Profiling is done using `gprof` and `mpicc`. It seems that these tools are not enough to profile MPI code and I couldn't find a proper way to do it. Here are the results:

serial

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.48      4.57     4.57                             main
  0.22      4.58     0.01        1    10.07    10.07  initMat
  0.00      4.58     0.00        1     0.00     0.00  printMat

                     Call graph

granularity: each sample hit covers 2 byte(s) for 0.22% of 4.58 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    4.57    0.01                 main [1]
                0.01    0.00       1/1           initMat [2]
                0.00    0.00       1/1           printMat [3]
-----------------------------------------------
                0.01    0.00       1/1           main [1]
[2]      0.2    0.01    0.00       1         initMat [2]
-----------------------------------------------
                0.00    0.00       1/1           main [1]
[3]      0.0    0.00    0.00       1         printMat [3]
-----------------------------------------------

Index by function name

   [2] initMat                 [1] main                    [3] printMat
```

As expected, the main computation takes up most of the time in serial implementation.

OMP

```
                Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 97.45      7.06     7.06        1     7.06     7.06  initMat
  0.42      7.09     0.03        1     0.03     0.03  printMat

                     Call graph

granularity: each sample hit covers 2 byte(s) for 0.14% of 7.09 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    7.09                 main [1]
                7.06    0.00       1/1           initMat [2]
                0.03    0.00       1/1           printMat [3]
-----------------------------------------------
```

```
                7.06    0.00       1/1           main [1]
[2]     99.6    7.06    0.00       1         initMat [2]
-----------------------------------------------
                0.03    0.00       1/1           main [1]
[3]      0.4    0.03    0.00       1         printMat [3]
-----------------------------------------------

Index by function name

   [2] initMat                 [3] printMat
```

In both MPI and OMP, the main function takes little to no time. `initMat`, the function that initializes the matrices , takes more time than the actual multiplication.

MPI

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
100.32      1.07     1.07                             main
```

CUDA

```
==75484== NVPROF is profiling process 75484, command: ./a.out 2

 Time taken is 23.954443
==75484== Profiling application: ./a.out 2
==75484== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 97.70%  22.3799s         1  22.3799s  22.3799s  22.3799s  multiply(float*, float*, float*)
  1.16%  265.92ms         1  265.92ms  265.92ms  265.92ms  [CUDA memcpy DtoH]
  1.14%  261.20ms         2  130.60ms  130.58ms  130.62ms  [CUDA memcpy HtoD]

==75484== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 99.55%  22.9089s         3  7.63629s  130.68ms  22.6470s  cudaMemcpy
  0.44%  101.02ms         3  33.673ms  419.65us  100.17ms  cudaMalloc
  0.00%  914.48us         3  304.83us  271.89us  366.54us  cudaFree
  0.00%  659.10us        83  7.9400us     502ns  311.76us  cuDeviceGetAttribute
  0.00%  58.119us         1  58.119us  58.119us  58.119us  cudaLaunch
  0.00%  48.038us         1  48.038us  48.038us  48.038us  cuDeviceTotalMem
  0.00%  41.055us         1  41.055us  41.055us  41.055us  cuDeviceGetName
  0.00%  9.2000us         3  3.0660us     315ns  7.9910us  cudaSetupArgument
  0.00%  3.9460us         1  3.9460us  3.9460us  3.9460us  cudaConfigureCall
  0.00%  3.5950us         2  1.7970us     701ns  2.8940us  cuDeviceGetCount
  0.00%  1.7160us         2     858ns     727ns     989ns  cuDeviceGet
  0.00%     494ns         1     494ns     494ns     494ns  cudaGetLastError
```

Here memory transfer takes considerable amount of time apart from the time taken by `multiply`. However, for smaller matrix sizes `memorycpy` takes more time than `multiply`.

