Cache e latência

- Memória que pode ser acessada mais rapidamente que a memória principal. Como preencher a cache?
- Localidade: O princípio que acessar uma posição é normalmente seguido pelo acesso a posição vizinha.
- Depois de acessar uma posição de memória (instrução ou dados), um programa irá tipicamente acessar a posição vizinha (localidade espacial) em um futuro próximo (localidade temporal).

Cache

- Explorar localidade: Faz sentido o acesso a memória operar em blocos de dados em lugar de dados individuais, ex. arrays são alocados em posições contiguas de memória;
- Blocos ou linhas de cache: assumindo linhas de 16 floats: z[0], z[1],..,z[15] estão na mesma linha.

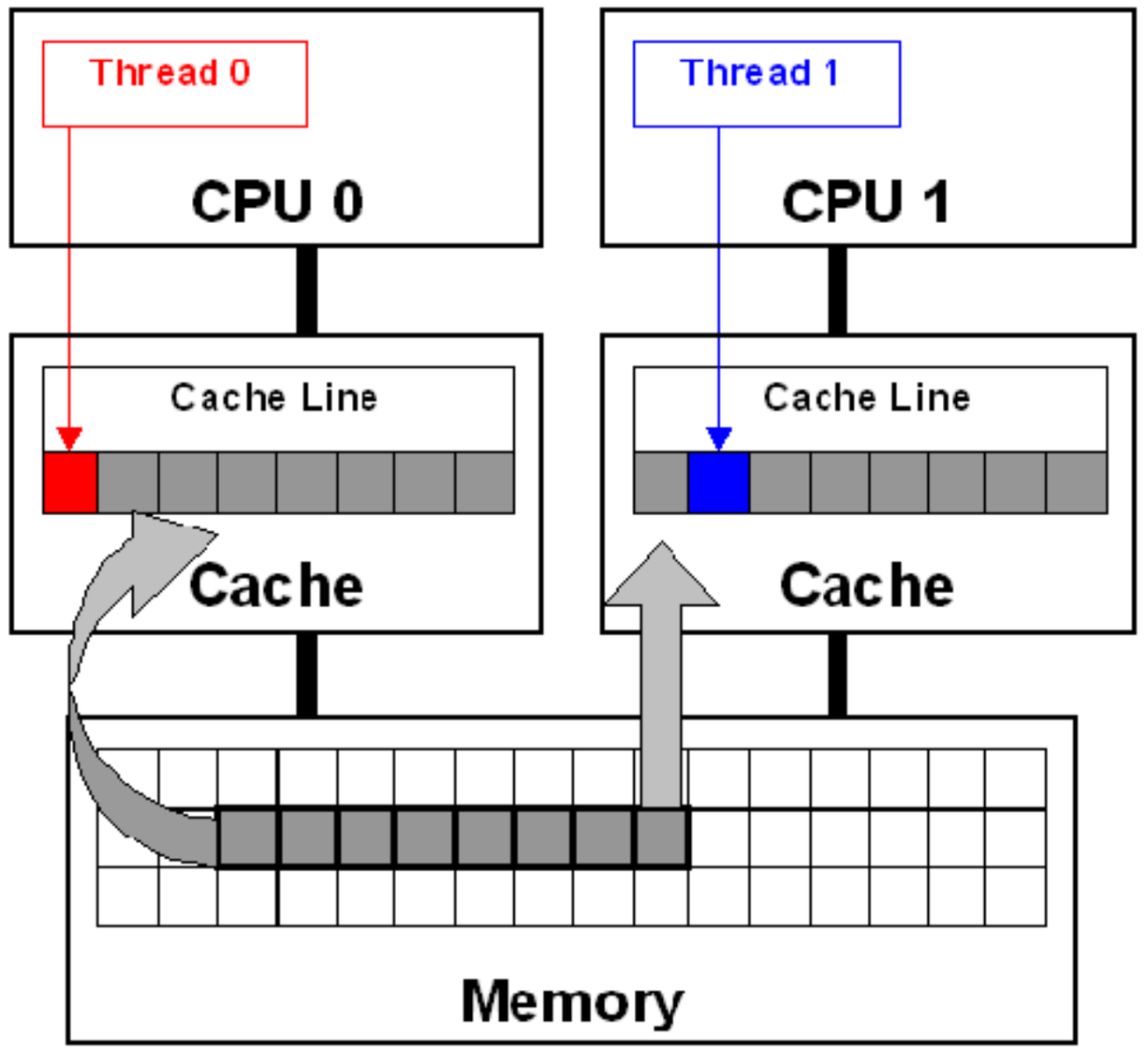
Cache

- False Sharing: Ocorre quando duas variáveis não relacionadas (usadas por processos diferentes) são colocadas na mesma unidade de compartilhamento (página, linha de cache, etc). **O comportamento das threads a respeito do acesso a memória se corresponde ao acesso a variáveis compartilhadas, ainda que não sejam.**
- Qual deve ser o comportamento quando uma variável compartilhada armazenada no cache é modificada por outra thread?
- Invalidação!

Coerencia de cache

- CPU0 escreve na cache. O valor na cache e na mem principal são inconsistentes

- Linha de cache é invalidada e deve ser substituída com uma nova linha da memória



False Sharing

```
int i, j, m, n; double y[m];

/* Assign y = 0 */ ...

for (i = 0; i < m; i++)

    for (j = 0; j < n; j++)

        y[i] += f(i,j);
```

False Sharing

```
/* Variáveis privadas*/ int i, j, iter_count;

/* Variáveis compartilhadas inicializadas por 1 core */
int m, n, core_count; double y[m];
Iter_count = m/core_count ;

/* Core 0 faz */
for (i = 0; i < iter_count; i++)
        for (j = 0; j < n; j++)
                y[i] += f(i,j);

/* Core 1 faz */
for (i = iter_count; i < 2*iter_count; i++)
        for (j = 0; j < n; j++)
                y[i] += f(i,j);

...
```

False Sharing

- Assumindo: 2 cores, m = 8, double = 8 bytes, cache lines = 64 bytes, 0 e 1 executam simultaneamente y[i] += f(i,j), y[0] está armazenado no inicio de uma linha de cache
