Assignment 4 - Raunak Borker

The matrix transpose operation was benchmarked on the Stanford FarmShare corn machine with the specification amd64 8-core Opteron 2384. The STREAM benchmark for this system yielded a bandwidth for copy of ~4.2 GB/s.

The simple transpose yielded a maximum effective bandwidth of 0.27 GB/s using 0 padding., which is 6.4% of the STREAM benchmark. Cachegrind yields the following results: total L1 misses of 9,448,446 (~9.4million) to read a total of 16,773,121 (~16.7million). The total reads is the 4096*4095 (excluding diagonal). When reading the row-wise elements the miss rate is 1 in 8 since a cache line fits 8 doubles. But reading column wise the miss rate is 8 in 8. Which amounts to 9/8*8.35million ~ 9.4million.

For the optimized transpose results are summarized in the following table:

Block size	Padding	Effective bandwidth (GB/s)
64 X 64	32	3.35
128 X 128	32	4.01
256 X 256	32	3.82
512 X 512	32	3.25
64 X 64	16	3.47
128 X 128	16	3.77
256 X 256	16	3.94
512 X 512	16	3.85

* table reports highest bandwidths obtained for given configuration in all it's runs
The L1 cache was expected to accommodate 2 blocks of sizes <= 64 at a time. But it was observed that using bigger block sizes than this was still beneficial.

These were the best combinations obtained, reaching to more than 90% of the stream benchmark.. Generally odd number for padding gave considerably poorer performance. Also smaller block sizes or smaller padding gave poor results. Cachegrind results without padding indicated a 100% miss rate for the reading of the blocks. This is attributed to cache thrashing. Hence with **8 elements** is just enough to offset this phenomenon as expected. But results are reported for higher padding values. The miss rate for the padded case with one of the above block sizes comes down to ~1/8 for both the blocks accessed in a single iteration. This is expected as now the full block is loaded onto cache and the row access explanation for simple transpose (above) applies here. This gives roughly

2million misses in 16million. There were similar number of misses observed in the swapping process, although it would be expected that this should be close to 0 since the matrices were already in the cache. But this miss rate in the swapping process implies that probably grabbing the first element was always a miss. Nevertheless performance close to STREAM was achieved.

Cachegrind has it’s limitations for example it simulates only the 1st and last levels of cache skipping anything in between. Also it doesn't account for the Translation lakeside buffer misses, which could lower the bandwidth a bit more. This could be partly responsible for the discrepancy between expected and observed bandwidth trends.

Note that Cachegrind screenshots haven’t been included due to page constraints.