Aula14: Hierarquia de Memória— *Misses*, 3 Cs e 7 Maneiras para Reduzir *Misses*

Revisão: Performance de Caches

CPU time = (ciclos de CPU + ciclos de stall de Memória) x período do clock

ciclos de stall de Memória = (Reads x MR p/ Read x MP p/ Read + Writes x MR p/ Write x MP p/ Write)

ciclos de stall de Memória = número de acessos de Memória x MR x MP

Revisão: Performance de Caches

CPUtime = IC x ($CPI_{execution}$ + acessos de Mem por instr x Miss rate x Miss penalty) x período do clock

Misses por instr = acessos de Mem por instr x Miss rate

CPUtime = IC x ($CPI_{execution}$ + Misses por instr x Miss penalty) x período do clock

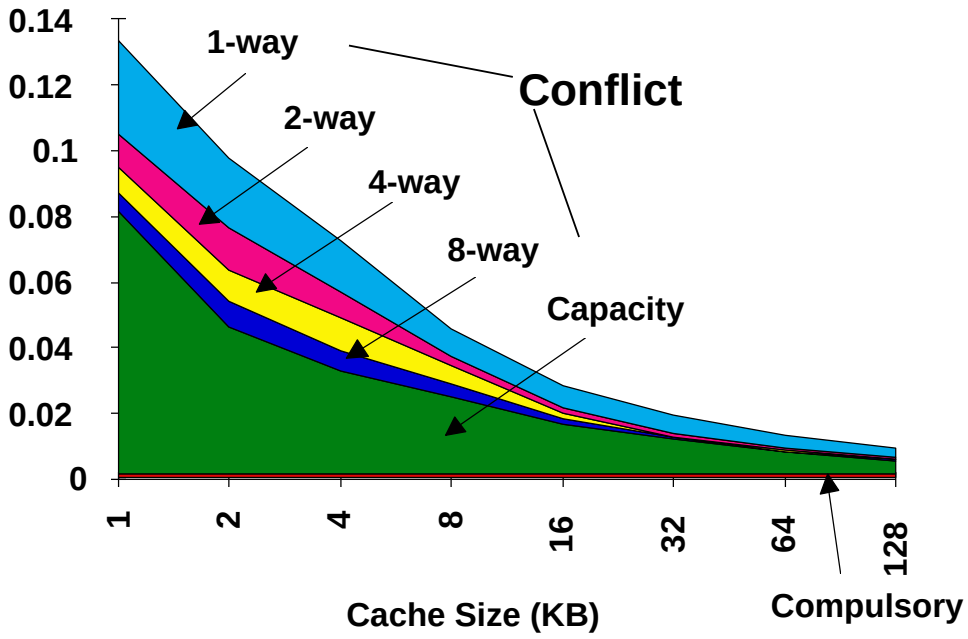
Revisão: Melhorando a Performance de Caches

1. *Reduzindo MR,*
2. Reduzindo MP, ou
3. Reduzindo o tempo de hit da cache.

