COMP 605: Introduction to Parallel Computing
Lecture : GPU/CUDA - Running Jobs on tuckoo

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Posted: 04/25/17
Last Update: May 9, 2017

Table of Contents

Running GPU/CUDA Jobs on tuckoo

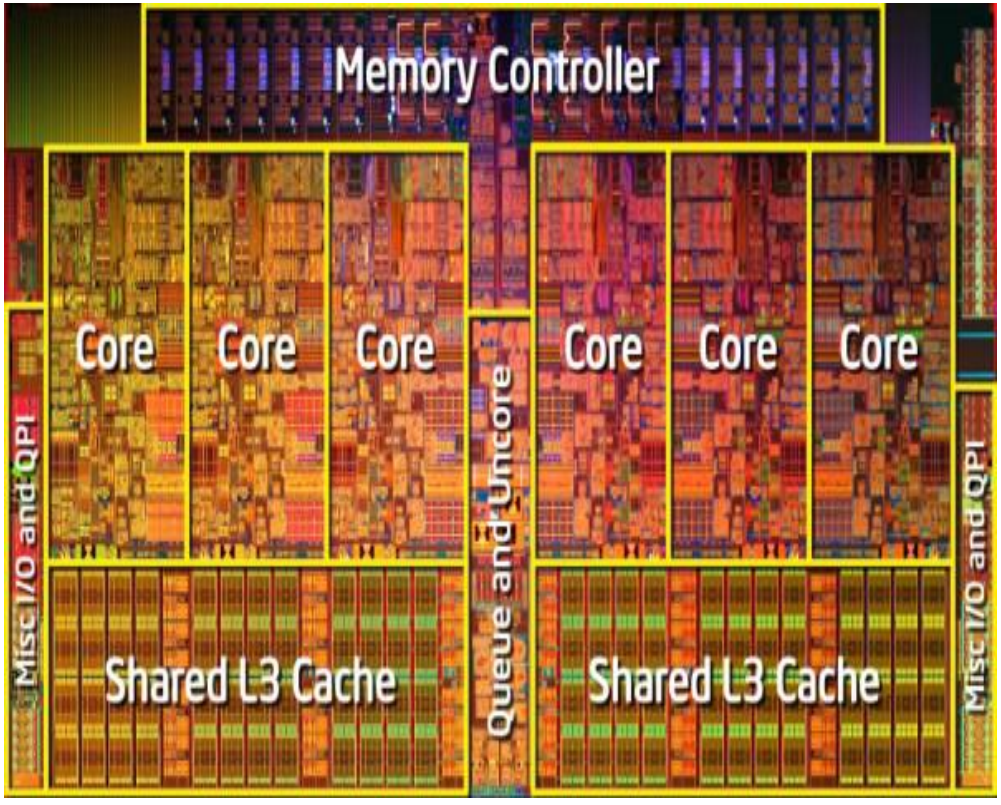
A CPU/GPU Cluster: tuckoo.sdsu.edu

```
Welcome to tuckoo (baby) -- a CSRC student cluster
-------------------------------------------------
CPUs & RAM
-----------------------------------------------
nodes 1,2,4       Xeon X3360   @ 2.83GHz, 8GB ea.
node5             Xeon E5420   @ 2.50GHz, 20GB
node6             Xeon E5-1650 @ 3.20GHz, 64GB
node7             Xeon X5650   @ 2.67GHz, 48GB
node8             Xeon E5620   @ 2.40GHz, 48GB
node9             Xeon E5-1660 @ 3.30GHz, 32GB
node11            Xeon E5-2650 @ 2.60GHz, 64GB

GPUs
-----------------------------------------------
node9  has 2     GTX 480 gpu cards (1.6GB dev ram ea.)
node8  has 2     C2075   gpu cards (  6GB dev ram ea.)
node7  has 2     C1060   gpu cards (  4GB dev ram ea.)
node11 has 1     K40     gpu card  (                 )

see files /examples/cuda/nodeX.SPECS (X=7,8,9,11)*
      and /examples/cuda/GPU.SPECS

*Note -- the specs appear to indicate that the C2075 cards only
         have 4GB ram; they should indeed have 6GB ram each.
```

