Heterogeneous programming with OpenACC

Alexei Strelchenko

LinkSCEEM User Meeting

Nicosia, 27 June, 2012

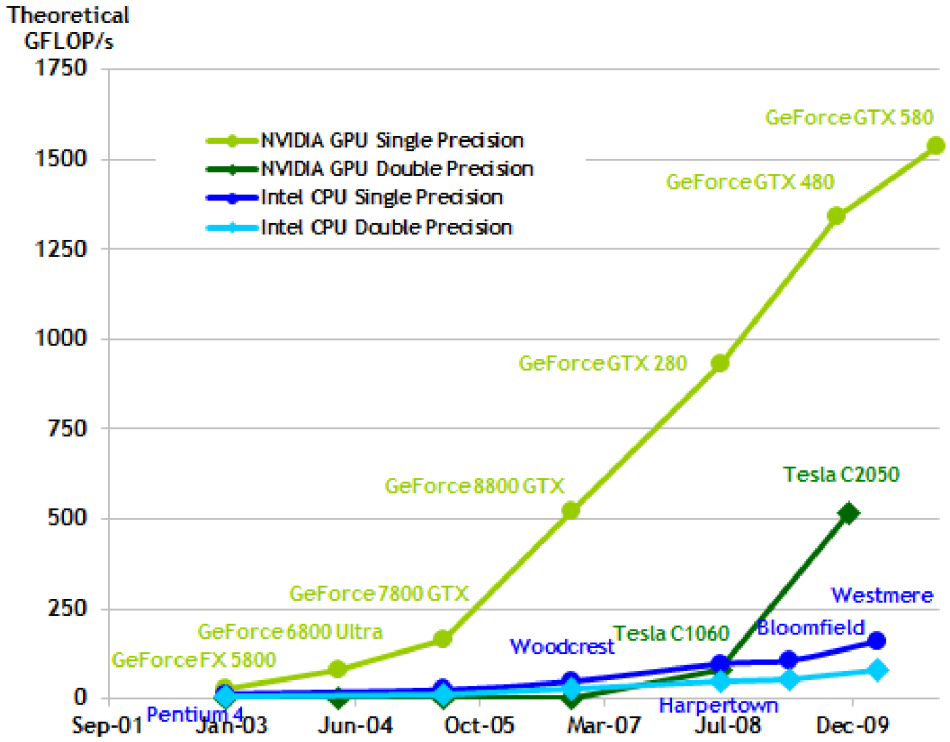
The Cyprus Institute

Plan:

- **Motivation**
  - Why are we interesting in heterogeneous systems?
  - GPU programming approaches and available tools
- **GPU programming model**
- **Overview of GPU libraries**
- **Programming with OpenACC directives**

Why GPUs?

- **CPUs vs GPUs (theoretical peak performance):**



Parallel computing challenges

- **General idea:**
  - simultaneous independent computations on several processing units with followed synchronization

