JLAB at 12 GeV - INT

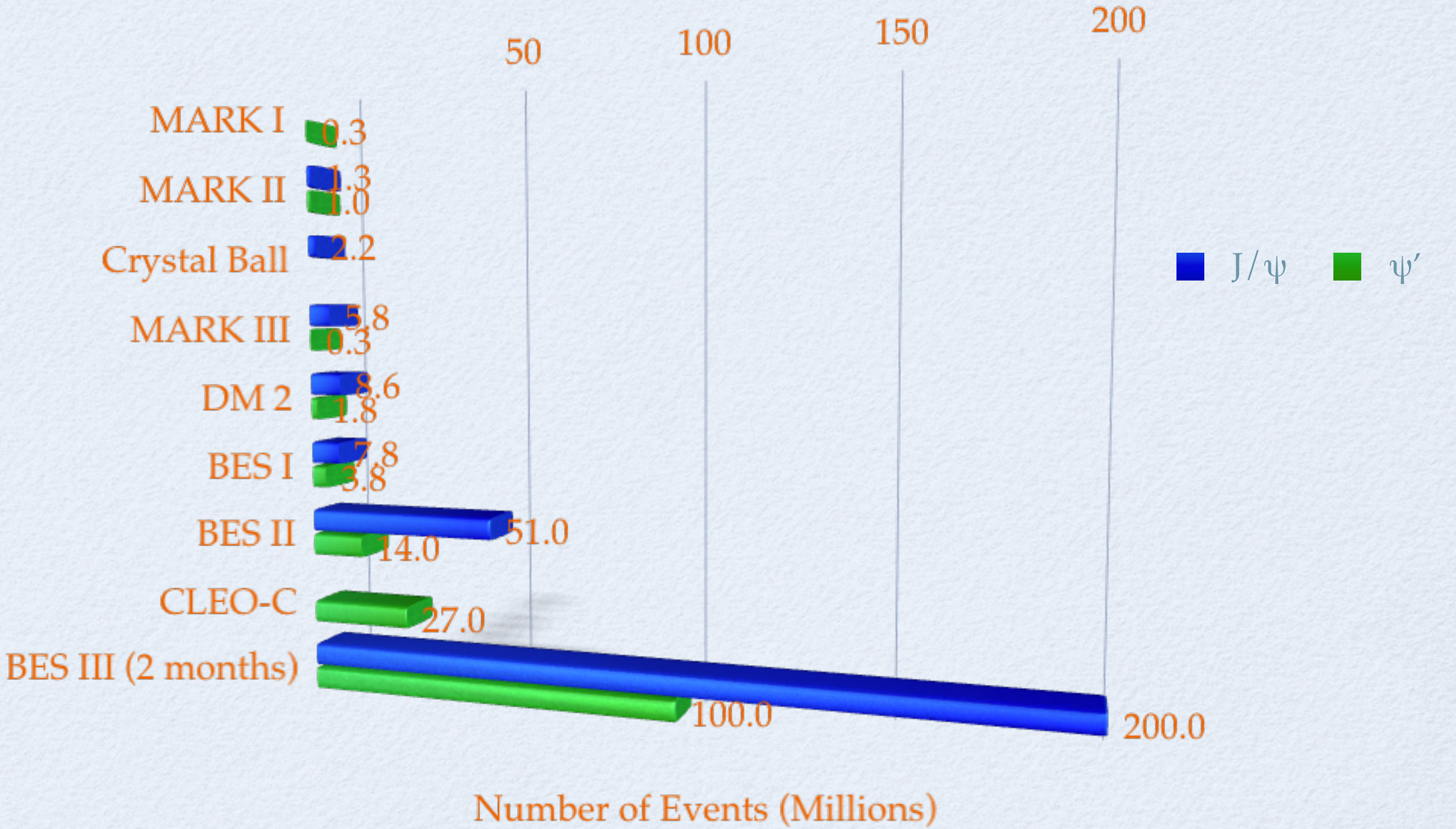
GENERAL PURPOSE GPU COMPUTING IN PARTIAL WAVE ANALYSIS

Hrayr Matevosyan - NTC, Indiana University

November 18/2009

COMPUTATIONAL CHALLENGES IN PWA

Rapid Increase in Available Data



COMPUTATIONAL CHALLENGES IN PWA

Rapid Increase in Available Data

- GLUE-X Projections:

MC Studies of $\gamma p \rightarrow \eta\pi^0 p$
(Blake Leverington)

With O(1) second/event for MC, 29 years!

Hadronic BG Rate

Signal Rate

$10^7\gamma/s$				
σ	rate	number of events		
		day^{-1}	$week^{-1}$	$year(10^7 s)^{-1}$
$124\,\mu b$	$1.55\,kHz$	1.34×10^8	9.37×10^8	1.55×10^{10}
$0.5\,\mu b$	$6.3\,Hz$	5.44×10^5	3.81×10^6	6.3×10^7
σ	rate	MC data record size (GB)		
		day^{-1}	$week^{-1}$	$year^{-1}$
$124\,\mu b$	$1.55\,kHz$	4.37×10^3	2.16×10^4	3.57×10^5
$124\,\mu b \times 4\% \times 40\%$		6.99×10^1	3.46×10^2	5.71×10^3
$0.5\,\mu b$	$6.3\,Hz$	1.25×10^1	8.76×10^1	1.45×10^3
$0.5\,\mu b \times 40\%$	$6.3\,Hz$	5.00×10^0	3.50×10^1	5.80×10^2

Table 2: Expected event rates for various production cross sections in photo-production. The σ column is the cross section. The rate is the equivalent experimental data rate given a beam rate of $10^7\gamma/s$ and the $30cm$ target. A beam rate of $10^8\gamma/s$ will increase the magnitudes of the events and data sets by one order.

Computational Challenges in PWA

Fitting and Plotting:

- Intensity for a Single Event:

Fit Parameters

$$I(\Omega) = \sum_{\alpha} \left| \sum_{\beta} V_{\alpha,\beta} A_{\alpha,\beta}(\Omega) \right|^2$$

- Unbinned Data Fit - Minimize Log Likelihood:

$$-2\ln(\mathcal{L}) = -2 \sum_{Data} \ln(I(\Omega_i)) + 2 \sum_{MC} I(\Omega_i)$$

Log-Intensity Sum Recalculated on Each Fit Iteration

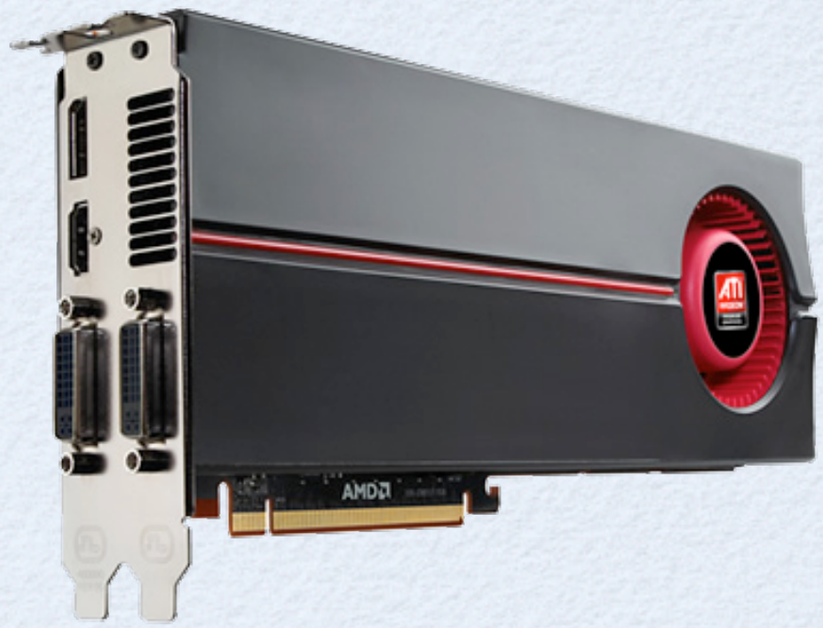
Intensity Calculated Only Once

More Sophisticated Amplitudes Will Have non-factorable Fit Parameters: Amplitudes Need Recalculation!

GPU ACCELERATED PWA

IU AmpTool

GPU PWA



GPU ACCELERATED PWA

IU AmpTool

User-Oriented

- Analysis Framework Independent:
  - User supplies data and defines amplitudes from 4-vector level
  - The Framework drives the fit, uses fitted parameters to plot data and Intensity-weighted Monte-Carlo.

