## Tesla Data Center & Workstation GPU Solutions

### Tesla M-series GPUs

**M2090 | M2070 | M2050**

- **Integrated CPU-GPU Servers & Blades**

<table>
<thead>
<tr>
<th></th>
<th>M2090</th>
<th>M2070</th>
<th>M2050</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cores</strong></td>
<td>512</td>
<td>448</td>
<td>448</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>6 GB</td>
<td>6 GB</td>
<td>3 GB</td>
</tr>
<tr>
<td><strong>Memory bandwidth</strong></td>
<td>177.6 GB/s</td>
<td>148.8 GB/s</td>
<td>148.8 GB/s</td>
</tr>
</tbody>
</table>

### Tesla C-series GPUs

**C2070 | C2050**

- **Workstations 2 to 4 Tesla GPUs**

<table>
<thead>
<tr>
<th></th>
<th>C2070</th>
<th>C2050</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory</strong></td>
<td>6 GB</td>
<td>3 GB</td>
</tr>
<tr>
<td><strong>Memory bandwidth</strong></td>
<td>144 GB/s</td>
<td>144 GB/s</td>
</tr>
<tr>
<td><strong>Peak Perf Gflops</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single Precision</td>
<td>1331</td>
<td>1030</td>
</tr>
<tr>
<td>Double Precision</td>
<td>665</td>
<td>515</td>
</tr>
</tbody>
</table>

© NVIDIA Corporation 2011
CUDA 4.0: Highlights

**Easier Parallel Application Porting**
- Share GPUs across multiple threads
- Single thread access to all GPUs
- No-copy pinning of system memory
- New CUDA C/C++ features
- Thrust templated primitives library
- NPP image/video processing library
- Layered Textures

**Faster Multi-GPU Programming**
- Unified Virtual Addressing
- NVIDIA GPUDirect™ v2.0
  - Peer-to-Peer Access
  - Peer-to-Peer Transfers
  - GPU-accelerated MPI

**New & Improved Developer Tools**
- Auto Performance Analysis
- C++ Debugging
- GPU Binary Disassembler
- cuda-gdb for MacOS
Easier Porting of Existing Applications

**Share GPUs across multiple threads**

- Easier porting of multi-threaded apps
  - pthreads / OpenMP threads share a GPU
- Launch concurrent kernels from different host threads
  - Eliminates context switching overhead
- New, simple context management APIs
  - Old context migration APIs still supported

**Single thread access to all GPUs**

- Each host thread can now access all GPUs in the system
  - One thread per GPU limitation removed
- Easier than ever for applications to take advantage of multi-GPU
  - Single-threaded applications can now benefit from multiple GPUs
  - Easily coordinate work across multiple GPUs (e.g. halo exchange)
No-copy Pinning of System Memory

- Reduce system memory usage and CPU memcpy() overhead
- Easier to add CUDA acceleration to existing applications
- Just register malloc’d system memory for async operations and then call cudaMemcpy() as usual

<table>
<thead>
<tr>
<th>Before No-copy Pinning</th>
<th>With No-copy Pinning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra allocation and extra copy required</td>
<td>Just register and go!</td>
</tr>
<tr>
<td>malloc(a)</td>
<td></td>
</tr>
<tr>
<td>cudaMallocHost(b)</td>
<td></td>
</tr>
<tr>
<td>memcpy(b, a)</td>
<td>cudaHostRegister(a)</td>
</tr>
<tr>
<td>cudaMemcpy() to GPU, launch kernels, cudaMemcpy() from GPU</td>
<td></td>
</tr>
<tr>
<td>memcpy(a, b)</td>
<td></td>
</tr>
<tr>
<td>cudaFreeHost(b)</td>
<td>cudaHostUnregister(a)</td>
</tr>
</tbody>
</table>

All CUDA-capable GPUs on Linux or Windows
- Requires Linux kernel 2.6.15+ (RHEL 5)
New CUDA C/C++ Language Features

- **C++ new/delete**
  - Dynamic memory management

- **C++ virtual functions**
  - Easier porting of existing applications

- **Inline PTX**
  - Enables assembly-level optimization
C++ Templatized Algorithms & Data Structures (Thrust)

- Powerful open source C++ parallel algorithms & data structures
  - Similar to C++ Standard Template Library (STL)
- Automatically chooses the fastest code path at compile time
  - Divides work between GPUs and multi-core CPUs
  - Parallel sorting @ 5x to 100x faster

<table>
<thead>
<tr>
<th>Data Structures</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>thrust::device_vector</td>
<td>thrust::sort</td>
</tr>
<tr>
<td>thrust::host_vector</td>
<td>thrust::reduce</td>
</tr>
<tr>
<td>thrust::device_ptr</td>
<td>thrust::exclusive_scan</td>
</tr>
<tr>
<td>Etc.</td>
<td>Etc.</td>
</tr>
</tbody>
</table>
Unified Virtual Addressing

