3.36pt

Sistemas Operacionais

André Luiz da Costa Carvalho

andre@icomp.ufam.edu.br

Aula 04 - Escalonamento de Processos

Aula de Hoje

3.36pt

1. Princípios do Escalonamento
   - Workload
   - Métricas de Escalonamento
2. Politicas de Escalonamento
   - FIFO
   - Shortest Job First
   - STCF
3. Preocupação com tempo de resposta
   - Round Robin

Aula de Hoje

3.36pt

1. Princípios do Escalonamento
   - Workload
   - Métricas de Escalonamento
2. Politicas de Escalonamento
   - FIFO
   - Shortest Job First
   - STCF
3. Preocupação com tempo de resposta
   - Round Robin

Introdução

Tudo que vimos até agora foi a respeito do **mecanismo** de troca de contexto.

Precisamos entender agora as políticas de escalonamento (ou disciplinas) que governam estas trocas. O escalonamento.

O termo escalonamento é anterior à computação; vindo das áreas de administração e engenharia de produção.

Pergunta básica: Como desenvolver uma política de escalonamento? O que devemos assumir? Que métricas são importantes? Quais eram as abordagens mais simples?

Carga de Trabalho

- A carga de trabalho (ou **workload**) é o conjunto de características de um trabalho que deve ser efetuado.
  - No caso do escalonamento, são os tipos de processos que devem ser rodados pelo S.O.
  - As características destes processos são essenciais para definir quais são as políticas de escalonamento mais adequadas.

- Nesta aula, começaremos com workloads pouco realísticos, e conforme estudamos políticas de escalonamento mais complexas, com mais variações no estilo dos processos que serão escalonados.

Workload inicial para hoje

1. Todos processos precisam rodar pela mesma quantidade de tempo para finalizar.
2. Todos os processos chegam no escalonador ao mesmo tempo.
3. Não há preempção, cada processo roda até terminar.
4. Todos os processos só usam a CPU (sem E/S)
5. O tempo necessário para a execução de um processo é conhecido.

Ainda na aula de hoje relaxaremos algumas destas características.

Métricas de Escalonamento

Uma parte importante da engenharia de um escalonador é uma forma objetiva de medir a qualidade do escalonamento: uma **métrica**.

Uma métrica é uma forma de medir algo, e em escalonamento podemos ter diversas métricas.

Métricas comuns:

- Tempo de execução (Turnaround)
- Tempo de espera
- Tempo de resposta
- Justiça
- Eficiência

Turnaround

O Tempo de execução (ou turnaround) é o tempo que o processo passou no sistema desde a sua criação.
Ou seja, é o tempo em que o processo terminou menos o tempo em que ele chegou no sistema:

$$T_{turnaround} = T_{completo} - T_{chegada}$$

No momento, como todos processos chegam juntos, $T_{chegada} = 0$, então o tempo de execução $T_{turnaround} = T_{completo}$.

Isto mudará mais adiante. Veremos outras métricas conforme as aulas alcançarem políticas de escalonamento mais complexas.

Aula de Hoje

3.36pt

1. Princípios do Escalonamento
   - Workload
   - Métricas de Escalonamento
2. Politicas de Escalonamento
   - FIFO
   - Shortest Job First
   - STCF
3. Preocupação com tempo de resposta
   - Round Robin

FIFO

A idéia mais simples para o escalonamento de processos é **uma fila**. Ou seja, os processos são escalonados por ordem de chegada.

Também conhecido como First Come,First Served - FCFS.

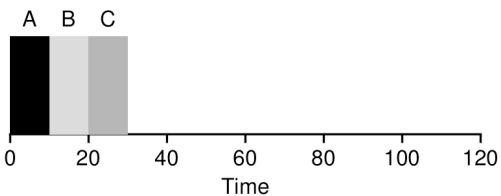
Tem a vantagem de ser extremamente simples de implementar, e se adequar bem ao workload simples que assumimos até agora.

Exemplo

3 processos A, B e C chegam virtualmente ao mesmo tempo ($T_{chegada} = 0$). Como a fila depende da ordem, digamos que A chegou ligeiramente a frente de B e B a frente de C, e que os processos rodam por 10 segundos. Qual será o tempo de execução (turnaround)?