GPU ACCELERATED PWA

IU AmpTool

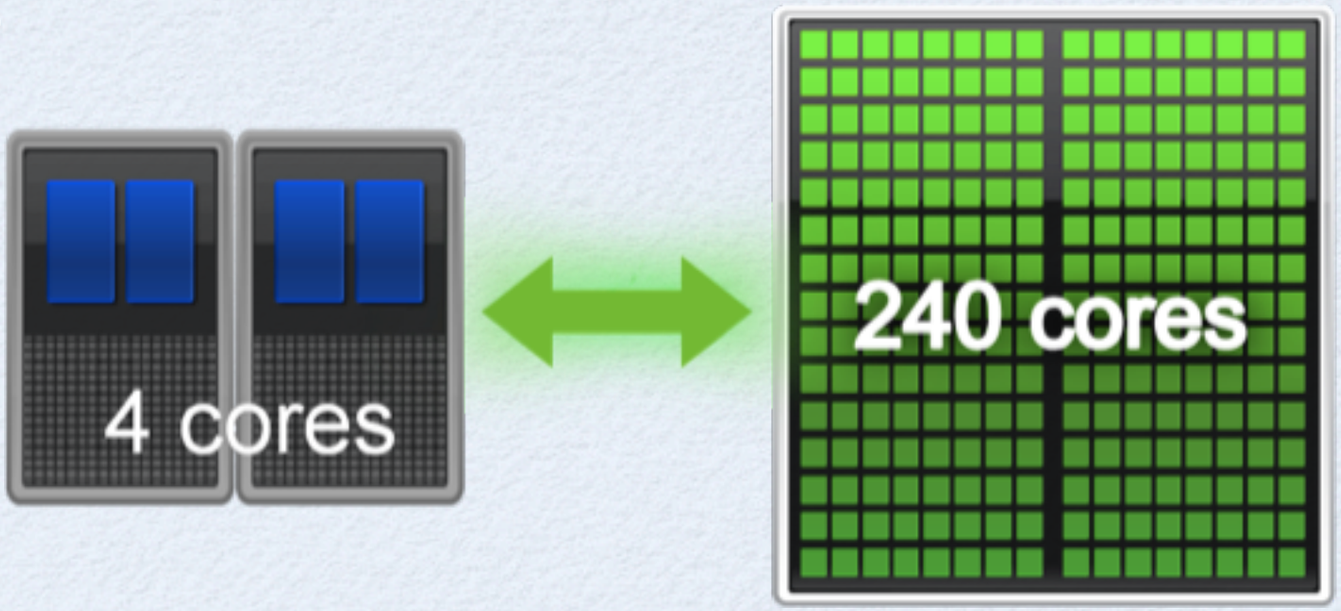
GPU Integration

- Turn on by a simple compile-time switch
- The framework completely manages the GPU initialization/memory / function calls.
- All GPU-enabled amplitudes are automatically called, the rest are evaluated on CPU. (for data and MC - norm integral and plotting)
- Log-intensity sum is calculated on the GPU.
- Integration with MPI allows using multiple machines with GPUs

WHY BOTHER ?

- Massively parallel architecture

- Large advantage in performance over CPU

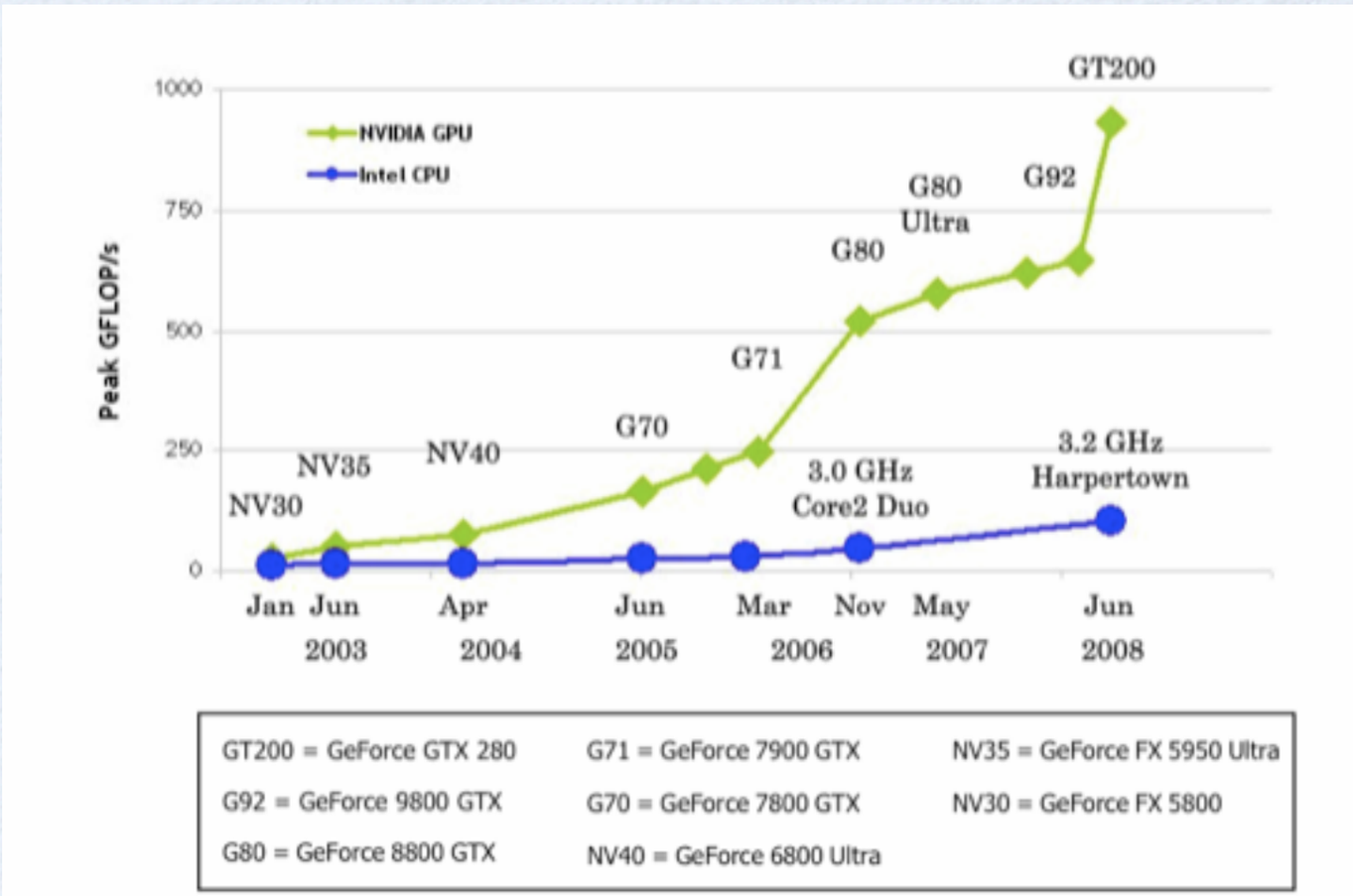


- A sound and rapidly developing CUDA (Compute Unite Device Architecture) toolkit as an extension of C language (in future C++, Java, Fortran, etc.)

WHY BOTHER ?

- Massively parallel architecture

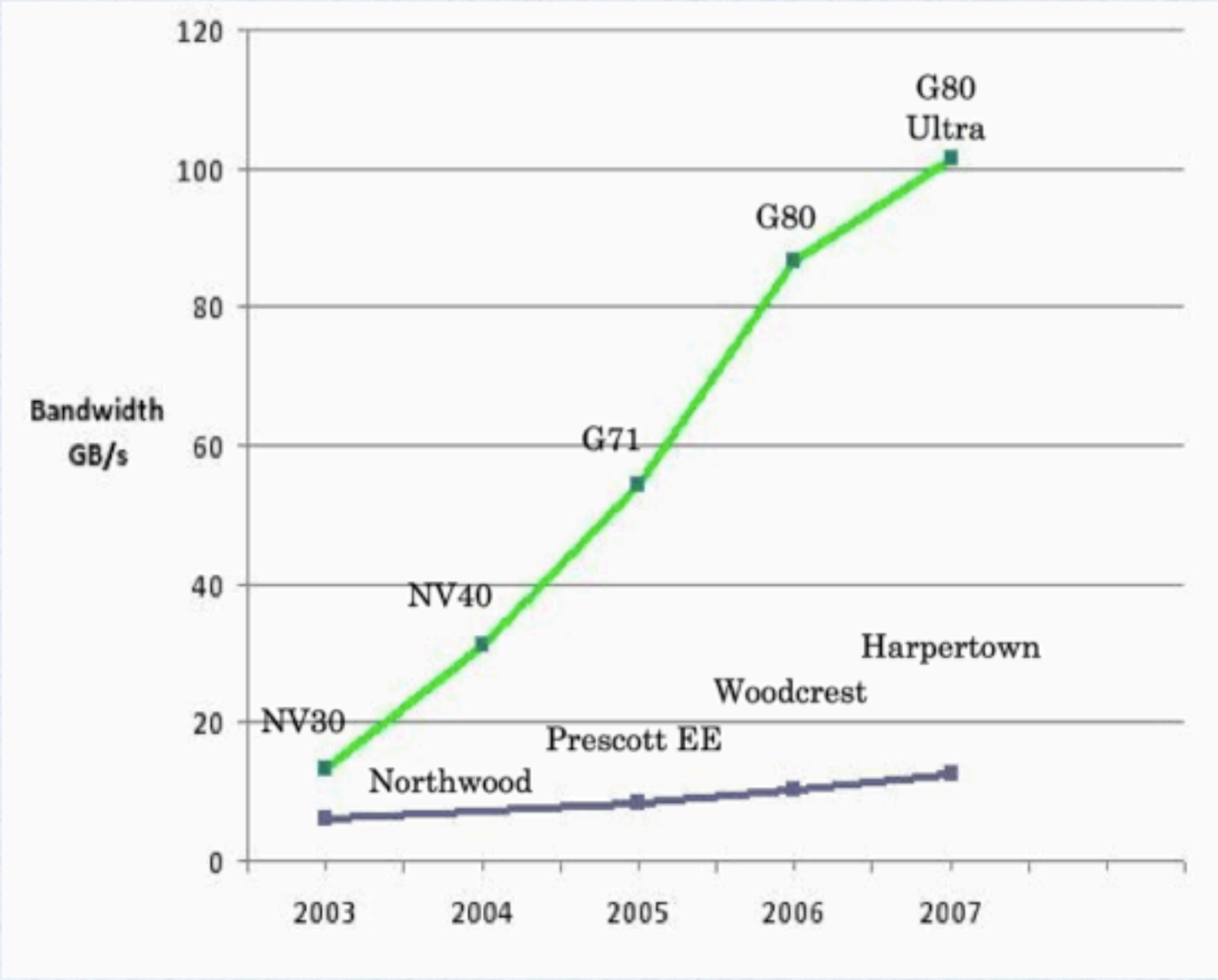
- Large advantage in performance over CPU