A CPU/GPU Cluster: tuckoo.sdsu.edu

```
-----------------------------------------
   to see available resources on the nodes see the output from "pbsnodes"
   to see mpirun help/options do "mpirun --help"
   example codes, makefiles, and batch scripts are in /examples
-----------------------------------------------------------------------
   the cluster system has 9 compute nodes with various CPUs:

    Node name               #Avail Cores   Node Properties**    GPUs
-----------------------------------------------------------------------
    node1,node2,node4  4ea.                core4, mpi            no
-----------------------------------------------------------------------
    node6                    6             core6, mpi            no
    node9                    6             core6, mpi            yes
-----------------------------------------------------------------------
    node5                    8             core8, mpi            no
    node8                    8             core8, mpi            yes
-----------------------------------------------------------------------
    node7                    12            core12,mpi            yes
-----------------------------------------------------------------------
    node11                   16            core16,mpi            yes
-----------------------------------------------------------------------
     **see the output from "pbsnodes -a".
```

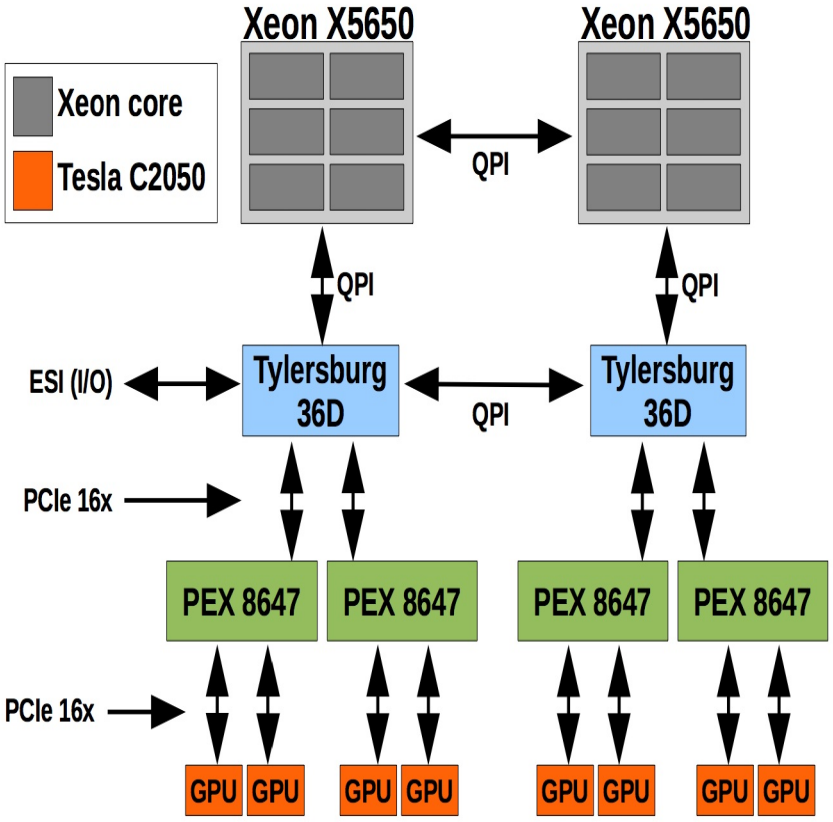
- Intel Xeon X5650 system contains six CPUs (Xeon 5650)
- QPI-PCIe bridge;
- PCI-e switch for GPUs.



Source: www

System with 2 Intel Xeon X5650 and 8 Nvidia GPU Teslas

- Intel Xeon X5650 system
- Two six-core CPUs (Xeon 5650)
- eight GPUs
- Tylesburg-36D, QPI-PCIe bridge
- PXE8647 PCI-e switch for GPU pairs.



