Discussion on

NVIDIA's Compute Unified Device Architecture
( CUDA )

Stan Tomov

03/25/2015

specifications and graphs taken from
CUDA Programming Guide Version 1.0
(references on: http://www.nvidia.com/object/cuda_get.html)
CUDA

CUDA Libraries

CUDA Runtime

CUDA Driver

Application

CPU

GPU

CUBLAS, CUFFT, ...

we can easily use LAPACK with CUBLAS

C like API
Programming model

A highly multithreaded coprocessor

* thread block
  ( a batch of threads with fast shared memory executes a kernel )

* Grid of thread blocks
  ( blocks of the same dimension, grouped together to execute the same kernel; reduces thread cooperation )

__global__ void MatVec( . . . ) {
  // Block index
  int bx = blockIdx.x;
  int by = blockIdx.y;

  // Thread index
  int tx = threadIdx.x;
  int ty = threadIdx.y;

  . . .
}

// set the grid and thread configuration
Dim3 dimBlock(3,5);
Dim3 dimGrid(2,3);

// Launch the device computation
MatVec<<<dimGrid, dimBlock>>>( . . . );
GPUs & Challenges

- Programming is 'easier' with NVIDIA's Compute Unified Device Architecture

Quadro FX 5600

Installed (at ICL) on a 4 x Dual Core AMD Opteron Processor 265 (1800 MHz, 1024 KB cache)

Some numbers:

- processors: 128 (total)          max performance: 346 GFlop/s
- registers: 8192 / block          memory bandwidth: 76.8 GB/s
- warp size: 32                      bandwidth to CPU: 8 GB/s
- max threads / block: 512
- among 8 processors on a multiproc.
GPUs & Challenges

- Programming is 'easier' with NVIDIA's Compute Unified Device Architecture

- 1. Get data into shared memory
- 2. Compute

For DLA the CI is about 32 (on IBM Cell about 64)
* not enough to get close to peak (346 GFlop/s)
* CUBLAS sgemm is about 120 Gflop/s
GPUs & Challenges

- Programming is 'easier' with NVIDIA's Compute Unified Device Architecture

http://mc.stanford.edu/cgi-bin/images/6/65/SC08_Volkov_GPU.pdf

Discussion

- Dense Linear Algebra
  - Matrix-matrix product
  - LAPACK with CUDA
- Sparse Linear Algebra
  - Sparse matrix-vector product
- Projects using CUDA