Tutoraggio di Sistemi Operativi

Matteo Federico

Lezione #5

TOR VERGATA

UNIVERSITÀ DEGLI STUDI DI ROMA

In the last Episode

Raccolta di frutta

Scrivere un programma C che simula la raccolta della frutta.
Il main thread deve creare N thread, divisi in 3 gruppi, ognuno addetto alla raccolta di un tipo di frutto diverso: mele, pere ed arance.
Dopo aver creato i thread figli il main thread si mette a generare randomicamente i diversi tipi di frutta in una singola variabile condivisa con tutti i thread.
I thread leggono questa variabile condivisa e "raccolgono" la frutta a loro corrispondente aggiungendo il valore del frutto in una loro variabile personale.
Dopo un certo tempo il main thread smette di generare frutta, manda un messaggio di uscita nella variabile condivisa e stampa i valori di frutta raccolti Ignorare problemi di sincronizzazione.

Esercizio Scheduling 2

Scheduling VRR: Esercizio 1 Esame 13/6/22

Si descriva lo scheduling di CPU round-robin virtuale.
Si consideri uno scenario in cui lo scheduler di CPU round-robin virtuale sia basato su un time-slice pari a $K > 10$ millisecondi, esista un unico processo CPU-bound A di durata infinita ed esista un unico processo I/O-bound B di durata infinita. Il processo B ha CPU burst di lunghezza pari a 10 millisecondi.
Si indichi il massimo valore di K che possa permettere al processo B di avere un tempo di attesa per l'esecuzione del CPU burst non più ampio del 250% rispetto alla durata del timeslice.

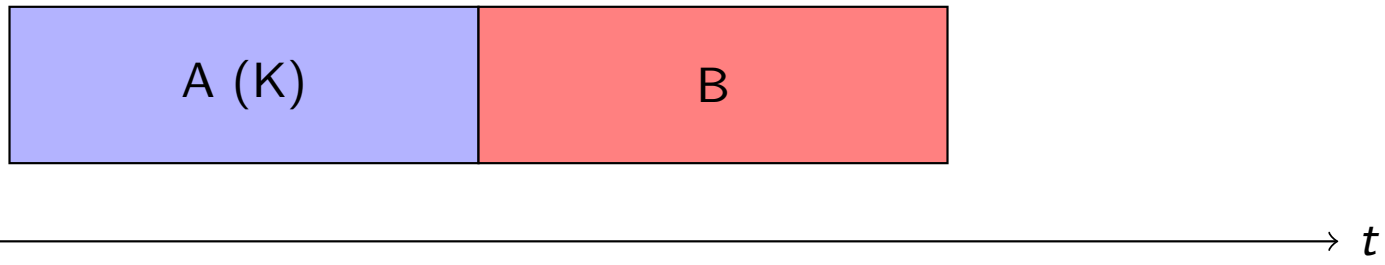
TOR VERGATA

Esercizio VRR 13/6/22 – Esecuzione

Dati

- $K > 10$ ms
- A: CPU-bound B: I/O Bound con CPU burst di 10ms
- $K < 250\% CPU\_Burst(B)$

Timeline (caso peggiore per B)