Source: http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview

GPU Performance Example: GEForce 8800 GTX

- Single-precision multiply-add ops
- Ops take place in a single instruction cycle.
- Peak Perf (all PEs busy all the time):

$$\begin{aligned} Perf_{peak} &= 16 SMs \times (8 cores/SM) \\ &\times (2 FLOPS/op/core) \\ &\times (1 \quad instruction/\text{clock cycle}) \\ &\times (1.35x10^9 \text{ clock cycles}/sec) \\ &= 345.6 \text{ GFLOPs / second} \end{aligned}$$

- With same clock speed, for the Kepler GK110, $Perf_{peak}$ 8 *TeraFlops*

Source: http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/lecture_notes/gpus.pdf

GPU Bandwidth Example: GEForce 8800 GTX

- Each GEForce SM has a local store of 16KB, & 8192 32-bit registers.
- Shared memory partitions:
  - 6 DDR3 DRAM (900 MHz)
  - 8-byte wide data path
  - 128 MB per DDRAM partition.
  - $Mem_{tot} = 768$ MB
- Peak bandwidth (double-data rate):

$$\begin{aligned} BW_{peak} &= 6x8\, bytes/transfer \times 2\, transfers/clockcycle \\ &\quad \times 0.9x10^9\, clockcycles/second \\ &= 6 \times 8 \times 2 \times 0.9 GB/second \\ &= 86.4 GB/second \end{aligned}$$

Source: http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/lecture_notes/gpus.pdf

GPU/CUDA Env on tuckoo

Check whether your computer has a capable GPU

- Identify the model name of your GPU.
- On Windows, use GPU-Z found here.
  Note: The listed capabilities of the card may be inaccurate on multi GPU systems.
- On linux, in a console use: lspci — grep VGA
- On Macintosh, Select About this Mac from the Apple menu, then click More Info. Under Hardware select Graphics/Displays.
- tuckoo is not a GPU node:

```
[mthomas@tuckoo ~]$  lspci | grep VGA
[mthomas@tuckoo]$ lspci | grep VGA
02:00.0 VGA compatible controller: Matrox Electronics Systems Ltd. MGA G200e [Pilot]
      ServerEngines (SEP1) (rev 02)
```

- Check a GPU node:

```
[mthomas@tuckoo]$ ssh node9 "/sbin/lspci | grep VGA"
01:00.0 VGA compatible controller: NVIDIA Corp.. NV44 [GeForce 6200 LE] (rev a1)
02:00.0 VGA compatible controller: NVIDIA Corp.. GF100 [GeForce GTX 480] (rev a3)
03:00.0 VGA compatible controller: NVIDIA Corp.. GF100 [GeForce GTX 480] (rev a3)
```

Check for tuckoo GPU nodes:

```
[mthomas@tuckoo cudatests]$ cat /etc/motd
. . .
  GPUs
   ------------------------------------------
   node9  has 2     GTX 480 gpu cards (1.6GB dev ram ea.)
   node8  has 2     C2075   gpu cards (  6GB dev ram ea.)
   node7  has 2     C1060   gpu cards (  4GB dev ram ea.)
   node11 has 1     K40     gpu card  (                 )

   see files /examples/cuda/nodeX.SPECS (X=7,8,9,11)*
         and /examples/cuda/GPU.SPECS

. . .
```

Table: Table of tuckoo CPU/GPU configurations (from /etc/motd & /examples/cuda)

node ID	**7**	**8**	**9**	**11**
GPU Type	**2 Tesla C1060**	**2 Tesla C2075**	**2 GTX 480**	**2 K40**
CPU Type	2 Xeon X5650	2 Xeon X5650	2 Xeon E5620	1 Xeon E5-2650
#CPU cores	6*2=12	4*2=8	3*2=6	2*8=16
#Multiprocessor (MP)	30/12	14/14	15/15	15
#SP/Cores	240	448/448	480/480	480
Max Thd/Block	512	1024	1024	1024
Max Grd Dim	$64k \times 64k \times 1$	$64k \times 64k \times 64$	$64k \times 64k \times 64$	$2G \times 64k \times 64k$

Warning: this is not necessarily correct. Clusters change frequently!!!

