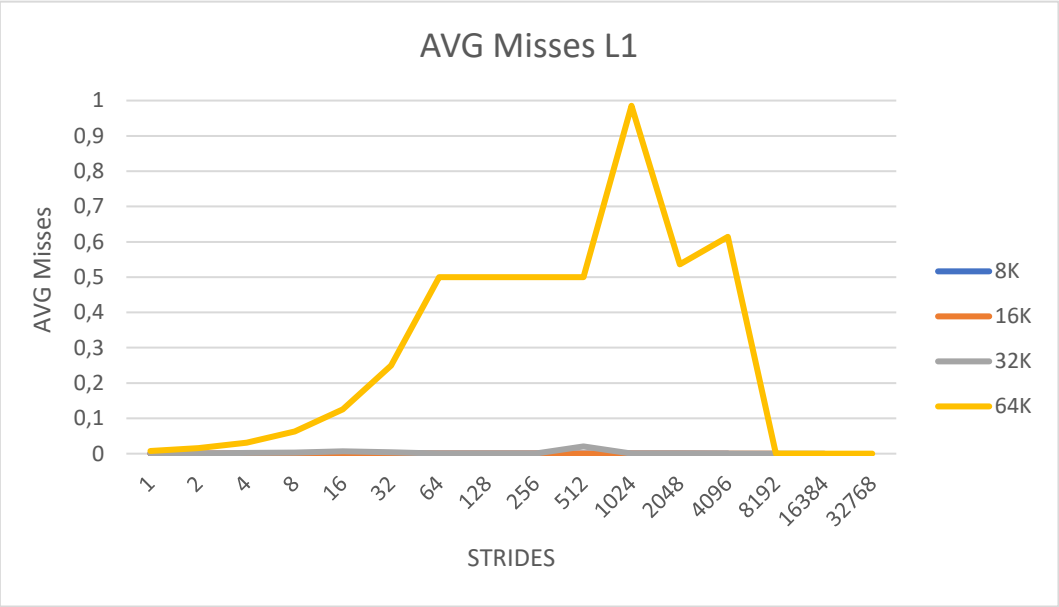
3.1.1.

a)
O evento PAPI_L1_DCM é o único evento adicionado à Performance Application Programming Interface (PAPI). Este evento indica que queremos contar o número de cache misses de dados na cache L1.

b)




c)

- 32KB. Pois a diferença de número de misses entre Array_Size 8KB e 16KB, e 16KB e 32KB é relativamente pequena quanto ao número de misses na passagem de Array_Size 32KB para 64KB.

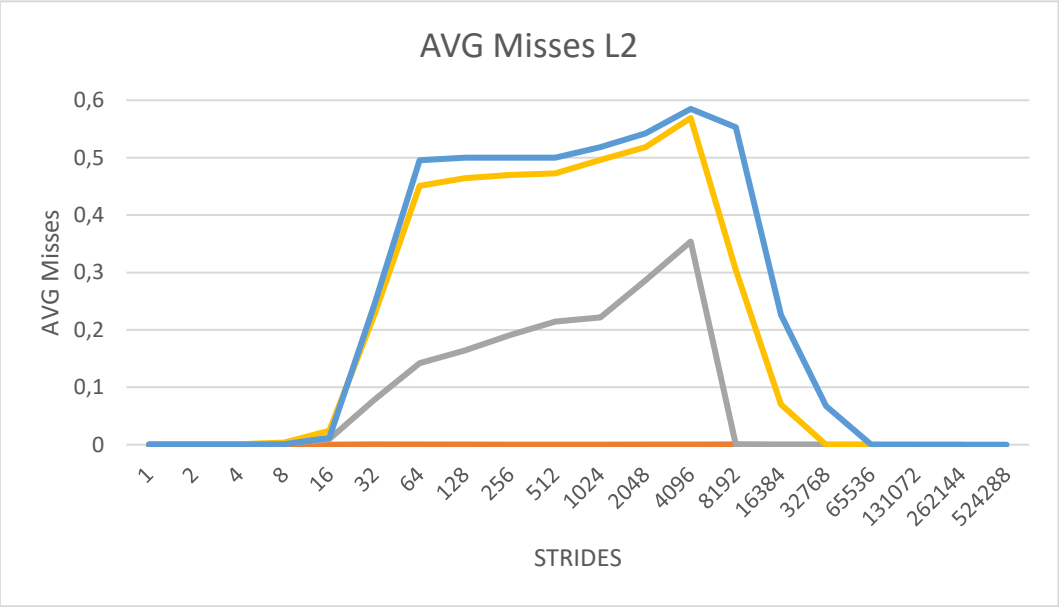
- 16B. Pois para Array_Size > Cache_Size o número de misses aumenta drasticamente de stride 16 para 32, pois ultrapassou o tamanho do bloco, dando mais misses.

- 8-Ways. Pois para Array_Size > Cache_Size (64KB) e considerando stride = 512, 512/2, 512/4 e 512/8, podemos denotar que o número de misses diminui á medida que o stride vai diminuído, concluído que existem 8-Ways. Usámos esta variação nos strides para ir 1 vez, 2 vezes, 4 vezes e 8 vezes à mesma linha de cache.

