

Introdução aos multiprocessadores

João Canas Ferreira

Abril 2018



Assuntos

- 1 Processamento paralelo
- 2 Multiprocessadores de memória partilhada
- 3 Modelo simplificado de desempenho

Multiprocessadores

- Objetivo: criar computadores potentes pela interligação de computadores mais simples
 - programas devem aproveitar um número variável de processadores
 - utilização de processadores mais simples e eficientes (em vez de complexos e ineficientes)
 - aumento da disponibilidade (robustez em face de falhas)
- execução simultânea de programas independentes: *parallelismo ao nível do processo*
- programa único em execução simultânea em vários processadores: *programa paralelo*
- **cluster:** computador paralelo composto por muitos processadores alojados em computadores independentes
- **processador multinúcleo:** vários processadores (núcleos) num único circuito integrado
- vários processadores (multinúcleo ou não) podem ser reunidos num único computador: multiprocessador

Concorrência e paralelismo

		Software	
		Sequencial	Concorrente
Hardware	“Serial”	Multiplicação de matrizes em Matlab (Pentium 4)	Windows a correr no processador Pentium 4
	Paralelo	Multiplicação de matrizes em Matlab (Xeon e5345)	Windows a correr no processador Xeon e5345

→ Grandes desafios:

- Como executar programas sequenciais eficientemente em *hardware* paralelo?
- Como executar programas concorrentes eficientemente à medida que o número de processadores aumenta? (Escalabilidade)

Desafio 1: Maior rapidez de processamento

■ Pretende-se obter um aumento de rapidez (*speed-up*) de 90 usando um computador com 100 processadores. Que percentagem dos cálculos pode ser sequencial?

■ Lei de Amdahl:

f : fator de aumento de rapidez

p : percentagem de tempo que o melhoramento pode ser aplicado:

$$S = \frac{1}{(1 - p) + \frac{p}{f}}$$

■ Fazendo:

$$90 = \frac{1}{(1 - p) + \frac{p}{100}}$$

■ Resultado: $p \approx 0,999$

■ Para obter um aumento de rapidez de 90 usando 100 processadores, apenas 0,1 % do tempo de execução pode ser sequencial.

Desafio 2: Escalabilidade (1)

- Pretende-se somar duas matrizes de dimensão 10×10 e realizar 10 somas com variáveis simples (com dependências). Qual é o aumento de rapidez obtido usando um multiprocessador com 10 ou com 100 elementos? (Seja t o tempo que demora fazer uma multiplicação uma adição).
- Tempo necessário num único processador sequencial: $110 \times t$
- Tempo com multiprocessador 10: $20 \times t \Rightarrow S = 5,5$ (55,5 % do ideal)
- Tempo com multiprocessador 100: $11 \times t \Rightarrow S = 10$ (10 % do ideal)

E se as matrizes forem 100×100 (10000 adições)?

- Tempo necessário num único processador sequencial: $10010 \times t$
- Tempo com multiprocessador 10: $1010 \times t \Rightarrow S = 9,9$ (99 % do ideal)
- Tempo com multiprocessador 100: $110 \times t \Rightarrow S = 91$ (>90 % do ideal)

Desafio 2: Escalabilidade (2)

⇒ No problema anterior, é mais fácil aproveitar os recursos no caso de problemas de maiores dimensões.

⇒ Escalabilidade de um multiprocessador

Escalabilidade forte aumento de rapidez obtido mantendo a dimensão do problema.

Escalabilidade fraca aumento de rapidez obtido aumentando o problema de forma proporcional ao número de processadores.

⇒ Seja M o espaço de memória e P o número de processadores:

- escalabilidade forte: memória por processador M/P (decresce)
- escalabilidade fraca: memória por processador M (mantém-se)

⇒ O tipo de escalabilidade relevante depende do problema.

Desafio 3: Equilíbrio

⇒ Até agora, assumiu-se que os cálculos podiam ser distribuídos de forma equilibrada por todos os processadores. Para 100 processadores, cada um executa 1 % dos cálculos.

O que sucede quando um dos processadores tratar de 2 % dos cálculos (no caso de matrizes 100×100)?

⇒ $2\% \times 10000 = 200$ adições são feitas por um processador; os restantes 99 repartem 9800 adições.

