Chen GU Assignment 3 CPSC 424

Building and running information

Development environment

Module loaded

```
Currently Loaded Modulefiles:
  1) Base/yale_hpc
  2) Langs/Intel/15
  3) MPI/OpenMPI/1.8.6-intel15
  4) Tools/TotalView/8.14.1-8
  5) GPU/Cuda/8.0
```

env command

```
MKLROOT=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/mkl
MANPATH=/home/apps/fas/Tools/TotalView/toolworks/totalview.8.14.1-8/man:/u
sr/local/cluster/hpc/MPI/OpenMPI/1.8.6-intel15/share/man:/home/apps/fas/La
ngs/Intel/2015_update2/composer_xe_2015.2.164/man/en_US:/home/apps/fas/Lan
gs/Intel/2015_update2/composer_xe_2015.2.164/debugger/gdb/intel64/share/ma
n/:/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/debugger
/gdb/intel64_mic/share/man/:/usr/share/man:/opt/moab/share/man:
GDB_HOST=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/de
bugger/gdb/intel64_mic/bin/gdb-ia-mic
HOSTNAME=compute-45-4.local
PBS_VERSION=TORQUE-4.2.9
IPPROOT=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/ipp
INTEL_LICENSE_FILE=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_201
5.2.164/licenses:/opt/intel/licenses:/home/apps/fas/Licenses/intel_site.li
c
TERM=xterm-256color
SHELL=/bin/bash
HISTSIZE=1000
GDBSERVER_MIC=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.1
64/debugger/gdb/target/mic/bin/gdbserver
```

```
PBS_JOBNAME=STDIN
LIBRARY_PATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/lib64:/usr/local/cluster/
hpc/GPU/Cuda/8.0/lib:/usr/local/cluster/hpc/MPI/OpenMPI/1.8.6-intel15/lib:
/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/ipp/../comp
iler/lib/intel64:/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.
2.164/ipp/lib/intel64:/home/apps/fas/Langs/Intel/2015_update2/composer_xe_
2015.2.164/compiler/lib/intel64:/home/apps/fas/Langs/Intel/2015_update2/co
mposer_xe_2015.2.164/mkl/lib/intel64:/home/apps/fas/Langs/Intel/2015_updat
e2/composer_xe_2015.2.164/tbb/lib/intel64/gcc4.4
PERL5LIB=/opt/moab/lib/perl5
FPATH=/usr/local/cluster/hpc/MPI/OpenMPI/1.8.6-intel15/include:/home/apps/
fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/mkl/include
PBS_ENVIRONMENT=PBS_INTERACTIVE
QTDIR=/usr/lib64/qt-3.3
QTINC=/usr/lib64/qt-3.3/include
INCLUDE_PATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/include
MIC_LD_LIBRARY_PATH=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_20
15.2.164/mpirt/lib/mic:/home/apps/fas/Langs/Intel/2015_update2/composer_xe
_2015.2.164/ipp/lib/mic:/home/apps/fas/Langs/Intel/2015_update2/composer_x
e_2015.2.164/compiler/lib/mic:/home/apps/fas/Langs/Intel/2015_update2/comp
oser_xe_2015.2.164/mkl/lib/mic:/opt/intel/mic/coi/device-linux-release/lib
:/opt/intel/mic/myo/lib:/home/apps/fas/Langs/Intel/2015_update2/composer_x
e_2015.2.164/tbb/lib/mic
PBS_O_WORKDIR=/lustre/home/client/fas/cpsc424/cg736/as/as6
ANT_HOME=/opt/rocks
LC_ALL=en_US
PBS_TASKNUM=1
USER=cg736
LD_LIBRARY_PATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/lib64:/usr/local/clust
er/hpc/MPI/OpenMPI/1.8.6-intel15/lib:/home/apps/fas/Langs/Intel/2015_updat
e2/composer_xe_2015.2.164/mpirt/lib/intel64:/home/apps/fas/Langs/Intel/201
5_update2/composer_xe_2015.2.164/ipp/../compiler/lib/intel64:/home/apps/fa