Locating CPU Devices on tuckoo.sdsu.edu

```
[mthomas@tuckoo:~/ParCompMatl/cuda] ./checknodes.sh
rsh node1 "/sbin/lspci | grep VGA"
0d:00.0 VGA compatible controller: Matrox Graphics, Inc. MGA G200e [Pilot] ServerEngines (SEP1) (rev 02)
----------------------------------
rsh node2 "/sbin/lspci | grep VGA"
0d:00.0 VGA compatible controller: Matrox Graphics, Inc. MGA G200e [Pilot] ServerEngines (SEP1) (rev 02)
----------------------------------
rsh node3 "/sbin/lspci | grep VGA"
node3: No route to host
----------------------------------
rsh node4 "/sbin/lspci | grep VGA"
0d:00.0 VGA compatible controller: Matrox Graphics, Inc. MGA G200e [Pilot] ServerEngines (SEP1) (rev 02)
----------------------------------
rsh node5 "/sbin/lspci | grep VGA"
01:03.0 VGA compatible controller: ATI Technologies Inc ES1000 (rev 02)
----------------------------------
rsh node6 "/sbin/lspci | grep VGA"
07:04.0 VGA compatible controller: Matrox Electronics Systems Ltd. MGA G200eW WPCM450 (rev 0a)
----------------------------------
rsh node7 "/sbin/lspci | grep VGA"
85:00.0 VGA compatible controller: nVidia Corporation GT215 [GeForce GT 240] (rev a2)
----------------------------------
rsh node8 "/sbin/lspci | grep VGA"
06:00.0 VGA compatible controller: nVidia Corporation Device 1096 (rev a1)
08:00.0 VGA compatible controller: nVidia Corporation Device 1096 (rev a1)
81:00.0 VGA compatible controller: nVidia Corporation G98 [GeForce 8400 GS] (rev a1)
----------------------------------
rsh node9 "/sbin/lspci | grep VGA"
01:00.0 VGA compatible controller: NVIDIA Corporation NV44 [GeForce 6200 LE] (rev a1)
02:00.0 VGA compatible controller: NVIDIA Corporation GF100 [GeForce GTX 480] (rev a3)
03:00.0 VGA compatible controller: NVIDIA Corporation GF100 [GeForce GTX 480] (rev a3)
----------------------------------
rsh node10 "/sbin/lspci | grep VGA"
node10: Unknown host
----------------------------------
rsh node11 "/sbin/lspci | grep VGA"
05:03.0 VGA compatible controller: Matrox Electronics Systems Ltd. MGA G200eW WPCM450 (rev 0a)
```

Check tuckoo compiler and ENV

```
[mthomas@tuckoo ~]$  lspci | grep VGA
02:00.0 VGA compatible controller:
Matrox Graphics, Inc. MGA G200e [Pilot] ServerEngines (SEP1) (rev 02)

[mthomas@tuckoo ~]$ which nvcc
/usr/local/cuda/bin/nvcc

[mthomas@tuckoo:~/jf] nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2013 NVIDIA Corporation
Built on Thu_Mar_13_11:58:58_PDT_2014
Cuda compilation tools, release 6.0, V6.0.1

[mthomas@tuckoo ~]$ cat /proc/version
Linux version 2.6.32-642.6.2.el6.x86_64
(mockbuild@worker1.bsys.centos.org)
(gcc version 4.4.7 20120313
(Red Hat 4.4.7-17) (GCC) ) #1 SMP
Wed Oct 26 06:52:09 UTC 2016
```

