Report

Normal matrix multiplication results

Observation1 (base observation)

- Cache block size used = 64B
- Array dim:64*64
- L1 cache size= 4KB, associativity=4
- L2 cache size = 1MB, associativity =16
- # of instruction references = 12816920
- # of data references = 5604424
- # of L1 instruction cache misses = 1672
- # of L1 data cache misses = 272201
- # of L2 cache references = 273873
- # of L2 cache misses = 3175
- Hit rate L1 = 95.1 %, L2 = 99.99% almost

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=4096,4,64 --D1=4096,4,64 --LL=1048576,16,64 ./a
.out 64
==25== Cachegrind, a cache and branch-prediction profiler
==25== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==25== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==25== Command: ./a.out 64
==25==
--25-- warning: L3 cache found, using its data for the LL simulation.
==25== error calling PR_SET_PTRACER, vgdb might block
==25==
==25== I   refs:      12,816,920
==25== I1  misses:         1,672
==25== LLi misses:           838
==25== I1  miss rate:       0.01%
==25== LLi miss rate:       0.01%
==25==
==25== D   refs:       5,604,424  (5,250,221 rd   + 354,203 wr)
==25== D1  misses:       272,201  (  270,655 rd   +   1,546 wr)
==25== LLd misses:         2,337  (    1,125 rd   +   1,212 wr)
==25== D1  miss rate:        4.9% (      5.2%     +     0.4%  )
==25== LLd miss rate:        0.0% (      0.0%     +     0.3%  )
==25==
==25== LL refs:          273,873  (  272,327 rd   +   1,546 wr)
==25== LL misses:          3,175  (    1,963 rd   +   1,212 wr)
==25== LL miss rate:         0.0% (      0.0%     +     0.3%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

Observation2 (associativity decreased from 4 to 2)

- Cache block size used = 64B
- Array dim:64*64
- L1 cache size= 4KB, associativity=2
- L2 cache size = 1MB, associativity =16
- # of instruction references = 12816920

- # of data references = 5604424
- # of L1 instruction cache misses = 1811
- # of L1 data cache misses = 298610
- # of L2 cache references = 300421
- # of L2 cache misses = 3175
- Hit rate L1 = 94.7 %, L2 = 99.99% almost

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=4096,2,64 --D1=4096,2,64 --LL=1048576,16,64 ./a
.out 64
==26== Cachegrind, a cache and branch-prediction profiler
==26== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==26== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==26== Command: ./a.out 64
==26==
--26-- warning: L3 cache found, using its data for the LL simulation.
==26== error calling PR_SET_PTRACER, vgdb might block
==26==
==26== I   refs:      12,816,920
==26== I1  misses:         1,811
==26== LLi misses:           838
==26== I1  miss rate:       0.01%
==26== LLi miss rate:       0.01%
==26==
==26== D   refs:       5,604,424  (5,250,221 rd   + 354,203 wr)
==26== D1  misses:       298,610  (  284,330 rd   +  14,280 wr)
==26== LLd misses:         2,337  (    1,125 rd   +   1,212 wr)
==26== D1  miss rate:        5.3% (      5.4%     +     4.0%  )
==26== LLd miss rate:        0.0% (      0.0%     +     0.3%  )
==26==
==26== LL refs:          300,421  (  286,141 rd   +  14,280 wr)
==26== LL misses:          3,175  (    1,963 rd   +   1,212 wr)
==26== LL miss rate:         0.0% (      0.0%     +     0.3%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

from observation 1 and observation 2 we can say that as we decrease the associativity at L1 there is decrease in hit rate at L1 since, as associativity decrease conflict misses may increase and hence misses may increase so hit rate may decrease.

Observation3 (matrix size increase to 256*256 with associativity 4)

- Cache block size used = 64B
- Array dim: 256*256
- L1 cache size= 4KB, associativity=4
- L2 cache size = 1MB, associativity =16
- # of instruction references = 782276467
- # of data references = 340763081
- # of L1 instruction cache misses = 1672
- # of L1 data cache misses = 16870441
- # of L2 cache references = 18872113

