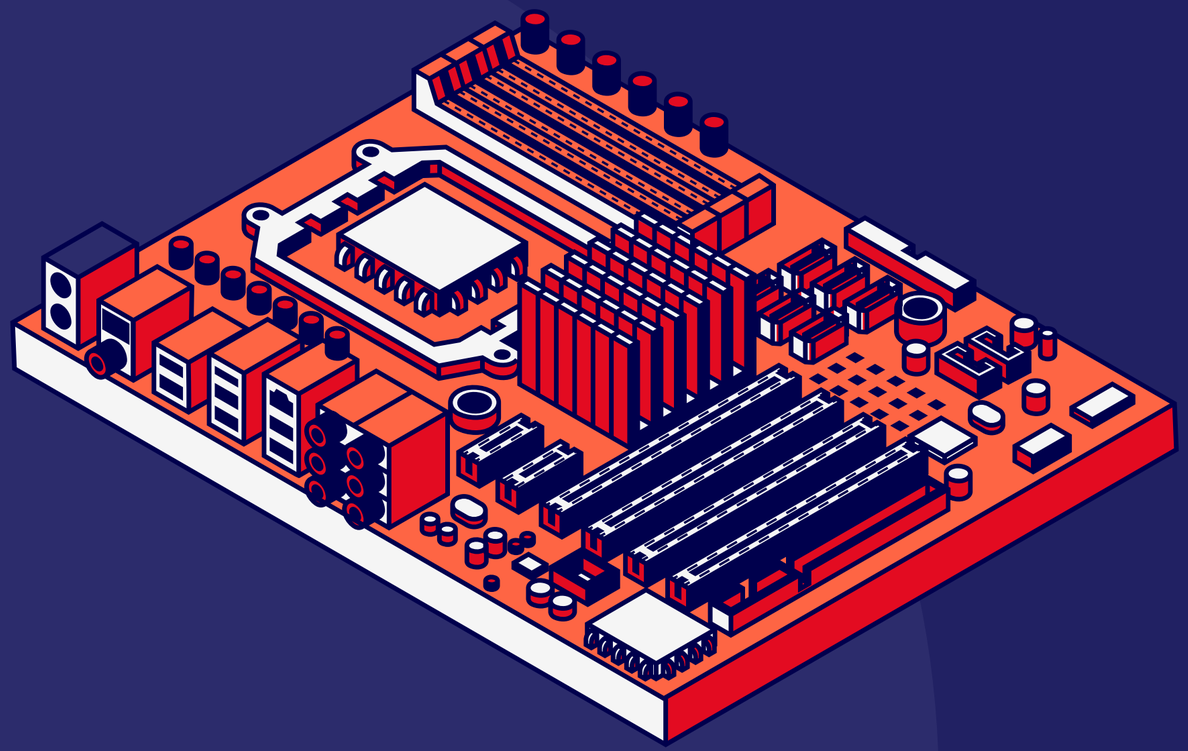
Análise de Desempenho de Cache L1:

