Desenvolvimento e Implementação de uma Arquitetura Multicore com Escalonamento e Gerência de Memória

Michel Pires, Centro Federal de Educação Tecnológica de Minas Gerais November 25, 2024

Observações:

1. O trabalho deve ser realizado individualmente, logo, qualquer indício de cópia será desconsiderada para avaliação.

2. O trabalho entregue deve obrigatoriamente executar em sistema operacional Linux com gcc/g++ na versão 11 ou superior, sendo de responsabilidade do aluno validar possíveis erros de compilação e execução. O processo de compilação deve seguir os padrões pré estabelecidos na disciplina.

3. As entregas devem ser realizadas **impreterivelmente** nas datas estipuladas. Atrasos serão permitidos, porém, implicarão na redução da nota em **50% para cada dia de atraso** ou a cada solicitação de prorrogação por parte do aluno.

Este trabalho visa aprofundar os conhecimentos sobre sistemas operacionais modernos, escalonamento de processos e gerência de memória, por meio de um simulador baseado em um *pipeline* de 5 estágios (*IF, ID, EX, MEM* e *WB*) implementado com o **MPIS**. A seguir, são apresentadas as etapas de desenvolvimento, as quais estão estruturadas em módulos que permitem uma abordagem incremental e modular do aprendizado.

Módulo 1: Arquitetura Multicore e Suporte a Preempção

Objetivo: Ampliar as funcionalidades do simulador para incluir suporte a uma arquitetura multicore e a execução de tarefas preemptivas, proporcionando uma simulação mais realista do comportamento para um sistemas operacionais modernos.

1. **Descrição das Alterações:**

   - Expandir o simulador para suportar múltiplos núcleos (i.e., *cores*), permitindo que cada núcleo execute processos de forma independente e simultânea.
   - Implementar a preempção, possibilitando a interrupção de processos em execução e a troca por outros, conforme definido pelas decisões do sistema operacional. Para fins de simulação inicial, a preempção será realizada com base no término do *quantum* alocado a cada processo.

- Adicionar os estados de ciclo de vida de processos, conforme descrito por Tanenbaum: *pronto*, *bloqueado* e *executando*.
- Integralizar o *Process Control Block* (PCB) ao simulador para armazenar as informações críticas de cada processo. O PCB deverá incluir, no mínimo:
  - Dados do processo (e.g., ID, estado atual, prioridade, etc..);
  - Informações de memória (e.g., endereço base, limite de alocação, banco de registradores, etc ..);
  - Dados sobre arquivos abertos e outros recursos associados ao processo.

2. **Conceitos Necessários:**

   - **Arquitetura Multicore:** Compreender os fundamentos de paralelismo e compartilhamento de recursos. A implementação utilizará `pthreads` para simular a execução concorrente. Será necessário considerar mecanismos como exclusão mútua, regiões críticas e sincronização para evitar condições de corrida e garantir a consistência dos dados compartilhados. Nesse sentido, tente modelar o simulador para abstrair esses conceitos, sendo possível aplicá-los sempre que necessário durante a execução de cada processo. Há duas alternativas a serem consideradas: (a) definir as regiões críticas antecipadamente, utilizando as mesmas durante a execução dos processos ou; (b) definir abstrações no simulador que podem ser chamadas pela linguagem de programação, o que gera o controle necessário quando isso for de fato exigido.
   - **Preempção:** Implementar a lógica de interrupção e troca de contexto, utilizando uma lista de identificadores únicos de processos (*IDs*) e seus respectivos PCBs. Esta lista será gerenciada pelo sistema operacional, que decidirá quais processos deverão ser executados. A ordem inicial que vamos considerar é a FCFS (i.e., *First Come First Service*).
   - **Estados do Ciclo de Vida dos Processos:**
     - Processos no estado *bloqueado* deverão retornar à fila de prontos após o término do período de bloqueio. Esses processos sempre entram no final da fila.
     - O sistema operacional deverá gerenciar solicitações de recursos, como I/O. Caso o recurso solicitado esteja ocupado, o processo será bloqueado até que o recurso seja liberado. Assim que for liberado, o recurso deve ser reservado para o processo (utilizar para isso o PCB) e o desbloqueio é realizado, retornando o processo para a lista de prontos.
     - O ciclo de vida do sistema operacional será implementado utilizando múltiplas threads, permitindo que tarefas como verificação de solicitações de recursos e gerenciamento de processos prontos/executando sejam realizadas de forma independente e em paralelo.