Check tuckoo ENV on one of the nodes

```
[mthomas@tuckoo cudatests]$ cat env.bat
#!/bin/sh
# this example batch script requests many processors...
# for more info on requesting specific nodes see
# "man pbs_resources"
#PBS -V
#PBS -l nodes=1:csrc-gpu
#PBS -N env
#PBS -j oe
#PBS -r n
#PBS -q batch
cd $PBS_O_WORKDIR

echo ------------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo ------------------------------------------------------
echo ------------------------------------------------------

mpiexec -np 1 -hostfile $PBS_NODEFILE /usr/local/cuda/bin/nvcc -V
echo ------------------------------------------------------
echo ------------------------------------------------------
mpiexec -np 1 -hostfile $PBS_NODEFILE /bin/cat  /proc/version
```

Check tuckoo ENV on one of the nodes

```
[mthomas@tuckoo cudatests]$ cat env.o33799
-----------------------------------------------------
Job is running on node node10
-----------------------------------------------------
PBS: qsub is running on tuckoo.sdsu.edu
PBS: originating queue is batch
PBS: executing queue is batch
PBS: working directory is /home/mthomas/pardev/cuda/cudatests
PBS: execution mode is PBS_BATCH
PBS: job identifier is 33799.tuckoo.sdsu.edu
PBS: job name is env
PBS: node file is /var/spool/torque/aux//33799.tuckoo.sdsu.edu
PBS: current home directory is /home/mthomas
PBS: PATH = /opt/pgi/linux86-64/2011/mpi/mpich/include/:/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:
/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/lib64/openmpi/bin:/usr/local/torque/bin:
/usr/local/torque/sbin:/usr/local/cuda/bin:/usr/local/tau/x86_64/bin:
/usr/local/vampirtrace/bin:/opt/pgi/linux86-64/11.0/bin:/home/mthomas/bin
-----------------------------------------------------
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2012 NVIDIA Corporation
Built on Thu_Apr__5_00:24:31_PDT_2012
Cuda compilation tools, release 4.2, V0.2.1221
-----------------------------------------------------
Linux version 2.6.32-220.17.1.el6.x86_64 (mockbuild@c6b5.bsys.dev.centos.org) (gcc version 4.4.6 20110731 (Red Hat 4.4.6-3) (
```

The CUDA Compiler: nvcc

- you can install CUDA toolkit, compile code without a GPU device.
- To compile use: *nvcc*
- **NOTE: CUDA does not support doubles on the *device* by default:** You need to add the switch "-arch sm_13" (or a higher compute capability) to your nvcc command:

```
[mthomas/dblTst]
[mthomas/dblTst]nvcc -o dblTst dblTst.cu
nvcc warning : The 'compute_10' and 'sm_10' architectures are
deprecated, and may be removed in a future release.
ptxas /tmp/tmpxft_00006578_00000000-5_dblTst.ptx, line 76;
warning : Double is not supported. Demoting to float
[mthomas/dblTst]
```

```
[mthomas/dblTst]
[mthomas/dblTst] nvcc -arch=sm_13 -o dblTst dblTst.cu
[mthomas/dblTst]
```

Compile & run CUDA code on the login node

- you can install CUDA toolkit, compile code without a GPU device.
- To compile use: *nvcc*
- You can RUN the code from the command line on some machines.

```
[mthomas@tuckoo hello]$ cat simple_hello.cu
/*
 * simple_hello.cu
 * Copyright 1993-2010 NVIDIA Corporation.
 *     All rights reserved.
 *
  */
#include <stdio.h>
#include <stdlib.h>

int main( void ) {
   int deviceCount;

   cudaGetDeviceCount( &deviceCount );
   printf("Hello, World! You have %d devices\n",
        deviceCount );
    return 0;
}
```

```
[mthomas@tuckoo chapter03]$ nvcc -o simple_hello simple_hello.cu
nvcc warning : The 'compute_10' and 'sm_10' architectures
      are deprecated, and may be removed in a future release.
```

```
[mthomas@tuckoo chapter03]$  ./hello_world
Hello, World! You have 0 devices
```

