



# EVALUAREA PERFORMANȚEI ALGORITMIILOR DE PLANIFICARE A PROCESELOR

Coordonator științific  
Conf. univ. dr. ing. Paul Irofti

Absolvent  
Stan Cătălin-Andrei

# Motivație

Procesul este entitatea cea mai importantă din cadrul unităților de calcul, sistemul de operare fiind răspunzător pentru gestionarea a zeci de mii de procese, astfel modul în care se alocă resursele devine un factor cheie în performanța unei astfel de unități. Prin urmare, un instrument prin care putem obține rapid o evaluare preliminară a unui algoritm se dovedește a fi o necesitate, iar ușurința cu care acesta este folosit cât și calitatea rezultatelor fiind elementele vitale ce diferențiază astfel de instrumente.

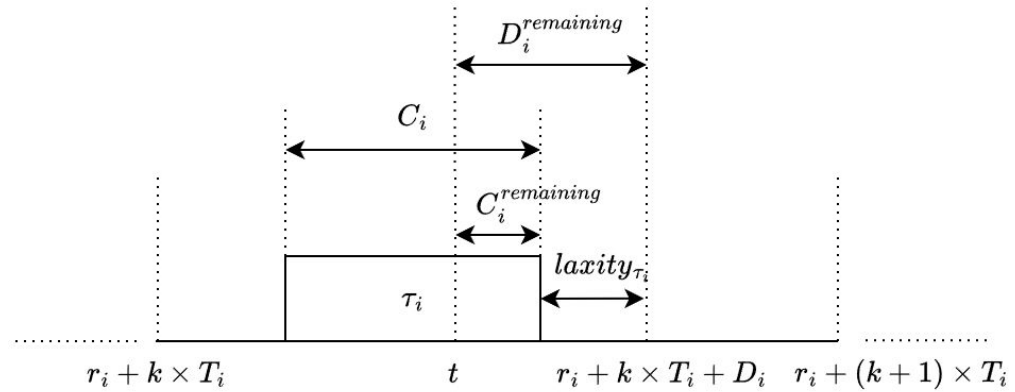


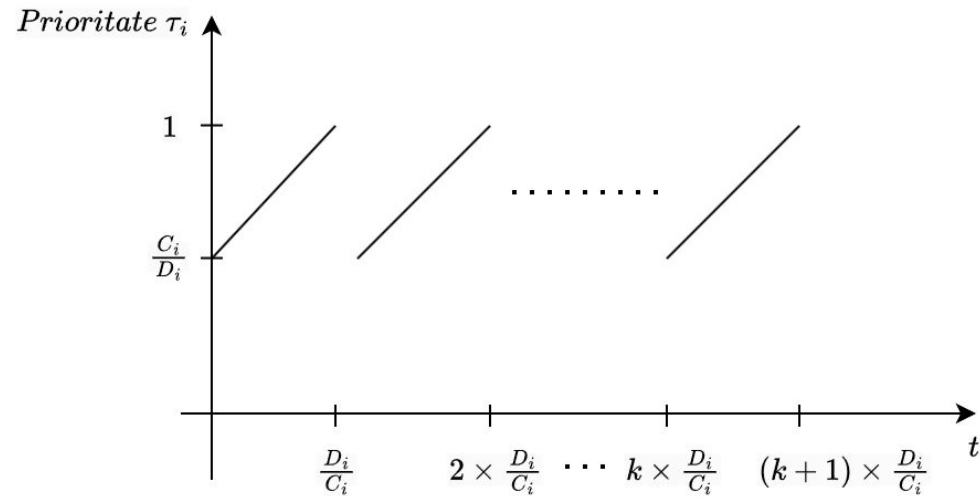
# Algoritmi - Single Core, Multi Core

- Efficient Dynamic Round Robin
- Mean Threshold Shortest Job Round Robin
- Min-Max Round Robin
- Best Time Quantum Round Robin
- Fair-Share Scheduling

# Algoritmi - Real Time

- Least Slack Time Rate first
- DynAmic Real-time Task Scheduling
- Linear Task Scheduling