s/Langs/Intel/2015_update2/composer_xe_2015.2.164/ipp/lib/intel64:/home/ap
ps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/ipp/tools/intel64/p
erfsys:/opt/intel/mic/coi/host-linux-release/lib:/opt/intel/mic/myo/lib:/h
ome/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/compiler/lib/
intel64:/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/mkl
/lib/intel64:/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.16
4/tbb/lib/intel64/gcc4.4:/home/apps/fas/Langs/Intel/2015_update2/composer_
xe_2015.2.164/debugger/ipt/intel64/lib
PBS_O_HOME=/home/fas/cpsc424/cg736
MIC_LIBRARY_PATH=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.
2.164/compiler/lib/mic:/home/apps/fas/Langs/Intel/2015_update2/composer_xe
_2015.2.164/mpirt/lib/mic:/home/apps/fas/Langs/Intel/2015_update2/composer
_xe_2015.2.164/tbb/lib/mic
ROCKS_ROOT=/opt/rocks
CPATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/include:/usr/local/cluster/hpc/M
PI/OpenMPI/1.8.6-intel15/include:/home/apps/fas/Langs/Intel/2015_update2/c
omposer_xe_2015.2.164/ipp/include:/home/apps/fas/Langs/Intel/2015_update2/
```

```
composer_xe_2015.2.164/mkl/include:/home/apps/fas/Langs/Intel/2015_update2
/composer_xe_2015.2.164/tbb/include
PBS_WALLTIME=14400
PBS_GPUFILE=/var/spool/torque/aux//5451302.rocks.omega.hpc.yale.internalgp
u
PBS_MOMPORT=15003
PBS_O_QUEUE=cpsc424gpu
YHPC_COMPILER=Intel
OMPI_MCA_orte_precondition_transports=f20cd2d28f432704-15e3f8c3bb8e89d6
NLSPATH=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/com
piler/lib/intel64/locale/%l_%t/%N:/home/apps/fas/Langs/Intel/2015_update2/
composer_xe_2015.2.164/ipp/lib/intel64/locale/%l_%t/%N:/home/apps/fas/Lang
s/Intel/2015_update2/composer_xe_2015.2.164/mkl/lib/intel64/locale/%l_%t/%
N:/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/debugger/
gdb/intel64_mic/share/locale/%l_%t/%N:/home/apps/fas/Langs/Intel/2015_upda
te2/composer_xe_2015.2.164/debugger/gdb/intel64/share/locale/%l_%t/%N
MAIL=/var/spool/mail/cg736
PBS_O_LOGNAME=cg736
PATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/samples/bin/x86_64/linux/release:
/usr/local/cluster/hpc/GPU/Cuda/8.0/bin:/home/apps/fas/Tools/TotalView/too
lworks/totalview.8.14.1-8/bin:/home/apps/fas/Tools/TotalView/toolworks/mem
oryscape.3.6.1-8/bin:/home/apps/fas/Tools/TotalView:/usr/local/cluster/hpc
/MPI/OpenMPI/1.8.6-intel15/bin:/home/apps/fas/Langs/Intel/2015_update2/com
poser_xe_2015.2.164/bin/intel64:/home/apps/fas/Langs/Intel/2015_update2/co
mposer_xe_2015.2.164/mpirt/bin/intel64:/home/apps/fas/Langs/Intel/2015_upd
ate2/composer_xe_2015.2.164/debugger/gdb/intel64_mic/bin:/home/apps/fas/La
ngs/Intel/2015_update2/composer_xe_2015.2.164/debugger/gdb/intel64/bin:/ho
me/apps/fas/Modules:/usr/lib64/qt-3.3/bin:/opt/moab/bin:/bin:/usr/bin:/usr
/local/sbin:/usr/sbin:/sbin:/usr/java/latest/bin:/opt/rocks/bin:/opt/rocks
/sbin:/home/fas/hpcprog/ahs3/bin:/home/apps/bin:/home/fas/cpsc424/cg736/bi
n
YHPC_COMPILER_MINOR=164
PBS_O_LANG=en_US.iso885915
PBS_JOBCOOKIE=73291E674DD8783304A24E81A99DA7D9
TBBROOT=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/tbb