Compiling & running CUDA code using batch script

```
[mthomas@tuckoo]$ cat simple_hello.cu
/*simple_hello.cu
 * Copyright 1993-2010 NVIDIA Corporation.
  */
#include <stdio.h>
#include <stdlib.h>

int main( void ) {
   int deviceCount;

   cudaGetDeviceCount( &deviceCount );
   printf("Hello, World! You have %d devices\n",
        deviceCount );
    return 0;
}
```

```
[mthomas@tuckoo chapter03]$ nvcc -o simple_hello simple_hello.cu
nvcc warning : The 'compute_10' and 'sm_10' architectures
      are deprecated, and may be removed in a future release.
```

```
[mthomas@tuckoo] cat hello_world.bat
#!/bin/sh
#PBS -V
#PBS -l nodes=node8:ppn=1
#PBS -N hello_world
#PBS -j oe
#PBS -r n
#PBS -q batch
cd $PBS_O_WORKDIR
./hello_world
```

```
[mthomas@tuckoo] qsub hello_world.bat
13882.tuckoo.sdsu.edu

[mthomas@tuckoo] cat hello_world.o13882
Hello, World! You have 3 devices
```

CUDA Example: simple_kernel.cu

```
[mthomas@tuckoo hello]$ cat simple_kernel.cu
/*
 * Copyright 1993-2010 NVIDIA Corporation.
 *     All rights reserved.
 *
  */
#include <stdio.h>

__global__ void kernel( void ) {
}

int main( void ) {
    int deviceCount;

    kernel<<<1,1>>>();

    cudaGetDeviceCount( &deviceCount );
    printf("Hello, World! You have %d devices\n",
           deviceCount );

    return 0;
}
```

```
[mthomas@tuckoo hello]$ nvcc -o simple_kernel simple_kernel.cu
nvcc warning : The 'compute_10' and 'sm_10'.....
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$  ./simple_kernel
Hello, World! You have 0 devices
```

CUDA Batch Script Example

```
[mthomas@tuckoo chapter03]$ cat hello_world.bat
#!/bin/sh
# this example batch script requests many processors...
# for more info on requesting specific nodes see
# "man pbs_resources"
#PBS -V
#PBS -l nodes=node8:ppn=1
#PBS -N simple_hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR

echo ------------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo ------------------------------------------------------
NCORES=‘wc -w < $PBS_NODEFILE‘
echo "hello_world_kernel using $NCORES cores..."
./simple_hello
```

Run Job using Batch Queue: hello_world.bat - OUTPUT

```
[mthomas@tuckoo chapter03]$ cat hello_world.o18574
-----------------------------------------------------
Job is running on node node9
-----------------------------------------------------
PBS: qsub is running on tuckoo.sdsu.edu
PBS: originating queue is batch
PBS: executing queue is batch
PBS: working directory is /home/mthomas/pardev/cuda/cuda_by_example/chapter03
PBS: execution mode is PBS_BATCH
PBS: job identifier is 18574.tuckoo.sdsu.edu
PBS: job name is hello_world
PBS: node file is /var/spool/torque/aux//18574.tuckoo.sdsu.edu
PBS: current home directory is /home/mthomas
PBS: PATH = /usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/openmpi/bin:/usr/lo
-----------------------------------------------------
hello_world using 1 cores...
Hello, World! You have 3 devices
```

CUDA: simple_add.cu

```
#include <stdio.h>

__global__ void my_first_kernel(float * x) {
    int tid = threadIdx.x + blockDim.x * blockIdx.x;
    x[tid] = (float) threadIdx.x;
}


int main(int argc, char ** argv) {
    float * h_x, * d_x; // h=host, d=device
    int nblocks=2, nthreads=8, nsize=2 * 8;

    h_x = (float *)malloc(nsize * sizeof(float));

    cudaMalloc((void **)&d_x,nsize * sizeof(float));

    my_first_kernel<<<nblocks,nthreads>>>(d_x);

    cudaMemcpy(h_x,d_x,nsize * sizeof(float),
              cudaMemcpyDeviceToHost);

    for (int n=0; n<nsize; n++)
        printf(" n, x = %d %f \n",n,h_x[n]);

    cudaFree(d_x); free(h_x);
}
```

```
[mthomas@tuckoo] cat simple_add.bat
#!/bin/sh
#PBS -V
#PBS -l nodes=node9:ppn=1
#PBS -N simple_add
#PBS -j oe
#PBS -r n
#PBS -q batch
cd $PBS_O_WORKDIR

./simple_add
```

```
[mthomas@tuckoo]simple_add.o13886
 n, x = 0 0.000000
 n, x = 1 0.000000
 n, x = 2 0.000000
 n, x = 3 0.000000
 n, x = 4 0.000000
 n, x = 5 0.000000
 n, x = 6 0.000000
 n, x = 7 0.000000
 n, x = 8 0.000000
 n, x = 9 0.000000
 n, x = 10 0.000000
 n, x = 11 0.000000
 n, x = 12 0.000000
 n, x = 13 0.000000
 n, x = 14 0.000000
 n, x = 15 0.000000
```