Osservazione

- B attende K di A

Vincolo

$$K \leq 25 \Rightarrow \boxed{K_{max} = 25 \text{ ms}}$$

Esercizio scheduling Unix

Scheduling: Unix

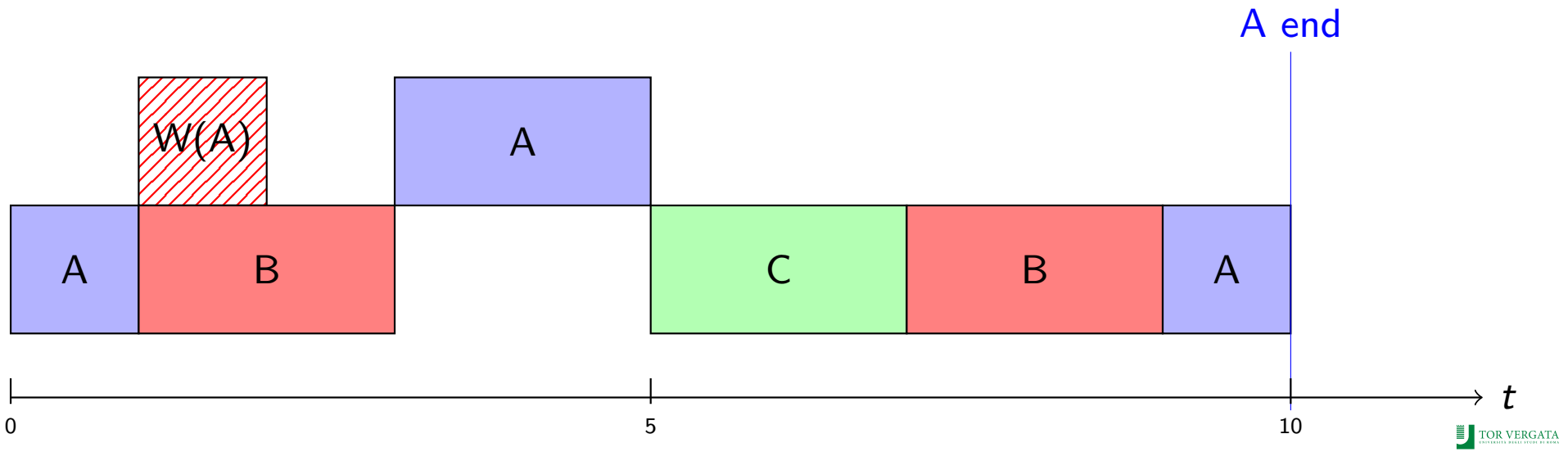
Si descriva lo scheduler di CPU Unix tradizionale. Si consideri inoltre un insieme di 3 processi, A, B e C, entrati in CPU in ordine aventi nice +19, di cui A è I/O Bound con due CPU-burst di 1 e 3 millisecondi prima di terminare, B è CPU bound di durata infinita, e C è I/O con 2 CPU-burst di 3 ms. Si indichi se A possa terminare entro 9 millisecondi considerando un quanto di tempo per lo scheduling di 4 oppure 2 millisecondi, considerando che il tempo di blocco di A e B tra i due CPU-burst è di 1 millisecondo. Per la soluzione dell'esercizio si consideri che il tempo di esecuzione dello scheduler e di altre parti del kernel sia trascurabile, e che ci sia un unico processore.

Esercizio Unix – Esecuzione

Dati

- A: I/O-Bound con 2 CPU-Burst 1ms e 3ms
- B: CPU-Bound
- C: I/O-Bound con 2 CPU-Burst 3ms e 3ms
- WaitTime 1ms
- ordine di arrivo iniziale A,B,C

Timeline (caso K=2ms)



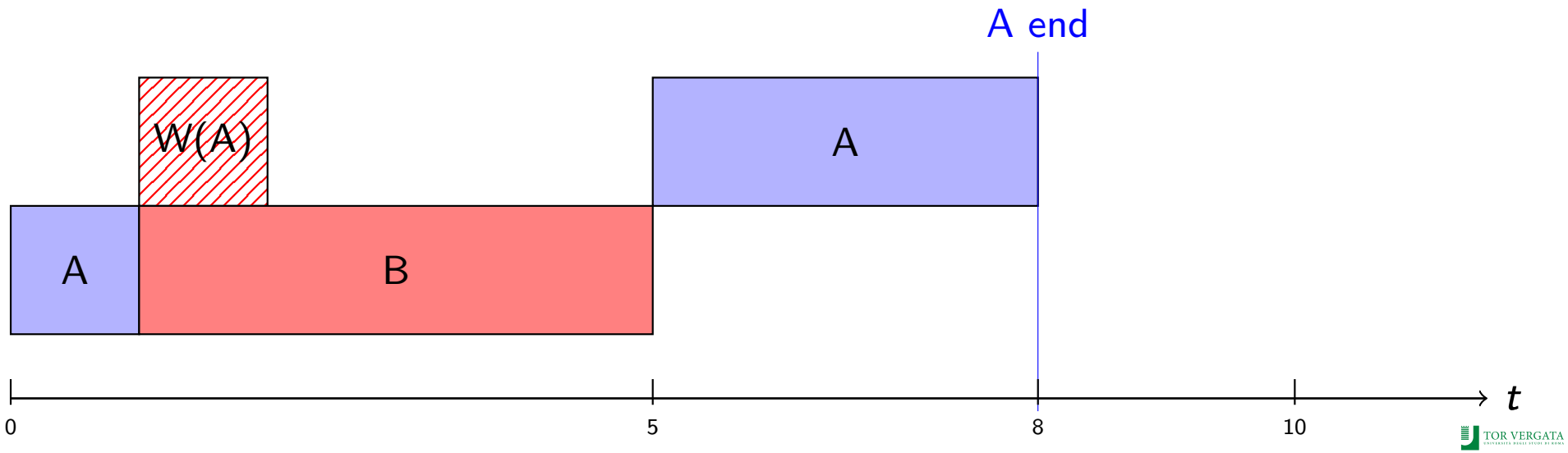
A termina a 10ms!