- A sound and rapidly developing CUDA (Compute Unite Device Architecture) toolkit as an extension of C language (in future C++, Java, Fortran, etc.)

WHY BOTHER ?

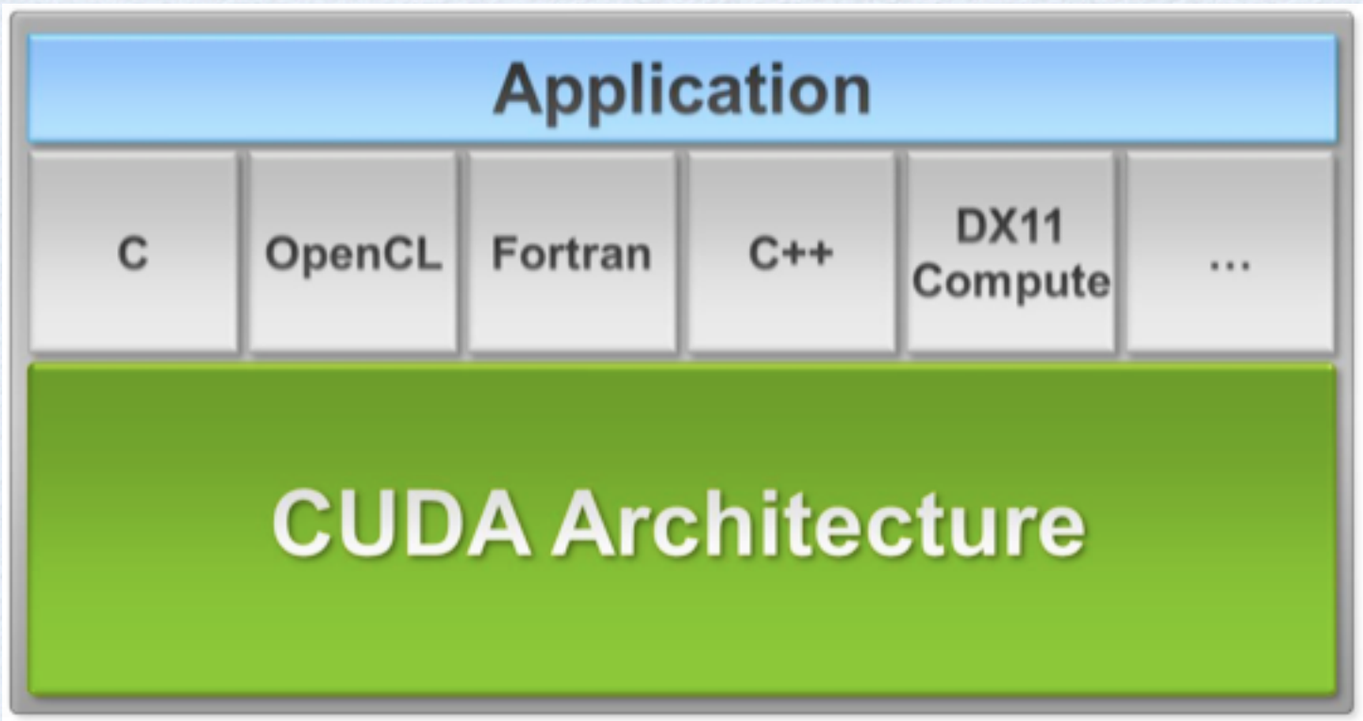
- Massively parallel architecture
- Large advantage in performance over CPU
- A sound and rapidly developing CUDA (Compute Unite Device Architecture) toolkit as an extension of C language (in future C++, Java, Fortran, etc.)



WHY BOTHER ?

- Massively parallel architecture

- Large advantage in performance over CPU



- A sound and rapidly developing CUDA (Compute Unite Device Architecture) toolkit as an extension of C language (in future C++, Java, Fortran, etc.)

GPU FEATURES

- Rapid advancements in GPUs driven by the gaming industry: ~$57 billion by 09!
- Cost-effective
  - ~ $350 for 1 TFlops
- Power-effective
  - < 200 Watts
- Scalable
  - Up to 4 cards in a MB

CUDA INITIATIVE

- The market for GPGPU in HPC is taking off!
- Support for double-precision floating point numbers starting GT 200 architecture.
- Dedicated TESLA boards: 4GB of DDR3 memory, GT200 GPU with 240 computing cores.
- Workstations and turn-key clusters available from various vendors

CUDA INITIATIVE

- The market for GPGPU in HPC is taking off!
- Support for double-precision floating point numbers starting GT 200 architecture.
- Dedicated TESLA boards: 4GB of DDR3 memory, GT200 GPU with 240 computing cores.
- Workstations and turn-key clusters available from various vendors
  - Some of rapidly extending CUDA features:

Fast floating point math functions

A large list of special math functions

FFT for CUDA

BLAS for CUDA

Fast linear interpolation using textures

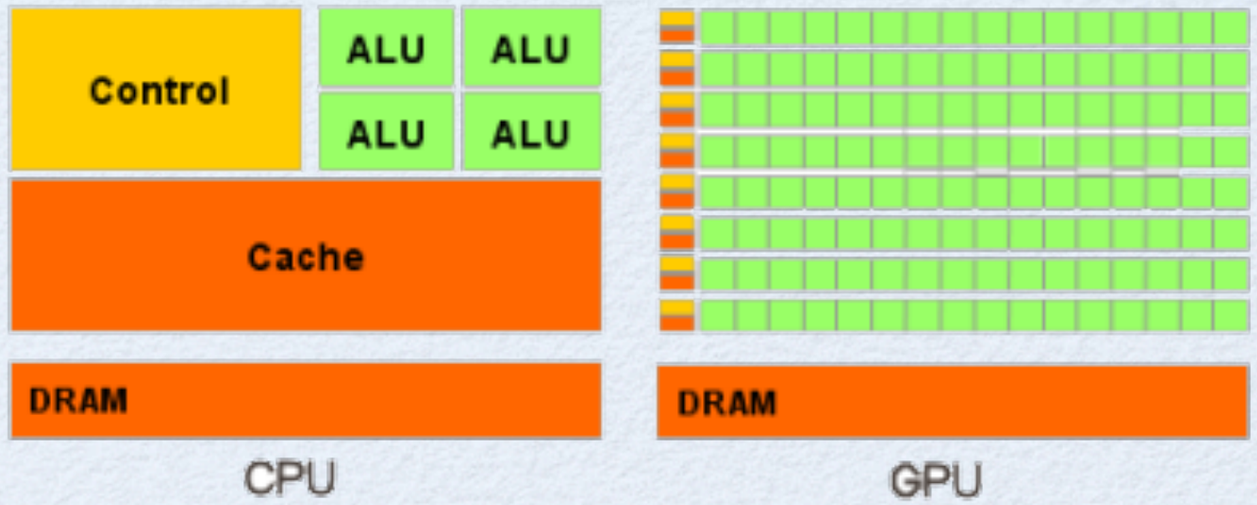
CUDA IN HPC

Some examples from various fields:

- ☑ Lattice QCD
- ☑ Image Processing / Tomography / Video Editing
- ☑ Molecular Dynamics / Computational Fluid Dynamics
- ☑ MATLAB Accelerator Engines (Matrices/Graphics)
- ☑ Computational Chemistry
- ☑ Computational Finance