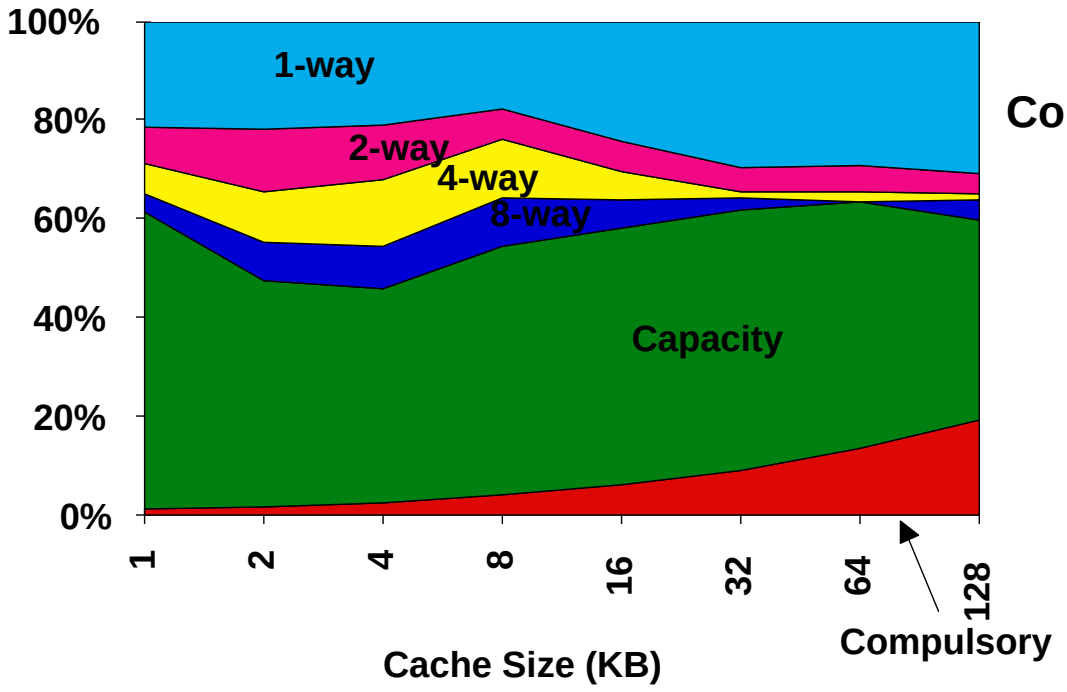
Reduzindo Misses

- Tipos de Misses: 3 Cs
  - *Compulsório*—O primeiro acesso a um bloco que não está na cache para que o bloco possa ser trazido pela primeira vez para a cache. Também chamado de *cold start misses* ou *misses de primeira referência*. *(Misses para Caches Infinitas)*
  - *Capacidade*—Se a cache não pode conter todos os blocos necessários durante a execução do programa, misses de capacidade ocorrerão devido aos blocos terem que ser descartados e depois trazidos novamente. *(Misses em Tamanho X Cache)*
  - *Conflito*—Se a estratégia de colocação de blocos e associativa por conjuntos ou mapeamento direto, misses de conflito ocorrerão se um bloco deve ser descartado porque muitos blocos mapearam para o conjunto. Também são chamados de *misses de colisão* ou *misses de interferência*.
    (Misses em N-way Associative, Tamanho X Cache)

Miss Rate Absoluto p/ 3Cs



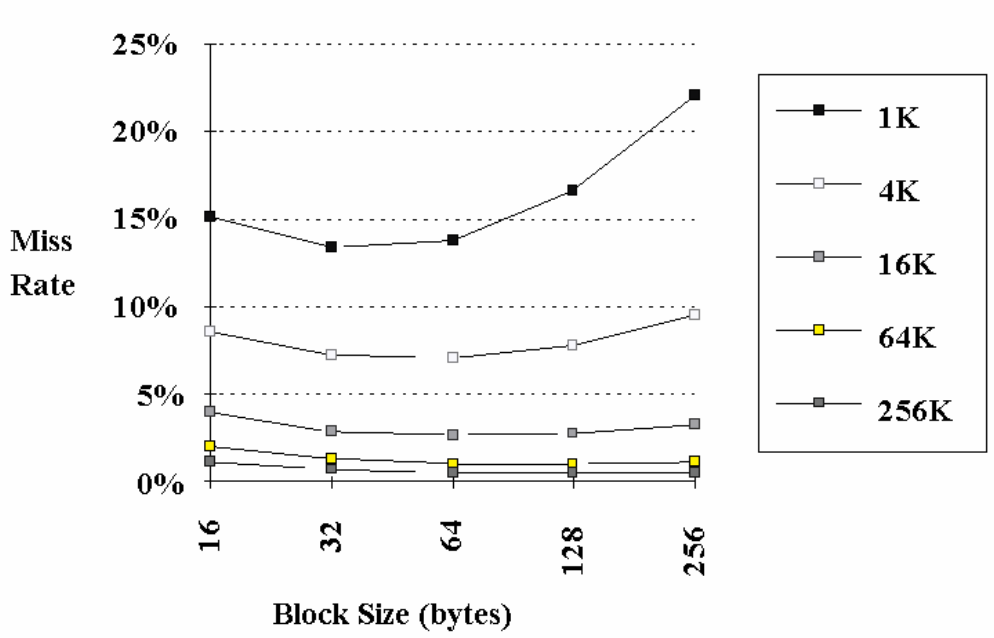
Miss Rate Relativo p/ 3Cs



Como Podemos Reduzir Misses?