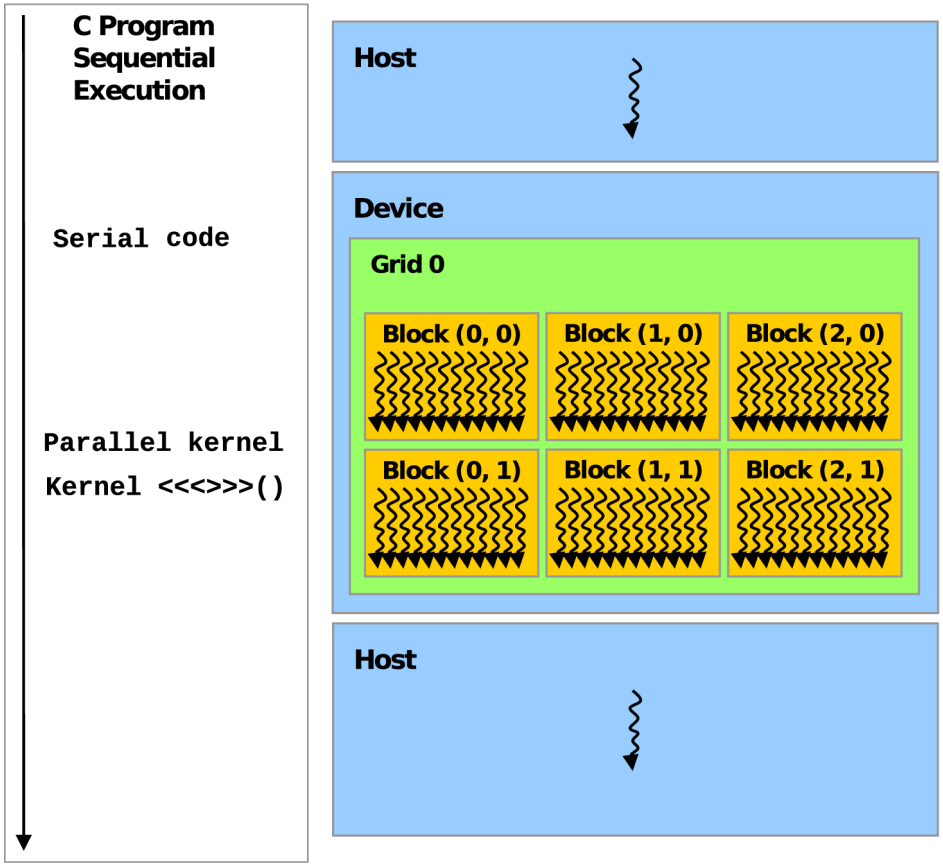
- **General problems:**
  - Amdahl's low: acceleration $N / (N * (1 - P) + P)$
    - P – portion of a code that can be made parallel
    - N – number of processing units
  - parallel code development more complicated

Parallel computing challenges

- **Latencies:**
  - HW latencies : DRAM (CAS) latency, cache latencies etc.
  - SW latencies : e.g., API functions calls etc.
- **Bandwidth:**
  - data caching may remedy too low BW
- **Arithmetical intensity (**# of arithm. opers /# of mem. opers**):**
  - relies on algorithm complexity: > O(n) is better

Hybrid programming model

- A typical hybrid application (CUDA API):

