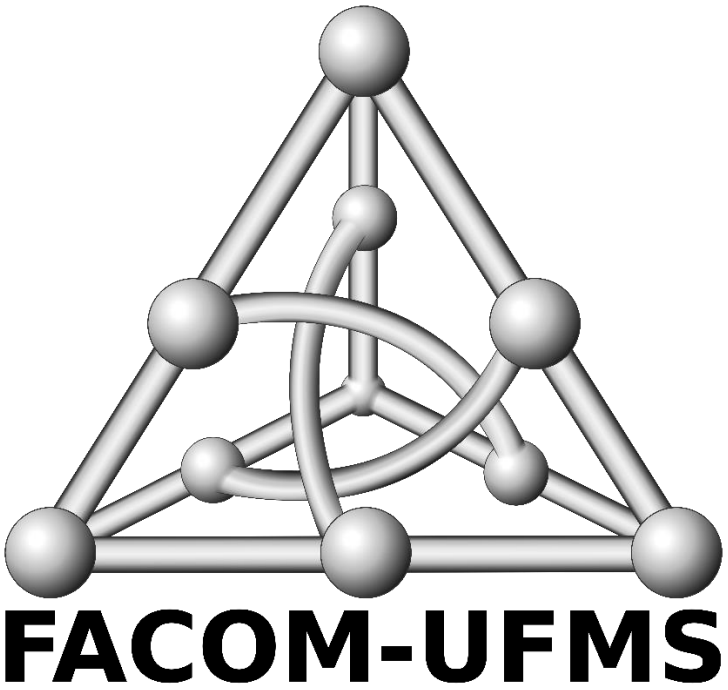
Universidade Federal do Mato Grosso do Sul

Engenharia da Computação

Relatório de Avaliação de Desempenho de Caches

Professora Nahri Moreano



Alunos: João Pedro Santos de Brito

Arthur Antonio Cavalari Cacciatore

Pedro e Silva Candia

Campo Grande - MS

Comandos de Compilação

- **simbasica:** gcc -o simbasica simbasica.c cache.c -lm
- **simsplit:** gcc -o simsplit simsplit.c cache.c -lm
- **simniveis:** gcc -o simniveis simniveis.c cache.c -lm

Experimento 1: Classificação de Falhas e Custo do Hardware

Trace utilizado: traceDijkstra.txt (63.159.737 acessos)

1. Resultados da Simulação com simbasica

Total de Acessos = 49.165.608 (Instruções) + 13.994.129 (Dados) = 63.159.737

Configuração	NBlocos	Associatividade	NPalavrasBloco	No Total Palavras	No Falhas L1	Taxa de Falhas L1
L1: 64-1-1 (Base)	64	1	1	64	22.040.383	0,348963
L1: 128-1-1	128	1	1	128	20.223.320	0,319959
L1: 256-1-1	256	1	1	256	17.299.430	0,273899
L1: 64-2-1	64	2	1	64	21.267.598	0,336727
L1: 64-4-1	64	4	1	64	19.937.043	0,315649
L1: 64-1-2	64	1	2	128	13.582.433	0,215049
L1: 64-1-4	64	1	4	256	8.519.771	0,134900

2. Análise dos Resultados

a) Em relação à taxa de falhas, o aumento de qual fator trouxe maior ganho de desempenho?

Analisando as variações a partir da configuração base (L1: 64-1-1, Taxa de Falhas: 0,348963):

- **Aumento do Número Total de Blocos (NBlocos):**
  - L1: 128-1-1 (NBlocos de 64 para 128): Taxa de Falhas = 0,319959. Redução = 0,348963 - 0,319959 = 0,029004
  - L1: 256-1-1 (NBlocos de 64 para 256): Taxa de Falhas = 0,273899. Redução = 0,348963 - 0,273899 = 0,075064

- **Aumento da Associatividade:**
  - L1: 64-2-1 (Associatividade de 1 para 2): Taxa de Falhas = 0,336727. Redução = 0,348963 - 0,336727 = 0,012236
  - L1: 64-4-1 (Associatividade de 1 para 4): Taxa de Falhas = 0,315649. Redução = 0,348963 - 0,315649 = 0,033314

- **Aumento do Número de Palavras por Bloco (NPalavrasBloco):**

- L1: 64-1-2 (NPalavrasBloco de 1 para 2): Taxa de Falhas = 0,215049. Redução = 0,348963 - 0,215049 = 0,133914
- L1: 64-1-4 (NPalavrasBloco de 1 para 4): Taxa de Falhas = 0,134900. Redução = 0,348963 - 0,134900 = 0,214063

O aumento do **Número de Palavras por Bloco** trouxe o maior ganho de desempenho. Passar de 1 para 4 palavras por bloco (L1: 64-1-4) reduziu a taxa de falhas em aproximadamente 0,214, uma melhoria significativamente maior em comparação com as variações nos outros fatores.

b) As falhas que ocorrem na simulação com a configuração 64-1-1 são, provavelmente, em maior parte compulsórias, de capacidade ou de conflito?