C_INCLUDE_PATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/include:/usr/local/clus
ter/hpc/MPI/OpenMPI/1.8.6-intel15/include
F90=ifort
PWD=/home/fas/cpsc424/cg736
_LMFILES_=/home/apps/fas/Modules/Base/yale_hpc:/home/apps/fas/Modules/Lang
s/Intel/15:/home/apps/fas/Modules/MPI/OpenMPI/1.8.6-intel15:/home/apps/fas
/Modules/Tools/TotalView/8.14.1-8:/home/apps/fas/Modules/GPU/Cuda/8.0
YHPC_COMPILER_MAJOR=2
JAVA_HOME=/usr/java/latest
GDB_CROSS=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/d
ebugger/gdb/intel64_mic/bin/gdb-mic
DOMAIN=omega
PBS_NODENUM=0
LANG=C
```

```
MODULEPATH=/home/apps/fas/Modules
MOABHOMEDIR=/opt/moab
CUDADIR=/usr/local/cluster/hpc/GPU/Cuda/8.0
YHPC_COMPILER_RELEASE=2015
LOADEDMODULES=Base/yale_hpc:Langs/Intel/15:MPI/OpenMPI/1.8.6-intel15:Tools
/TotalView/8.14.1-8:GPU/Cuda/8.0
KDEDIRS=/usr
PBS_NUM_NODES=1
F77=ifort
PBS_O_SHELL=/bin/bash
LM_LICENSE_FILE=/home/apps/fas/Tools/TotalView/license.dat
PBS_JOBID=5451302.rocks.omega.hpc.yale.internal
MPM_LAUNCHER=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.16
4/debugger/mpm/bin/start_mpm.sh
CXX=icpc
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
INTEL_PYTHONHOME=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.
2.164/debugger/python/intel64/
SHLVL=1
HOME=/home/fas/cpsc424/cg736
PBS_O_HOST=login-0-0.local
FC=ifort
PBS_VNODENUM=0
LOGNAME=cg736
QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
PBS_QUEUE=cpsc424gpu
MODULESHOME=/usr/share/Modules
LESSOPEN=||/usr/bin/lesspipe.sh %s
PBS_MICFILE=/var/spool/torque/aux//5451302.rocks.omega.hpc.yale.internalmi
c
PBS_O_MAIL=/var/spool/mail/cg736
arch=intel64
INFOPATH=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/de
bugger/gdb/intel64/share/info/:/home/apps/fas/Langs/Intel/2015_update2/com
poser_xe_2015.2.164/debugger/gdb/intel64_mic/share/info/
CC=icc
PBS_NP=5
PBS_NUM_PPN=1
PBS_O_SERVER=rocks.omega.hpc.yale.internal
INCLUDE=/home/apps/fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/mkl
/include
MPI_PATH=/usr/local/cluster/hpc/MPI/OpenMPI/1.8.6-intel15
G_BROKEN_FILENAMES=1
PBS_NODEFILE=/var/spool/torque/aux//5451302.rocks.omega.hpc.yale.internal
PBS_O_PATH=/usr/local/cluster/hpc/GPU/Cuda/8.0/samples/bin/x86_64/linux/re
lease:/usr/local/cluster/hpc/GPU/Cuda/8.0/bin:/home/apps/fas/Tools/TotalVi
ew/toolworks/totalview.8.14.1-8/bin:/home/apps/fas/Tools/TotalView/toolwor
```

```
ks/memoryscape.3.6.1-8/bin:/home/apps/fas/Tools/TotalView:/usr/local/clust
er/hpc/MPI/OpenMPI/1.8.6-intel15/bin:/home/apps/fas/Langs/Intel/2015_updat
e2/composer_xe_2015.2.164/bin/intel64:/home/apps/fas/Langs/Intel/2015_upda
te2/composer_xe_2015.2.164/mpirt/bin/intel64:/home/apps/fas/Langs/Intel/20
15_update2/composer_xe_2015.2.164/debugger/gdb/intel64_mic/bin:/home/apps/
fas/Langs/Intel/2015_update2/composer_xe_2015.2.164/debugger/gdb/intel64/b
in:/home/apps/fas/Modules:/usr/lib64/qt-3.3/bin:/opt/moab/bin:/usr/local/b
in:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/java/latest/bin:/opt
/rocks/bin:/opt/rocks/sbin:/home/fas/hpcprog/ahs3/bin:/home/apps/bin:/home
/fas/cpsc424/cg736/bin
BASH_FUNC_module()=() {  eval `/usr/bin/modulecmd bash $*`
}
_=/bin/env
```

