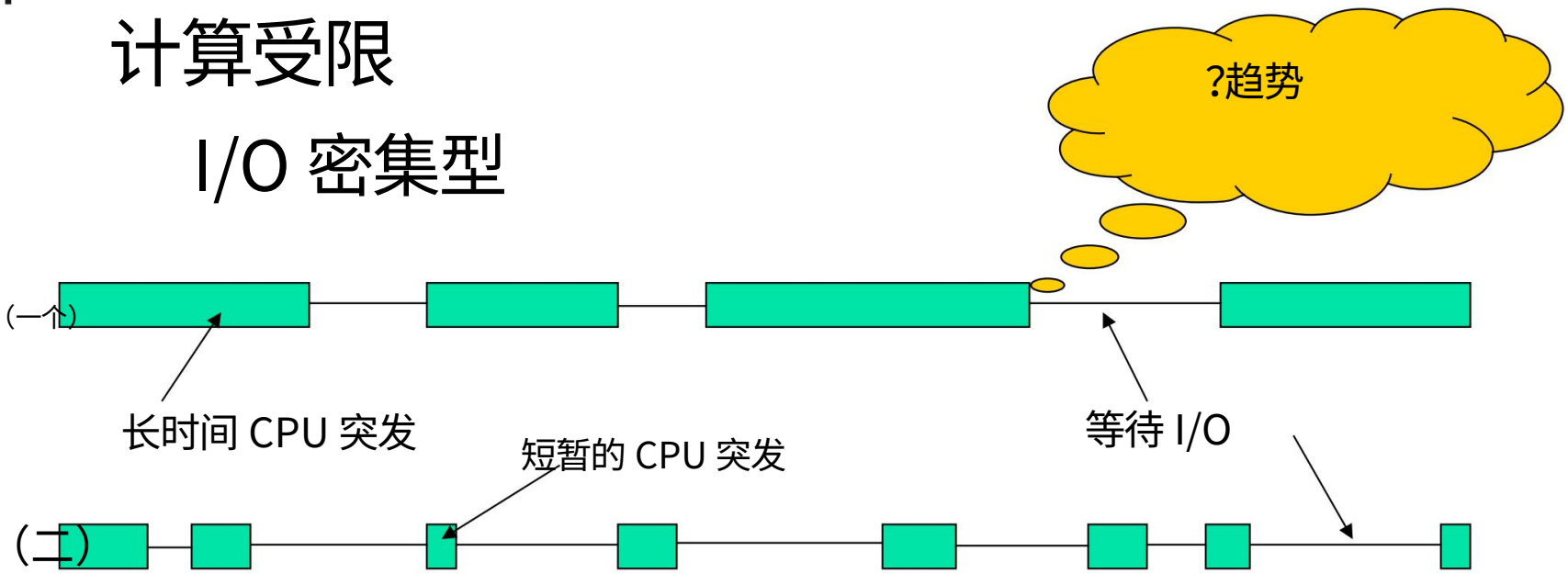


进程中的 CPU 和 I/O 突发

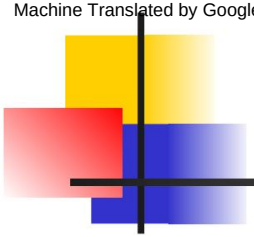
进程行为

计算受限

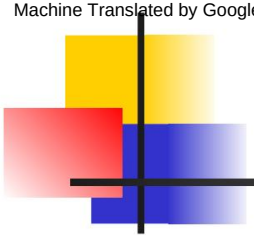
I/O 密集型



(a)计算密集型 (b)I/O密集型



多进程跟踪



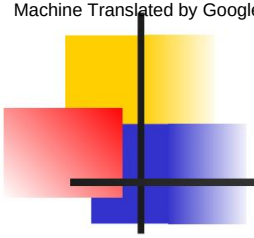
何时安排

创建一个新的进程

进程退出

进程因 I/O、信号量或其他原因而阻塞

发生 I/O 中断



调度员

将 CPU 控制权交给短期调度程序选择的进程的模块

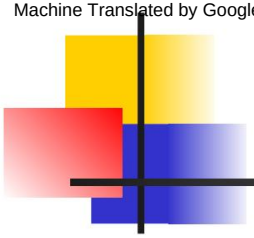
切换上下文

切换到用户模式

跳转到用户程序中的正确位置以重新启动该程序

调度延迟 调度延迟

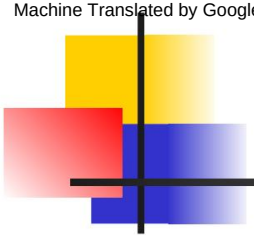
调度程序停止一个进程并启动另一个进程运行所需的时间



调度模式

抢占式

非抢占式　非抢占式,非增量式



调度算法的分类

批量

互动

实时

调度标准

CPU 利用率　　整个吞吐量

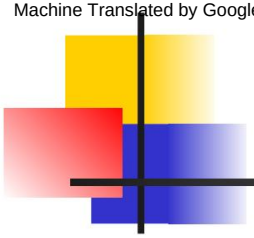
周转时间　　周转时间

等待进入内存　　在就绪队列中等待

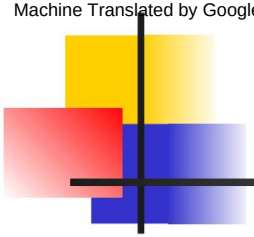
　　在 CPU 上执行　　进行 I/O　　等

待时间　　响应时间

…



调度算法目标