CAVEATS

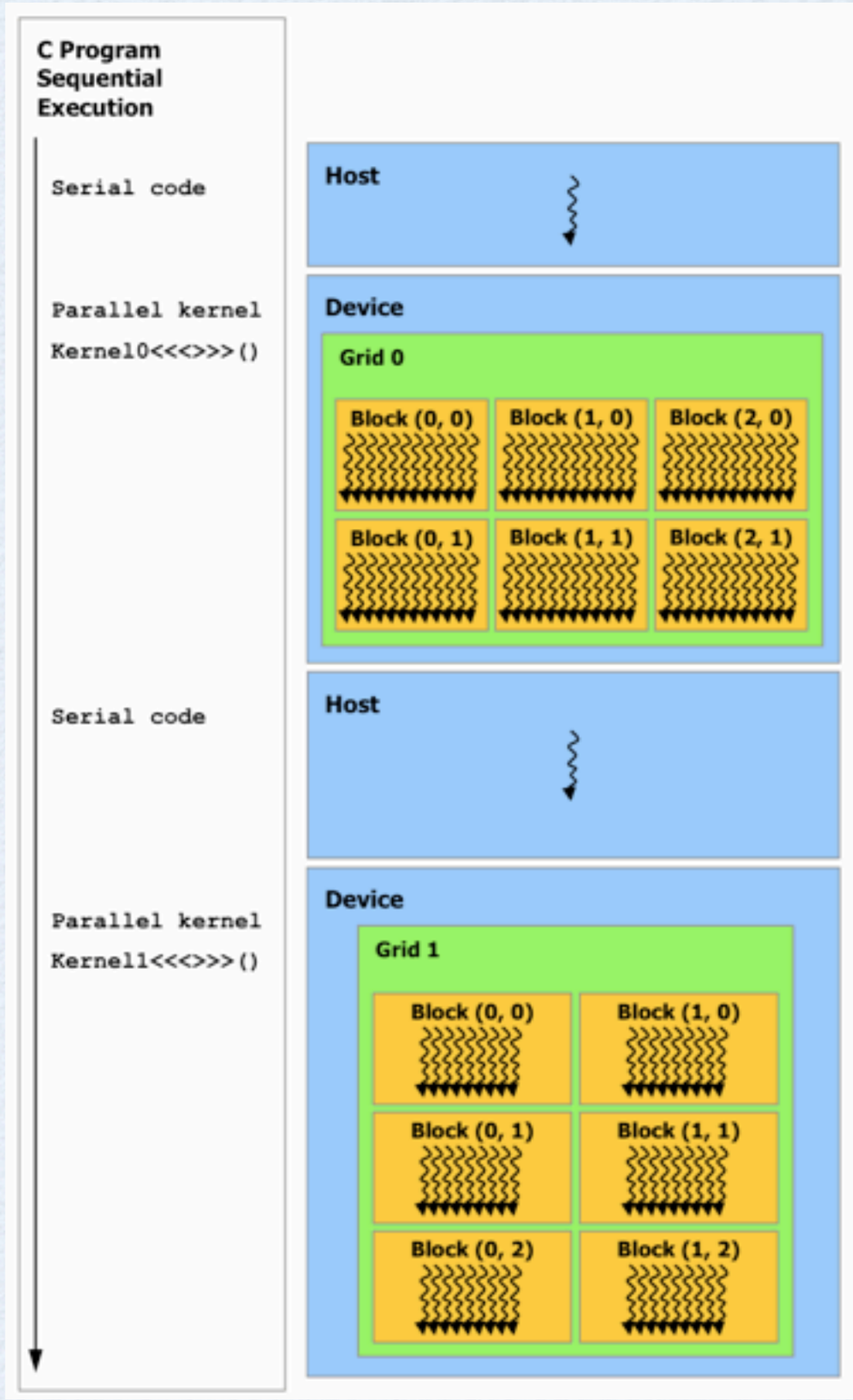
GPU architecture dictates:



- Demands massively-parallel tasks for optimal performance, to hide memory access latencies, etc.
- Requires extensive optimization at programming level: programmer has to worry about optimal memory access patterns/ availability, hardware limitations, adopt effective parallel algorithms.
- Favors **arithmetically intensive** problems.
- Weak* performance on doubles: only 1 DP core for every 8 SP

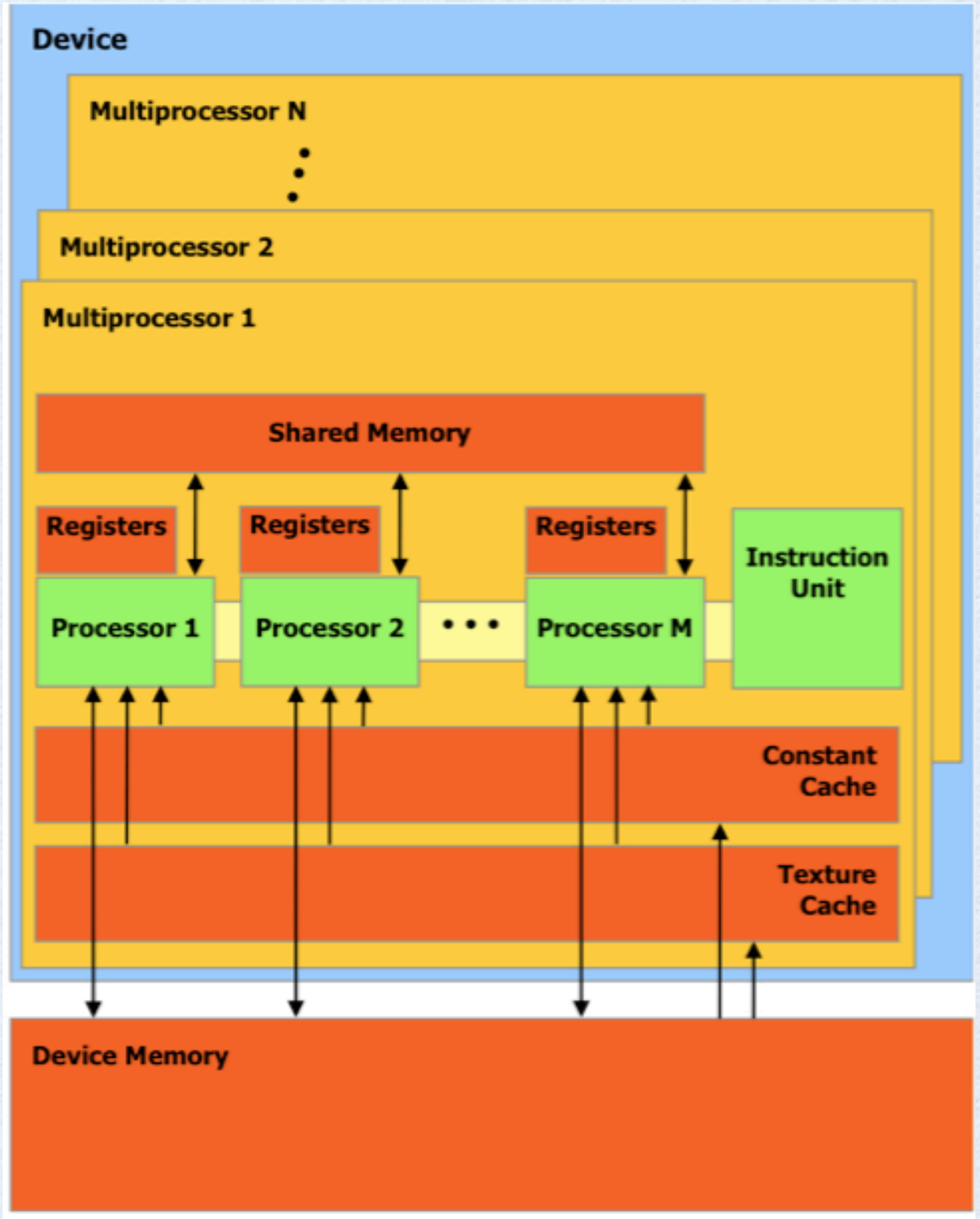
CUDA PROGRAMMING MODEL

- Identify parallelizable parts of the code and port them to GPU
- In GPU cores are divided into several multi processors (MP). Each MP has 8 scalar processor (SP) cores.
- Compute threads are divided into Blocks. All the blocks constitute the Grid of threads.
- All threads in blocks are executed at once - assigned to a single MP, scheduling is done automatically by the GPU driver.
- Max blocks/threads per MP: 8 / 1024 !
- Each MP has 16KB registers and 16KB shared memory - this limits number of concurrently executed Blocks per MP.
- All processes access FAST, read-only CONSTANT memory.
- Block structure provides Automatic Scalability, provided by CUDA driver.