Esercizio Unix – Esecuzione

Dati

- A: I/O-Bound con 2 CPU-Burst 1ms e 3ms
- B: CPU-Bound
- C: I/O-Bound con 2 CPU-Burst 3ms e 3ms
- WaitTime 1ms
- ordine di arrivo iniziale A,B,C

Timeline (caso K=4ms)



A termina a 8ms!

Esercizio Scheduling 8/9/2022

Esercizio 1 - 8/9/2022

Si descriva lo scheduler di CPU multi-level feedback-queue.
Inoltre, si consideri uno scenario in cui all'istante T0 esistano 2 processi P1 e P2. Il processo P1 é CPU-bound di durata infinita, mentre il processo P2 é I/O-bound ed interagisce infinite volte con un dispositivo D prima di completare la sua esecuzione.
Il CPU-burst di P2 sia di 5 millisecondi. Nel caso lo scheduler multi-level feedback-queue abbia 2 livelli di prioritá, si determini motivando la risposta quale sia il massimo valore di time-slice che garantisca un tempo di attesa di P2 non superiore a 10 millisecondi. Si assuma che la latenza di esecuzione dello scheduler di CPU sia nulla.

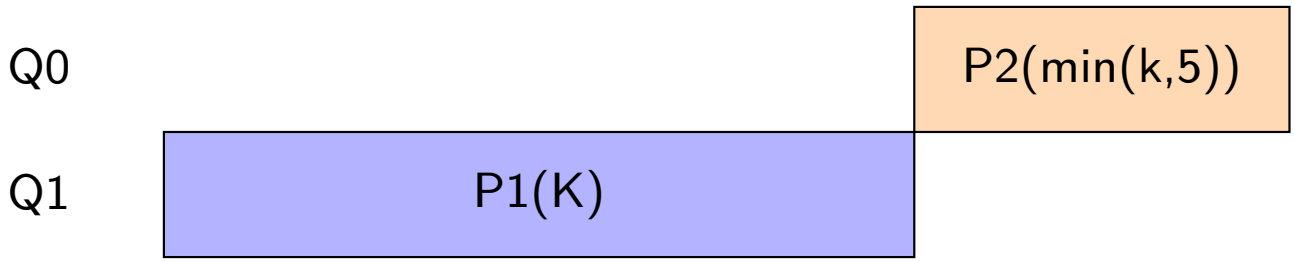
TOR VERGATA

MLFQ – 8/9/2022

Assunzioni

- P1: CPU-Bound
- P2: I/O-Bound con CPU-Burst di 5ms
- Q0 quanto K
- Q1 quanto 2*K
- wait(P2)<10ms

Timeline



Osservazione:

- il tempo di attesa di iniziare P2 dipende solo da P1
- e il tempo di esecuzione di P1 dipende in quale livello sta eseguendo

nel caso pegiore il tempo di attesa maggiore avviene se P1 si trova nel livello piu basso.