⇒ Tempo com multiprocessador 100:

$$\max\left(\frac{9800 \times t}{99}, \frac{200 \times t}{1}\right) + 10t = 210t \quad \Rightarrow S = 48$$

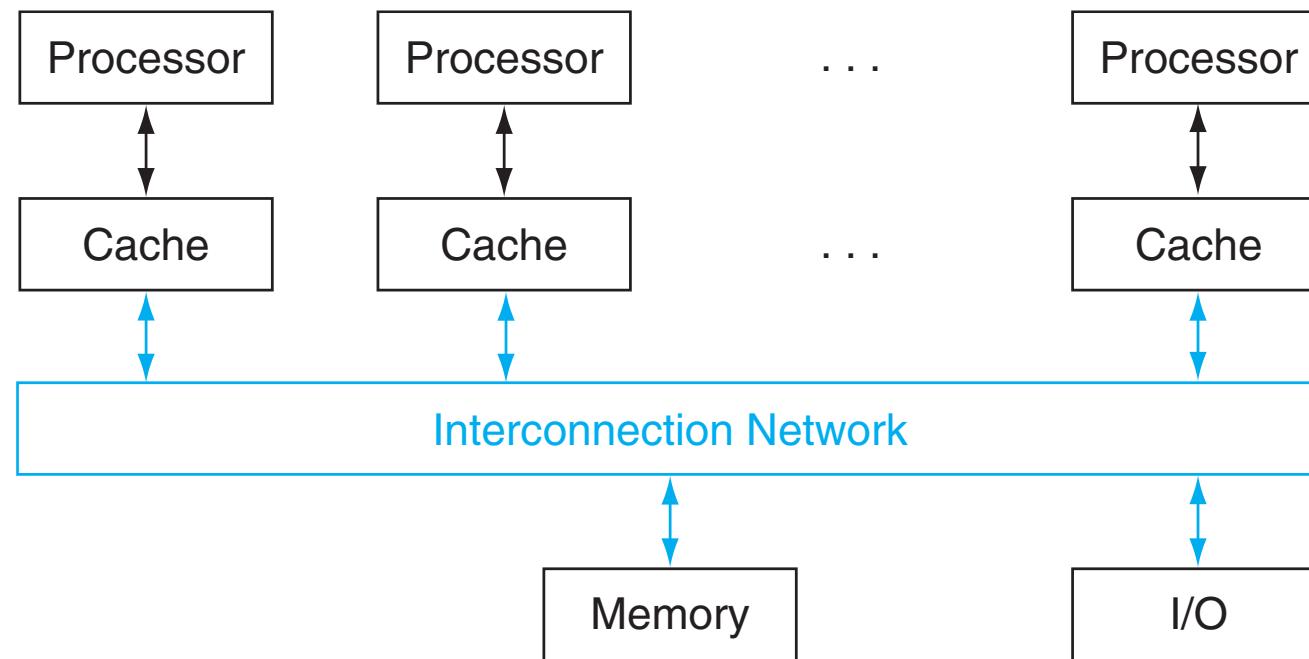
Cerca de metade da situação equilibrada!