- Mudar tamanho do bloco? Quais 3Cs são afetados?
- Mudar associatividade? Quais 3Cs são afetados?
- Mudar Compilador? Quais 3Cs são afetados?

1. Reduzindo Misses Aumentando Tamanho do Bloco



2. Reduzindo Misses Aumentando Associatividade

- Regra 2:1 para caches:
  - Miss Rate DM p/ tamanho de cache N = Miss Rate de cache 2-way Associative com tamanho N/2
- Cuidado: Tempo de execução é a unidade final de medição!
  - O período do clock vai aumentar?
  - Hill [1988] sugeriu que o tempo de hit de uma cache aumentava +10% para cache externa e +2% para cache interna quando mudamos a associatividade de 1-way para 2-way

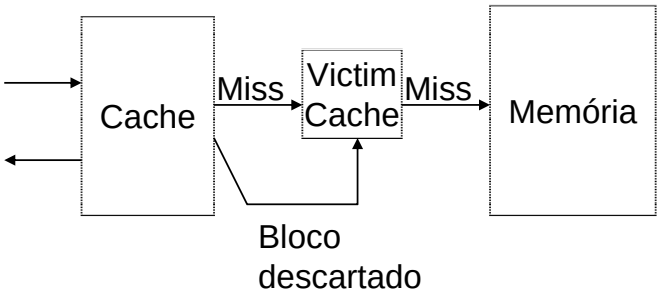
Exemplo: *Avg. Memory Access Time* (AMAT) x *Miss Rate*

- Assuma Clock = 1.10 p/ 2-way, 1.12 p/ 4-way, 1.14 p/ 8-way x Clock p/ mapeamento direto

Tamanho da Cache (KB)	Associatividade 1-way	2-way	4-way	8-way
1	2.33	2.15	2.07	2.01
2	1.98	1.86	1.76	1.68
4	1.72	1.67	1.61	1.53
8	1.46	1.48	1.47	1.43
16	1.29	1.32	1.32	1.32
32	1.20	1.24	1.25	1.27
64	1.14	1.20	1.21	1.23
128	1.10	1.17	1.18	1.20

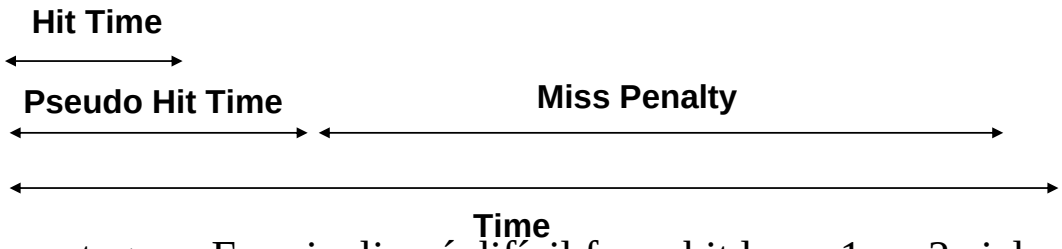
3. Reduzindo Misses com *Victim Cache*

- Como combinar hit rápido de mapeamento direto e ainda evitar misses de conflito?
- Adicionando buffer para colocar dados descartados pela cache
- Jouppi [1990]: *victim cache* com 4 blocos removeu 20% a 95% dos conflitos para uma cache DM de 4KB