simple_kernel_params.cu

```
#include "../common/book.h"

__global__ void add( int a, int b, int *c ) {
    *c = a + b;
}

int main( void ) {
    int c;
    int *dev_c;
    HANDLE_ERROR( cudaMalloc( (void**)&dev_c, sizeof(int) ) );

    add<<<1,1>>>( 2, 7, dev_c );

    HANDLE_ERROR( cudaMemcpy( &c, dev_c, sizeof(int),
              cudaMemcpyDeviceToHost ) );
    printf( "2 + 7 = %d\n", c );
    HANDLE_ERROR( cudaFree( dev_c ) );

    return 0;
}
```

simple_kernel_params.cu

```
[mthomas@tuckoo chapter03]$ cat simple_kernel_params.o3901
-----------------------------------------------------
Job is running on node csrc-gpu
-----------------------------------------------------
PBS: qsub is running on tuckoo.sdsu.edu
PBS: originating queue is batch
PBS: executing queue is batch
PBS: working directory is /home/mthomas/pardev/cuda/cuda_by_example/chapter03
PBS: execution mode is PBS_BATCH
PBS: job identifier is 3901.tuckoo.sdsu.edu
PBS: job name is simple_kernel_params
PBS: node file is /var/spool/torque/aux//3901.tuckoo.sdsu.edu
PBS: current home directory is /home/mthomas
PBS: PATH = /usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/
-----------------------------------------------------
simple_kernel_params using 1 cores...
2 + 7 = 9
2 + 7 = 9
...
2 + 7 = 9
2 + 7 = 9
2 + 7 = 9
```

obtaining device information: enum_gpu.cu (1)

```
#include "../common/book.h"
int main( void ) {
    cudaDeviceProp  prop;
    int count;
    HANDLE_ERROR( cudaGetDeviceCount( &count ) );
    for (int i=0; i< count; i++) {
        HANDLE_ERROR( cudaGetDeviceProperties( &prop, i ) );
        printf( "   --- General Information for device %d ---\n", i );
        printf( "Name:  %s\n", prop.name );
        printf( "Compute capability:  %d.%d\n", prop.major, prop.minor );
        printf( "Clock rate:  %d\n", prop.clockRate );
        printf( "Device copy overlap:  " );
        if (prop.deviceOverlap)
            printf( "Enabled\n" );
        else
            printf( "Disabled\n");
        printf( "Kernel execution timeout :  " );
        if (prop.kernelExecTimeoutEnabled)
            printf( "Enabled\n" );
        else
            printf( "Disabled\n" );
```