1 Processamento paralelo

2 Multiprocessadores de memória partilhada

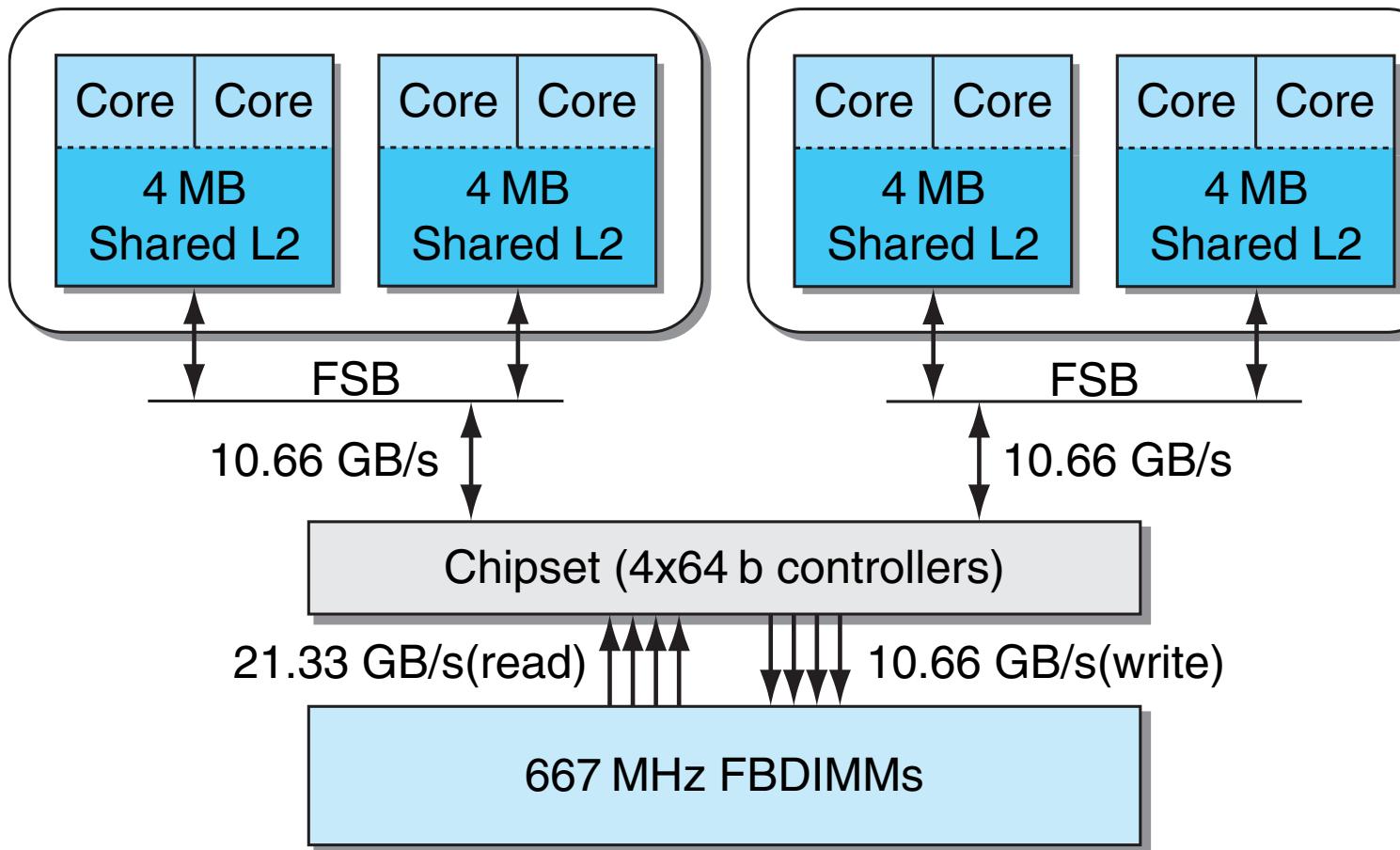
3 Modelo simplificado de desempenho

Organização



Fonte: [COD4]

Exemplo: Xeon e5345



Fonte: [COD4]

Características

- Espaço físico de endereços **único**
- Processadores comunicam através de variáveis partilhadas em memória: todos os processadores podem aceder a qualquer posição de memória via instruções *load/store*
- Dois tipos de organização:
 - o tempo de acesso a memória é idêntico para todos os processadores e todas as posições de memória: UMA (*uniform memory access*)
 - alguns acessos a memória são mais rápidos que outros (dependendo do processador e da posição de memória acedida): NUMA (*non-uniform memory access*)
- Coordenação no acesso aos dados partilhados por código em execução em processadores diferentes: **sincronização**.

Programa para memória partilhada (1)

- ➡ Tarefa: Somar 100000 números num multiprocessador NUMA com 100 processadores.
- ➡ Seja P_n o identificador do processador ($0 \leq P_n \leq 99$)
- ➡ Todos os processadores começam por executar o seguinte ciclo, que processa uma parte dos dados ($100000/100 = 1000$ números)

```
sum[Pn] = 0;  
  
for (i = 1000*Pn; i < 1000*(Pn+1) ; i=i+1)  
    sum[Pn] = sum[Pn] + A[i];
```