- # of L2 cache misses = 14696
- Hit rate L1 = 95 %, L2 = 99.99% almost

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=4096,4,64 --D1=4096,4,64 --LL=1048576,16,64 ./a
.out 256
==27== Cachegrind, a cache and branch-prediction profiler
==27== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==27== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==27== Command: ./a.out 256
==27==
--27-- warning: L3 cache found, using its data for the LL simulation.
==27== error calling PR_SET_PTRACER, vgdb might block
==27==
==27== I   refs:      782,276,467
==27== I1  misses:          1,672
==27== LLi misses:            838
==27== I1  miss rate:        0.00%
==27== LLi miss rate:        0.00%
==27==
==27== D   refs:      340,763,081  (322,664,430 rd   + 18,098,651 wr)
==27== D1  misses:     16,870,441  ( 16,857,375 rd   +     13,066 wr)
==27== LLd misses:         13,858  (      1,126 rd   +     12,732 wr)
==27== D1  miss rate:         5.0% (        5.2%     +        0.1%  )
==27== LLd miss rate:         0.0% (        0.0%     +        0.1%  )
==27==
==27== LL refs:        16,872,113  ( 16,859,047 rd   +     13,066 wr)
==27== LL misses:          14,696  (      1,964 rd   +     12,732 wr)
==27== LL miss rate:          0.0% (        0.0%     +        0.1%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

Observation4 (matrix size increase to 256*256 with associativity 2)

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=4096,2,64 --D1=4096,2,64 --LL=1048576,16,64 ./a
.out 256
==29== Cachegrind, a cache and branch-prediction profiler
==29== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==29== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==29== Command: ./a.out 256
==29==
--29-- warning: L3 cache found, using its data for the LL simulation.
==29== error calling PR_SET_PTRACER, vgdb might block
==29==
==29== I   refs:      782,276,467
==29== I1  misses:          1,811
==29== LLi misses:            838
==29== I1  miss rate:        0.00%
==29== LLi miss rate:        0.00%
==29==
==29== D   refs:      340,763,081  (322,664,430 rd   + 18,098,651 wr)
==29== D1  misses:     18,531,194  ( 17,514,666 rd   +  1,016,528 wr)
==29== LLd misses:         13,858  (      1,126 rd   +     12,732 wr)
==29== D1  miss rate:         5.4% (        5.4%     +        5.6%  )
==29== LLd miss rate:         0.0% (        0.0%     +        0.1%  )
==29==
==29== LL refs:        18,533,005  ( 17,516,477 rd   +  1,016,528 wr)
==29== LL misses:          14,696  (      1,964 rd   +     12,732 wr)
==29== LL miss rate:          0.0% (        0.0%     +        0.1%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

- Cache block size used = 64B
- Array size = 256*256
- L1 cache size= 4KB, associativity=2

- L2 cache size = 1MB, associativity =16
- # of instruction references = 782276467
- # of data references = 340763081
- # of L1 instruction cache misses = 1811
- # of L1 data cache misses = 18531194
- # of L2 cache references = 18533005
- # of L2 cache misses = 14696
- Hit rate L1 = 94.6 %, L2 = 99.99% almost

By comparing observation 3 and 4 with 1 and 2 we can conclude that even if the size of array increases from 64*64 to 256*256 hit rate remains almost the same at L1 cache because hit rate do not depend on the array size it depends on the cache size, associativity and block size.

Observation5 (cache size increased)

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=65536,2,64 --D1=65536,2,64 --LL=1048576,16,64 .
/a.out 64
==32== Cachegrind, a cache and branch-prediction profiler
==32== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==32== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==32== Command: ./a.out 64
==32==
--32-- warning: L3 cache found, using its data for the LL simulation.
==32== error calling PR_SET_PTRACER, vgdb might block
==32==
==32== I   refs:      12,816,920
==32== I1  misses:           840
==32== LLi misses:           838
==32== I1  miss rate:       0.01%
==32== LLi miss rate:       0.01%
==32==
==32== D   refs:       5,604,424  (5,250,221 rd   + 354,203 wr)
==32== D1  misses:         2,556  (    1,265 rd   +   1,291 wr)
==32== LLd misses:         2,337  (    1,125 rd   +   1,212 wr)
==32== D1  miss rate:        0.0% (      0.0%     +     0.4%  )
==32== LLd miss rate:        0.0% (      0.0%     +     0.3%  )
==32==
==32== LL refs:            3,396  (    2,105 rd   +   1,291 wr)
==32== LL misses:          3,175  (    1,963 rd   +   1,212 wr)
==32== LL miss rate:         0.0% (      0.0%     +     0.3%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