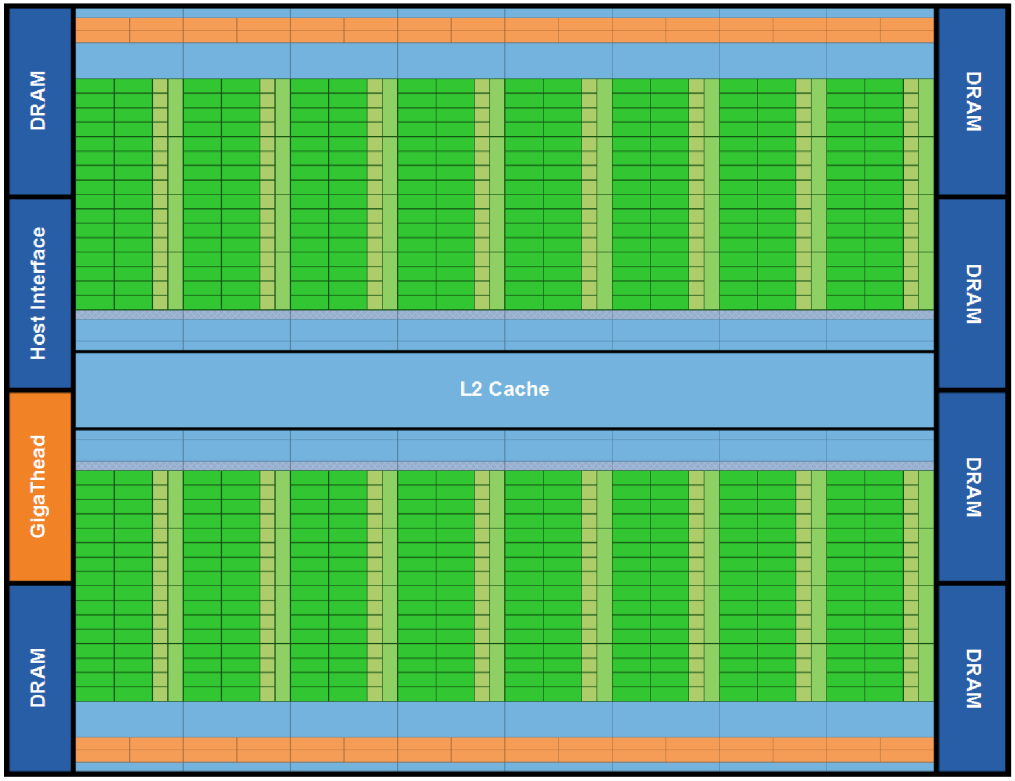


GPU computing approaches

- **Programming :**
  - High Level : CUDA runtime API ('CUDA C'), OpenCL
  - Intermediate level : CUDA driver API ('PTX')
- **Libraries and SW packages:**
  - CUBLAS, CUSPARSE, CUFFT, MATLAB etc.
- **Compiler directives:**
  - PGI Accelerator Directives, OpenACC

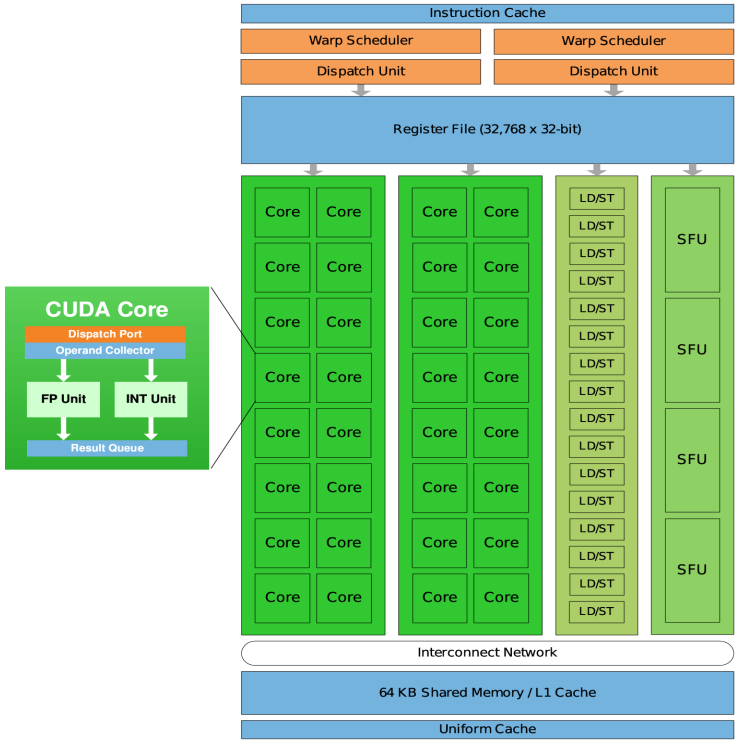
GPU anatomy

- **NVIDIA Fermi GF100 die scheme:**



GPU anatomy

- **NVIDIA Fermi GF100 SM scheme:**



GPU anatomy

- **NVIDIA Fermi GF100 highlights:**

– CUDA cores:	512 (grouped into 16 SM)
– DP capability:	256 FMA operations per clock
– SP capability:	512 FMA operations per clock
– Special function units:	4
– Shared memory (per SM):	configurable 16KB or 48KB
– L1 cache (per SM):	configurable 16KB or 48KB
– L2 cache:	768KB
– Concurrent kernels:	up to 16
– Load/Store address width:	64 bit
– ECC memory support	

CUDA API highlights

- **CUDA runtime API**
  - implemented in cudart dynamic library
  - minimal set of extensions to the C language
  - functions for memory management, system management etc.
  - build on top of a lower-level CUDA driver API

- **CUDA driver API**
  - additional of control over system
  - explicit initialization, context, module (analog. dll) management
  - semi-assembly language, *PTX*

CUDA execution model

- **Kernels are executed across the global *grid of threads*:**
  - thread grid defines the range of the computation
  - each CUDA thread executes the same code
- **Threads are grouped into local *thread blocks*:**
  - each thread in the block has unique ID: threadIdx.{x,y,z}
  - each block in the grid has unique ID: blockIdx.{x,y}
  - threads in a block can communicate
  - threads within a block can be synchronized