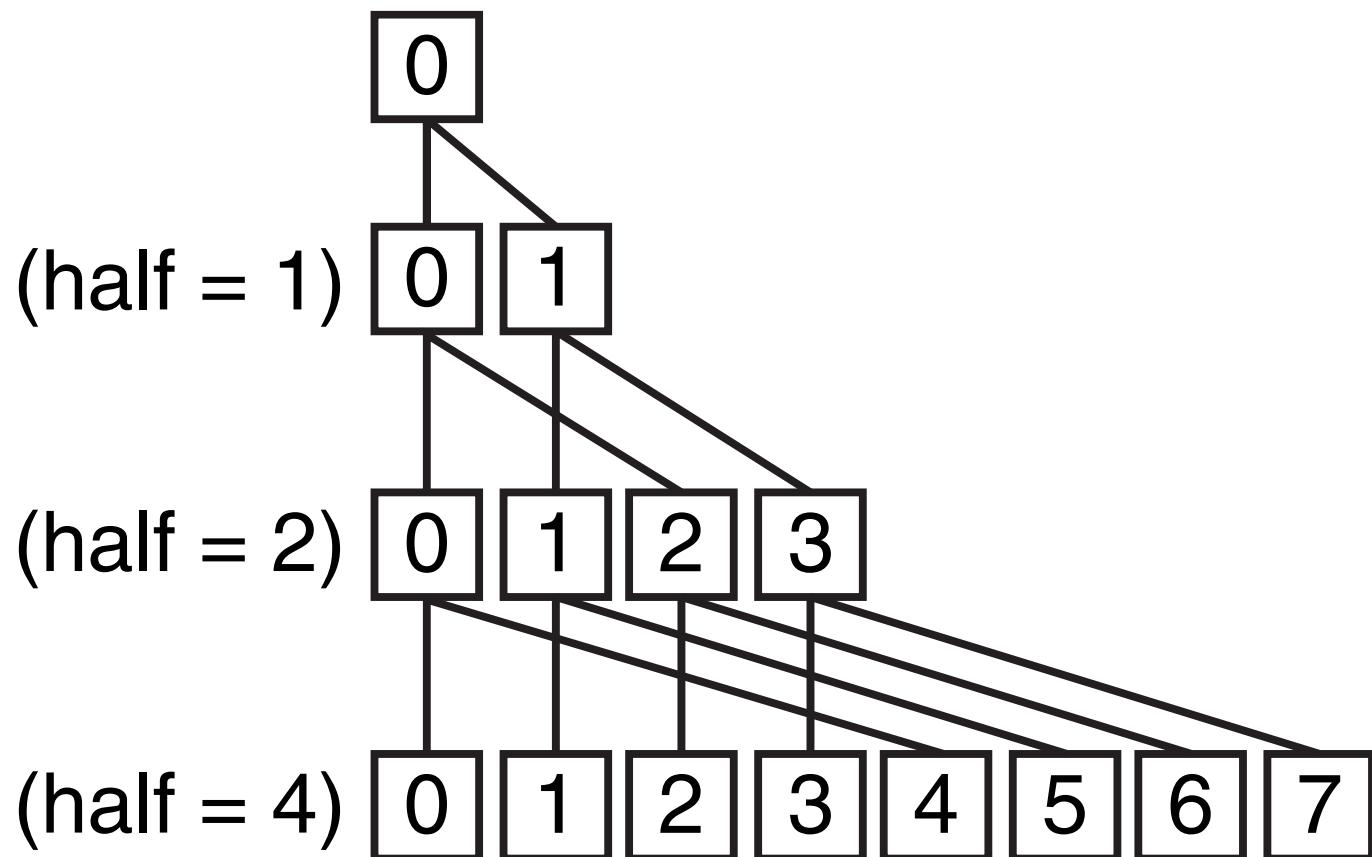
Programa para memória partilhada (2)

⇒ O segundo passo faz a adição das somas parciais (redução)

- 1/2 dos processadores somam pares de números
- 1/4 dos processadores somam as novas somas parciais
- 1/8 dos processadores somam as novas somas parciais, etc.

```
half = 100;  
repeat  
    synch(); // esperar pela conclusão das somas parciais  
    if (half % 2 != 0 && Pn == 0) // half é ímpar  
        sum[0] = sum[0] + sum[half-1];  
    half = half / 2;  
    if (Pn < half)  
        sum[Pn] = sum[Pn] + sum[Pn+half];  
until (half == 1);
```

Programa para memória partilhada (3)



Fonte: [COD4]

- ➡ Os quatro últimos níveis da operação de *redução* que soma os resultados de cada processador (da base para o topo).