Impacto da Associatividade vs. Carga de Trabalho

Ana Beatriz Tavares, Gustavo Randi



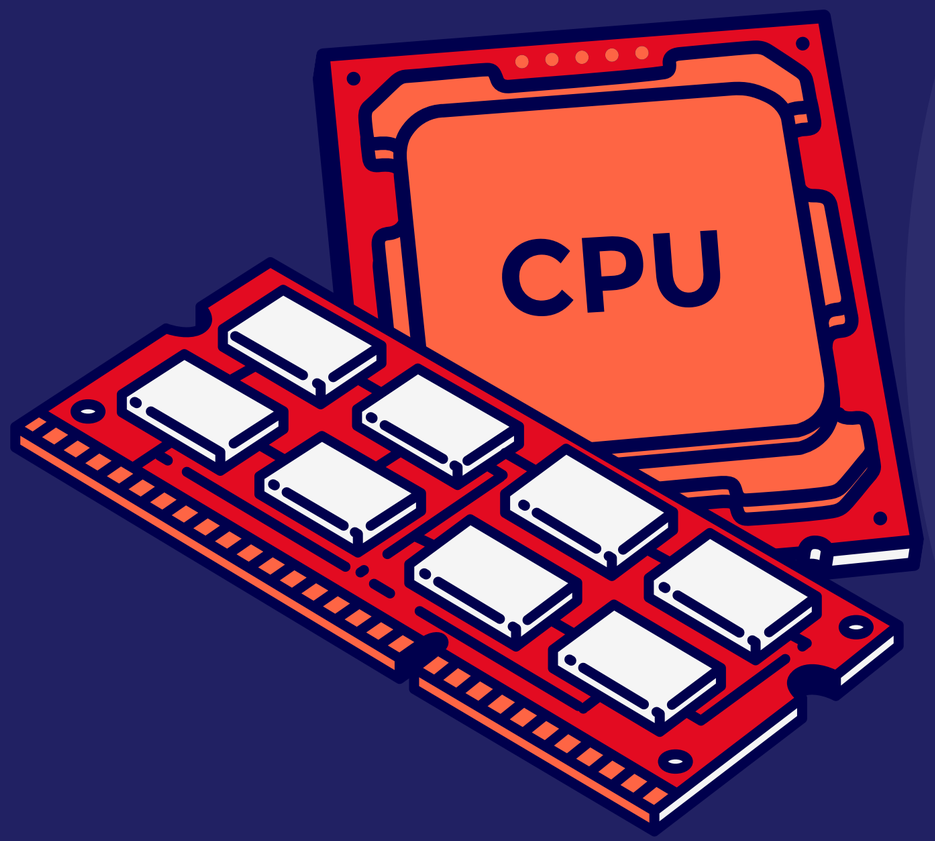
1.Introdução

Como sabemos, a velocidade dos processadores modernos é imensa, mas eles são constantemente limitados pelo tempo que leva para buscar dados na Memória Principal (RAM). Esse problema é conhecido como o "gargalo de von Neumann".

Para resolver isso, utilizamos a Hierarquia de Memória, e o componente mais crítico dessa hierarquia é a Memória Cache: uma memória pequena, mas extremamente rápida, que armazena os dados mais prováveis de serem usados pelo processador.

No entanto, o design dessa cache não é simples. Se a cache não for bem projetada, o processador pode sofrer "Cache Misses" (ou falhas), o que o força a buscar dados na RAM lenta, anulando o ganho de velocidade.

Nosso trabalho foca em um dos parâmetros de design mais cruciais: a Associatividade.



2.Objetivo

Nosso objetivo central é estudar, usando simulação, como a variação da Associatividade (a flexibilidade da cache) lida com diferentes Buffer Sizes (a carga de trabalho), especialmente em um cenário de alto conflito que criamos propositalmente.

3.Metodologia

Nosso experimento foi projetado como um estudo bidimensional, variando os seguintes parâmetros:

A. Variáveis da Metodologia

Parâmetro	Variável	Valores Testados
Associatividade	Estrutural	1-way, 2-way, 4-way e 8-way
Buffer Size	Carga de Trabalho	8KB, 16KB, 32KB, 64KB e 128KB
Padrão de Acesso	Carga de Trabalho	sequential e random

B. Configuração da Cache (Arquivos XML)

Para isolar o efeito da Associatividade, mantivemos o Tamanho Total da Cache L1 fixo em 32 KB para todos os testes. Para isso, o número de sets foi ajustado inversamente à associatividade, conforme a tabela:

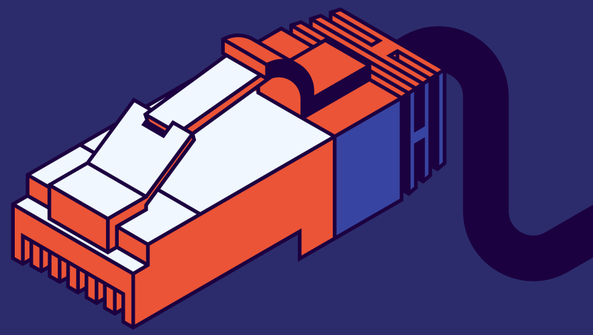
Associatividade	Sets	Line Size	Política	Tamanho Total
1-way	512	64 bytes	write_back	32 KB
2-way	256	64 bytes	write_back	32 KB
4-way	128	64 bytes	write_back	32 KB
8-way	64	64 bytes	write_back	32 KB

4.Execução

Utilizamos o simulador cache-Sim fornecido pelo professor. O experimento total consistiu em **40 execuções** (4 Assocs x 5 Buffers x 2 Padrões de Acesso).

Todo o processo foi automatizado usando:

- Um script Bash (run_full_logs.sh) para executar as 40 simulações e salvar os logs completos.
- Um script Python (extrai_dados.py) para ler os 40 logs, extrair as métricas (Miss Ratio e Total Cycles) e salvá-las em um arquivo CSV.
- Um script Python (gera_graficos.py) para gerar os gráficos finais.