Na configuração de cache 64-1-1 (pequena e com mapeamento direto), a alta taxa de falhas (0,348963) provavelmente resulta de uma combinação significativa de falhas de capacidade (cache muito pequena para o volume de dados do trace) e de conflito (devido ao mapeamento direto), além das falhas compulsórias iniciais. As melhorias observadas ao aumentar a associatividade e o número de blocos reforçam a influência desses dois tipos de falha (conflito e capacidade, respectivamente).

c) O programa correspondente ao trace provavelmente apresentou mais localidade temporal ou espacial?

O maior ganho de desempenho (redução na taxa de falhas) foi obtido ao aumentar o **Número de Palavras por Bloco**. Isso indica que, quando um dado é acessado, dados próximos a ele também são acessados em seguida.

Embora o aumento da associatividade e do tamanho total da cache também tenham reduzido as falhas (indicando alguma localidade temporal e mitigação de conflitos/capacidade), a melhoria mais drástica com o aumento do tamanho do bloco sugere que a **localidade espacial é predominante** ou, pelo menos, muito significativa no programa correspondente ao trace traceDijkstra.txt.

d) Em relação ao no total de palavras da cache L1, o aumento de qual fator causou maior aumento do custo do hardware?

O custo do hardware, estimado pelo "No total de palavras da cache" (NBlocos × NPalavrasBloco), aumentou da seguinte forma a partir da base (64 palavras):

- **Aumento do Número Total de Blocos (NBlocos):**
  - L1: 128-1-1: Total Palavras = 128. Aumento = 128 - 64 = 64 palavras.
  - L1: 256-1-1: Total Palavras = 256. Aumento = 256 - 64 = 192 palavras.

- **Aumento da Associatividade:**
  - L1: 64-2-1: Total Palavras = 64. Aumento = 64 - 64 = 0 palavras (a associatividade, por si só, não aumenta o número de palavras de dados, mas aumenta a complexidade dos comparadores e da lógica de substituição).
  - L1: 64-4-1: Total Palavras = 64. Aumento = 64 - 64 = 0 palavras.

- **Aumento do Número de Palavras por Bloco (NPalavrasBloco):**
    - L1: 64-1-2: Total Palavras = 128. Aumento = 128 - 64 = 64 palavras.
    - L1: 64-1-4: Total Palavras = 256. Aumento = 256 - 64 = 192 palavras.

Considerando o “No total de palavras da cache” como métrica de custo, tanto o aumento do **Número Total de Blocos** (para 256) quanto o aumento do **Número de Palavras por Bloco** (para 4) resultaram no mesmo maior aumento de custo: 192 palavras adicionais em relação à configuração base.

e) Se você deseja uma cache com desempenho alto, mesmo que o custo do hardware seja alto, qual cache você escolherá?

Para obter alto desempenho, mesmo com custo elevado, a cache L1: 64-1-4 (64 blocos, associatividade 1, 4 palavras/bloco) é a melhor escolha entre as testadas, pois apresentou a menor taxa de falhas (0,134900) com um custo de 256 palavras.**3. Modificações no simbasica para Classificação de Falhas**

Além disso, para modificar o simulador simbasica.c de forma a calcular separadamente o número de falhas compulsórias, de capacidade e de conflito, seriam necessárias as seguintes alterações:

Estruturas de Dados Adicionais:

1. **Tabela de Blocos Já Acessados (para Falhas Compulsórias):**
    - Manter uma estrutura de dados (ex: uma hash table ou um bitmap gigante) que registre todos os endereços de blocos de memória que já foram trazidos para a cache pelo menos uma vez durante a simulação.
    - bool blocos_ja_vistos; pra agir como uma lista de presença

2. **Simulador de Cache Ideal (Totalmente Associativa e LRU com Tamanho Infinito - para Falhas de Capacidade):**

- Para ajudar a identificar falhas de capacidade (quando a cache é pequena demais), o simulador precisaria de uma "cache de referência ideal". Essa cache de referência teria o mesmo tamanho total da cache principal que está sendo testada, mas com uma organização perfeita: totalmente associativa (qualquer bloco pode ir em qualquer lugar) e usando a política de substituição LRU.
- A ideia não é simular uma cache de tamanho infinito, mas sim uma cache do mesmo tamanho porém com a melhor organização possível. Se um acesso falha na cache principal e também falharia nessa cache de referência ideal, isso indica que o problema é realmente falta de espaço (capacidade), e não uma má organização da cache principal.