1 Processamento paralelo

2 Multiprocessadores de memória partilhada

3 Modelo simplificado de desempenho

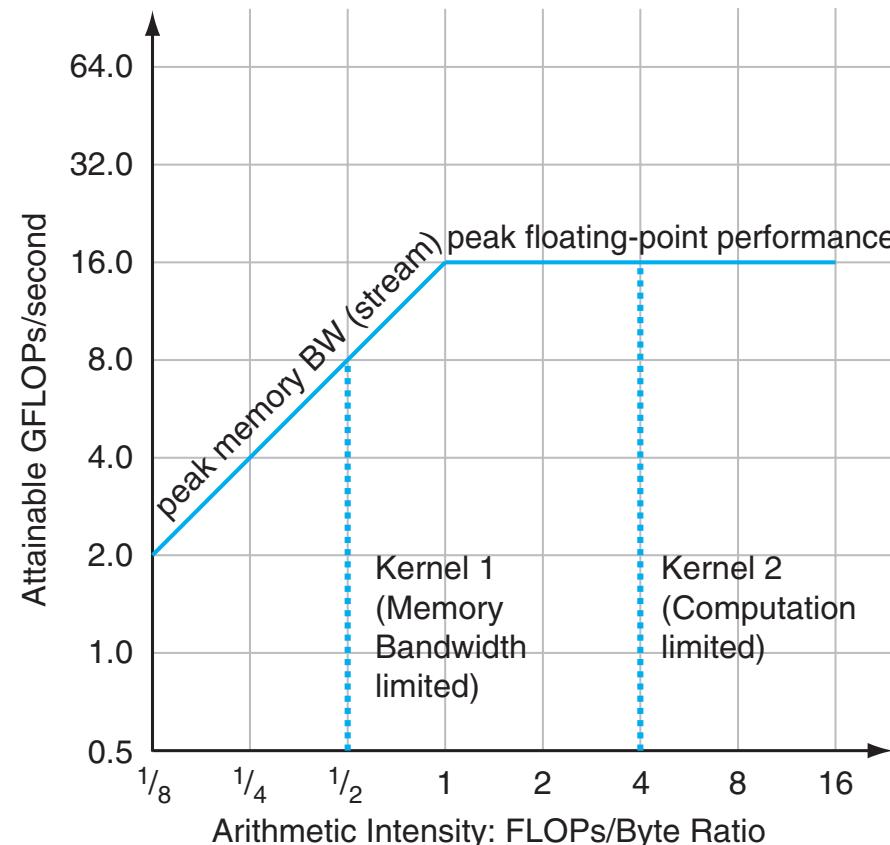
Simplificações

- Domínio: cálculo científico
- Dados em vírgula flutuante (VF): FLOP (*floating-point op per second*)
- Problemas de grandes dimensões compostos apenas por cálculos VF e transferências de dados
- **Intensidade aritmética:**
Operações vírgula flutuante / bytes transferidos de memória
- Impacto sobre o sistema de memória:

$$\frac{\text{Operações vírgula flutuante / segundo}}{\text{Operações de vírgula flutuante/byte}} = \text{bytes/segundo}$$

- bytes/segundo: taxa de transferência de dados que o sistema de memória deve suportar (*memory bandwidth*)
- Existe um valor máximo para o desempenho (valor de pico)