O Cenário de Conflito

Para forçar a cache a falhar e expor suas fraquezas, usamos parâmetros extremos nos 10.000.000 acessos de cada teste:

stride de 8.192 (8 KB):

Este valor foi calculado (128 sets x 64 linesize) para forçar que acessos sequenciais mapeassem para o mesmo conjunto da cache, criando o pior cenário de Miss de Conflito para a Associatividade 1-way.

write ratio de 50:

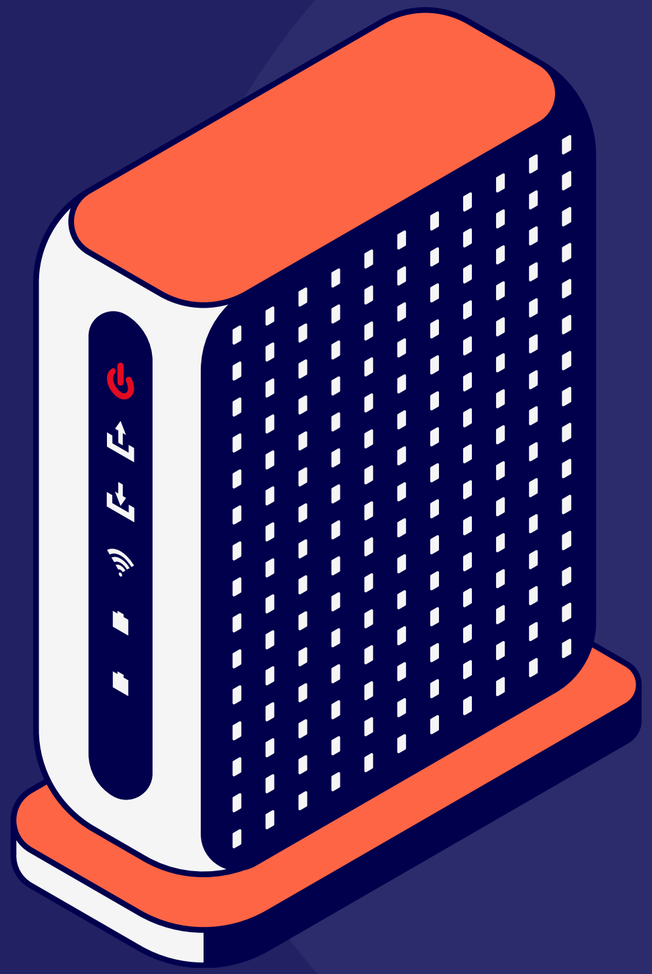
Uma carga balanceada de leituras e escritas.