Cachegrind analysis: The instruction miss rates and data miss rates are mentioned below. We can see that serial and OMP has similar statistics where as in MPI instruction and data-miss rates are considerably higher.

serial

```
==2927== Cachegrind, a cache and branch-prediction profiler
==2927== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==2927== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==2927== Command: ./serial
==2927==
--2927-- warning: L3 cache found, using its data for the LL simulation.

 Time taken is 235.619629
==2927==
==2927== I   refs:      33,037,538,463
==2927== I1  misses:             1,819
==2927== LLi misses:             1,690
==2927== I1  miss rate:           0.00%
==2927== LLi miss rate:           0.00%
==2927==
==2927== D   refs:      14,019,014,062  (13,015,728,675 rd   + 1,003,285,387 wr)
==2927== D1  misses:        62,772,493  (    62,644,822 rd   +       127,671 wr)
==2927== LLd misses:        62,761,087  (    62,633,938 rd   +       127,149 wr)
==2927== D1  miss rate:            0.4% (           0.5%     +           0.0%  )
==2927== LLd miss rate:            0.4% (           0.5%     +           0.0%  )
==2927==
==2927== LL refs:           62,774,312  (    62,646,641 rd   +       127,671 wr)
==2927== LL misses:         62,762,777  (    62,635,628 rd   +       127,149 wr)
==2927== LL miss rate:             0.1% (           0.1%     +           0.0%  )
```

OMP

```
==3133== Cachegrind, a cache and branch-prediction profiler
==3133== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==3133== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==3133== Command: ./omp
==3133==
--3133-- warning: L3 cache found, using its data for the LL simulation.

 Time taken is 244.495880
==3133==
==3133== I   refs:      33,037,538,399
==3133== I1  misses:             1,816
==3133== LLi misses:             1,687
==3133== I1  miss rate:           0.00%
==3133== LLi miss rate:           0.00%
==3133==
==3133== D   refs:      14,019,014,043  (13,015,728,664 rd   + 1,003,285,379 wr)
==3133== D1  misses:        62,772,305  (    62,644,708 rd   +       127,597 wr)
==3133== LLd misses:        62,761,087  (    62,633,937 rd   +       127,150 wr)
==3133== D1  miss rate:            0.4% (           0.5%     +           0.0%  )
==3133== LLd miss rate:            0.4% (           0.5%     +           0.0%  )
==3133==
==3133== LL refs:           62,774,121  (    62,646,524 rd   +       127,597 wr)
==3133== LL misses:         62,762,774  (    62,635,624 rd   +       127,150 wr)
==3133== LL miss rate:             0.1% (           0.1%     +           0.0%  )
```

MPI

```
==3225== Cachegrind, a cache and branch-prediction profiler
==3225== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==3225== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==3225== Command: mpirun -np 8 ./mpi
==3225==
--3225-- warning: L3 cache found, using its data for the LL simulation.
[warn] Epoll ADD(4) on fd 1 failed.  Old events were 0;
read change was 0 (none); write change was 1 (add): Operation not permitted

 Time taken is 3.978104
==3225==
==3225== I   refs:      60,582,190
==3225== I1  misses:        83,243
==3225== LLi misses:        10,336
==3225== I1  miss rate:       0.14%
==3225== LLi miss rate:       0.02%
==3225==
==3225== D   refs:      24,637,760  (16,965,706 rd   + 7,672,054 wr)
==3225== D1  misses:       849,555  (   795,670 rd   +    53,885 wr)
==3225== LLd misses:        76,610  (    48,015 rd   +    28,595 wr)
==3225== D1  miss rate:        3.4% (       4.7%     +       0.7%  )
==3225== LLd miss rate:        0.3% (       0.3%     +       0.4%  )
==3225==
==3225== LL refs:          932,798  (   878,913 rd   +    53,885 wr)
==3225== LL misses:         86,946  (    58,351 rd   +    28,595 wr)
==3225== LL miss rate:         0.1% (       0.1%     +       0.4%  )
```