How to run the code

The source code and scripts of different tasks are organized in sub directory `task1`, `task2` and `task3` respectively. For each task, I wrote a script which can run the code automatically.

To run the program, you can enter each sub directory and submit the job to running queue by typing:

```
qsub xxx.sh
```

Note: here xxx should be replaced by specific script name.
For `task3`, I do not provide a script. Since **NTB** is defined as macro, you have change the value of **NTB** and rebuild the program and rerun.

Output and evaluation

Throughout the assignment, I used **M2090** for all the final timings.

Task 1

Part a

I ran my kernel and also my "kij" code using the matrix dimension described in the table below.

	Run1	Run2	Run3	Run4	Run5
n	1024	8192	1024	8192	8192
m	1024	8192	1024	8192	1024
p	1024	8192	8192	1024	8192

The following table shows the best timing for each case along with the corresponding block and grid dimension to achieve the best timing.

	Grid Dimension	Block Dimension	GPU TIME (ms)	CPU TIME (ms)
Run1	64 * 64	16 * 16	37.986271	884.213074
Run2	256 * 256	32 * 32	18498.746094	472161.750000
Run3	32 * 32	32 * 32	304.956665	7073.050781
Run4	256 * 256	32 * 32	2283.919189	59344.769531
Run5	32 * 256	32 * 32	2415.672852	66298.593750

Part b

	Grid Dimension	Block Dimension	Single Precision GPU TIME (ms)	Double Precision GPU TIME (ms)
m=p=n8192	256 * 256	32 * 32	18498.746094	28609.435547

We can see from the table above that GPU is efficient for single precision calculation. The time used in calculating double precision matrix multiplication increases by about 50%.

Part c

The global memory of M2090 is **6GB**. Each float is **4B** and we have to store **3** matrices in the memory (A, B and C). Below is the formula to calculate the largest matrix dimension

```
max = sqrt(6*2^30 / 4 / 3) = 23179
```

However, in reality, this max dimension cannot be achieved since some of memory is

occupied for special purposes (e.g. thread local memory).

I wrote a script to increase the dimension by 1000 at a time and see if the timing is still larger than some number. Once the timing drops to near zero, I break the loop and report the largest size with reasonable timing. The largest size I found is **n = p = m = 20684**. The corresponding GPU TIME is **341605.875000 ms**.

Task 2

Part a

The following table shows the best timing for each case along with the corresponding block and grid dimension and tile size to achieve the best timing.

	Grid Dimension	Block Dimension	Tile Size	GPU TIME (ms)
Run1	32 * 32	32 * 32	32	14.724096
Run2	256 * 256	32 * 32	32	7456.747559
Run3	32 * 32	32 * 32	32	118.315903
Run4	256 * 256	32 * 32	32	951.742554
Run5	32 * 256	32 * 32	32	946.339355

Part b

	Grid Dimension	Block Dimension	Single Precision GPU TIME (ms)	Double Precision GPU TIME (ms)
m=p=n8192	256 * 256	32 * 32	7456.747559	12797.792969

With tiled calculation, both single and double precision calculation accelerate. However, single precision calculation is still much faster than double precision calculation (about 60%).

Part c

The largest matrix dimension is constrained by the global memory size. The theoretical max value is the same as that in task1 (about 23179).