Quando uma falha ocorre na cache principal (L1), o simulador segue estes passos para descobrir o tipo da falha:

1. É a primeira vez que este bloco é acessado? (Falha Compulsória)
    - O simulador verifica uma lista de "blocos já vistos".

- Se o bloco atual não está nessa lista, é uma falha compulsória. O bloco é então adicionado à lista, e o contador de falhas compulsórias aumenta.

2. Se não for compulsória, é por falta de espaço ou por má organização? (Capacidade vs. Conflito)
    - O simulador então testa o mesmo acesso em uma "cache ideal" (totalmente associativa, mesmo tamanho, LRU).
    - Se o acesso *também falhar* nessa cache ideal, significa que o problema é falta de espaço geral. É uma falha de capacidade. O contador de falhas de capacidade aumenta.
    - Se o acesso *acertar* na cache ideal, significa que o bloco caberia se a cache principal fosse perfeitamente organizada. Portanto, a falha na cache principal é por causa de sua organização limitada (ex: mapeamento direto). É uma falha de conflito. O contador de falhas de conflito aumenta.

Finalmente, o simulador precisaria de contadores separados para cada tipo de falha e garantir que as estruturas de dados (lista de blocos vistos e cache ideal) sejam corretamente inicializadas.

Experimento 2: Cache Unificada × Cache Split

Trace utilizado: traceDijkstra.txt (63.159.737 acessos) Total de Acessos = 49.165.608 (Instruções) + 13.994.129 (Dados) = 63.159.737

1. Simulação com simbasica (Cache Unificada)

- Configuração L1: 64-1-1 (NBlocos=64, Associatividade=1, NPalavrasBloco=1)
- Resultados:
    - nAcessosI: 49.165.608
    - nAcessosD: 13.994.129
    - nFalhasL1: 22.040.383

Configuração Unificada	No Total Palavras	Taxa de Falhas L1
L1: 64-1-1	64	0,348963

Cálculo da Taxa de Falhas L1 = 22.040.383 / 63.159.737 = 0,348963

2. Simulação com simsplit (Cache Split)

- **Configuração I1: 32-1-1** (NBlocos=32, Associatividade=1, NPalavrasBloco=1)
- **Configuração D1: 32-1-1** (NBlocos=32, Associatividade=1, NPalavrasBloco=1)
- **Resultados**:
    - nAcessosI: 49.165.608
    - nAcessosD: 13.994.129
    - nFalhasI (I1): 30.854.510
    - nFalhasD (D1): 7.001.576

Configuração Split	No Total Palavras (I1+D1)	Taxa de Falhas Combinada I1D1
I1: 32-1-1, D1: 32-1-1	64 (32+32)	0,600000

Cálculo da Taxa de Falhas Combinada I1D1 = (nFalhasI + nFalhasD) / (nAcessosI + nAcessosD) = (30.854.510 + 7.001.576) / (49.165.608 + 13.994.129) = 37.856.086 / 63.159.737 = 0,599371 (aproximadamente 0,600000)

3. Análise dos Resultados

a) Qual a vantagem de usar uma cache unificada, como a cache L1 modelada no simulador simbasica?

Uma vantagem da cache unificada é a **flexibilidade na alocação do espaço da cache**. O espaço total da cache pode ser dinamicamente compartilhado entre instruções e dados, conforme a necessidade do programa em um determinado momento. Se um programa tem muitos acessos a dados e poucos a instruções (ou vice-versa) durante uma fase de sua execução, a cache unificada pode alocar mais espaço para o tipo de acesso mais frequente, potencialmente levando a uma melhor taxa de acerto global em comparação com uma cache split de mesmo tamanho total onde as partições são fixas. Além disso, caches unificadas podem ter um design ligeiramente mais simples em termos de portas de acesso, se apenas um acesso (instrução ou dado) for permitido por ciclo.

b) Qual a vantagem de usar uma cache split, como as caches I1 e D1 modeladas no simulador simsplit?

A principal vantagem de usar uma cache split (separada para instruções e dados) é o **aumento da largura de banda da cache**. Como existem caches separadas e, idealmente, portas de acesso separadas para instruções e dados, o processador pode buscar uma instrução e acessar um dado simultaneamente no mesmo ciclo de clock. Isso é crucial para arquiteturas pipeline modernas, que buscam maximizar o paralelismo. Outra vantagem é que as caches podem ser otimizadas independentemente para as características específicas dos fluxos de acesso a instruções e dados.

c) Comparando: o numero total de palavras da cache unificada com o nr total de palavras da cache split (soma do no total de palavras de I1 e D1);

Mesmo com o mesmo tamanho total (64 palavras), a cache unificada (L1: 64-1-1, taxa de falhas 0,348963) teve um desempenho significativamente melhor que a cache split.