- Uma linha de cache pode armazenar 8 doubles, y ocupa uma linha inteira;
- A atualização de y[i] invalida uma linha inteira de cache, o outro core precisará buscar a linha atualizada na memória ainda que estejam acessando elementos diferentes;
- Boa parte dos acessos serão a memória principal!
- Os resultados são corretos mas o desempenho é muito ruim.

Desempenho na mult de matrizes

```
1  void *Pth_mat_vect(void* rank) {
2     long my_rank = (long) rank;
3     int i, j;
4     int local_m = m/thread_count;
5     int my_first_row = my_rank*local_m;
6     int my_last_row = (my_rank+1)*local_m - 1;
7
8     for (i = my_first_row; i <= my_last_row; i++) {
9        y[i] = 0.0;
10       for (j = 0; j < n; j++)
11           y[i] += A[i][j]*x[j];
12    }
13
14    return NULL;
15 }  /* Pth_mat_vect */
```

$$E = \frac{S}{t} = \frac{\left(\frac{T_{\text{serial}}}{T_{\text{parallel}}}\right)}{t} = \frac{T_{\text{serial}}}{t \times T_{\text{parallel}}}.$$

Table 4.5 Run-Times and Efficiencies of Matrix-Vector Multiplication (times are in seconds)

	Matrix Dimension					
	$8{,}000{,}000 \times 8$		8000×8000		$8 \times 8{,}000{,}000$	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0.393	1.000	0.345	1.000	0.441	1.000
2	0.217	0.906	0.188	0.918	0.300	0.735
4	0.139	0.707	0.115	0.750	0.388	0.290

False Sharing: como resolvê-lo

- Padding: Encher o final do vetor com espaços. Garantir que a atualização em uma thread não afete a linha de cache de outra thread.
- Cada thread pode usar armazenamento local durante o laço e ao terminar atualizar o armazenamento compartilhado.
- Detecção: Intel Vtune Performance Analyzer
- int arr[PARALLEL * 16] __attribute__ ((aligned (8))); (http://stackoverflow.com/questions/18236603/cache-lines-false-sharing-and-alignment)
- Quando o código é compilado com opções de otimização, o compilador elimina false sharing usando variáveis temporais privadas à thread. Run-time false sharing poderia ser um problema se na hora da compilação a otimização for inabilitada.