$$2 * K < 10 \rightarrow K < 5 \rightarrow Kmax = 5$$

Esercizio Scheduling 19/7/2021

Esercizio 1 - 19/7/2021

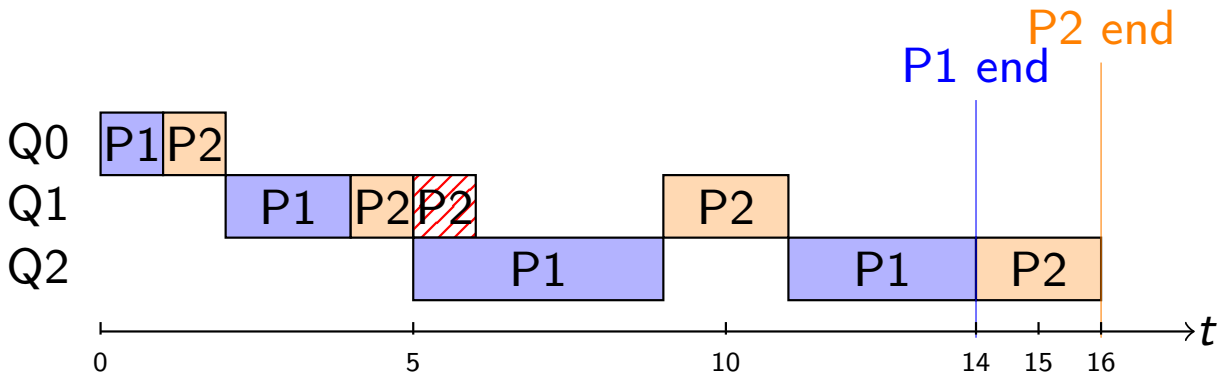
Si descriva lo scheduler di CPU multi-level-feedback-queue. Inoltre, supponendo che il quanto di tempo base di tale scheduler sia pari a 1 millisecondo e che al tempo $T = 0$ siano creati in ordine 2 processi P1 e P2 tali che P1 abbia un solo CPU-burst di durata pari a 10 millisecondi, mentre P2 abbia due differenti CPU-burst di durata pari a 2 e 4 millisecondi, si determini il tempo di completamento di P2 considerando il caso in cui lo scheduler abbia 3 differenti livelli di priorità e tra i suoi due CPU-burst P2 rimanga in stato "wait" per 0.5 millisecondi. Si assuma che la latenza per eseguire attività di sistema operativo sia trascurabile

MLFQ – 19/7/2021

Assunzioni

- P1: CPU-burst 10ms
- P2: 2 CPU-Burst: 2ms e 4ms e wait time di 0,5ms
- Q0 quanto 1
- Q1 quanto 2
- Q2 quanto 4