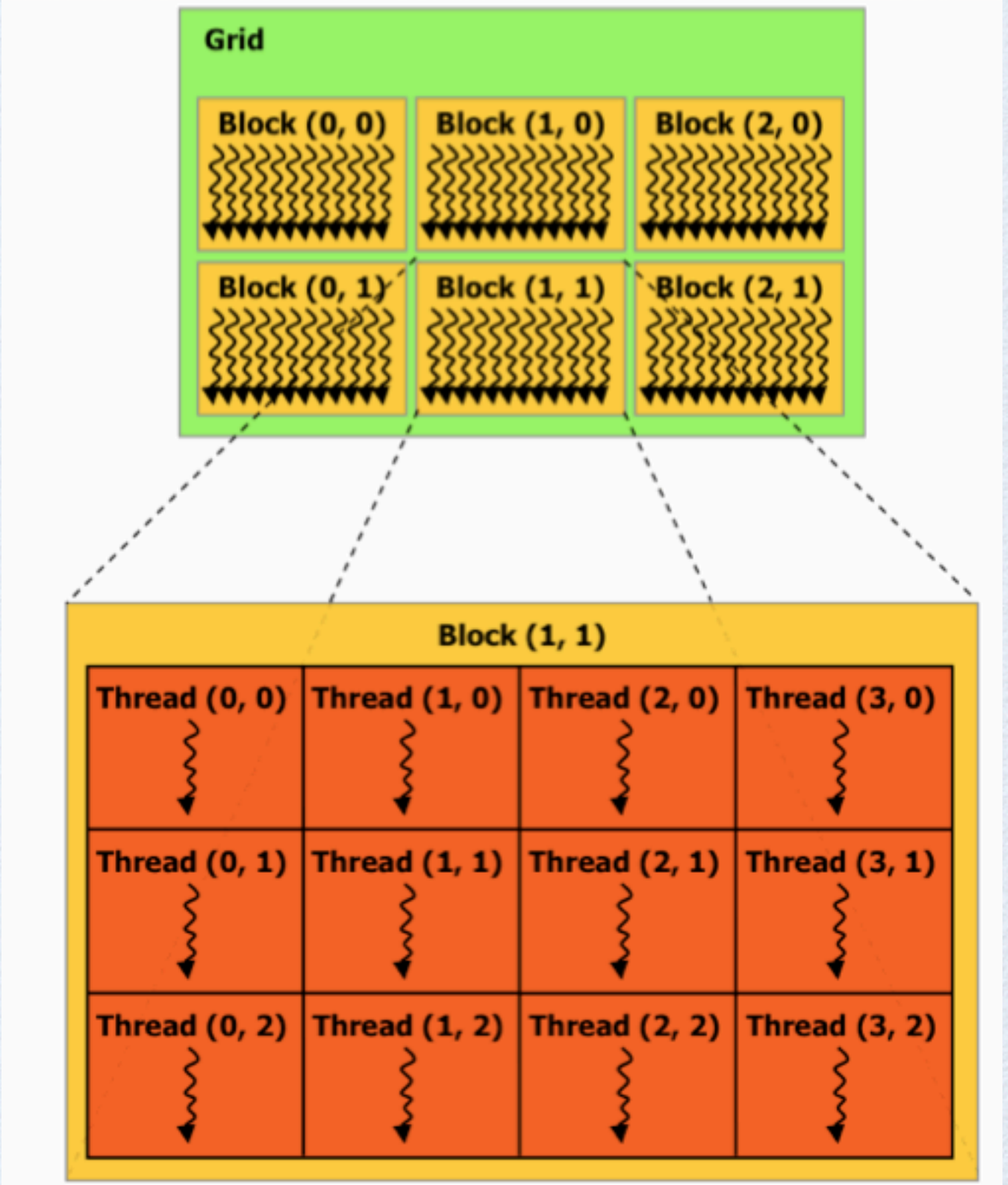
CUDA PROGRAMMING MODEL

- Identify parallelizable parts of the code and port them to GPU
- In GPU cores are divided into several multi processors (MP). Each MP has 8 scalar processor (SP) cores.
- Compute threads are divided into Blocks. All the blocks constitute the Grid of threads.
- All threads in blocks are executed at once - assigned to a single MP, scheduling is done automatically by the GPU driver.
- Max blocks/threads per MP: 8 / 1024 !
- Each MP has 16KB registers and 16KB shared memory - this limits number of concurrently executed Blocks per MP.
- All processes access FAST, read-only CONSTANT memory.
- Block structure provides Automatic Scalability, provided by CUDA driver.



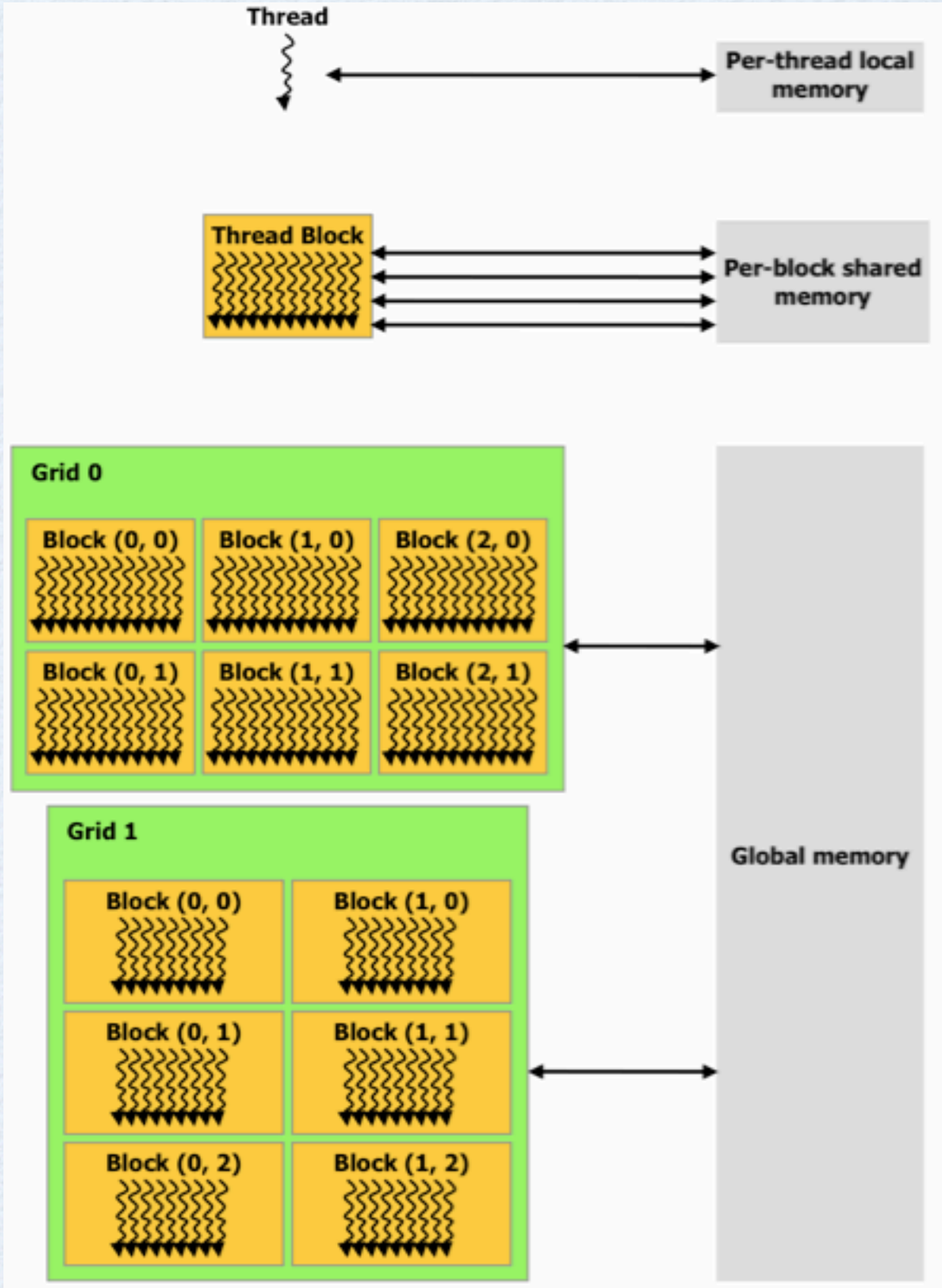
CUDA PROGRAMMING MODEL

- Identify parallelizable parts of the code and port them to GPU
- In GPU cores are divided into several multi processors (MP). Each MP has 8 scalar processor (SP) cores.
- Compute threads are divided into Blocks. All the blocks constitute the Grid of threads.
- All threads in blocks are executed at once - assigned to a single MP, scheduling is done automatically by the GPU driver.
- Max blocks/threads per MP: 8 / 1024 !
- Each MP has 16KB registers and 16KB shared memory - this limits number of concurrently executed Blocks per MP.
- All processes access FAST, read-only CONSTANT memory.
- Block structure provides Automatic Scalability, provided by CUDA driver.