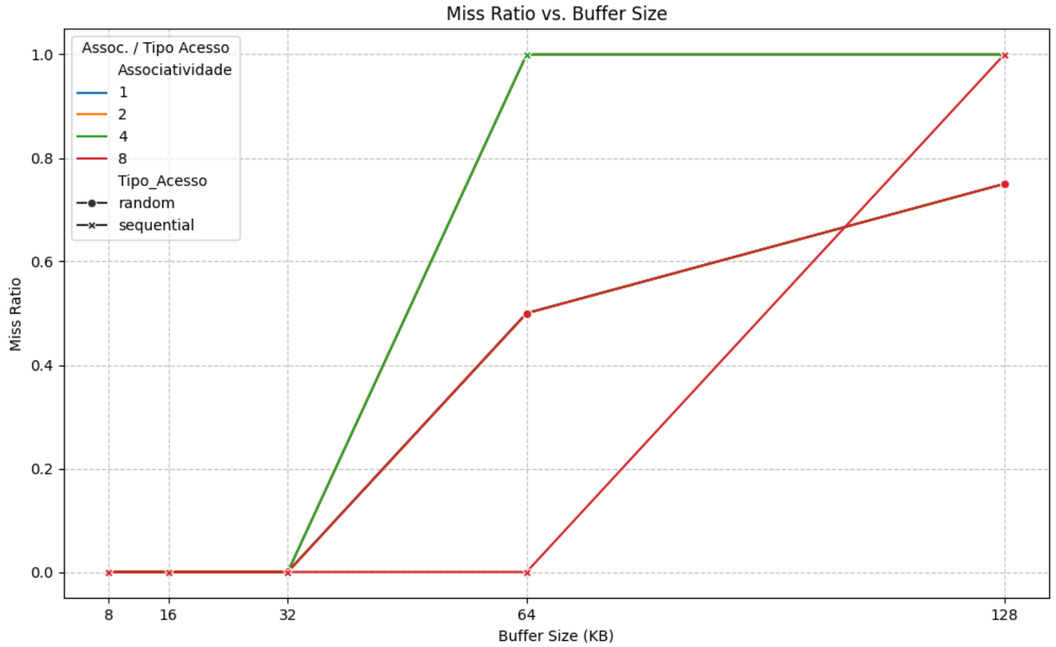
5.Resultados

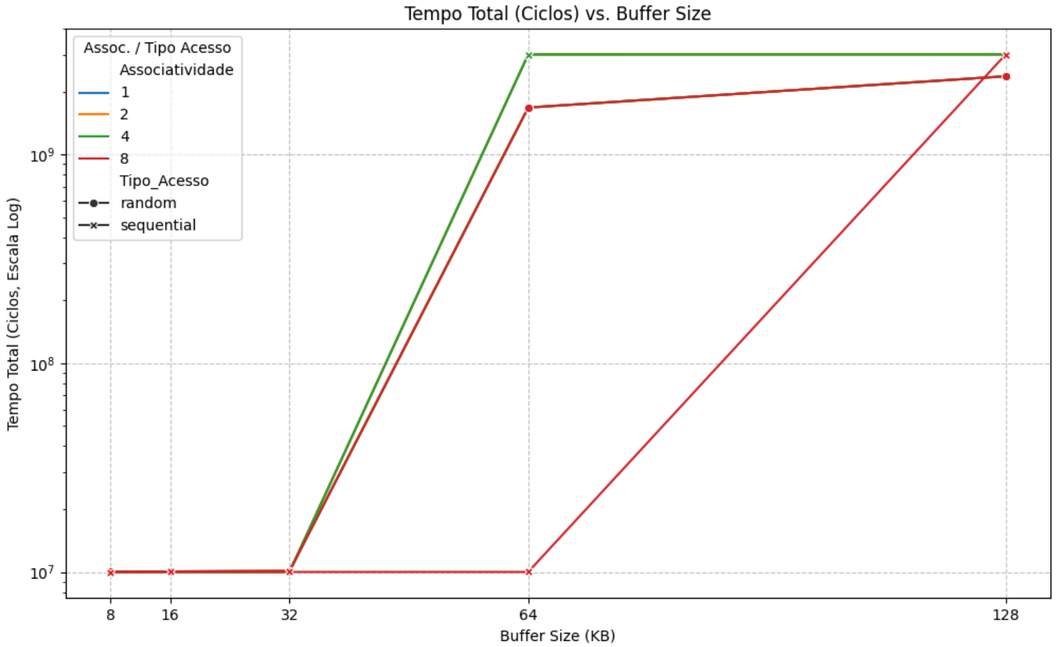
Após a execução dos logs, usamos o script em python para gerar:

- Os gráficos separados entre *miss_ratio* e *cycles*, cada um mostrando como foi o desempenho para cada associatividade e tipo de acesso.

- Dois gráficos gerais que mostram o desempenho para *miss_ratio vs buffer_size* e outro *total_cycles vs buffer_size*.



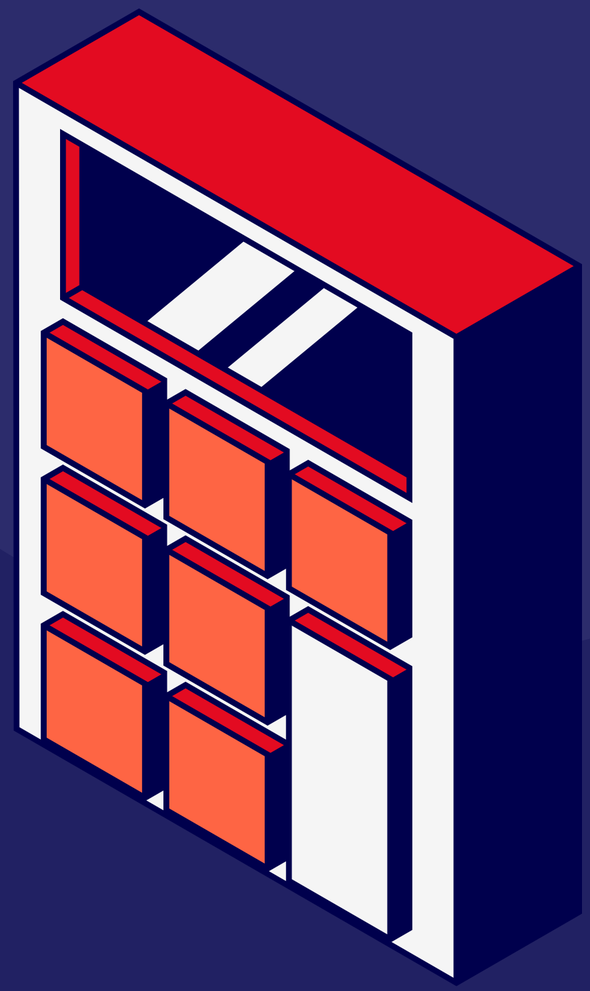

Miss Ratio vs. Buffer Size
Assoc. / Tipo Acesso
Associatividade
1
2
4
8
Tipo_Acesso
random
sequential
Miss Ratio
1.0
0.8
0.6
0.4
0.2
0.0
8
16
32
64
128
Buffer Size (KB)


Tempo Total (Ciclos) vs. Buffer Size
Assoc. / Tipo Acesso
Associatividade
1
2
4
8
Tipo_Acesso
random
sequential
Tempo Total (Ciclos, Escala Log)
10^9
10^8
10^7
8
16
32
64
128
Buffer Size (KB)

O que aconteceu com nossos testes?

Os resultados obtidos foram um tanto controversos com o que a equipe esperava obter. Tivemos que recorrer ao professor para que ele nos auxiliasse.