Timeline



Risposta: P2 termina a T=16ms

Esercizio Scheduling 21/2/2022

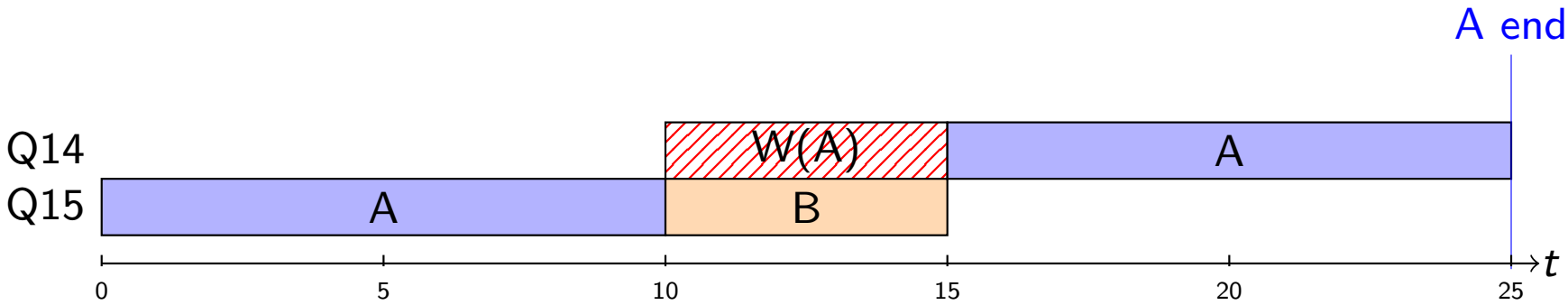
Esercizio 1 - 21/2/2022

Si descriva lo scheduler di CPU Windows. Si consideri inoltre un insieme di 2 processi A e B single-thread appartenenti alla classe "variable". Il processo A esegue 2 burst in CPU di 10 millisecondi ciascuno, andando in blocco al termine di ciascun burst per un tempo pari a 5 millisecondi. Il processo B è CPU bound di durata infinita. Supponendo che all'istante T0 i due processi/thread abbiano la stessa priorità minore di 15, e che il processo A sia schedulato per primo in CPU, si determini il tempo di completamento di A supponendo che il sistema sia fornito di una sola unità di processamento, che il costo di esecuzione della attività del sistema operativo Windows sia trascurabile e che il time-slice utilizzato dallo scheduler di CPU sia 20 millisecondi.

Assunzioni

- A: I/O-Bound con due burst da 10ms e un wait wait Time di 5ms
- B: CPU-Bound
- Priorità(A,T0)=Priorità(B,T0)=15

Timeline



Risposta: A termina a 25ms!