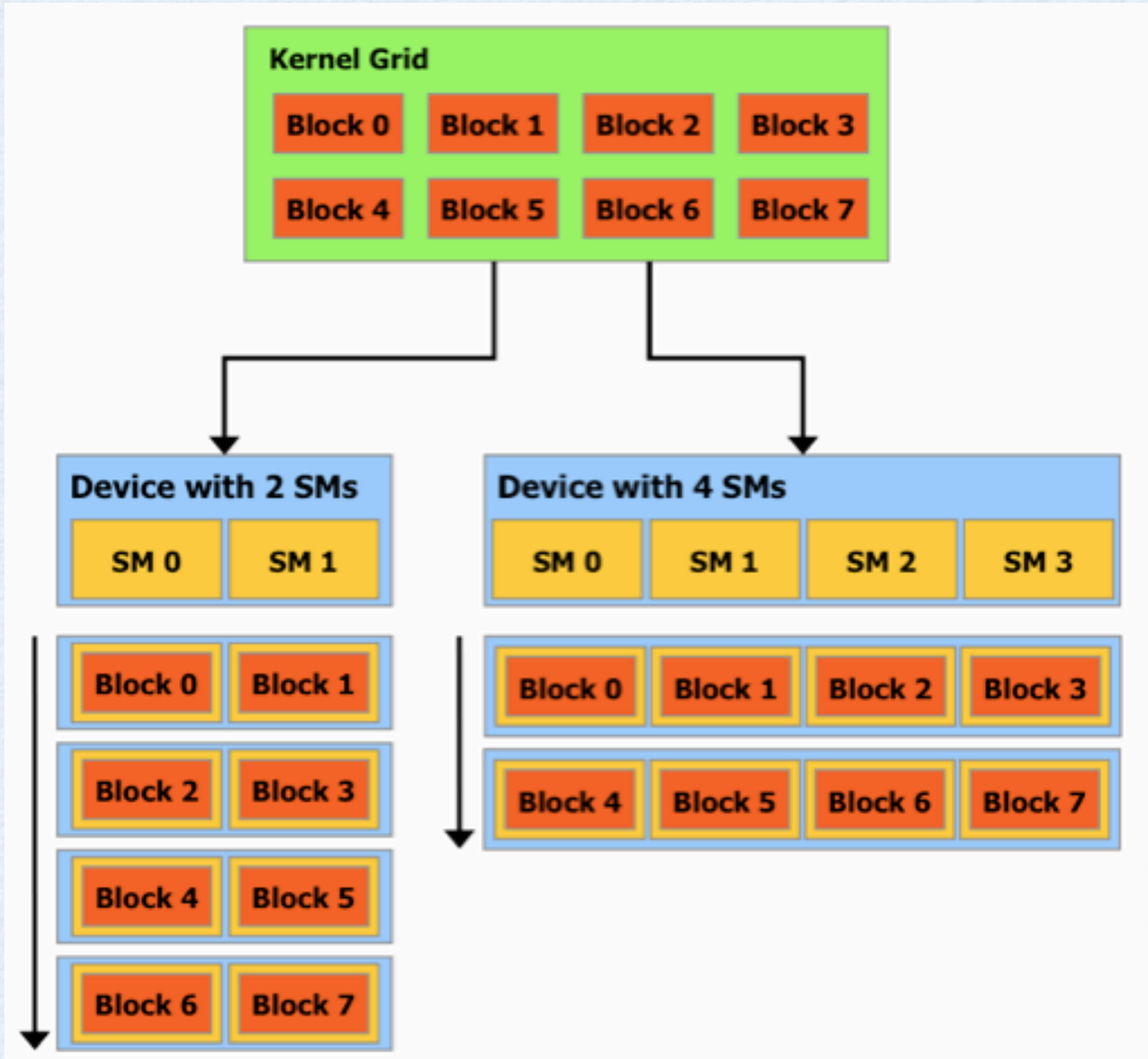
CUDA PROGRAMMING MODEL

- Identify parallelizable parts of the code and port them to GPU
- In GPU cores are divided into several multi processors (MP). Each MP has 8 scalar processor (SP) cores.
- Compute threads are divided into Blocks. All the blocks constitute the Grid of threads.
- All threads in blocks are executed at once - assigned to a single MP, scheduling is done automatically by the GPU driver.
- Max blocks/threads per MP: 8 / 1024 !
- Each MP has 16KB registers and 16KB shared memory - this limits number of concurrently executed Blocks per MP.
- All processes access FAST, read-only CONSTANT memory.
- Block structure provides Automatic Scalability, provided by CUDA driver.



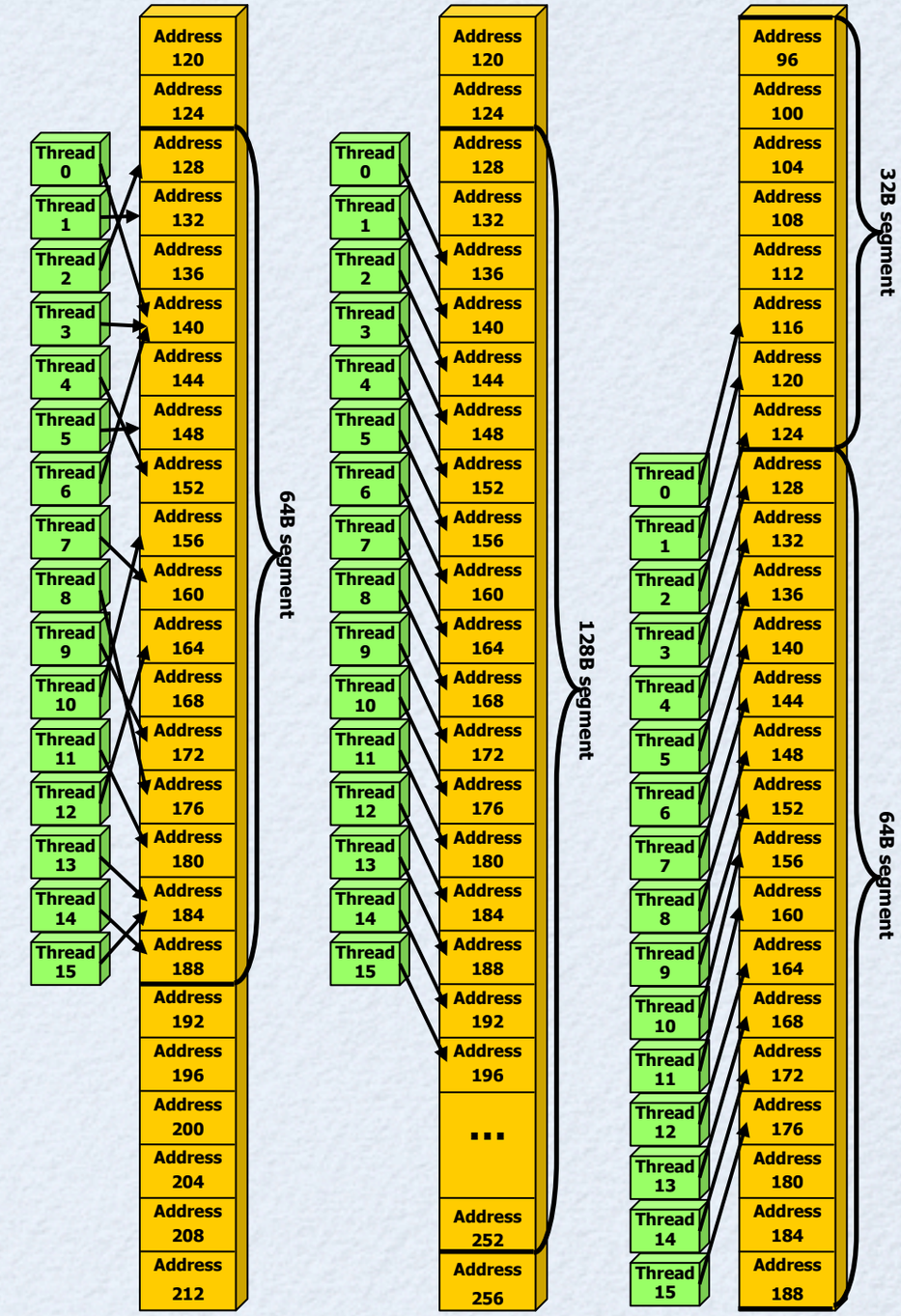
CUDA PROGRAMMING MODEL

- Identify parallelizable parts of the code and port them to GPU
- In GPU cores are divided into several multi processors (MP). Each MP has 8 scalar processor (SP) cores.
- Compute threads are divided into Blocks. All the blocks constitute the Grid of threads.
- All threads in blocks are executed at once - assigned to a single MP, scheduling is done automatically by the GPU driver.
- Max blocks/threads per MP: 8 / 1024 !
- Each MP has 16KB registers and 16KB shared memory - this limits number of concurrently executed Blocks per MP.
- All processes access FAST, read-only CONSTANT memory.
- Block structure provides Automatic Scalability, provided by CUDA driver.



MEMORY ACCESS PATTERN

- Global memory access has large latency (~100 x slower than local and shared).
- All the processes in a warp should access global memory within 128B for a single transfer read/write.
- Registers and Shared memory are a scares resource.
- Shared memory access should also be arranged to avoid bank conflicts.



Left: random `float` memory access within a 64B segment, resulting in one memory transaction.
Center: misaligned `float` memory access, resulting in one transaction.
Right: misaligned `float` memory access, resulting in two transactions.

CUDA FOR C

- Is a superset of C programming language. Additional, very limited set of GPU primitives. GPU functions (kernels) can only call other GPU functions (no system calls, yet).
- Supports some C++ constructs. Full support for the announced, next-gen cards.
- The compiler - nvcc and GPU driver available for most common flavors of OSes - Linux/Mac/Windows. CUDA code is universal across the platforms.
- Comes with toolkit, manual and SDK - easy learning curve.
- GDB debugger for Linux, VS integrated NEXUS debugger for Windows.
- Rapidly Evolving!