Após uma revisão, tanto nos parâmetros utilizados pelos alunos, quanto para como o simulador cache-sim estava se comportando, chegamos a conclusão de que:

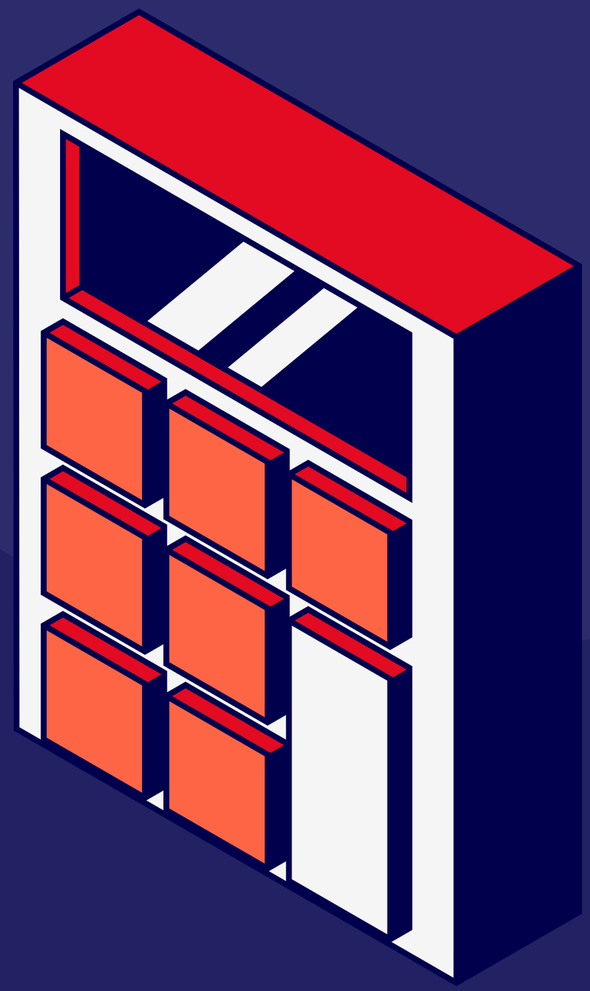


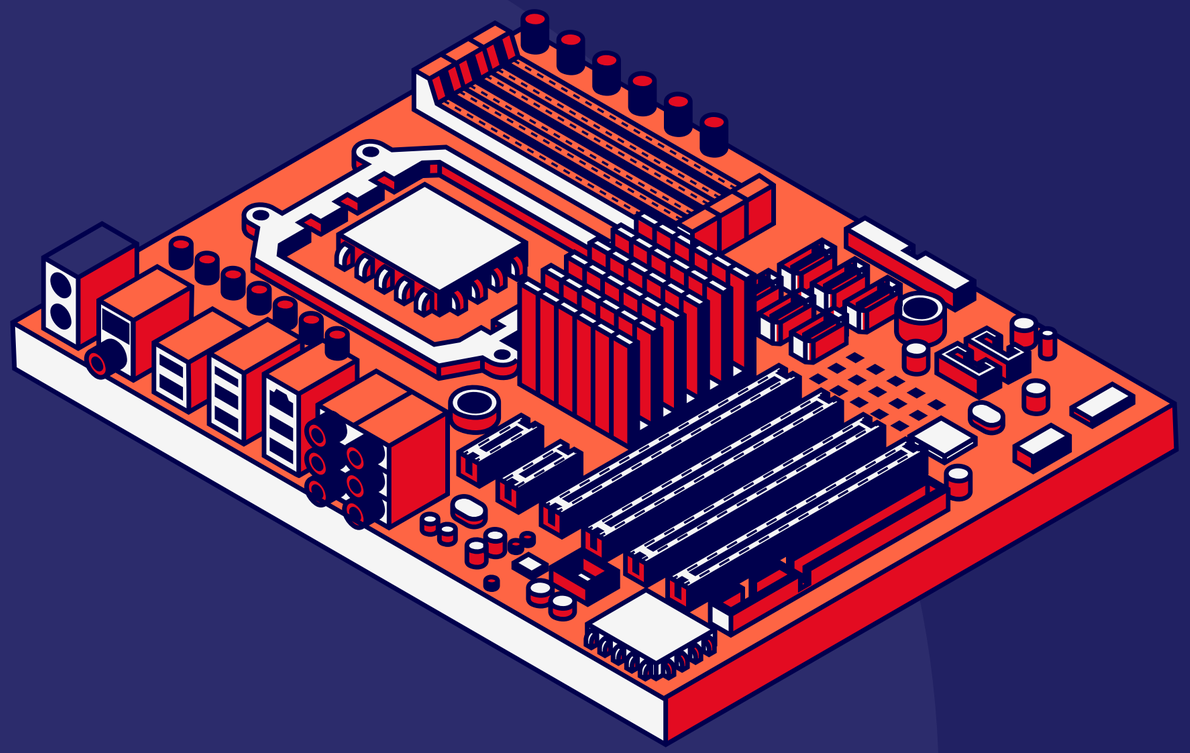
- Padrão **random**: Distribuiu os acessos de forma perfeitamente uniforme entre todos os conjuntos. Ele não criou os "pontos quentes" desbalanceados do mundo real.