Exemplo

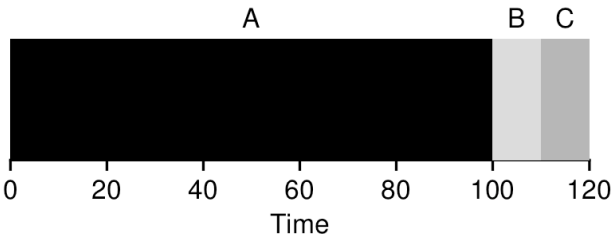
3 processos A, B e C chegam virtualmente ao mesmo tempo ($T_{chegada} = 0$). Como a fila depende da ordem, digamos que A chegou ligeiramente a frente de B e B a frente de C, e que os processos rodam por 10 segundos. Qual será o tempo de execução (turnaround)?



Tempo de execução

Agora assuma que os processos podem demorar tempos diferentes para terminar de executar. Como é a performance do FIFO? Qual workload pode levar ele a ter um desempenho ruim?

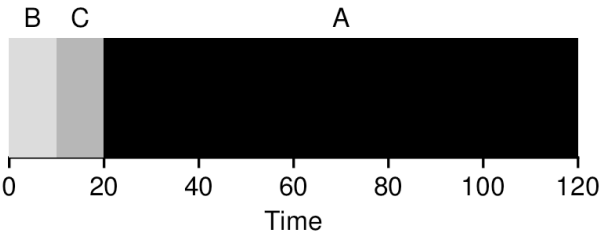
FIFO



Efeito de comboio (convoy efect): Onde poucos consumidores leves ficam presos esperando poucos consumidores peso-pesado.

Shortest Job First - SJF

Processo mais curto primeiro: o escalonador escolhe o processo que precisa de menos tempo para completar, depois o segundo, seguindo nesta ordem. No exemplo anterior:



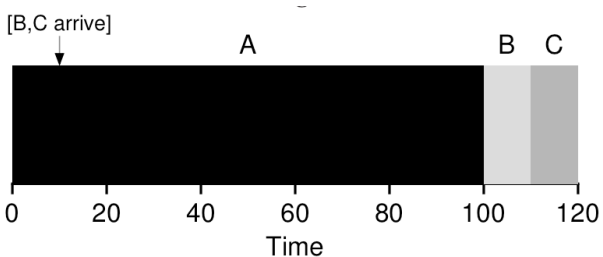
Qual o tempo de execução?

SJF

No nosso workload inicial, pode-se dizer que o SJF é **ótimo**. Contudo, na vida real temos tudo menos um workload organizado como esse.

Por exemplo, que problemas poderiam surgir se não mais assumíssemos que os processos não chegam todos ao mesmo tempo?

SJF



Qual o tempo de espera agora?

Shortest Time-to-Completion-First (STCF)

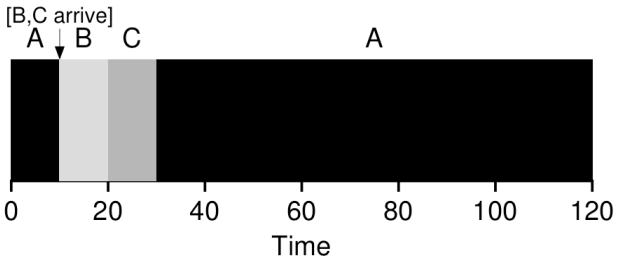
Vamos relaxar a restrição que os processos devem rodar até o fim.

Vamos também adicionar o conceito de interrupção por temporizador e **preempção**.

Shortest Time-to-completion First (STCF) ou Preemptive SJF é a versão com preempção do SJF.

Toda vez que um Processo novo chega no sistema, o escalonador define qual dos processos (incluindo o que acabou de chegar) precisa de menos tempo para acabar.

STCF



Qual o tempo de espera?

Aula de Hoje

3.36pt

1. Princípios do Escalonamento
   - Workload
   - Métricas de Escalonamento
2. Politicas de Escalonamento
   - FIFO
   - Shortest Job First
   - STCF
3. Preocupação com tempo de resposta
   - Round Robin

Métrica:Tempo de resposta

Se pudéssemos estimar o tamanho de um processo, e eles só usassem CPU, e nossa métrica fosse apenas tempo de execução, o STCF seria uma política ótima.

Em antigos sistemas operacionais em *batch* esses algoritmos eram normais.

Contudo, em sistemas com compartilhamento de tempo, é importante para um usuário que os processos que ele inicia comecem o mais rápido possível.