Esercizio Scheduling 18/6/2020

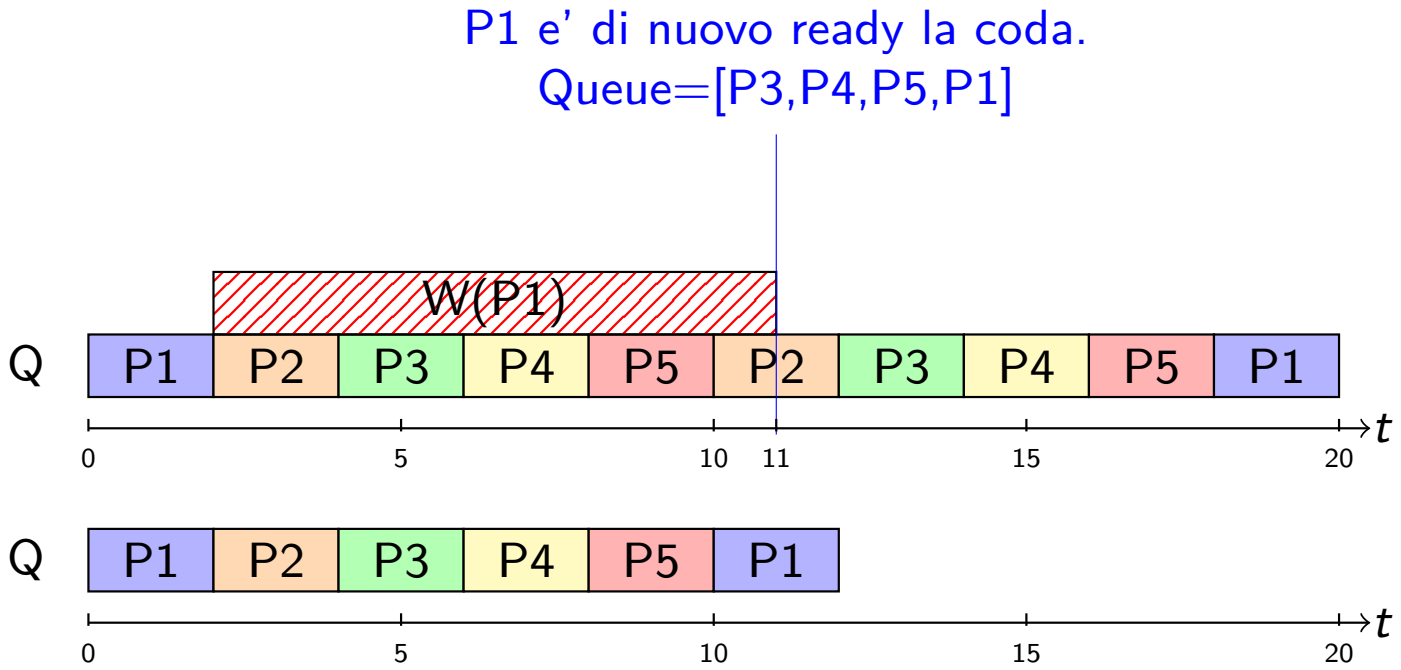
Esercizio 1 - 18/6/2020

Descrivere l'algoritmo di di scheduling di CPU Round-Robin (RR), discutendo l'impatto della scelta del time-slice sul comportamento dei processi I/O bound.
Si consideri inoltre il caso in cui vengono attivati allo stesso istante temporale 5 processi P1, ... , P5. P1 e' un processo I/O bound che interagisce con un dispositivo D per N=10 volte e poi termina (il processo termina non appena l'ultima interazione e' completata).
Tutti gli altri processi sono CPU bound di durata infinita.
Supponendo che (1) la lunghezza di ogni CPU burst di P1 prima di una nuova richiesta di I/O sia nota a priori e pari a 5 ms., (2) il dispositivo D impieghi 9 ms. per ogni interazione, (3) il tempo di CPU per il dispatcher RR sia nullo.
Si determini (motivando la risposta) quale tra i seguenti valori del time-slice minimizza il tempo di attesa di P1 in caso di dispatching RR: 2 ms., 6 ms., 18 ms.
Si calcoli inoltre il tempo di attesa almeno nel caso del time-slice che minimizza tale tempo

Assunzioni

- P1: I/O-Bound : per 10 volte esegue 5ms e attende 9ms.
- P2,P3,P4,P5: CPU-Bound

Timeline (2ms)

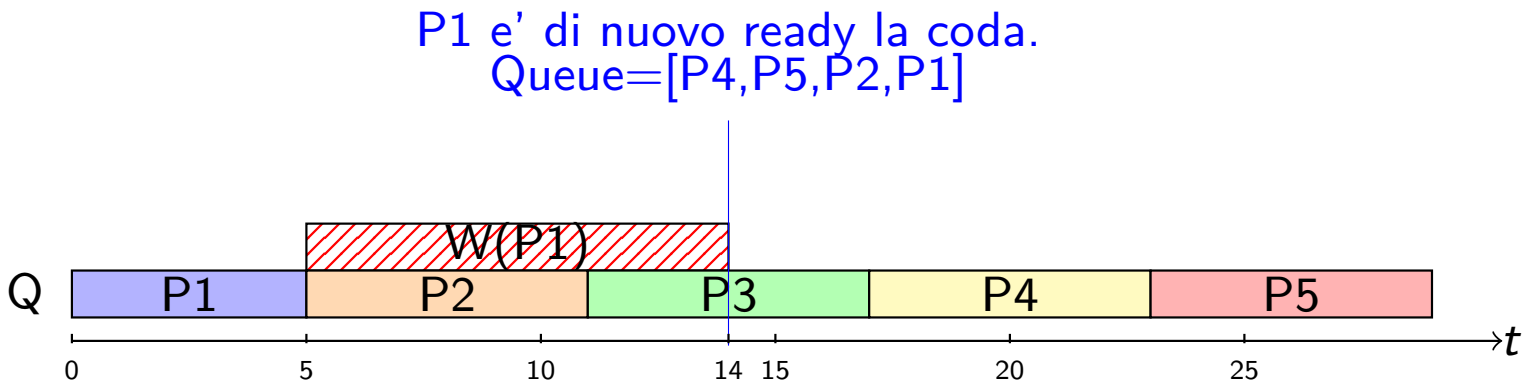


Risposta: l'attesa massima in questo caso é 8ms!

Assunzioni

- P1: I/O-Bound : per 10 volte esegue 5ms e attende 9ms.
- P2,P3,P4,P5: CPU-Bound

Timeline (6ms)