- Padrão **sequential**: Também criou um padrão previsível, que não simula esse desbalanceamento.

Ou seja, a associatividade maior (4-way, 8-way) pode parecer ter **menos efeito** em nossos testes do que teria no **mundo real**.

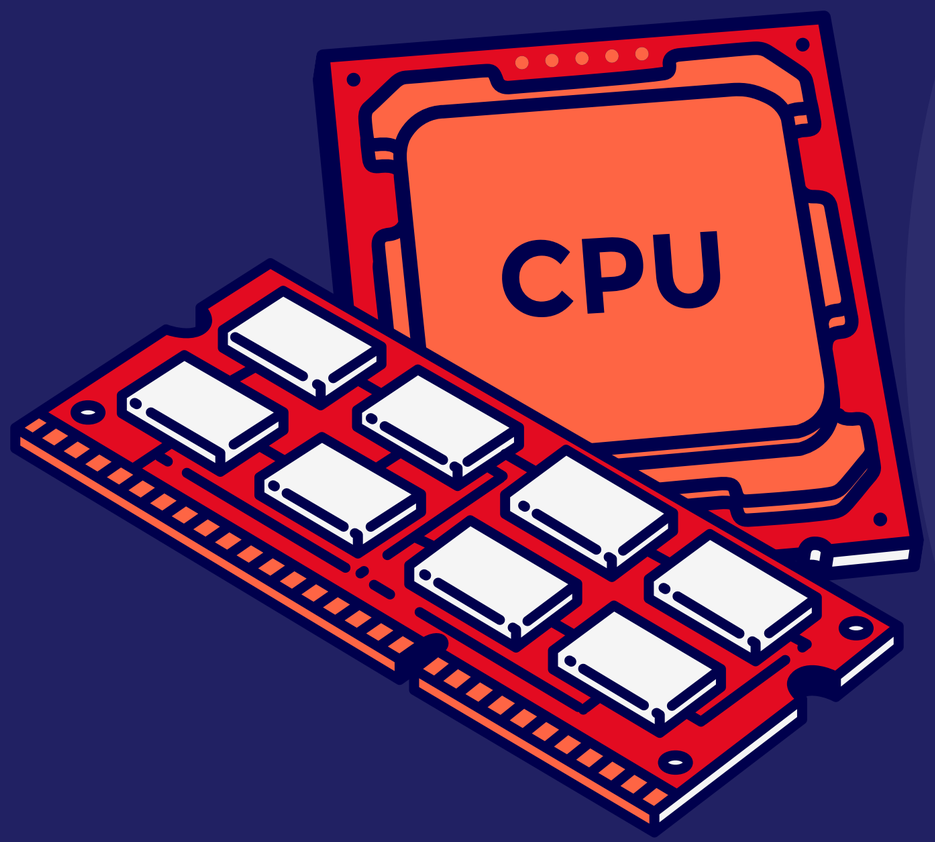
Motivo: Nossos testes não simularam a principal condição que a associatividade foi criada para resolver:
o desbalanceamento caótico da carga de trabalho no mundo real.





