

UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO – UFRSA

Centro de Ciências Exatas e Naturais – CCEN

Departamento de Computação - DC

Graduação em Ciência da Computação

Disciplina: Estrutura de Dados II

Prof.: Paulo Henrique Lopes Silva

Prática Offline 1

1. Cache Eviction.

- **Cache eviction** é o processo de remoção de itens do cache quando ele atinge sua capacidade máxima, a fim de liberar espaço para novos dados.
- Esse processo é essencial para manter a eficiência do cache, evitando que ele fique cheio e incapaz de armazenar novos dados.

2. Principais Políticas de Cache Eviction.

- **LRU (Least Recently Used):**
 - Remove o item que não foi acessado há mais tempo.
 - Baseia-se na suposição de que os dados acessados recentemente serão acessados novamente em breve.
 - Aplicações: Sistemas operacionais (gestão de páginas de memória), navegadores web (gestão de cache de recursos).
- **FIFO (First In, First Out):**
 - Remove o item mais antigo no cache, ou seja, aquele que foi inserido primeiro.
 - Simples de implementar, mas pode não ser tão eficiente quanto outras políticas em termos de acesso frequente.
 - Aplicações: Sistemas de filas de processamento, algumas implementações de cache de hardware.
- **LFU (Least Frequently Used):**
 - Remove o item que foi acessado com menos frequência.
 - Baseia-se na suposição de que os dados menos utilizados são menos importantes.
 - Aplicações: Caches de dados em bases de dados, alguns tipos de cache em sistemas distribuídos.
- **Remoção aleatória:**
 - Remove um item aleatório do cache.
 - Simples de implementar e pode ser útil em algumas situações onde a carga de trabalho é imprevisível.
 - Aplicações: Algumas caches de hardware, algoritmos de amostragem.
- **MRU (Most Recently Used):**
 - Remove o item mais recentemente acessado.
 - Usado em casos específicos onde o item mais recentemente utilizado é menos provável de ser reutilizado.
 - Aplicações: Algumas caches de banco de dados com padrões de acesso específicos.

3. Aplicações:

- **Cache eviction** é aplicada em várias áreas, incluindo:
 - Sistemas Operacionais: Gestão de memória virtual, caches de disco, buffers de

arquivos.

- Navegadores Web: Cache de recursos como imagens, scripts e páginas web.
- Bancos de Dados: Caches de consultas, caches de dados frequentemente acessados.
- Sistemas de Arquivos: Caches de diretórios, caches de metadados.
- Redes de Distribuição de Conteúdo (CDNs): Caches de conteúdo web para reduzir a latência e melhorar a performance.
- Hardware: Caches de CPU (L1, L2, L3), caches de discos rígidos e SSDs.

4. Operações:

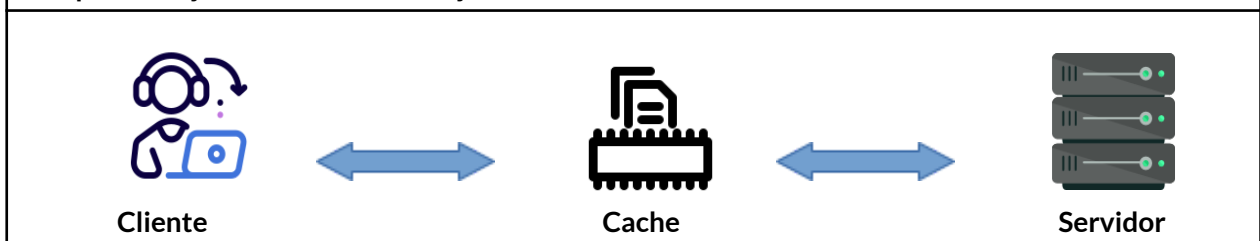
- **Hit**: procurar e achar um dado na cache.
- **Miss**: procurar um dado na base de dados quando este não for encontrado na cache.
- Principal objetivo é criar políticas que maximizem **hits** e minimizem **miss**.

5. Considere a necessidade do desenvolvimento de uma base de dados sobre ordens de serviço (OS) para reparar falhas em equipamentos.

- **Tarefa:**
 - **Simulação de uma aplicação cliente/servidor, onde:**
 - Uma classe ou conjunto de classes implementa o cliente que pode fazer buscas na base de dados;
 - Uma classe ou conjunto de classes implementa o servidor que controla a base de dados.
 - **Sobre as entidades:**
 - Cada ordem de serviço possui dados como código, nome, descrição e hora da solicitação.
 - **Sobre a cache e a base de dados:**
 - A base de dados deve ser implementada como uma Árvore AVL.
 - A modelagem de cada nó da árvore faz parte da interpretação do projeto (use orientação a objetos).
 - A cache pode ser implementada como uma lista de tamanho 20 e deve utilizar uma política baseada em FIFO.
 - **Requisitos da simulação:**
 - Tanto o cliente como o servidor vão ser executados no mesmo computador (simulados).
 - Entre o cliente e o servidor existe a cache.
 - Clientes podem fazer consultas (**buscas**) usando o código.
 - Outras operações que um cliente pode fazer são:
 - **Cadastrar OS.**
 - **Listar** ordens de serviço (todos os dados de todas as ordens de serviço).
 - **Alterar OS.**
 - **Remover OS.**
 - **Acessar a quantidade** de registros.
 - Servidor atende às requisições do cliente.
 - O servidor deve manter uma espécie de log informando a altura da árvore a cada inserção ou remoção e se houve alguma rotação ou não (sugiro inserir essas informações em um arquivo).
 - Além disso, deve mostrar os itens da cache a cada operação.
 - Para mostrar a política de **cache eviction** funcionando.
 - Um cliente pode fazer a autenticação para acessar a aplicação?
 - É facultativo.
 - Tratamento de erros e exceções onde for possível (interpretativo).
 - **Execução da simulação:**
 - Inicia o programa:
 - 60 ordens de serviço devem ser adicionadas na base de dados do servidor.

- Após isso, mostrar a interface do cliente.
- Realizar:
 - Três consultas.
 - Uma listagem.
 - Um cadastro.
 - Uma listagem.
 - Um cadastro.
 - Uma listagem.
 - Uma alteração.
 - Uma listagem.
 - Uma remoção.
 - Uma listagem.
 - Uma remoção.
 - Uma listagem.

2. Representação Gráfica da Simulação.



3. Avaliação:

- O trabalho vale 30% da nota da 1ª unidade.
- Deve ser desenvolvido na linguagem Java usando orientação a objetos.
 - Data de entrega: 12/08/2024, somente via sigaa.
 - Formato de envio: seu-nome-pratica-off-1.zip.
- **O projeto é individual.**
 - Em caso de alguma dúvida, o(a) aluno(a) poderá ser chamado para esclarecimentos.
- **Em caso de comprovação de cópia ou fraude, os trabalhos envolvidos receberão a nota ZERO.**
- Entregou fora do prazo?
 - O desconto na nota será de 20%.
 - Data limite: 14/08/2024.
 - Entregas não serão aceitas após a data limite.
 - Não há reposição de trabalhos práticos.