3.1.2.

a)
Alterar o evento para PAPI_L2_DCM. Modificar o mínimo e máximo para contabilizar com o tamanho de L1 por forma a chegar a L2.
Minimo: 64KB
Máximo: 1024KB

b)




c)

- 2556KB. Pois a diferença de número de misses entre Array_Size 64KB e 128KB, e 128KB e 256KB é relativamente pequena quanto à passagem de Array_Size de 256KB para 512KB.
- 16B. Pois para Array_Size > Cache_Size o número de misses aumenta drasticamente de stride 16 para 32, pois ultrapassou o tamanho do bloco, dando mais misses.
- 8-Ways. Pois para Array_Size > Cache_Size (256KB) e considerando stride = 4096, 4096/2, 4096/4, 4096/8, podemos denotar que o número de misses diminui á medida que o stride vai diminuído, concluído que existem 8-Ways. Usámos esta variação nos strides para ir 1 vez, 2 vezes, 4 vezes e 8 vezes à mesma linha de cache.

3.2.1.

a)
Cada matriz ocupa 1024 * 1024 * sizeof(short) = 2 * 2^20 = 2MB

b)

| | |
|---|---|
| Total number of L1 data cache misses | 1076.467548 * 10^6 |
| Total number of load / store instructions | 2149.586492 * 10^6 |
| Total number of clock cycles | 4902.835582 * 10^6 |
| Elapsed Time | 1.489160 seconds |

c)
(Total number of L1 data cache misses) / (Total load/store instructions)
= 1076.467548 * 10^6 / (2149.586492 * 10^6) = 50.078% Miss-Rate
Hit-Rate = 1 – Miss-Rate = 49.922%

3.2.2.

a)

| | |
|---|---|
| Total number of L1 data cache misses | 33.577801 * 10^6 |
| Total number of load / store instructions | 2149.597310 * 10^6 |
| Total number of clock cycles | 1498.111800 * 10^6 |
| Elapsed Time | 0.455028 seconds |

b)
(Total number of L1 data cache misses) / (Total load/store instructions)
= 33.577801 * 10^6 / (2149.597310 * 10^6) = 1.562% Miss-Rate
Hit-Rate = 1 – Miss-Rate = 98.438%

c)

| | |
|---|---|
| Total number of L1 data cache misses | 33.56490 * 10^6 |
| Total number of load / store instructions | 2149.583464 * 10^6 |
| Total number of clock cycles | 1498.805818 * 10^6 |
| Elapsed Time | 0.492233 seconds |

O tempo de execução aumenta pois estamos a contabilizar a tempo da realização da transposta. No entanto melhora o tempo em comparação ao da alínea anterior pois o miss rate é muito menor.

d)
ΔHit-Rate = Hit-Rate[mm2] – Hit-Rate[mm1] = 98.438 – 39.922 = 48.516%
Speedup(#Clocks) = #Clocks[mm1] / #Clocks[mm2] = 4902.835582 / 1498.111800 = 3.273

Com esta mudança conseguimos um speedup de aproximadamente 3x, este aumento deve-se ao facto de ao transpormos a matriz estamos a aceder ao mesmo bloco do for interior nos dois vetores, diminuindo assim drasticamente o número de mises.

3.2.3.

a)
1 Bloco em cache = 16B
16 / sizeof(short) = 16 / 2 = 8 elementos

b)

| | |
|---|---|
| Total number of L1 data cache misses | 99.118185 * 10^6 |
| Total number of load / store instructions | 3363.885451 * 10^6 |
| Total number of clock cycles | 2847.546882 * 10^6 |
| Elapsed Time | 0.864899 seconds |

c)
(Total number of L1 data cache misses) / (Total load/store instructions)
= 99.118185 * 10^6 / (3363.885451 * 10^6) = 2.947% Miss-Rate
Hit-Rate = 1 – Miss-Rate = 97.053%

d)
ΔHit-Rate = Hit-Rate[mm3] – Hit-Rate[mm1] = 97.053 – 39.922 = 47.131%
Speedup(#Clocks) = #Clocks[mm1] / #Clocks[mm3] = 4902.835582 / 2847.546882 = 1.722

Em relação ao mm1, conseguimos um speedup de 1.722. Isto deve-se ao facto de estarmos a aproveitar sabermos o tamanho do bloco, fazendo a multiplicação de uma linha por todas as colunas de tamanho de um bloco, permitindo assim reduzir o número de misses.

e)
ΔHit-Rate = Hit-Rate[mm3] – Hit-Rate[mm2] = 97.053 – 98.438 = -1.385%
Speedup(#Clocks) = #Clocks[mm2] / #Clocks[mm3] = 1498.111800 / 2847.546882 = 0.526

Embora conseguirmos melhorar o speedup em relação a mm1, não o conseguimos fazer com mm2, aumentando o número de misses e duplicando o Elapsed Time. Isto pode dever-se ao facto de apesar de termos reduzido os misses conflict podemos ter ainda miss capacity da cache. Pode não ser possível guardar todos os elementos na cache. Um benefício em relação ao mm2 é de não ter de fazer a transposta, mas temos de saber o tamanho do bloco.

3.2.3.

| | Lscpu | Meus dados |
|---|---|---|
| L1-d | 32KB | 32KB |
| L2 | 256KB | 256KB |

Pelo que está de acordo com os dados dado pelo CPU.