obtaining device information: enum_gpu.cu (2)

```
        printf( "   --- Memory Information for device %d ---\n", i );
        printf( "Total global mem:  %ld\n", prop.totalGlobalMem );
        printf( "Total constant Mem:  %ld\n", prop.totalConstMem );
        printf( "Max mem pitch:  %ld\n", prop.memPitch );
        printf( "Texture Alignment:  %ld\n", prop.textureAlignment );
        printf( "   --- MP Information for device %d ---\n", i );
        printf( "Multiprocessor count:  %d\n",
                    prop.multiProcessorCount );
        printf( "Shared mem per mp:  %ld\n", prop.sharedMemPerBlock );
        printf( "Registers per mp:  %d\n", prop.regsPerBlock );
        printf( "Threads in warp:  %d\n", prop.warpSize );

        printf( "Max threads per block:  %d\n",
                    prop.maxThreadsPerBlock );
        printf( "Max thread dimensions:  (%d, %d, %d)\n",
                    prop.maxThreadsDim[0], prop.maxThreadsDim[1],
                    prop.maxThreadsDim[2] );
        printf( "Max grid dimensions:  (%d, %d, %d)\n",
                    prop.maxGridSize[0], prop.maxGridSize[1],
                    prop.maxGridSize[2] );
        printf( "\n" );
        }
}
```

```
------------------------------------------------
--- General Information for device 0 ---
Name:  Tesla C1060
Compute capability:  1.3
Clock rate:  1296000
Device copy overlap:  Enabled
Kernel execution timeout :  Disabled
--- Memory Information for device 0 ---
Total global mem:  4294770688
Total constant Mem:  65536
Max mem pitch:  2147483647
Texture Alignment:  256
--- MP Information for device 0 ---
Multiprocessor count:  30
Shared mem per mp:  16384
Registers per mp:  16384
Threads in warp:  32
Max threads per block:  512
Max thread dimensions:  (512, 512, 64)
Max grid dimensions:  (65535, 65535, 1)
------------------------------------------------
--- General Information for device 2 ---
Name:  GeForce GT 240
Compute capability:  1.2
Clock rate:  1340000
Device copy overlap:  Enabled
Kernel execution timeout :  Disabled
--- Memory Information for device 2 ---
Total global mem:  1073020928
Total constant Mem:  65536
Max mem pitch:  2147483647
Texture Alignment:  256
```

```
------------------------------------------------
--- General Information for device 1 ---
Name:  Tesla C1060
Compute capability:  1.3
Clock rate:  1296000
Device copy overlap:  Enabled
Kernel execution timeout :  Disabled
   --- Memory Information for device 1 ---
Total global mem:  4294770688
Total constant Mem:  65536
Max mem pitch:  2147483647
Texture Alignment:  256
   --- MP Information for device 1 ---
Multiprocessor count:  30
Shared mem per mp:  16384
Registers per mp:  16384
Threads in warp:  32
Max threads per block:  512
Max thread dimensions:  (512, 512, 64)
Max grid dimensions:  (65535, 65535, 1)
------------------------------------------------
   --- MP Information for device 2 ---
Multiprocessor count:  12
Shared mem per mp:  16384
Registers per mp:  16384
Threads in warp:  32
Max threads per block:  512
Max thread dimensions:  (512, 512, 64)
Max grid dimensions:  (65535, 65535, 1)
```

GPU/CUDA Env on local host: OS X

Download CUDA SDK for OS X (Fall'12)

- CUDA SDK: https://developer.nvidia.com/gpu-computing-sdk
- Instructions:
  `http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-mac-os-x/index.html`
- Check that system has CUDA GPU system
- MacBookPro link:
  `http://www.geforce.com/hardware/notebook-gpus/geforce-gt-650m`
- Install on OS X:

```
Set ENV variables (csh):
# for CUDA
set path = (/Developer/NVIDIA/CUDA-5.0/bin $path)
set DYLD_LIBRARY_PATH = /Developer/NVIDIA/CUDA-5.0/lib
```

Check for location of the compiler:

```
[gidget:~] mthomas% which nvcc
/Developer/NVIDIA/CUDA-5.0/bin/nvcc

[gidget:~] mthomas% nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2012 NVIDIA Corporation
Built on Fri_Sep_28_16:10:16_PDT_2012
Cuda compilation tools, release 5.0, V0.2.1221

See if the CUDA Driver is installed correctly on OS X:

[gidget:~] mthomas% kextstat | grep -i cuda
  123    0 0xffffff7f825b9000 0x2000
  0x2000     com.nvidia.CUDA (1.1.0) <4 1>
```