Módulo 2: Implementação do Escalonador de Processos

Objetivo: Cada aluno deve desenvolver um escalonador de processos capaz de selecionar quais *jobs* ou tarefas serão executados. Essa tarefa envolve a criação de duas abstrações principais:

1. **Mecanismo de Escalonamento:** Este componente será responsável pela lógica central de seleção e execução de tarefas. O mecanismo deve ser genérico e modular, permitindo a integração com diferentes políticas de escalonamento.
2. **Política de Escalonamento:** Define as regras específicas para a priorização e execução dos processos como FCFS, *Round Robin*, baseada em prioridades, entre outras. As políticas devem ser implementadas de forma independente, facilitando sua substituição ou extensão sem afetar o mecanismo de escalonamento.

A integração entre essas duas abstrações deve ser feita por meio de um design modular, onde a política de escalonamento é "acoplada" ao mecanismo por uma interface bem definida.

1. **Tarefas a realizar:**
    - Adicionar um escalonador ao simulador. O escalonador deve gerenciar os processos considerando:
        - **Quantum**: tempo máximo que um processo pode usar a CPU antes de ser removido.
        - **Timestamp**: tempo de vida do processo no sistema, atualizado em cada ciclo.
    - Simular o comportamento do escalonador em diferentes cenários.
2. **Políticas de escalonamento a serem implementadas:** Cada aluno deve selecionar e implementar algumas das políticas de escalonamento apresentadas por Tanenbaum, considerando sua adequação às necessidades do sistema e seu impacto no desempenho. As políticas escolhidas serão utilizadas em experimentos comparativos, com o objetivo de analisar e discutir como diferentes características dos *jobs* ou tarefas podem influenciar a eficácia de cada abordagem. Essa análise experimental deve permitir a identificação de cenários em que certas decisões do sistema operacional proporcionam benefícios, enquanto outras podem resultar em degradação de desempenho. O aluno deve, portanto, refletir sobre as condições específicas que favorecem ou prejudicam cada política, promovendo uma compreensão profunda das implicações práticas de cada escolha de escalonamento.
3. **Entrega:** Um artigo técnico seguindo o **IEEE conference template** no Overleaf, descrevendo:
    - A implementação de cada política.
    - Cenários de teste utilizados.

- Comparação entre as políticas e os resultados obtidos.

4. **Conceitos necessários:**
    - Escalonamento preemptivo e não preemptivo.
    - Métricas de escalonamento: *throughput*, tempo de espera, tempo de resposta.
    - Tanenbaum: ciclo de vida do processo e tabelas de controle de processos (PCB).

Módulo 3: Gerenciamento de Cache e Escalonamento Baseado em Similaridade

Objetivo: Explorar o impacto do gerenciamento de cache na execução de tarefas e avaliar sua aplicabilidade em escalonamento.

1. **Tarefas a realizar:**
    - Implementar uma política de reaproveitamento de dados na cache para acelerar a execução de *jobs* similares.
    - Avaliar se esta abordagem pode ser integrada ao escalonador para priorizar processos que compartilham dados na cache.
2. **Discussões teóricas:**
    - Benefícios do uso de cache em arquiteturas modernas.
    - Similaridade entre *jobs*: identificação e possíveis ganhos em escalonamento.
3. **Entrega:** Subseção de resultados no artigo que cubra os seguintes tópicos:
    - Implementação da política de cache.
    - Avaliação experimental do impacto no desempenho do simulador.
    - Discussão sobre a viabilidade de integrar a política ao escalonador.
4. **Conceitos necessários:**
    - Hierarquia de memória: cache, memória principal e virtual.
    - Políticas de cache (*LRU, FIFO*).
    - Técnicas de escalonamento.