Linear Task Scheduling

# Simulator

Evenimente

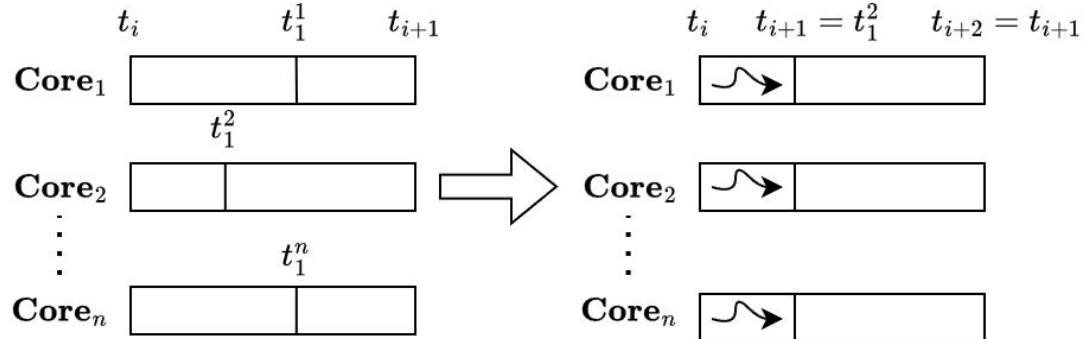
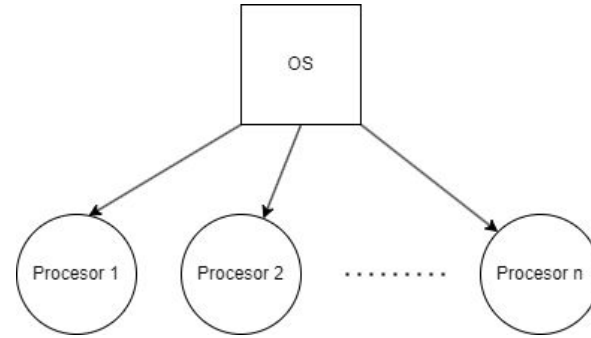
```
class Event {  
private:  
    eventType type;  
    int time;  
    Process process;
```

```
enum eventType {  
    ARRIVAL,  
    CPUBURSTCOMPLETE,  
    TIMEREXPIRED,  
    PREEMT,  
    REALTIME,  
    TICK,  
    LOADBALANCE,  
};
```



Este implementat un simulator de evenimente discrete. Există o entitate globală care simulează intrările proceselor în sistem, iar fiecare **Core** simulează execuția.

# Simulator



# Generare date

Aleator sau din fișier

## Single Core + Multi Core

Sunt puse la dispoziție maxim 3 distribuții normale configurabile de utilizator pentru generarea timpilor de execuție

Timpii de intrare și prioritățile generate din distribuții uniforme

## Real Time

Intrare în sistem la timpul 0

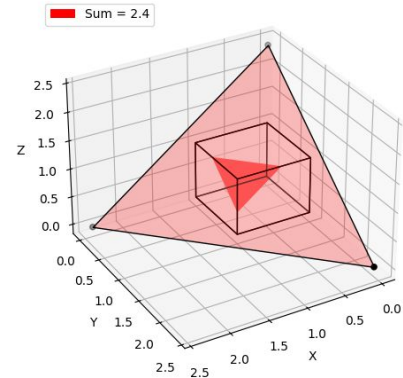
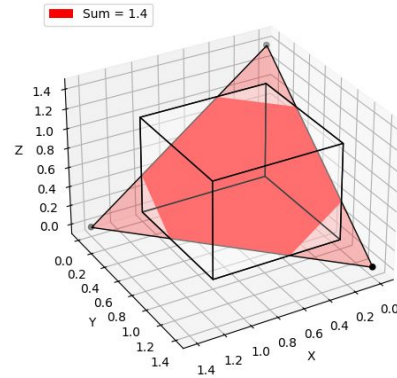
Perioada este egală cu deadline-ul

Timpii de execuție și perioade generate cu *randfixedsum*



# Randfixedsum

$$u_i = \frac{C_i}{D_i}, U = \sum_i u_i$$



# Evaluare

- Timpi de răspuns, de așteptare.
  - Număr schimbări de context
  - Turnaround time
- Pentru o utilizare țintă  $U$  câte mulțimi reușește algoritmul să planifice
- Diferența maximă de virtual runtime

# API

- virtual std::vector processArrived(std::vector p, int time, Metrics &stats)
  - virtual std::vector processCPUComplete(Process p, int time, Metrics &stats)
  - virtual std::vector schedule(int time, Metrics &stats, bool timerExpired)
- 
- virtual int loadBalance(int time)
  - virtual int assignCPU(Process p)



Multumesc!