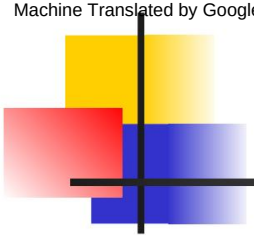


批处理系统中的调度

先到先得

最短任务优先剩余

时间最短优先



先到先得

平均等待时间

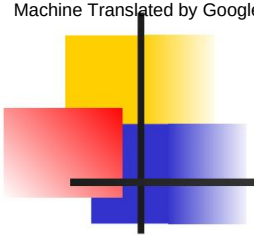
重量百分比=(0+24+27)/3=17

啊=?

最短作业优先

非抢占式
2. 先发制人

图 2-40。最短作业优先调度的示例。(a)按原始顺序运行四个作业。(b)按最短作业优先顺序运行它们。



测验

圣杰夫： 韋斯特儀=?

FCFS: AWT=?

最短作业优先

如何预测下一个 CPU 的长度爆裂？

指数平均值n +(1－a)Tn－1，

Tn+1 = 0 a 1

吨n 第 n 个 CPU 突发的长度

Tn+1下一个 CPU 突发的预测值

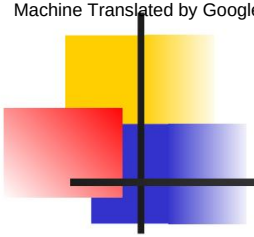
Tn+1=atn+(1－a)at n－1+…+(1－a)杰 在n－j+…+(1－a) n+1 T0

剩余时间最短的下一个

即抢占式 SJF 调度

非抢占式 SJF 调度:AWT=?

抢占式 SJF 调度:AWT=?

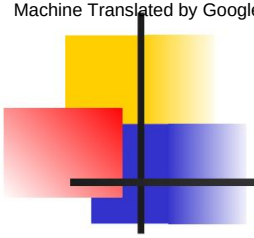


交互系统中的调度

循环调度优先级调度多队列最短

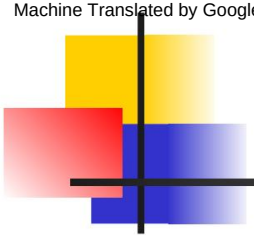
进程优先

保证调度抽签调度公平份额调度



循环调度

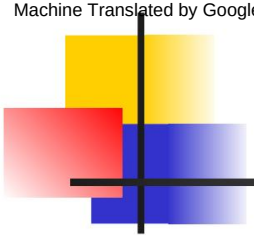
图 2-41 循环调度。(a)可运行进程列表。
(b)B 用完其时间片后的可运行进程
列表。