Módulo 4: Gerência de Memória e PCB

Objetivo: Ampliar o bloco de controle de processos (PCB) com conceitos de gerenciamento de memória e discutir os benefícios de endereçamento virtual.

1. **Tarefas a realizar:**

- Incorporar informações de gerenciamento de memória no PCB, incluindo:
  - Endereço base e limite.
  - Mapeamento de páginas para suporte a memória virtual.
- Implementar o suporte ao endereçamento virtual, simulando a tradução de endereços lógicos para físicos.

2. **Discussões teóricas:**
   - Conceitos de memória virtual: paginação, segmentação e tabelas de páginas.
   - Benefícios do uso de endereçamento virtual em arquiteturas modernas.
   - Impacto do gerenciamento de memória no desempenho de sistemas multicore.
3. **Entrega:** Seção do artigo, discutindo sobre memória e seus impactos no sistema operacional.
   - Alterações realizadas no PCB.
   - Benefícios observados ao utilizar memória virtual no simulador.
   - Discussão sobre a interação entre escalonamento e gerenciamento de memória.
4. **Conceitos necessários:**
   - Tabela de páginas e *TLB (Translation Lookaside Buffer)*.
   - Troca de contexto e impacto no gerenciamento de memória.
   - Tanenbaum: sistemas operacionais modernos (capítulos sobre memória e processos).

Cronograma de Entregas

Módulo	Tarefa	Data de Entrega	Valor
1	Arquitetura multicore	04/01/2025	10 pontos
2	Escalonador de processos	15/01/2025	10 pontos
3	Gerenciamento de cache	01/02/2025	10 pontos
4	Gerência de memória	18/02/2025	10 pontos
5	Artigo final	19/02/2025	10 pontos

Na data de entrega de cada etapa, os seguintes itens deverão ser apresentados para avaliação:

(a) a implementação completa do algoritmo desenvolvido;

(b) o artigo científico parcial detalhando o trabalho realizado.

O artigo deve seguir o modelo *IEEE Conference Template*, disponível em https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn, e conter, no mínimo, as seguintes seções: Resumo, Introdução, Metodologia, Resultados e Discussão, Conclusão e Referências. Considere um único artigo para compor o trabalho todo, sendo esse enviado em versões parciais conforme o calendário de datas apresentado.

1 Considerações Importantes

- A seção de resultados constitui a parte central do artigo e deve descrever com clareza os experimentos realizados, os passos metodológicos seguidos (i.e., relacionar com a seção de metodologia), os resultados obtidos e suas implicações. Esses resultados devem ser discutidos de maneira crítica, destacando sua relação com os objetivos propostos e, quando possível, comparando-os com abordagens já existentes na literatura.
- O algoritmo implementado deve ser devidamente citado no artigo e disponibilizado em um repositório público, como o GitHub. O link para o repositório deve ser incluído nas referências em conformidade com o padrão adotado (ABNT ou internacional).
- Não é necessário elaborar um arquivo `README.md` completo no repositório, uma vez que o artigo deve conter todas as informações necessárias para a compreensão da implementação. No repositório, deve-se incluir o link para o artigo e instruções básicas sobre o processo de compilação e execução da solução apresentada.
- No AVA, o aluno deverá submeter exclusivamente o artigo, renovando a submissão a cada data estipulada. Todas as informações indispensáveis para a análise e reprodução do trabalho, incluindo pseudocódigos que sigam os padrões científicos, devem estar descritas no artigo ou em um arquivo auxiliar. Além disso, no repositório disponibilizado para o algoritmo, devidamente citado no artigo, deve conter as instruções para execução e as informações básicas para a compreensão da arquitetura proposta de execução e produção dos resultados.
- A avaliação será composta por dois elementos: a entrega do simulador e o artigo científico.

Avaliação

A avaliação será baseada na qualidade da implementação e na análise apresentada no artigo técnico, bem como na clareza das discussões realizadas. A execução modular permitirá o acompanhamento contínuo do progresso e a aplicação prática dos conceitos estudados.