CUDA kernels

- **Standard host C-function (compute y[i]=a*x[i]+y[i]):**

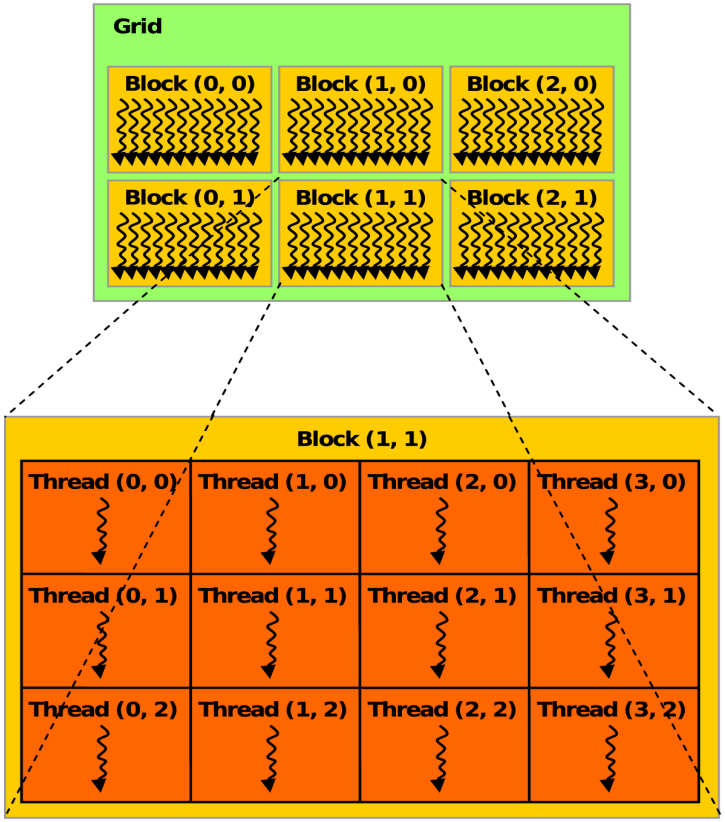
```
void axpy(const int a,  const int *x, int *y, const int N){
    int idx;
    for (idx = 0; idx < N; idx++)
        y[idx] += a * x[idx];
}
```

- **Equivalent CUDA device code:**

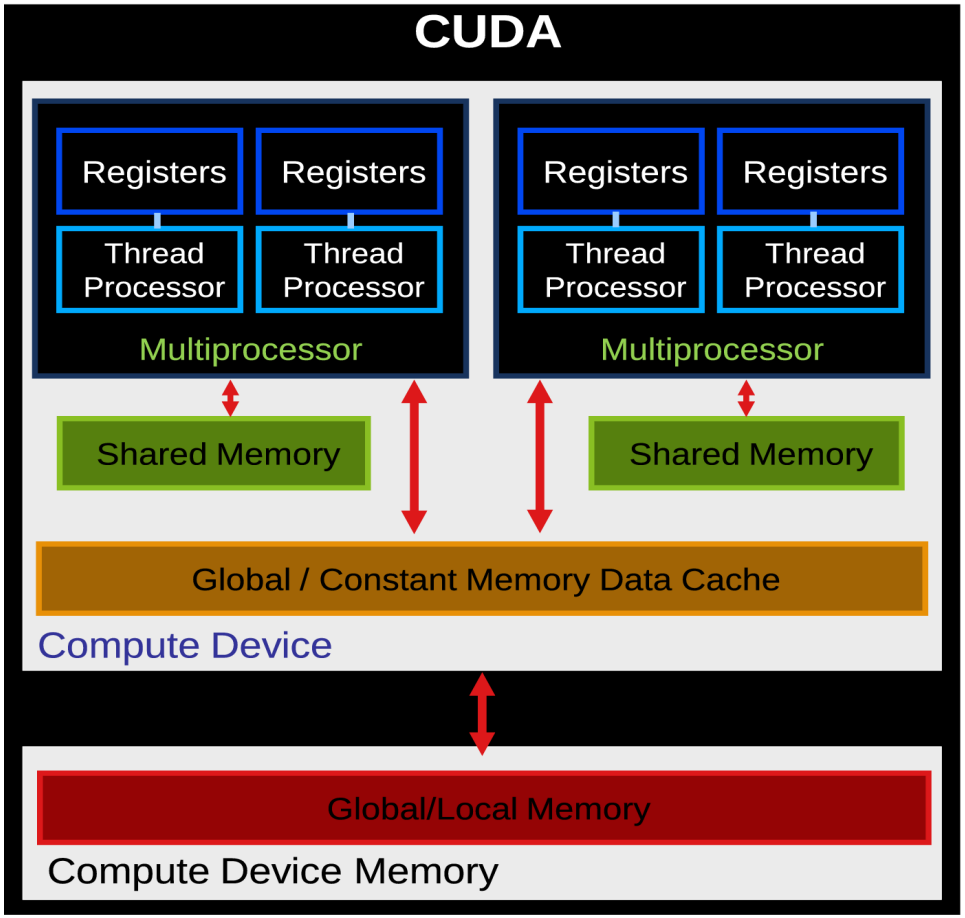
```
__global__ void *axpy(const int a,  const int *x, int *y, const int N){
    int idx = threadIdx.x + blockIdx.x * blockDim.x;
    y[idx]  += a * x[idx];
}
```