4. Reduzindo Misses com Pseudo-Associatividade

- Como combinar hit rápido de mapeamento direto e ter a mesma taxa de miss de conflito de uma cache 2-way SA?
- Dividir a cache: em um miss, cheque outra metade da cache para ver se bloco está lá (pseudo-hit ou slow hit)



- Desvantagem: Em pipeline é difícil fazer hit levar 1 ou 2 ciclos
  - Melhor para caches que não estão diretamente ligadas ao processador

5. Reduzindo Misses com HW Prefetching de Instrução & Dados

- E.g., Prefetching de Instrução
  - Alpha 21064 busca 2 blocos em um miss
  - Bloco extra é colocado em *stream buffer*
  - Caso ocorra um miss, cheque *stream buffer* antes de gerar acesso à memória
- Também funciona com dados:
  - Jouppi [1990] 1 *stream buffer* de dados pegou 25% misses em cache de 4KB; 4 *streams* pegou 43%
  - Palacharla & Kessler [1994] para programas científicos com 8 *streams* pegou 50% a 70% dos misses de 2 caches 64KB, 4-way set associative
- Prefetching assume bandwidth da memória é maior e pode ser usado sem penalidade

VERLab
Visão e Robótica

6. Reduzindo Misses com SW Prefetching de Dados

- Prefetch de Dados
  - Carrega dados em registrador (HP PA-RISC loads)
  - *Cache Prefetch*: carrega em cache (MIPS IV, PowerPC, SPARC v. 9)
  - Instruções especiais de prefetching não podem causar faltas; uma forma de execução especulativa
- Executando instruções de prefetch leva tempo
  - Custo de prefetch < economia em número menor de misses?

VERLab
Visão e Robótica

7. Reduzindo Misses com Otimizações de Compiladores

- Instruções
  - Reorganizar procedimentos em memória para reduzir misses
  - Profiling para checar conflitos
  - McFarling [1989] reduziu misses de cache em 75% em 8Kb mapeamento direto com blocos de 4 bytes
- Dados
  - *Merging Arrays*: melhora localidade espacial colapsando 2 vetores em um único vetor
  - *Loop Interchange*: muda aninhamento de loops para acessar dados na ordem de armazenamento em memória
  - *Loop Fusion*: Combina 2 loops independentes que possuem mesmo controle de loop e alguma sobreposição de variáveis
  - *Blocking*: Melhora localidade temporal acessando blocos que dependem dos mesmos dados repetidamente vs. varrendo todas as colunas e linhas das matrizes.

Exemplo: *Merging Arrays*

```
/* Before */
int val[SIZE];
int key[SIZE];

/* After */
struct merge {
   int val;
   int key;
};
struct merge merged_array[SIZE];
```

Reduzindo conflitos entre val & key

Exemplo: *Loop Interchange*

```
/* Before */
for (k = 0; k < 100; k = k+1)
   for (j = 0; j < 100; j = j+1)
      for (i = 0; i < 5000; i = i+1)
            x[i][j] = 2 * x[i][j];
/* After */
for (k = 0; k < 100; k = k+1)
   for (i = 0; i < 5000; i = i+1)
      for (j = 0; j < 100; j = j+1)
            x[i][j] = 2 * x[i][j];
```

Acessos sequenciais ao invés de acessá-los sequencialmente a cada 100 palavras

Exemplo: *Loop Fusion*

```
/* Before */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
      a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
      d[i][j] = a[i][j] + c[i][j];
/* After */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
  {   a[i][j] = 1/b[i][j] * c[i][j];
      d[i][j] = a[i][j] + c[i][j];}
```

2 misses por acesso de a e c vs. um miss por acesso

Exemplo: *Blocking*

```
/* Before */
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
      {r = 0;
       for (k = 0; k < N; k = k+1){
         r = r + y[i][k]*z[k][j];};
       x[i][j] = r;
      };
```