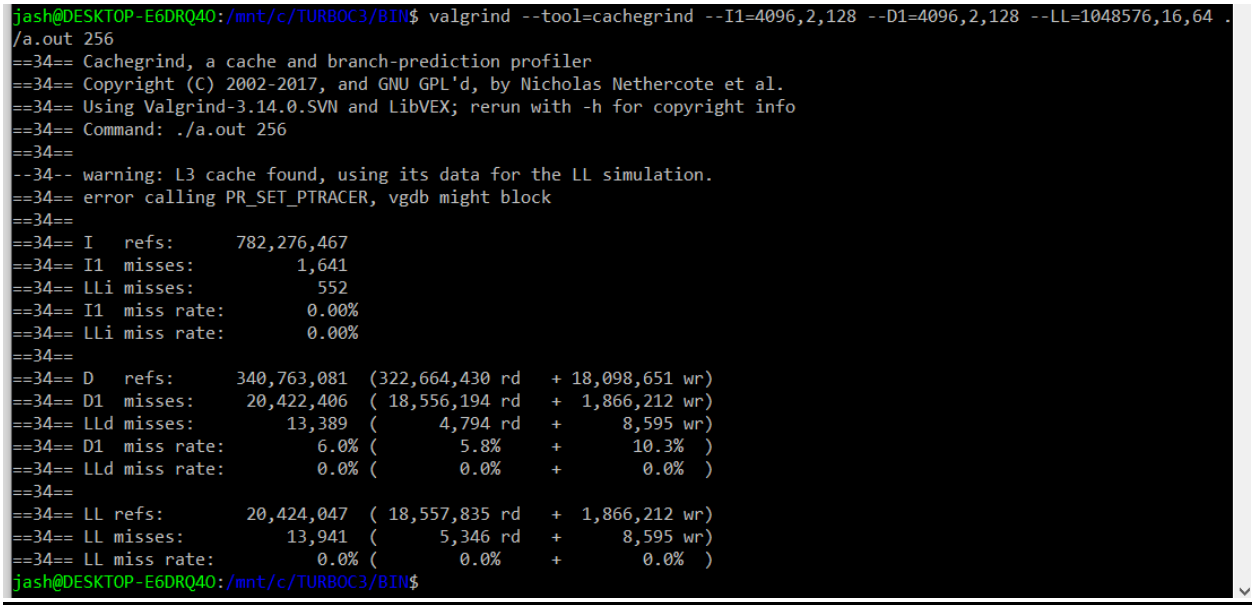
- Cache block size used = 64B
- Array size = 64*64
- L1 cache size= 4KB, associativity=2
- L2 cache size = 1MB, associativity =16
- # of instruction references = 12816920
- # of data references = 5604424
- # of L1 instruction cache misses = 840

- # of L1 data cache misses = 2556
- # of L2 cache references = 3396
- # of L2 cache misses = 3175
- Hit rate L1 = 99.99%, L2 = 99.99% almost

By comparing observation 5 and 2 we can say that as the cache size increase hit rate increase as the hit rate of L1 cache in observation 5 is 99.99% almost and hit rate of L1 cache in observation 2 is 94.7% when the size in observation 5 increases from 4KB to 64KB.

Observation6 (increase block size)

- Cache block size used = 128B
- Array size = 256*256
- L1 cache size= 4KB, associativity=2
- L2 cache size = 1MB, associativity =16
- # of instruction references = 782276467
- # of data references = 340763081
- # of L1 instruction cache misses = 1641
- # of L1 data cache misses = 20422406
- # of L2 cache references = 20424047
- # of L2 cache misses = 13941
- Hit rate L1 = 94 %, L2 = 99.99% almost



Now if we compare observation 3 and 6 then we can say that block size is increase in observation 6 from 64B to 128B but hit rate is reduced from 94.6% to 94% reason behind this is when the block size increase cold start miss decrease but capacity and conflict misses may increase as a result the overall hit rate may increase or decrease it happen that in this case it decreased.