循环调度

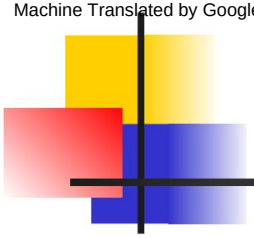
4 毫秒的时间段

平均重量=(6+4+7)/
3=5.66



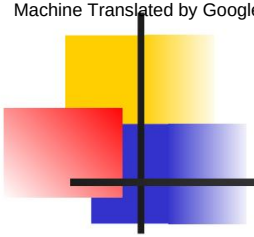
量子值

较小的时间量如何增加上下文切换



优先级调度

图。具有四个优先级类别的调度算法。



多级反馈队列调度

理念

具有不同 CPU 突发特性的独立进程

允许进程在队列之间移动

多级队列调度

前台（交互)进程后台（批处理)进程

周转时间

多级反馈队列调度

调度器由下列参数定义：

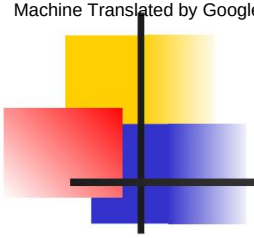
队列数量

各个队列的调度算法

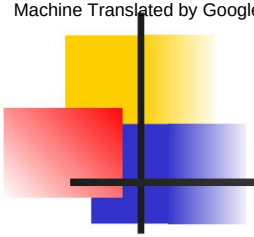
用于确定何时将进程升级到更高优先级队列的方法

用于确定何时将进程降级到较低优先级队列的方法

当进程需要服务时,用于确定该进程进入哪个队列的方法



彩票安排



更多调度算法

保证调度

公平份额调度

…

实时系统中的调度

第一类

硬实时

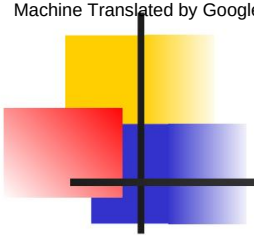
软实时

类别 II 定期

非周期性类别

III 静态

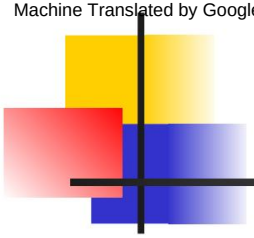
动态



实时 CPU 调度

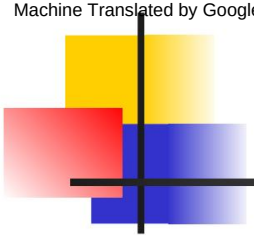
Minimizing Latency 最小化延迟

中断延迟、调度延迟



实时 CPU 调度基于优先级的调度

例如:P2 的优先级比 P1 高



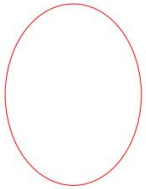
实时 CPU 调度单调速率调度单一速率

具有抢占功能的静态优先级策略

例子：
P1的优先级高于P2；P1的周期比
P2的周期短

实时 CPU 调度最早截止时间优先调度根

据截止时间动态分配优先级



采用单调速率调度时错过截止期限

最早截止时间优先调度

leexudong@nankai.edu.cn

实时 CPU 调度

- 按比例的份额调度
  - 比例共享调度器通过在所有应用程序之间分配 T 份额来运行
  - 一个应用程序可以获得N份时间，从而确保应用程序拥有总处理器时间的 N/T

leexudong@nankai.edu.cn

实时 CPU 调度

POSIX 实时调度

SCHED_FIFO

SCHED_RR

SCHED_OTHER

pthread attr_getsched_policy(pthread_attr_t *attr， int *policy)

pthread attr_setsched_policy（pthread_attr_t *属性， int 策略)

