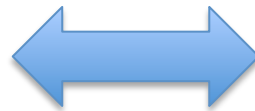


# Memory Mountain

- It performs memory accesses with different locality patterns.
- Simple approach:
  - Allocate array of size “ $W$ ” words
  - Loop over the array with stride index “ $S$ ” and measure speed of memory accesses
  - Vary  $W$  and  $S$  to estimate cache characteristics
- Changing  $W$  varies the total amount of memory accessed by the program.
  - As  $W$  gets larger than one level of the cache, performance of the program will drop.
- Changing  $S$  varies the spatial locality of each access.
  - If  $S$  is less than the size of a cache line, sequential accesses will be fast.
  - If  $S$  is greater than the size of a cache line, sequential accesses will be slower.

# Transpose

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

# Fast Transpose - Step 1

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2
5	6

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

- Copy the data on the buffer block

# Fast Transpose - Step 2

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	5
2	6

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

- Transpose the block

# Fast Transpose - Step 3

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	5
2	6

1	5	0	0
2	6	0	0
0	0	0	0
0	0	0	0

- Copy the transposed block from the buffer block to the destination matrix

# Fast Transpose - Step 4

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

9	10
13	14

1	5	0	0
2	6	0	0
0	0	0	0
0	0	0	0

- Iterates over blocks

# Fast Transpose - Step 5

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

9	13
10	14

1	5	0	0
2	6	0	0
0	0	0	0
0	0	0	0

- Iterates over blocks

# Fast Transpose - Step 6

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

9	13
10	14

1	5	9	13
2	6	10	14
0	0	0	0
0	0	0	0

- Iterates over blocks



# Lab Exercises

- Play and visualize (plot, open-office, etc...) results of the memory-mountain program
- Write a code that performs a matrix transpose and measure its performance.
- Write an optimized version using the Fast Transpose (see slides)
- Use different matrix sizes (1024, 2048, 4096) and play with the block size. Plot the time of execution vs block size. Does the performance gain reach a plateau? Why?
- Using perf (see [this](#), or [here](#) ), visualize cache activity. Plot the number of cache hit vs the block size and discuss the finding. ( for cache profiling use something like `perf stat -e L1-dcache-load-misses {your command}` )