PWA IS A GOOD CANDIDATE?

- Partial Wave Analysis seems to be a ideal candidate for GPU computing!
  - The quantities of interest are additive for each event - Trivially Parallelizable!
    - Amplitudes calculations grow in arithmetical complexity.
    - Maximum likelihood fits involve a large number of fitting iterations - only a few numbers transferred to GPU per computationally intensive calculation!
    - Large number of events leverage GPU's multi-core, multi-thread capabilities.

GPU DRIVING CODE

- Setting and Querying the GPU:

```
///////CUDA INITIALIZATION
cudaSetDevice(0);
//Query the device
cudaDeviceProp devProp;
cudaGetDeviceProperties( &devProp, 0 );
```

- Allocating memory on GPU and copying data to it:

switch between single and double prec.

```
int m_iVArrSize=sizeof(GDouble)*m_iNAmps;
//Device memory
cudaMalloc((void**) &m_pfDevAmpRe, m_iAmpArrSize);
//Memory transfer CPU -> GPU
cudaMemcpy(m_pfDevAmpRe, m_pfAmpRe, m_iAmpArrSize, cudaMemcpyHostToDevice);
```

- Calling function on GPU:

```
void GPU_ExecCalcAmp(dim3 dimGrid,dim3 dimBlock,GDouble* pfDevData,GDouble* pcDevAmp,int iNEvents, int iNParticles,
    GDouble* pfPars, int iNPars)
{
    //casting Amp array to WCUComplex for 8 or 16 bit write operation of both real and complex parts at once
    ampcalc_kernel<<< dimGrid, dimBlock >>>( pfDevData, (WCUComplex*) pcDevAmp, iNEvents, iNParticles);
}
```

CUDA AMPLITUDE

- CUDA Amplitude Template in IU AmpTool:

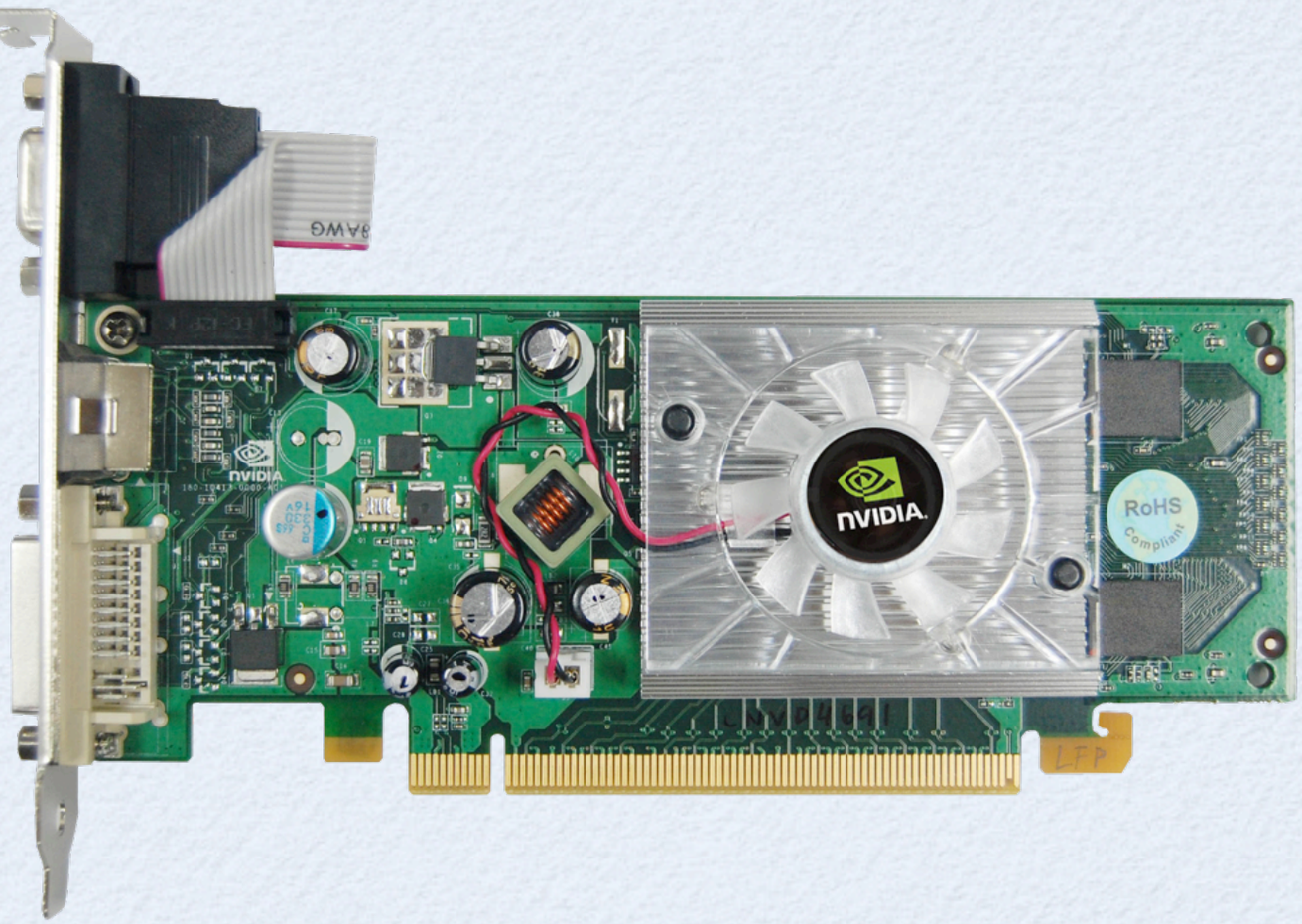
```
__global__ void
ampcalc_kernel(GDouble* pfDevData, WCUComplex* pcDevAmp, int iNEvents, int iNParticles)
{
    int iEvent = threadIdx.x+GPU_BLOCK_SIZE_X*threadIdx.y+
                                (blockIdx.x+blockIdx.y*gridDim.x)*GPU_BLOCK_SIZE_SQ;

    WCUComplex m_cS;
    WCUComplex m_cSij[3];

    GDouble dV1[4];
    GDouble dV2[4];
    GDouble dV3[4];

    dV1[0]=pfDevData[iEvent];
    dV1[1]=pfDevData[iNEvents+iEvent];
    dV1[2]=pfDevData[2*iNEvents+iEvent];
    dV1[3]=pfDevData[3*iNEvents+iEvent];


    ...

    pcDevAmp[iEvent]=cAmp;
}
```

Here goes the code for the event number "iEvent"

FEASIBLE TEST PLATFORM?

- For a Start:

GeForce 8400 GT ~ $30



Or Even a Laptop's NVIDIA Card.

FEASIBLE TEST PLATFORM?

THE BEAST

- Specs:
  - CPU: Core i7 920 - 2.66 GHz Quad-Core
  - Memory: 6 GB DDR3 1600 MHz
  - Overall Cost ~ $1600
  - Expandable - up to 3 GPUs!
  - OS - Fedora 10 x86_64

- CUDA GPU:
  EVGA GTX 285 SSC ~ 1 TFlops !
  1584 MHz core clock
  240 Processing Cores

GPU MEMORY:
1024 MB, 512 bit DDR3
169.3 GB/s Memory Bandwidth
GPU to RAM: ~ 5.6 GB/s measured!

IS IT WORKING? Yes !!!

$$L = \sum_{n \in N_{Events}} \log \left(\sum_{i=1, j \le i}^{i \le N_{Amps}} V_i V_j^* A_i(n) A_j^*(n) \right)$$

Standalone Benchmarks:

1 core, SSE VECTORIZED!

Events	Amps	CPU	GPU	GPU/CPU x
~1.*10^5	40	38.5s/500	29.1s/5000	13
~2.*10^5	20	21.4s/500	11.5s/5000	18.5
~2.*10^5	8	5.3s/500	2.1s/5000	25
~2*10^6	8	53s/500	19.7/5000	27
~1.*10^7	4	29.5s/100	5.78s/1000	51
~1.*10^7	1	17.3s/100	1.85s/1000	95
~2.*10^7	2	45.8s/100	4.56s/1000	100.8

SOME MORE BENCHMARKS

GPU timings include GPU-> CPU memory transfer time, C++ Compiler - > ICC 11 with Auto-Vectorization !

Amps Calc Time: sec/Iteration (with AmpFactor->Amp transform on CPU)

Events	CPU - 1 Amp	GPU - 1 Amp	CPU - 3 Amps	GPU - 3 Amps
2581	$5.1*10^{-3}$	$1.3*10^{-4}$	0.015	$3*10^{-4}$
$\sim 1.*10^{6}$	2	0.012	6	0.037
$\sim 8.*10^{6}$	16	0.097	48	0.29

Log-Like Sum: sec/Iteration

Events	CPU - 1 Amp	GPU - 1 Amp	CPU - 3 Amps	GPU - 3 Amps
2581	$1.1*10^{-4}$	$6.8*10^{-5}$	$1.8*10^{-4}$	$6.8*10^{-5}$
$\sim 1.*10^{6}$	0.049	$2.5*10^{-4}$	0.074	$4.6*10^{-4}$
$\sim 8.*10^{6}$	0.37	$1.5*10^{-3}$	0.59	$3.2*10^{-3}$

WHAT'S AHEAD?