CUDA thread hierarchy

- Each thread uses ID to decide what data to work on :



CUDA memory architecture



OpenCL API highlights

- **Platform layer API**
  - Provides management for computational resources
  - Query, select and initialize computational devices
  - Create context(s) and command queue(s)
- **Runtime API**
  - Submit device code for execution
  - Define execution domain for compute kernels
  - Manage scheduling, compute and memory resources

Execution model

- **Kernels are executed across the global domain of *work-items*:**
  - global domain defines the range of the computation
  - each work-item executes the same code
  - each work-item has unique (global and local) ID

- **Work-items are grouped into local *work-groups*:**
  - local dimensions defines work-group size
  - each work-group has unique ID
  - work-items in a work-group can communicate
  - work-items within a work-group can be synchronized

Execution model: device code example

- **Standard host C-function (compute y[i]=a*x[i]+y[i]):**

```
void axpy(const int a,  const int *x, int *y, const int N){
   int i;
   for (i = 0; i < N; i++)
        y[i] += a*x[i];
}
```

- **Equivalent OpenCL device code:**

```
const char *axpy_src =
"kernel void axpy(const int a, const global int *x,  global int *y, const int N){      \n"
"    int tid = get_global_id(0);                                                        \n"
"    if(tid < N)                                                                        \n"
"        y[tid] += a*x[tid];                                                            \n"
"}                                                                                      \n";
```

General purpose libraries

- **CUDA 4.x high performance math routines:**
  - cuBLAS : Complete BLAS library
  - cuSPARSE : Sparse Matrix library
  - cuRAND : RNG library
  - cuFFT : FFT library

- **3d party math libraries :**
  - MAGMA : BLAS (linear solvers, eigenval. solvers)
  - OpenNL : Sparse matrix library
  - IMSL : Fortran Numerical Lib (BLAS & statistics)
  - ViennaCL : LA routines (for OpenCL platforms)

General purpose libraries

- **Mathematica 8 (CUDA & OpenCL support):**
  - multiple-GPU support
  - access to *Mathematica's* flexible programming language
  - access to *Mathematica's* computable data, visualization etc.
  - GPU accelerated functions (BLAS, FFT, image processing)

- **MATLAB (CUDA support):**
  - data manipulation on NVIDIA GPU
  - GPU accelerated MATLAB operations
  - integration of CUDA kernels into MATLAB apps.
  - use of multiple GPUs on the desktop and a computer cluster

Links:

- www.gpgpu.org
  - papers, news, forum
- www.hgpu.org
  - lots of material on GPU computing
- www.openacc-standard.org
  - OpenACC docs, forum, news

Let's turn to OpenACC

The Cyprus Institute