leexudong@nankai.edu.cn

多处理器调度

多 CPU

负载共享成为可能　但调度

问题也相应变得更加复杂

多处理器调度方法

非对称多处理

所有调度决策、I/O 处理和其他系统活动均由单个处理器（主服务器）处理。 其他处理器仅执行用户代码。

对称多处理（SMP）

每个处理器都是自调度的。所有进程可能位于一个公共就绪队列中，或者每个处理器都有自己的就绪进程专用队列。

leexudong@nankai.edu.cn

多处理器调度

- Processor Affinity 处理器亲和性

  考虑当一个进程在特定处理器上运行时,缓存内存会发生什么情况。
  进程最近访问的数据填充了处理器的缓存。

  因此,进程的连续内存访问
  通常在高速缓存中满足。

处理器亲和性的尊重形式

软亲和力

当操作系统有策略试图保留

在同一处理器上运行的进程 但不保证它会这样做

硬亲和性

sched_setaffinity() 系统调用

leexudong@nankai.edu.cn

Linux 调度程序

/kernel/sched/core.c

静态 void __sched notrace __schedule(bool preempt)

*

静态 __always_inline struct rq

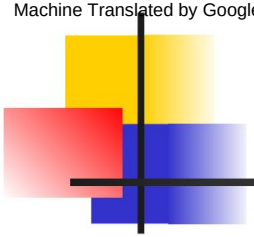
context_switch(struct rq *rq, struct task_struct *prev,

结构 task_struct *next,结构 rq_flags *rf)

静态内联结构 task_struct *

pick_next_task（结构rq *rq,结构 task_struct *prev,结构 rq_flags *rf)

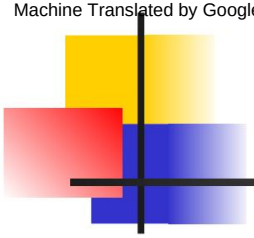
leexudong@nankai.edu.cn



Linux 调度程序（续）

pick_next_task

leexudong@nankai.edu.cn



线程调度

两级并行线程调度程序

用户级线程

内核级线程

（超线程）

线程调度

用户级线程:进程争用范围（PCS）
进程范围

(a)可能以 50 毫秒的间隔调度用户级线程
每 CPU 突发运行 5 毫秒的进程量子和线程。

线程调度

内核级线程:系统争用范围(SCS)

(b)可能使用相同的调度机制来调度内核级线程
特征为（a）。

线程调度案例

```
#包括 <pthread.h> #包括 <stdio.h>
#定义线程数 5
int main(int argc， char *argv[]){
 int 我；
pthread_t tid[NUM THREADS];
pthread_attr t attr; /*
获取默认属性*/ pthread_attr
init(&attr); /*设置调度算法为
PROCESS或SYSTEM */ pthread_attr setscope(&attr,
PTHREAD_SCOPE_SYSTEM);
/* 设置调度策略 - FIFO、RT 或其他 */
pthread attr setschedpolicy(&attr， SCHED OTHER);
/* 创建线程 */
对于 (i = 0; i < NUM THREADS; i++)
pthread_create(&tid[i],&attr,runner,NULL);
```

线程调度案例

```
/*现在加入每个线程*/for (i = 0; i
< NUM THREADS; i++) pthread_join(tid[i],
   NULL);
}

 /* 每个线程将在此函数中开始控制 */ void *runner(void
*param) {

    printf(    我是一个线程\n    );
    pthread_exit(0);
}
```

线程调度

P线程调度

PTHREAD_SCOPE_PROCESS使用 PCS 调度来调度线程

PTHREAD_SCOPE_SYSTEM使用 SCS 调度来调度线程

pthread attr setscope(pthread attr t *attr， int scope)

pthread attr getscope(pthread attr t *attr， int *scope)

政策与机制

- 调度机制
- 调度机制
- 调度策略
- 调度策略

概括

调度器

过程行为

调度模式调度准则调度

算法线程调度

...

问答?