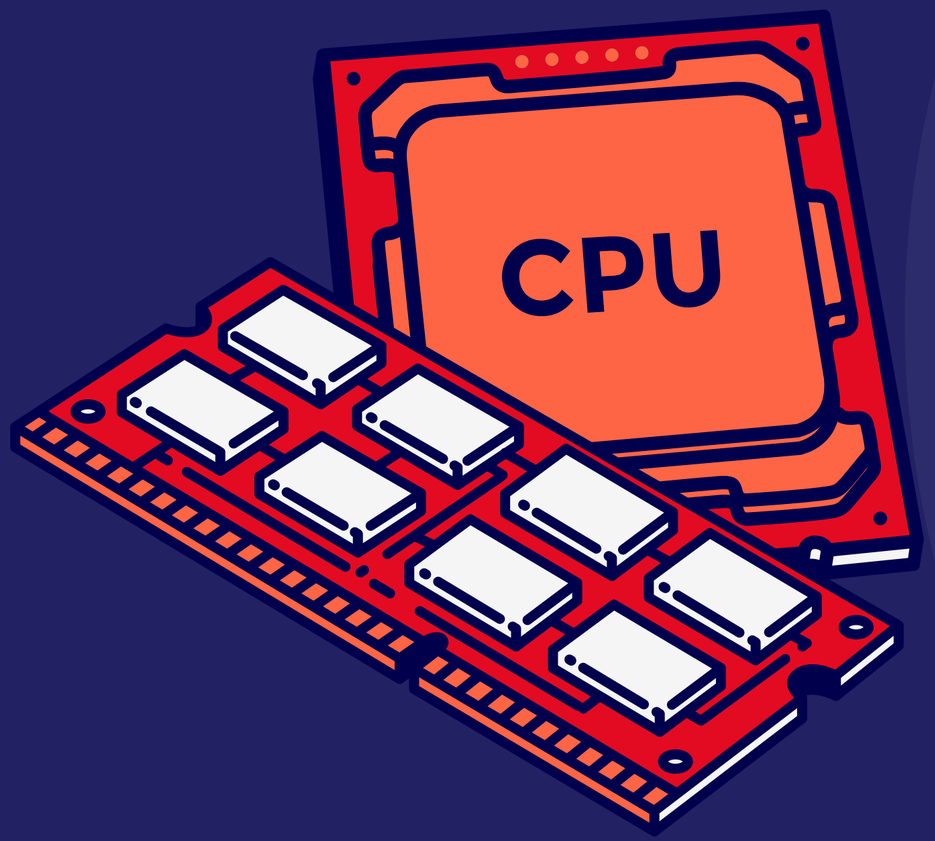
No mundo real, os acessos à memória são **caóticos e imprevisíveis**. Múltiplas variáveis, threads e funções competem pelos mesmos recursos. Alguns conjuntos da cache ficam **sobrecarregados** ("pontos quentes"), enquanto outros ficam quase vazios.

A associatividade N-way (ex: 8-way) **permite** que aquele conjunto sobrecarregado **armazene até 8 blocos diferentes**, **mitigando** os misses de conflito.



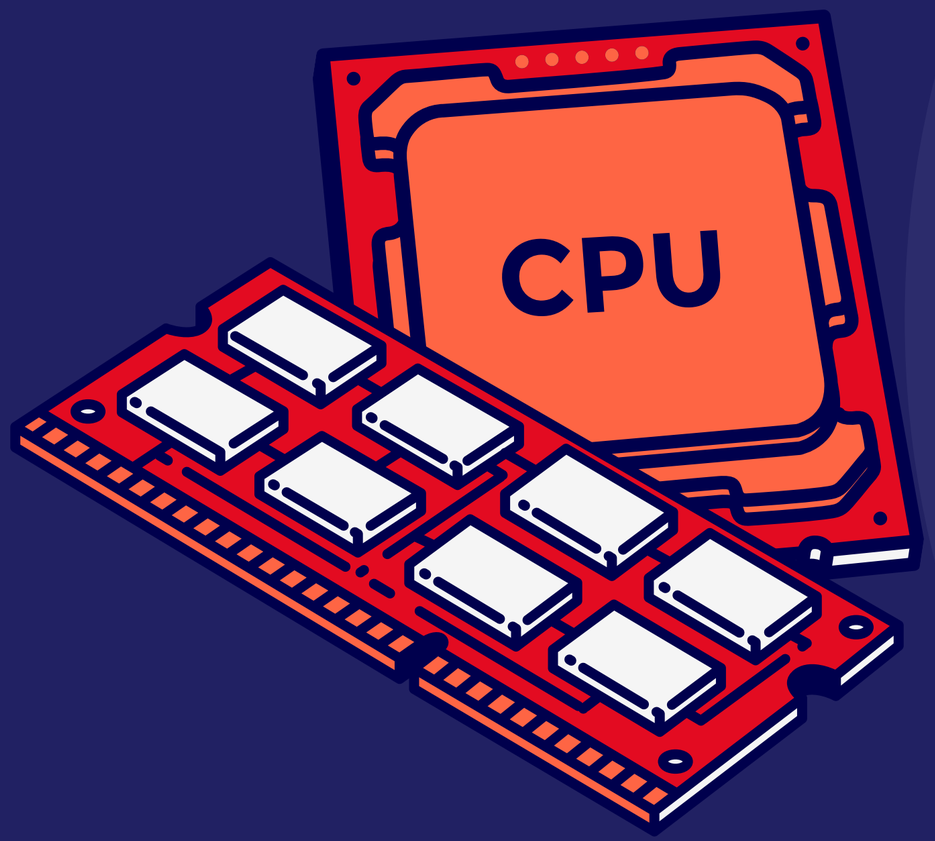
6.Análise dos Gráficos Gerados

Observando os gráficos, notamos...



