GPU Computing with Dirac

Hemant Shukla
Architectural Differences

CPU

Control Logic

ALU

Cache

DRAM

Less than 20 cores
1-2 threads per core
Latency is hidden by large cache

GPU

DRAM

512 cores
10s to 100s of threads per core
Latency is hidden by fast context switching
CUDA  (Compute Unified Device Architecture)

OpenCL

Microsoft's DirectCompute

Third party wrappers are also available for Python, Perl, Fortran, Java, Ruby, Lua, MATLAB, IDL, and Mathematica

Compilers from PGI, RCC, HMPP, Copperhead, OpenACC
**Hardware**

44 Nodes
- 1 Tesla C 2050 (Fermi) GPU
- 2 Nehalem Quad Core
- 24 GB of DDR3

4 Nodes
- 1 Tesla C 1060 (Fermi) GPU
- 2 Nehalem Quad Core
- 24 GB of DDR3

1 Node
- 4 Tesla C 1060 GPU
- 2 Nehalem Quad Core
- 24 GB DDR3

1 Node
- 4 Tesla C 2050 GPU
- 2 Nehalem Quad Core
- 24 GB DDR3

**Interconnect**
- 4X QDR InfiniBand (32 Gb/s)
Request an account of Dirac

http://www.nersc.gov/users/computational-systems/dirac/getting-started/request-for-dirac-account/

For questions contact NERSC Consultants at consult@nersc.gov
DIRAC Access

CARVER IBM iDataPlex

Dirac nodes reside on Carver

SINGLE NODE ACCESS

> ssh -l <username>@carver.nersc.gov
> qsub -I -V -q dirac_int -l nodes=1:ppn=8:fermi
qsub: waiting for job 1822582 cvrsvc09 ib to start
qsub: job 1822582 cvrsvc09 ib ready

[dirac28] >

1. Log into Carver
2. Start interactive node on Dirac
3. Dirac node prompt
```c
#include <stdio.h>
int main()
{
    int noOfDevices;
    /* get no. of device */
    cudaGetDeviceCount (&noOfDevices);
    cudaDeviceProp prop;
    for (int i = 0; i < noOfDevices; i++)
    {
        /* get device properties */
        cudaGetDeviceProperties (&prop, i);

        printf ("Device Name:\t %s\n", prop.name);
        printf ("Total global memory:\t %ld\n", prop.totalGlobalMem);
        printf ("No. of SMs:\t %d\n", prop.multiProcessorCount);
        printf ("Shared memory / SM:\t %ld\n", prop.sharedMemPerBlock);
        printf ("Registers / SM:\t %d\n", prop.regpsPerBlock);
    }
    return 1;
}
```

First CUDA Program

Compilation (on Carver)

> module load cuda
OR
> module load cuda/<version>

> nvcc whatDevice.cu -o whatDevice

Log into Dirac and run the code

> qsub -I -V -q dirac_int -l nodes=1:ppn=8:fermi

[dirac27] > ./whatDevice

Output

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Tesla C2050</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total global memory</td>
<td>2817982464</td>
</tr>
<tr>
<td>No. of SMs:</td>
<td>14</td>
</tr>
<tr>
<td>Shared memory / SM:</td>
<td>49152</td>
</tr>
<tr>
<td>Registers / SM:</td>
<td>32768</td>
</tr>
</tbody>
</table>
CUDA + OpenMP + MPI

```
> gcc -fopenmp someCode.cu -o someCode         [include omp.h]
> nvcc -Xcompiler -fopenmp -lgomp someCode.cu -o someCode

> mpicc -c main.c -o main.o
> nvcc -c kernel.cu -o kernel.o

> mpicc main.o kernel.o -L/usr/local/cuda/lib64 -lcudart
```
>> N = 5000;
>> a = rand (N,N);
>> b = gpuArray(a);

>> tic; r1 = fft2(a); toc;
Elapsed time is 1.135059 seconds.

>> tic; gather (r1); toc;
Elapsed time is 0.015606 seconds.

>> tic; r2 = fft2(b); toc;
Elapsed time is 0.423533 seconds.

>> tic; gather (r2); toc;
Elapsed time is 0.253305 seconds.
Running Batch Jobs

```bash
#PBS -q dirac_reg
#PBS -l nodes=16:ppn=8:fermi
#PBS -l walltime=00:10:00
#PBS -A gpgpu
#PBS -N my_job
#PBS -e my_job.$PBS_JOBID.err
#PBS -o my_job.$PBS_JOBID.out
#PBS -V

cd $PBS_O_WORKDIR
mpirun -np 128 ./my_executable

> qsub submit.job
```
Programming Models on Dirac

CUDA
OpenCL
Matlab
HMPP
MINT
RCC
PGI (CUDA-FORTRAN)
OpenACC
Adaptive Mesh Refinement

Data stored in Octree data structure

Refinement with $2^l$ spatial resolution per level

$8^3$ cells per patch

Identical spatial geometry (same kernel)

Uniform and individual time-steps

Figure - Hsi-Yu Schive et al., 2010
Results

Large scale Cosmological Simulations with GAMER

Hemant Shukla, Hsi-Yu Schive et al. SC 2011
For More Information

http://www.nersc.gov/users/computational-systems/dirac/