109590004 呂育瑋 作業系統 HW2 hand-written part

4.1

多線程適合高密集計算與大量IO操作，因此以下兩種情形並不會比單線程較率還高：

1. 高密集計算、少量IO操作：在計算上相比單線程，需要額外的線程管理與增加上下文切換與同步開銷。
2. 低密集計算、大量IO操作：這時跟CPU性能較無關，而是IO操作速度，而單線程能更好發揮IO操作的異步性能。

4.3

所有線程都共享：Heap memory, Global variables

各線程單獨持有：Register values, Stack memory

4.4

在單一處理器上運行多線程會需要將線程作時間切割來共享CPU資源，增加了上下文切換與線程調度成本，降低了效率。

在多處理器上運行多線程能讓不同線程在不同CPU上同時執行，節省上下文切換與線程之間的等待，有比單一處理器更好的效能表現。

5.2(a)

CPU利用率越高代表可以將更多進程調度到CPU上執行，一個進程長時間使用CPU時，會讓其他進程的響應時間便久。

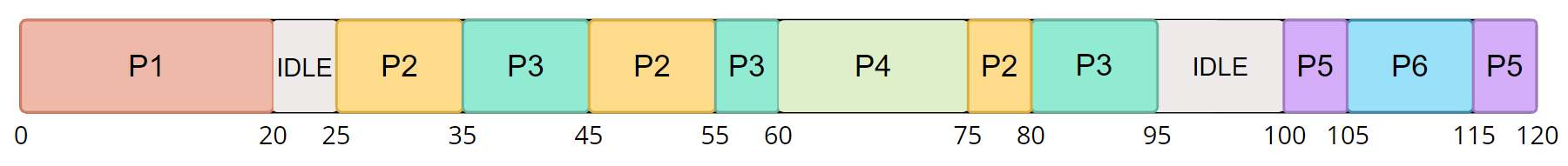
5.2 (b)

透過優先調度短進程讓平均周轉時間縮短，但可能讓長周轉時間的進程等待時間變長。

5.2 (c)

如果I/O使用率高，進程在等待I/O操作完成前，CPU不能使用I/O設備，CPU利用率降低；CPU利用率高則I/O使用率降低，I/O較常處於閒置狀態，會導致進程要等I/O操作完成才能繼續執行，造成更久的等待時間。

5.8(a)



一張含有 資料表 的圖片

自動產生的描述5.8(b)

各進程周轉時間

P1 = 20 – 0 = 20

P2 = 80 – 25 = 55

P3 = 95 – 35 = 60

P4 = 75 – 60 = 15

P5 = 120 – 100 = 20

P6 = 115 – 105 = 10

(c) What is the waiting time for each process?

(d) What is the CPU utilization rate?

5.8(c)

各進程等待時間

P1: 0 – 0 = 0

P2: 25 – 25 = 0

P3: 30 – 25 = 5

P4: 60 – 60 = 0

P5: 100 – 100 = 0

P6: 105 – 110 = 0

5.8(d)

CPU利用率 = (105 / 120) \* 100%

Sdadw

5.14: Consider a preemptive priority scheduling algorithm based on dynamically changing priorities.

•Larger priority numbers imply higher priority

•When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α

•When it is running, its priority changes at a rate β

•All processes are given a priority of 0 when they enter the ready queue

•The parameters α and β can be set to give many different scheduling algorithms

(a) What is the algorithm that results from β > α > 0?

(b) What is the algorithm that results from α < β < 0?

Sdadw

5.15: Explain the differences in how much the following scheduling algorithms discriminate in favor of short processes:

(a) FCFS (b) RR (c) Multilevel feedback queues

Sdadw

6.4: Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in user-level programs.

Sdadw

6.6: The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.

Sdadw

6.10: The implementation of mutex locks provided in Section 6.5 suffers from busy waiting.

–Describe what changes would be necessary so that a process waiting to acquire a mutex lock would be blocked and placed into a waiting queue until the lock became available.

Sdadw