A. Padrão de Acesso Sequencial

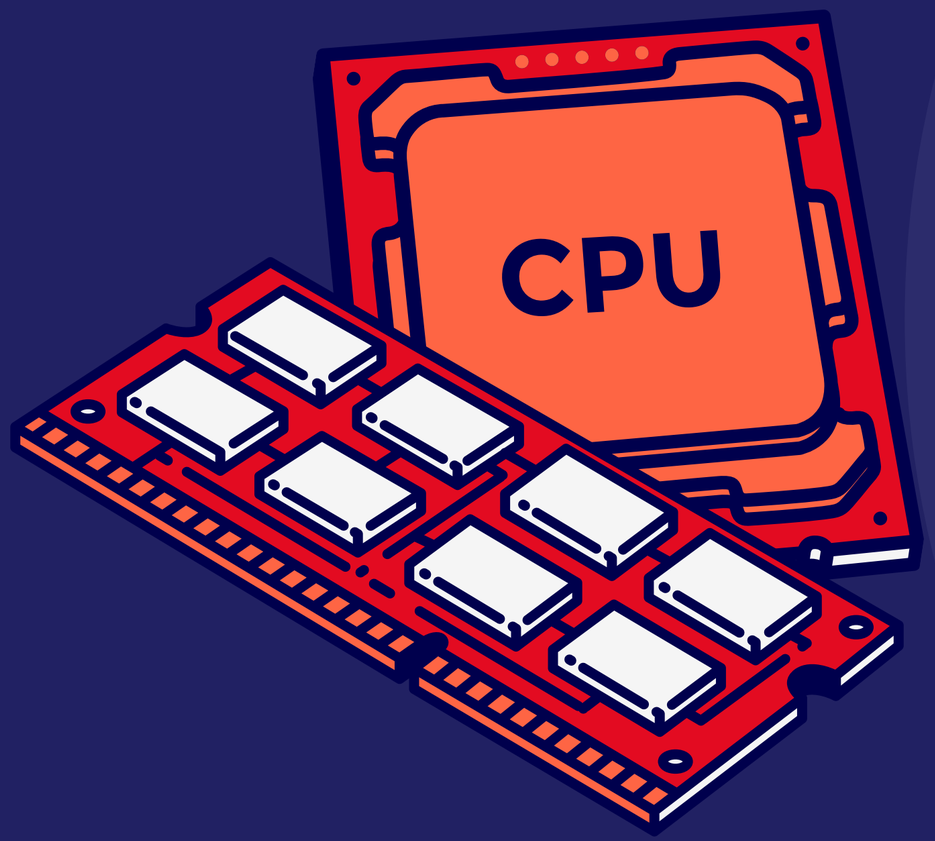
- Até 32 KB (Limite da Cache): Todas as caches (1-way a 8-way) tiveram desempenho perfeito (Miss Ratio 0.0), pois a carga cabia na cache.
- O Colapso (Acima de 32 KB): No Buffer Size de 64 KB, as caches 1-way, 2-way e 4-way falharam catastroficamente. O Miss Ratio salta para 100% (1.0).
- Por quê? Isso prova nossa hipótese do Stride. A falta de flexibilidade combinada com o Stride de 8KB (que força o mapeamento para o mesmo set) fez com que os blocos se substituíssem continuamente (thrashing).



A. Padrão de Acesso Sequencial

A Solução: A cache 8-way mitiga isso.

Ela também sofre (o Miss Ratio sobe), mas sua flexibilidade impede o colapso de 100%, pelo menos até 64KB.



B. Padrão de Acesso Aleatório

- Na realidade: O acesso aleatório é o pior inimigo da cache. O Miss Ratio começa a crescer para todas as caches a partir do momento em que a carga atinge o limite (32 KB).
- A Superioridade da Flexibilidade: Em todos os pontos de sobrecarga (32KB a 128KB), a cache 8-way (linha vermelha) teve o menor Miss Ratio e, consequentemente, o menor Tempo Total em Ciclos.
- Por quê? Para dados imprevisíveis, ter mais "opções" (8 slots por conjunto) oferece a melhor chance de encontrar um espaço livre, evitando um Miss.

7. Conclusão

Com base nos 40 testes e na análise dos gráficos, concluímos que:

A Associatividade não importa se a carga de trabalho é pequena. Se o Buffer Size cabe na cache (abaixo de 32 KB), uma cache 1-way barata tem o mesmo desempenho de uma 8-way cara.

Caches de baixa associatividade são frágeis. Elas são extremamente vulneráveis a Misses de Conflito (como provado pelo nosso Stride) e entram em colapso total de desempenho (100% de Miss Ratio) sob estresse.

A Alta Associatividade é a melhor defesa. Em cenários realistas (cargas de trabalho grandes e acessos imprevisíveis/aleatórios), a flexibilidade das caches 8-way é o que impede o colapso do sistema, mantendo o Miss Ratio e o Tempo de Execução em níveis gerenciáveis.

Trade-Off: O custo de hardware para implementar uma cache 8-way é justificado pela resiliência e estabilidade que ela oferece contra os piores cenários de acesso.

Obrigado!

repository