A principal razão para isso é a **flexibilidade da cache unificada**. Ela pode alocar dinamicamente seu espaço entre instruções e dados conforme a necessidade do programa. Se um programa precisa de mais espaço para dados em um momento e mais para instruções em outro, a cache unificada se adapta.

Já a **cache split tem partições fixas** (32 palavras para instruções e 32 para dados neste caso). Se a demanda por um tipo de acesso (instruções ou dados) ultrapassa sua partição dedicada, ocorrem falhas, mesmo que a outra partição esteja ociosa. No **traceDijkstra.txt**, a cache de instruções (I1) da configuração split teve uma taxa de falhas muito alta (60%), indicando que 32 palavras foram insuficientes para as instruções. A cache unificada, podendo usar todos os 64 blocos de forma flexível, evitou muitas dessas falhas.

Em suma, a capacidade de **balanceamento dinâmico do espaço** deu à cache unificada a vantagem neste cenário específico.

Experimento 3: Cache 1 Nível × Cache 2 Níveis

Trace: traceDijkstra.txt

Configurações de Simulação:

1. **simbasica (Cache L1 Unificada):** Configuração L1: 64-1-2
2. **simniveis (Cache 2 Níveis):** Configuração I1: 32-1-2, D1: 32-1-2, L2: 64-2-8

Parâmetros de Tempo (Ciclos):

- Acerto L1 (unificada, I1, D1): 1
- Acerto L2: 10
- Acesso à Memória Principal (penalidade de falha): 100
- CPI Ideal: 1
- No de Instruções: Total de acessos a instruções do trace.

Cálculos Necessários:

Para a Cache Unificada L1 (simulada com simbasica):

1. Taxa de falhas L1
2. Tempo efetivo de acesso à memória (TEAM L1)
3. Falhas por instrução (FPI L1)
4. Ciclos em stall pela memória por instrução (StallCyclesPI L1)
5. CPI real do processador (CPI_real L1)

Para a Cache de 2 Níveis (simulada com simniveis):

1. Taxa de falhas combinada I1D1
2. Taxa de falhas local L2
3. Taxa de falhas global L2
4. Tempo efetivo de acesso à memória (TEAM 2 Níveis)
5. Falhas por instrução I1D1 (FPI_I1D1)
6. Falhas por instrução L2 (FPI_L2 - falhas que vão para memória)
7. Ciclos em stall pela memória por instrução (StallCyclesPI 2 Níveis)

8. CPI real do processador (CPI_real 2 Níveis)

Resumo Comparativo

Métrica	Cache Unificada L1 (64-1-2)	Cache 2 Níveis (I1/D1:32-1-2, L2:64-2-8)
Taxa de Falhas L1 (ou I1D1 combinada)	0,215049	0,200959
Taxa de Falhas Local L2	-	0,332679
Taxa de Falhas Global L2	-	0,066859
Tempo Efetivo de Acesso à Memória (ciclos)	22,5049	9,6949
Falhas L1 (ou I1D1) por Instrução	0,276285	0,258158
Falhas L2 (memória) por Instrução	-	0,085883
Ciclos em Stall pela Memória por Instrução	27,6285	11,16988
CPI Real do Processador	28,6285	12,16988

Análise:

A introdução de uma cache L2 melhorou significativamente o desempenho do sistema de memória.

- O **Tempo Efetivo de Acesso à Memória (TEAM)** foi drasticamente reduzido de 22,5 ciclos para 9,7 ciclos. Isso ocorre porque muitas das falhas que iriam para a memória principal (100 ciclos de penalidade) agora são capturadas pela L2 (com uma penalidade efetiva menor a partir da L1).
- Os **Ciclos em Stall pela Memória por Instrução** caíram de 27,6 para 11,2. Isso reflete diretamente a redução no tempo médio que o processador espera pela memória.
- Consequentemente, o **CPI Real do Processador** melhorou de 28,6 para 12,2. Um CPI menor indica maior eficiência do processador (mais instruções por ciclo, ou menos ciclos por instrução).

Apesar da taxa de falhas combinada de I1/D1 (0,200959) ser apenas ligeiramente melhor que a taxa de falhas da L1 unificada (0,215049), o impacto da L2 em tratar uma grande porção dessas falhas L1 (Taxa de Falhas Local L2 de 0,332679 significa que ~66,7% das falhas L1 são resolvidas pela L2) é o que impulsiona a melhoria de desempenho. A Taxa de Falhas Global L2 de 0,066859 mostra que apenas cerca de 6,7% dos acessos originais do processador resultam em uma busca na memória principal, em comparação com 21,5% no sistema com apenas L1.

Isso demonstra a eficácia da hierarquia de múltiplos níveis de cache em reduzir a latência média de acesso à memória e melhorar o desempenho geral do processador.