I reused the script I used in task 1. Again, the largest size I found is **n = p = m = 20684**. The

corresponding GPU TIME is **164544.390625 ms**. The largest size is the same as that in task 1. But the corresponding GPU TIME decreases significantly (from **341605.875000** ms to **164544.390625 ms**)

Note

In task 2, I assume that the tiles and blocks are square and of the same size, but the matrix dimensions can be arbitrary. Below the is part of code that implement arbitrary matrix size.

```
/* if p is not an multiple of TW and it is the last tile,
  * the number of remaining elememnt is in this tile is p % TW
  * Otherwise, it is a full tile with TW elements*/
 int boarder = ((p%TW) !=0 && i == tile_num-1) ? p % TW : TW;
 int indexa=ty*TW, indexb=tx;
 for(; indexa<ty*TW+boarder; indexa++,indexb+=TW) {
   cvalue += atile[indexa] * btile[indexb];
 }
```

Task 3

I implemented the multi-tiled strategy described in task 3. I use **NTB** to denote the number of adjacent tiles handled at one time. I started trials with **NTB** starting from 2 and incremented **NTB** at a time before **NTB** reaches 16. After that, I multiply **NTB** by 2 at a step. The following table shows the successful trials.

	Grid Dimension	Block Dimension	Tile Size	NTB	GPU TIME (ms)
n=p=m=8192	256 * 256	32 * 32	32	2	8068.263672
n=p=m=8192	256 * 256	32 * 32	32	3	8188.750488
n=p=m=8192	256 * 256	32 * 32	32	4	8025.918945
n=p=m=8192	256 * 256	32 * 32	32	5	7560.078125
n=p=m=8192	256 * 256	32 * 32	32	6	7890.470215
n=p=m=8192	256 * 256	32 * 32	32	7	7868.088867
n=p=m=8192	256 * 256	32 * 32	32	8	7410.684570

n=p=m=8192	256 * 256	32 * 32	32	9	7819.804688
n=p=m=8192	256 * 256	32 * 32	32	10	7373.133789
n=p=m=8192	256 * 256	32 * 32	32	11	7782.720215
n=p=m=8192	256 * 256	32 * 32	32	12	7753.938965
n=p=m=8192	256 * 256	32 * 32	32	13	7322.179199
n=p=m=8192	256 * 256	32 * 32	32	14	7746.424805
n=p=m=8192	256 * 256	32 * 32	32	15	7727.256348
n=p=m=8192	256 * 256	32 * 32	32	16	7274.133789
n=p=m=8192	256 * 256	32 * 32	32	32	7229.791504
n=p=m=8192	256 * 256	32 * 32	32	64	7210.930664
n=p=m=8192	256 * 256	32 * 32	32	128	7086.891602
n=p=m=8192	256 * 256	32 * 32	32	256	**7068.958008**

We can see when **NTB = 256**, it achieves the besting timing (7068.958008 ms).

The table below shows the comparison timing between tiled multiplication and multi-tiled multiplication. Multi-tiled calculation is slightly fasster than tile calculation when **NTB** is large. Overall, the timing of these 2 approaches is quite close. As described the assigment6.pdf, though the number of tiles loaded from global memory into shared memory would be reduced by ~25%. Unfortunately, multi-tiled approach does increase the usage of registers and shared memory, so the benefit of using multi-tiled calculation may not always be significant.

	Grid Dimension	**Block Dimension**	**Tile Size**	**Tiled GPU TIME (ms)**	**Multi-Tiled (NTB=256) GPU TIME (ms)**
n=p=m=8192	256 * 256	32 * 32	32	7456.747559	7068.958008

Note

In task 3, I assume that the matrix dimensions are multiples of the tile width. But my code

can handle the case where the final block has fewer adjacent tiles than the other blocks. The following code does the job

```
int remaining_NTB = (Grid_Dim_X%NTB!=0&&blockIdx.x==gridDim.x-1)? Grid_Dim
_X%NTB : NTB;
  ...

while(row < n && col < m && kt < remaining_NTB){
    c[index] = cvalue[kt];
    kt++;
    index += TW;
}
```