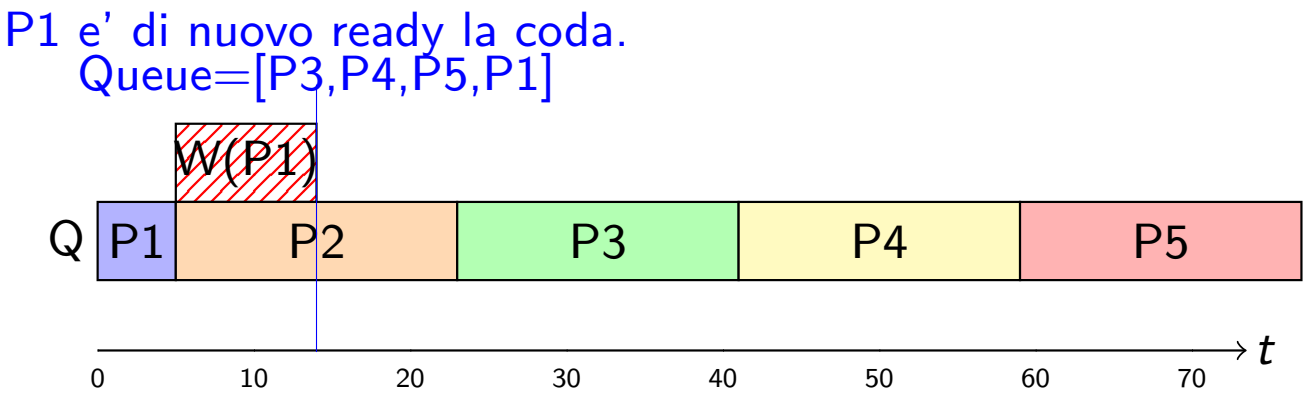


Risposta: l'attesa massima in questo caso é 21ms!

Assunzioni

- P1: I/O-Bound : per 10 volte esegue 5ms e attende 9ms.
- P2,P3,P4,P5: CPU-Bound

Timeline (18ms)



Risposta: l'attesa massima in questo caso é 63ms!

Timing (Linux & Windows)

Linux

- `gettimeofday()`
  - Tempo reale (wall-clock)
  - Risoluzione $\sim \mu$s
  - influenzato da cambi di sistema (NTP, clock)
- `clock_gettime(CLOCK_MONOTONIC)`
  - Tempo monotono (non retrocede)
  - Alta precisione (ns)
  - stabile per misure temporali
- `rdtsc`
  - Conta cicli CPU
  - altissima risoluzione

Windows

- `QueryPerformanceCounter()`
  - Alta precisione (hardware counter)
  - equivalente a $\texttt{clock}_g\textit{ettime}$
  - `stabile tra core`
- `GetTickCount()`
  - Tempo in ms dall'avvio
  - semplice ma bassa precisione

Sintesi

- Benchmark preciso: `rdtsc`
- Misure robuste: `clock_gettime`/ QPC
- Tempo Attuale: `gettimeofday`

TOR VERGATA

Linux: API usate nei thread

Gestione CPU e scheduling

- `sched_setaffinity(pid, size, mask)`
  - Imposta l'affinità del thread (CPU su cui può eseguire)
  - Usato per forzare la migrazione tra core
- `sched_getcpu()`
  - Restituisce la CPU corrente su cui si sta eseguendo! attenzione puó portare ad errori!

Nota:

- L'affinità non è immediata $\rightarrow$ migrazione asincrona
- Lo scheduler decide quando spostare il thread

TOR VERGATA

Windows: API equivalenti

Gestione CPU

- `SetThreadAffinityMask(handle, mask)`
  - Imposta la CPU su cui il thread può eseguire
  - Analogo a `sched_setaffinity`
- `GetCurrentProcessorNumber()`
  - Restituisce la CPU corrente
  - Analogo a `sched_getcpu`
- `Sleep(10)`
  - Sospende il thread per 10 ms
  - Simile a `usleep`

Nota:

- Anche qui l'affinità non è immediata
- Limite classico: 64 CPU (mask a 64 bit)

TOR VERGATA

Challenge dello Scheduling

PowerNap dei thread

Creare un programma che: Dato un numero N, crea N thread, ogni thread dovrá dormire su tutte le cpu del processore.
Su ogni CPU ogni thread dovrá dormire 5ms.
quando un thread si sveglia dovrá estrearre randomicamente la prossima cpu su cui dormrie!
I thread termina quando ha visitato tutte le cpu.
Il main thread dopo aver creato i thread rimane in attesa fin tanto che tutti i thread creati non terminano.

TOR VERGATA

Side Quest

Side Quest 1

Misurare il tempo impiegato da ogni thread e far stampare al main thread la classifica.

Potete inviare le vostre soluzioni a
matteo.federico@uniroma2.it.
Scrivete nell oggetto della mail [SO] challenge Scheduling.
Vedremo le più interessanti (in positivo e negativo) la prossima lezione.
(Sarò anonimo nel mostrare le soluzioni)

TOR VERGATA