Modelo simplificado

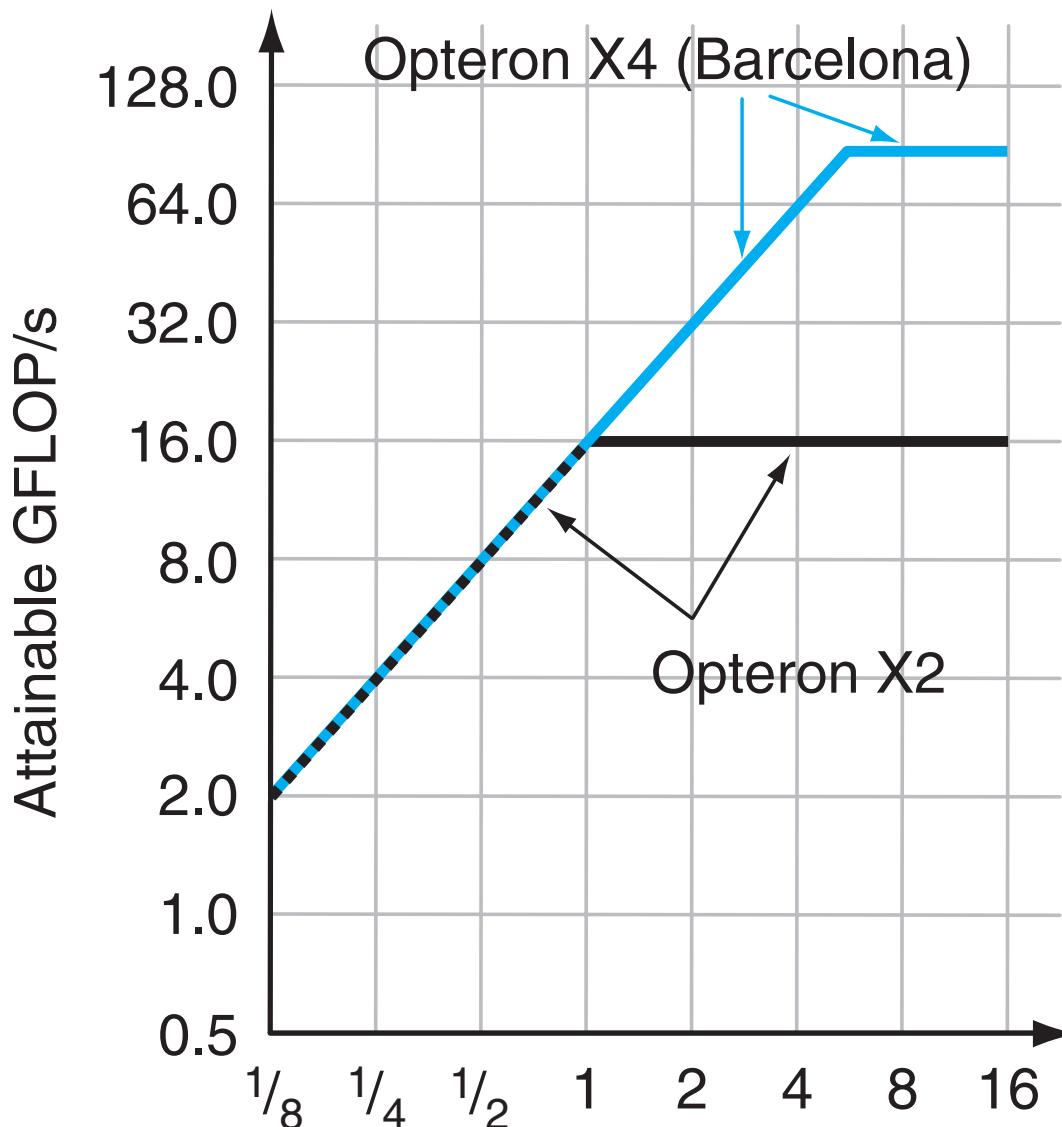


Fonte: [COD4]

➡ Desempenho é o valor mínimo entre:

- taxa de transferência de dados × intensidade aritmética
- valor máximo do desempenho assumindo que todas as unidades de cálculo estão permanentemente ocupadas (desempenho de pico)