Blocked matrix multiplication

Observation 7 (all cache parameters same as base Observation i.e observation1)

- Cache block size used = 64B
- Array dim:64*64
- L1 cache size= 4KB, associativity=4
- L2 cache size = 1MB, associativity =16
- # of instruction references = 5508834160
- # of data references = 2910874
- # of L1 instruction cache misses = 1678
- # of L1 data cache misses = 2910874
- # of L2 cache references = 2912552
- # of L2 cache misses = 3180
- Hit rate L1 = 99.90 %, L2 = 99.99% almost

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ gcc blockedmatmul.c
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=4096,4,64 --D1=4096,4,64 --LL=1048576,16,64 ./a
.out 64 8
==66== Cachegrind, a cache and branch-prediction profiler
==66== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==66== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==66== Command: ./a.out 64 8
==66==
--66-- warning: L3 cache found, using its data for the LL simulation.
==66== error calling PR_SET_PTRACER, vgdb might block
==66==
==66== I   refs:      5,508,834,160
==66== I1  misses:            1,678
==66== LLi misses:              842
==66== I1  miss rate:          0.00%
==66== LLi miss rate:          0.00%
==66==
==66== D   refs:      2,807,692,525  (2,296,254,890 rd   + 511,437,635 wr)
==66== D1  misses:        2,910,874  (    2,905,485 rd   +       5,389 wr)
==66== LLd misses:            2,338  (        1,153 rd   +       1,185 wr)
==66== D1  miss rate:           0.1% (          0.1%     +         0.0%  )
==66== LLd miss rate:           0.0% (          0.0%     +         0.0%  )
==66==
==66== LL refs:           2,912,552  (    2,907,163 rd   +       5,389 wr)
==66== LL misses:             3,180  (        1,995 rd   +       1,185 wr)
==66== LL miss rate:            0.0% (          0.0%     +         0.0%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

Now comparing observation 1 and 7 we come to a genuine question that why hit rate increased from 95.1% to 99.90% when we applied blocked matrix multiplication rather than normal matrix multiplication even if all the other cache parameters were same?

Answer: If we consider one iteration of blockedmatmul.c program with fixed *i*0, *j*0, and *k*0, we see that (by the three inner loops) 3*(BLK)^2 data are referred to. Now we can choose *BLK* small enough so that these 3*(BLK)^2 data will fit in the local memory and thus achieve *BLK*-fold reuse. In simple words blocking organizes the 2-D array in a program into large chunks called blocks of small 2-D arrays. The program is structured so that it loads a chunk into the L1 cache, does all the reads and writes that it needs to on that chunk, then discards the chunk, loads in the next chunk, and so on. This thing improves the hit rate because whenever the data is to be accessed in the innermost loop of the program blockedmatmul.c it will always cause hit once the chunk/block is loaded. Thus, we can say that using blocking we have improved the temporal locality of innermost loop and hence the hit rate.

Observation 8 (all cache parameters same as Observation3 and array size: 256*256)

```
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$ valgrind --tool=cachegrind --I1=4096,4,64 --D1=4096,4,64 --LL=1048576,16,64 ./a
.out 256 8
==67== Cachegrind, a cache and branch-prediction profiler
==67== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==67== Using Valgrind-3.14.0.SVN and LibVEX; rerun with -h for copyright info
==67== Command: ./a.out 256 8
==67==
--67-- warning: L3 cache found, using its data for the LL simulation.
==67== error calling PR_SET_PTRACER, vgdb might block
==67==
==67== I   refs:      390,940,034,379
==67== I1  misses:              1,678
==67== LLi misses:                842
==67== I1  miss rate:            0.00%
==67== LLi miss rate:            0.00%
==67==
==67== D   refs:      199,151,757,422  (163,022,426,475 rd   + 36,129,330,947 wr)
==67== D1  misses:      5,878,087,138  (  5,878,012,629 rd   +         74,509 wr)
==67== LLd misses:             13,859  (          1,238 rd   +         12,621 wr)
==67== D1  miss rate:             3.0% (            3.6%     +            0.0%  )
==67== LLd miss rate:             0.0% (            0.0%     +            0.0%  )
==67==
==67== LL refs:         5,878,088,816  (  5,878,014,307 rd   +         74,509 wr)
==67== LL misses:              14,701  (          2,080 rd   +         12,621 wr)
==67== LL miss rate:              0.0% (            0.0%     +            0.0%  )
jash@DESKTOP-E6DRQ4O:/mnt/c/TURBOC3/BIN$
```

- Cache block size used = 64B
- Array dim: 256*256
- L1 cache size= 4KB, associativity=4
- L2 cache size = 1MB, associativity =16
- # of instruction references = 390940034379
- # of data references = 199151757422
- # of L1 instruction cache misses = 1678
- # of L1 data cache misses = 5878087138

- # of L2 cache references = 5878088816
- # of L2 cache misses = 14701
- Hit rate L1 = 97 %, L2 = 99.99% almost

Hit rate improved from 95% to 97% in observation 8 due to same reason mentioned in the observation 7.