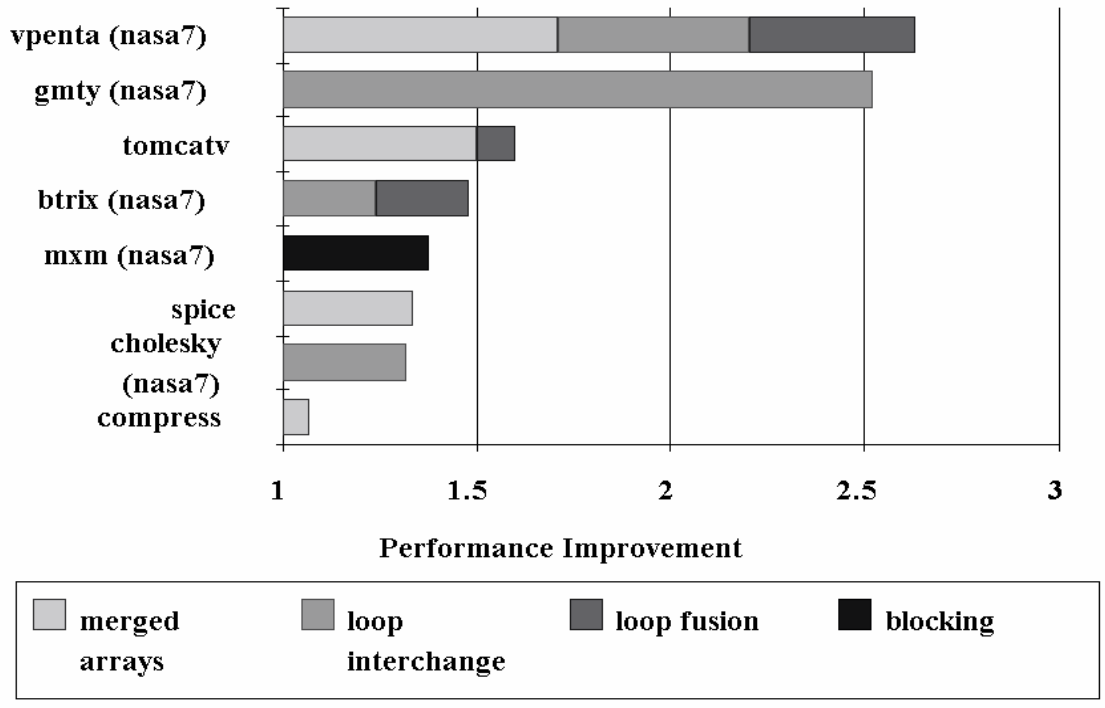
- Dois loops mais internos:
  - Lêem todos NxN elementos de z[]
  - Lêem N elementos de 1 linha de y[] repetidamente
  - Escrevem N elementos de 1 linha de x[]
- Misses de Capacidade são uma função de N e tamanho de cache (3 NxN)
- Idéia: calcular submatriz BxB que cabe na cache

Exemplo: *Blocking*

```
/* After */
for (jj = 0; jj < N; jj = jj+B)
for (kk = 0; kk < N; kk = kk+B)
for (i = 0; i < N; i = i+1)
    for (j = jj; j < min(jj+B-1,N); j = j+1)
     {r = 0;
      for (k = kk; k < min(kk+B-1,N); k = k+1) {
        r = r + y[i][k]*z[k][j];};
      x[i][j] = x[i][j] + r;
     };
```

- Misses de capacidade de $2N^3 + N^2$ a $2N^3/B + N^2$
- B é chamado de *Blocking Factor*
- E quanto a misses de conflito?

Resumo das Técnicas de Redução de Misses



Resumo

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{Memory\ \ accesses}{Instruction} \times \textbf{Miss rate} \times Miss\ \ penalty \right) \times Clock\ \ cycle\ \ time$$

- 3 Cs: Compulsório, Capacidade, Conflito
  1. Reduzindo misses com blocos maiores
  2. Reduzindo misses com maior associatividade
  3. Reduzindo misses com *victim cache*
  4. Reduzindo misses com pseudo-associatividade
  5. Reduzindo misses com HW prefetching de instruções e dados
  6. Reduzindo misses com SW prefetching de dados
  7. Reduzindo misses com otimizações do compilador