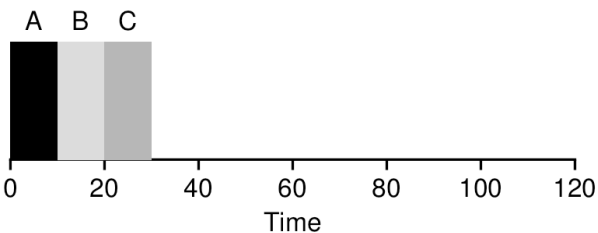
Por isto é utilizada também uma outra métrica: O **tempo de resposta**.

Tempo de Resposta

Tempo de resposta:

$$T_{resposta} = T_{primeiraExec} - T_{chegada}$$

Exemplo: Qual o tempo de resposta do SJF abaixo:

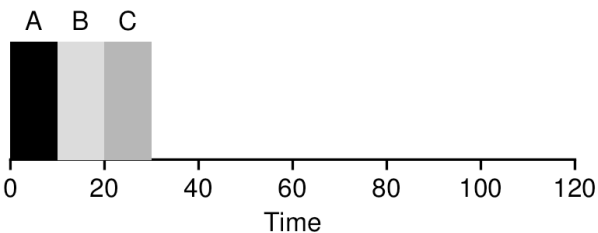


Tempo de Resposta

Tempo de resposta:

$$T_{resposta} = T_{primeiraExec} - T_{chegada}$$

Exemplo: Qual o tempo de resposta do SJF abaixo:



Qual característica de workload é terrível para tempo de resposta nos SJF?

Round Robin

Ao invés de rodar os processos até o fim, eles são rodados em rodízios, de acordo com uma fatia de tempo (o ***quantum***), até que todos processos terminem.

Usa uma **fila circular**.

O quantum deve ser um múltiplo do timer.

Exemplo

- A,B e C chegam ao mesmo tempo, e precisam de 5 segundos.
- SJF roda até o fim.
- *quantum* de 1 segundo.

Qual o tempo de resposta?

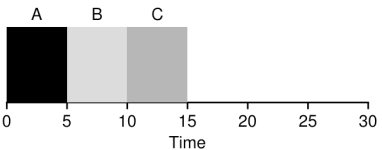


Figure 7.6: **SJF Again (Bad for Response Time)**

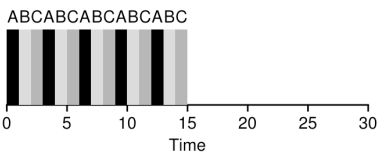


Figure 7.7: **Round Robin (Good for Response Time)**

Quantum

O tamanho do quantum é essencial para o round robin.

Quanto menor, melhor o tempo de resposta.(**Por que?**)

Quantum

O tamanho do quantum é essencial para o round robin.

Quanto menor, melhor o tempo de resposta.(**Por que?**)

Contudo, as trocas de contexto não são gratuitas do ponto de vista do sistema.

Quanto menor o quantum, mais tempo do sistema é gasto com trocas de contexto.

Amortização de Custos

Amortização pode ser usada para diminuir os custos de uma operação fixa.

Fazer a operação menos vezes.

No contexto de escalonamento com time-share, uma forma de amortizar é aumentar o quantum para diminuir os custos das trocas de contextos.

Ex: Se uma troca custa 1ms e o quantum é 10ms, aproximadamente 10% do tempo da CPU será gasto com trocas.

Se o Quantum for 100ms, esse tempo será mais próximo de 1%

Custos de Trocas

O custo de uma troca de contexto vai além de salvar e carregar registradores.

Quando um programa roda, ele usa também vários recursos para cache, TLBs, predição de branch entre outros.

Ao trocar contexto, todos estes estados são perdidos.

Round Robin

Round Robin é excelente para tempo de resposta.

Mas e **tempo de execução**?

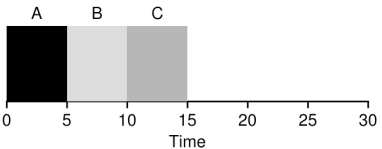


Figure 7.6: **SJF Again (Bad for Response Time)**

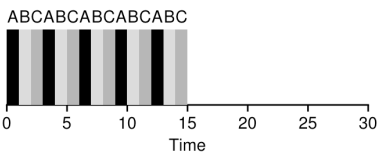


Figure 7.7: **Round Robin (Good for Response Time)**

Conclusões sobre políticas

Round Robin é justo, divide a CPU irmamente entre os processos rapidamente, o que é bom para o tempo de resposta mas ruim para o tempo de execução.

Trade-off entre os dois.

SJF e STCF são bons para execução mas ruins para resposta.

Ainda falta relaxar duas condições no workload (E/S) e tempo conhecido.

Próxima aula :-)