- OpenCL ?
  - ☑ Oriented towards consumer market.
  - ☑ Still in early stages of development - lots of

  - ☑ Easy Transition CUDA -> OpenCL
  - ☑ Slow to evolve :

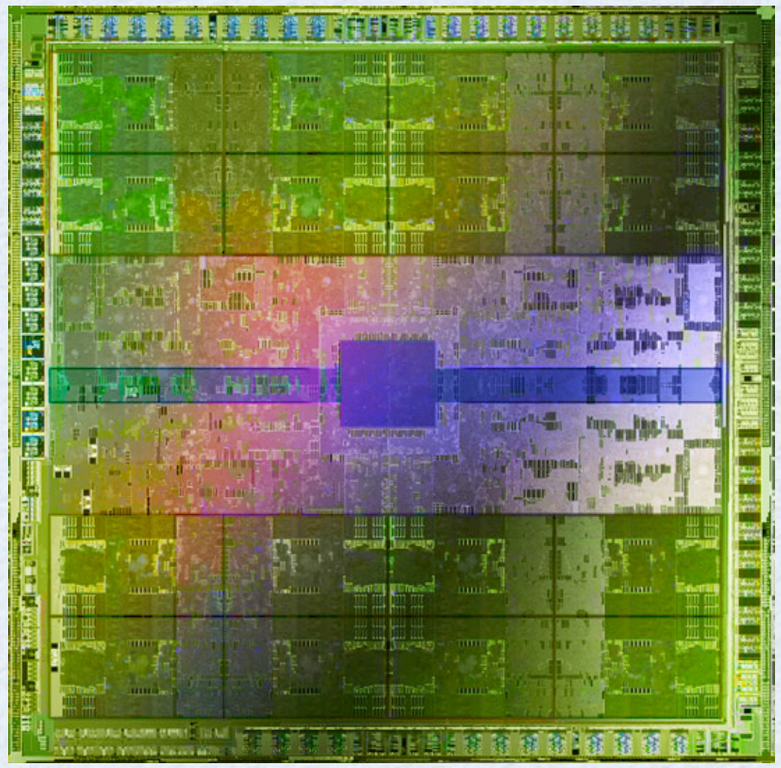
WHAT'S AHEAD?

- OpenCL ?
  - ☑ Oriented towards consumer market.
  - ☑ Still in early stages of development - lots of
  - ☑ Easy Transition
  - ☑

WHAT'S AHEAD?

FERMI

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Warp schedulers (per SM)	1	1	2
Special Function Units (SFUs) / SM	2	2	4
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache (per SM)	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit



Consumer Cards Coming Early 2010!

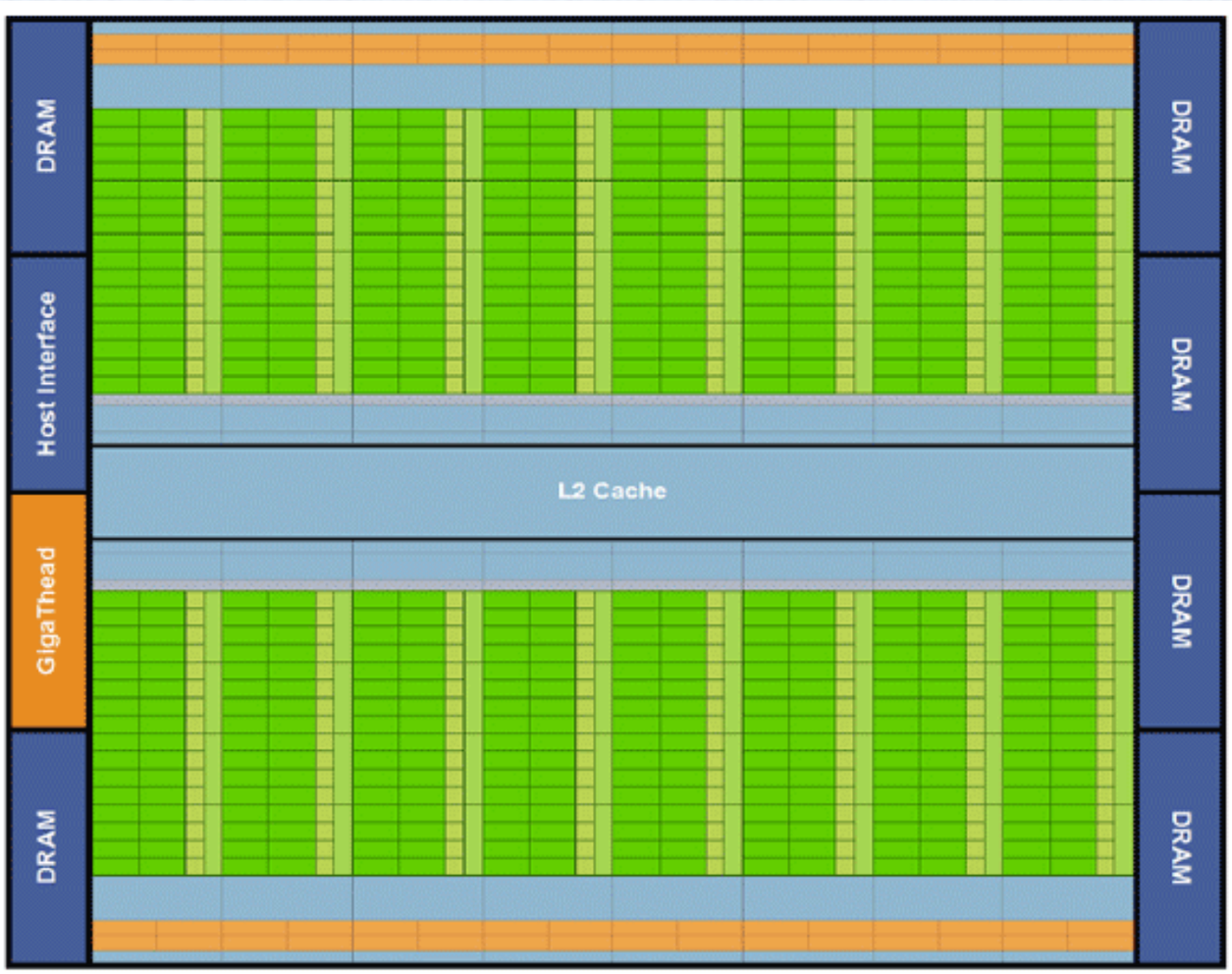
Price~ $400-$500

WHAT'S AHEAD?

- GT300 Improvements:
  - ☑ Support for the next generation IEEE 754-2008 double precision floating point standard
  - ☑ Up to 1 terabyte of faster (GDDR5) memory
  - ☑ 64-bit virtual address space (possibly unified CPU-GPU memory mapping)
  - ☑ ECC (error correcting codes) - not available on consumer cards
  - ☑ Multi-level cache hierarchy with L1 and L2 caches
  - ☑ Support for the C++ programming language
  - ☑ System calls and recursive functions
  - ☑ Concurrent kernel execution, fast context switching, 10x faster atomic instructions
  - ☑ More cores, 1:2 ratio for double-precision cores.

WHAT'S AHEAD?

- GT300 Improvements:
  - ☑ Support for ision floating point standard
  - ☑ Up to 1 terab
  - ☑ 64-bit virtua emory mapping)
  - ☑ ECC (error c ards
  - ☑ Multi-level c
  - ☑ Support for
  - ☑ System calls
  - ☑ Concurrent k ster atomic instructions
  - ☑ More cores, 1:2 ratio for double-precision cores.



Fermi's 16 SM are positioned around a common L2 cache. Each SM is a vertical rectangular strip that contain an orange portion (scheduler and dispatch), a green portion (execution units), and light blue portions (register file and L1 cache).

JUST ANNOUNCED

- Tesla C2050 & C2070 GPU Computing Processors
  - Single GPU PCI-Express Gen-2 cards for workstation configurations
  - Up to 3GB and 6GB (respectively) on-board GDDR5 memory
  - Double precision performance in the range of 520GFlops - 630 GFlops
  - Price - $2,499 and $3,999

- Tesla S2050 & S2070 GPU Computing Systems
  - Four Tesla GPUs in a 1U system product for cluster and datacenter deployments
  - Up to 12 GB and 24 GB (respectively) total on board GDDR5 memory
  - Double precision performance in the range of 2.1 TFlops - 2.5 TFlops
  - Price - $12,995 and $18,995

SUMMARY

- PWA is becoming increasingly computationally intensive due to massive amount of new data and use of more sophisticated amplitudes.
- GPGPU brings a computational power of a cluster to a desktop computer.
- GPU acceleration in IU AmpTools PWA framework proved to be highly successful.