Comparação de multiprocessadores



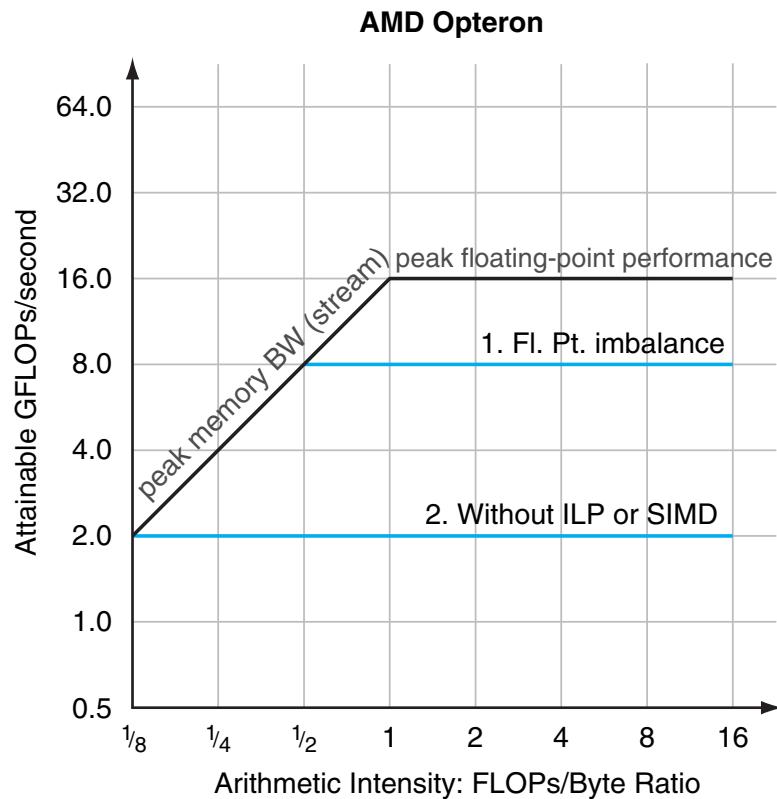
Fonte: [COD4]

Otimizações úteis

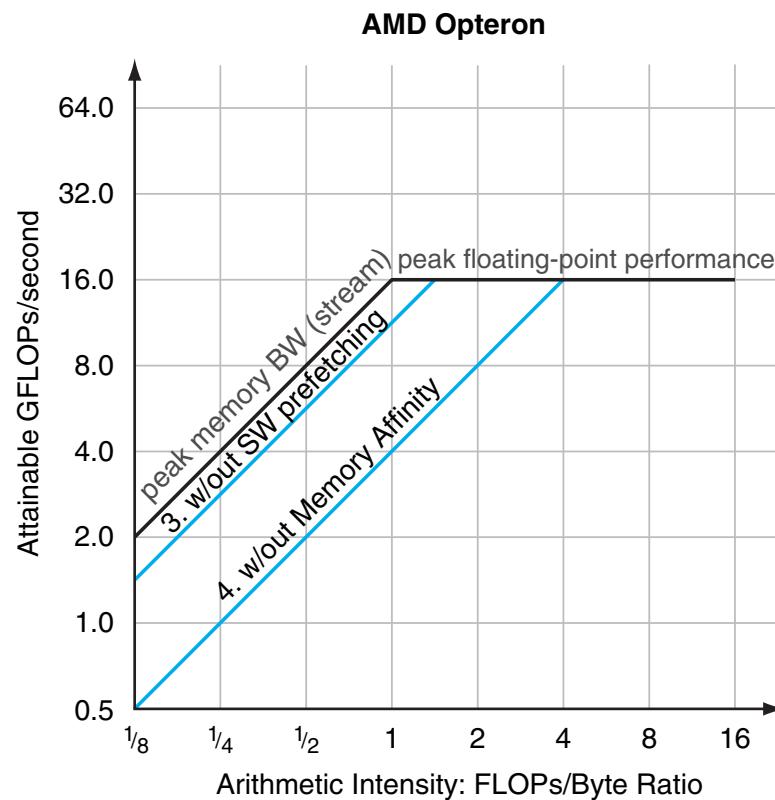
- ➡ As seguintes otimizações (realizadas pelo compilador) permitem frequentemente aumentar o desempenho.
- ➡ Computação:
 - ① Otimizar combinações de operações de vírgula flutuante
Tipicamente, CPUs requerem um número quase igual de somas e multiplicações VF simultâneas para atingir o desempenho de pico
 - ② Execução simultânea de instruções (*ILP=instruction level parallelism*)
O aproveitamento eficaz desta possibilidade também depende da organização do código
- ➡ Acesso a memória:
 - ① Usar instruções especiais para pré-carregar dados de memória (antes de serem necessários): *software prefetching*
 - ② Aproveitar afinidade de memória: colocar os dados em locais de memória com acesso mais eficiente para o processador que os trata.

Modelo com indicação de otimizações

Otimizações de cálculo



Otimizações de memória



Fonte: [COD4]

Referências

COD4 D. A. Patterson & J. L. Hennessy, Computer Organization and Design, 4 ed.

Os tópicos tratados nesta apresentação são abordados nas seguintes secções de [COD4]:

- 7.1–7.3, 7.10