One address space for all CPU and GPU memory
- Determine physical memory location from pointer value
- Enables libraries to simplify their interfaces (e.g. cudaMemcpy)

<table>
<thead>
<tr>
<th>Before UVA</th>
<th>With UVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate options for each permutation</td>
<td>One function handles all cases</td>
</tr>
<tr>
<td>cudaMemcpyHostToDevice</td>
<td>cudaMemcpyDefault</td>
</tr>
<tr>
<td>cudaMemcpyHostToDevice</td>
<td>(data location becomes an implementation detail)</td>
</tr>
<tr>
<td>cudaMemcpyDeviceToDevice</td>
<td></td>
</tr>
<tr>
<td>cudaMemcpyDeviceToDeviceToHost</td>
<td></td>
</tr>
</tbody>
</table>

Supported on Tesla 20-series and other Fermi GPUs
- 64-bit applications on Linux and Windows TCC
Unified Virtual Addressing

*Easier to Program with Single Address Space*

No UVA: Multiple Memory Spaces

- System Memory
  - 0x0000 0xFFFF
  - 0x0000 0xFFFF
- GPU0 Memory
  - 0x0000 0xFFFF
- GPU1 Memory
  - 0x0000 0xFFFF

CPU

GPU0

GPU1

PCI-e

UVA: Single Address Space

- System Memory
  - 0x0000
- GPU0 Memory
  - 0xFFFF
- GPU1 Memory
  - 0xFFFF

CPU

GPU0

GPU1

PCI-e

© NVIDIA Corporation 2011
NVIDIA GPUDirect™: Towards Eliminating the CPU Bottleneck

**Version 1.0**
*for applications that communicate over a network*

- Direct access to GPU memory for 3rd party devices
- Eliminates unnecessary sys mem copies & CPU overhead
- Supported by Mellanox and Qlogic
- Up to 30% improvement in communication performance

**Version 2.0**
*for applications that communicate within a node*

- Peer-to-Peer memory access, transfers & synchronization
- Less code, higher programmer productivity

Details @ [http://www.nvidia.com/object/software-for-tesla-products.html](http://www.nvidia.com/object/software-for-tesla-products.html)
Before NVIDIA GPUDirect™ v2.0

Required Copy into Main Memory

Two copies required:
1. cudaMemcpy(GPU2, sysmem)
2. cudaMemcpy(sysmem, GPU1)
NVIDIA GPUDirect™ v2.0: Peer-to-Peer Communication

Direct Transfers between GPUs

Only one copy required:
1. cudaMemcpy(GPU2, GPU1)
GPUDirect v2.0: Peer-to-Peer Communication

- Direct communication between GPUs
  - Faster - no system memory copy overhead
  - More convenient multi-GPU programming

- Direct Transfers
  - Copy from GPU₀ memory to GPU₁ memory
  - Works transparently with UVA

- Direct Access
  - GPU₀ reads or writes GPU₁ memory (load/store)

- Supported only on Tesla 20-series (Fermi)
  - 64-bit applications on Linux and Windows TCC
Automated Performance Analysis in Visual Profiler

Summary analysis & hints
- Session
- Device
- Context
- Kernel

New UI for kernel analysis
- Identify limiting factor
- Analyze instruction throughput
- Analyze memory throughput
- Analyze kernel occupancy
New Features in cuda-gdb

- Breakpoints on all instances of templated functions
- C++ symbols shown in stack trace view
- `info cuda threads` automatically updated in DDD
- Fermi disassembly with `cuobjdump`
- Now available for both Linux and MacOS

NVIDIA Parallel Nsight for Compute

Parallel Debugger
Debug compute kernels directly on GPU hardware
Examine thousands of threads executing in parallel in Visual Studio
Use conditional breakpoints to correct errors in massively parallel code

System Analysis
Capture CPU and GPU level events on a single correlated timeline
Timeline inspection tools allow for the examination of workload dependencies
Profile CUDA kernels using GPU performance counters
Misc

- **NVIDIA Management Library (NVML)**
  - cross platform C shared library
  - provides access to the query and management functionality
  - Management/monitoring applications can link against this library
  - nvidia-smi continues to exist

- **CUDA Profiling Tools Interface (CUPTI)**
  - enables the creation of profiling and tracing tools
  - Callback API and event API.
CUDA Registered Developer Program

All GPGPU developers should become NVIDIA Registered Developers

Benefits include:

- Early Access to Pre-Release Software
  - Beta software and libraries
- Submit & Track Issues and Bugs
  - Interact directly with NVIDIA QA engineers
- New benefits in 2011
  - Exclusive Q&A Webinars with NVIDIA Engineering
  - Exclusive deep dive CUDA training webinars
  - In-depth engineering presentations on pre-release software

Sign up Now: www.nvidia.com/ParallelDeveloper
CUDA 4.0

ZKI AK Supercomputing, May, 2011
Axel Koehler- NVIDIA