Kepler GPU Architecture and Benefits to Earth System Modeling

Peter Messmer  |  Developer Technology Group  |  pmessmer@nvidia.com
Stan Posey  |  HPC Industry and Applications  |  sposey@nvidia.com
<table>
<thead>
<tr>
<th></th>
<th>2011</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAM-SE</td>
<td>COSMO</td>
<td>CAM-SE</td>
</tr>
<tr>
<td>COSMO</td>
<td>GEOS-5</td>
<td>COSMO</td>
</tr>
<tr>
<td>GEOS-5</td>
<td>HOMME</td>
<td>GEOS-5</td>
</tr>
<tr>
<td>HOMME</td>
<td>ICON</td>
<td>NIM</td>
</tr>
<tr>
<td>ICON</td>
<td>NIM</td>
<td>NIM</td>
</tr>
<tr>
<td>NIM</td>
<td>gNEMO</td>
<td>WRF</td>
</tr>
<tr>
<td>gNEMO</td>
<td>WRF</td>
<td>WRF</td>
</tr>
<tr>
<td>WRF</td>
<td>RRTM</td>
<td>KernelGen</td>
</tr>
<tr>
<td>RRTM</td>
<td></td>
<td>OpenACC</td>
</tr>
</tbody>
</table>

### PU Progress Reported at This Workshop

<table>
<thead>
<tr>
<th>2011</th>
<th>2012</th>
<th>Other GPU Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAM-SE</td>
<td>CAM-SE</td>
<td>Model</td>
</tr>
<tr>
<td>COSMO</td>
<td>COSMO</td>
<td>ASUCA</td>
</tr>
<tr>
<td>GEOS-5</td>
<td>GEOS-5</td>
<td>CFSv2/GFS</td>
</tr>
<tr>
<td>HOMME</td>
<td>NIM</td>
<td>GALES</td>
</tr>
<tr>
<td>ICON</td>
<td>NIM</td>
<td>GRAPES</td>
</tr>
<tr>
<td>NIM</td>
<td>NIM</td>
<td>HARMONIE</td>
</tr>
<tr>
<td>gNEMO</td>
<td>NIM</td>
<td>HIRLAM</td>
</tr>
<tr>
<td>WRF</td>
<td>WRF</td>
<td>NEMO</td>
</tr>
<tr>
<td>- RRTM</td>
<td>- KernelGen</td>
<td>- OpenACC</td>
</tr>
<tr>
<td>- OpenACC</td>
<td>- OpenACC</td>
<td></td>
</tr>
</tbody>
</table>

VIDIA Motivation and Strategy

Motivation

- Financial – GPUs a good fit for computations of Earth system modeling
  - IDC: dedicated climate/NWP systems $852M in 2011; Drives other very large systems
  - GPUs in-line with trend of higher resolution modeling with manageable compute costs
- Challenges of climate research will drive continuous HPC innovation
  - Benefits derived for other HPC domains; Improves vendor product quality and reliability

Strategy

- Alliances to develop Fortran-based GPU compilers and tools
  - Commercial vendors PGI, CAPS, and Cray; OpenACC membership; Research initiatives
- Customer collaborations in applications engineering
  - NVIDIA technical contributions to CAM-SE/HOMME, COSMO, WRF, and NEMO
- GPU integration with systems from all major vendors
  - IBM, Cray, HP, SGI and many others; Kepler based-systems available during 1Q 2013
  - Technical collaboration on large system projects such Titan/ORNL, TSUBAME/Titech, etc.
AM-SE a Critical Application for Titan at ORNL
World’s Largest Open Science Computing Research Facility

~14,500 NVIDIA Tesla GPUs
20+ PetaFlops
~90% of flops from GPUs

CAM-SE
Model global climate change & explore mitigation strategies
Kepler GK110/CUDA 5 Highlights

- SMX
  - New instructions for programmability

- CUDA Dynamic Parallelism (CDP)
  - More responsibility for GPU

- Hyper-Q
  - Improved GPU utilization
## SMX Balance of Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Kepler GK110 vs Fermi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating point throughput</td>
<td>2-3x</td>
</tr>
<tr>
<td>Max Blocks per SMX</td>
<td>2x</td>
</tr>
<tr>
<td>Max Threads per SMX</td>
<td>1.3x</td>
</tr>
<tr>
<td>Register File Bandwidth</td>
<td>2x</td>
</tr>
<tr>
<td>Register File Capacity</td>
<td>2x</td>
</tr>
<tr>
<td>Shared Memory Bandwidth</td>
<td>2x</td>
</tr>
<tr>
<td>Shared Memory Capacity</td>
<td>1x</td>
</tr>
</tbody>
</table>
New High-Performance SMX Instructions

SHFL (shuffle) -- Intra-warp data exchange

ATOM -- Broader functionality, Faster

Compiler-generated, high performance instructions:
- bit shift
- bit rotate
- fp32 division
- read-only cache
Improving Programmability

- Library Calls from Kernels
- Simplify CPU/GPU Divide
- Batching to Help Fill GPU
- Dynamic Load Balancing
- Data-Dependent Execution
- Recursive Parallel Algorithms

Dynamic Parallelism

Programmability

Occupancy

Execution
What Does “Dynamic Parallelism” Mean?

GPU as Co-Processor

Autonomous, Dynamic Parallelism
CDP Simplifies Complex Execution Models

```c
int main() {
    float *data;
    setup(data);

    A <<< ... >>> (data);
    B <<< ... >>> (data);
    C <<< ... >>> (data);

    cudaThreadSynchronize();
    return 0;
}

__global__ void B(float *data) {
    do_stuff(data);

    X <<< ..., stream >>> (data);
    Y <<< ..., stream >>> (data);
    Z <<< ..., stream >>> (data);
    cudaStreamSynchronize(stream);

    do_more_stuff(data);
}
```
Fermi Concurrency

Fermi allows 16-way concurrency

- Up to 16 grids can run at once
- But CUDA streams multiplex into a single queue
- Overlap only at stream edges
Kepler Improved Concurrency

Kepler allows 32-way concurrency
- One work queue per stream
- Concurrency at full-stream level
- No inter-stream dependencies
Hyper-Q Helps to Better Utilize the GPU

- Streams from multiple CPU processes can execute concurrently
- Use as many MPI ranks as in CPU-only case
  
  \(\Rightarrow\) smaller impact of CPU work
- Particularly interesting for strong scaling
Thank You, Questions?

Peter Messmer | Developer Technology Group | pmessmer@nvidia.com
Stan Posey | HPC Industry and Applications | sposey@nvidia.com
New Instruction: SHFL

Data exchange between threads within a warp

- Avoids use of shared memory
- One 32-bit value per exchange
- 4 variants:

  - `__shfl()`
  - `__shfl_up()`
  - `__shfl_down()`
  - `__shfl_xor()`
**Improved: ATOM instructions**

- **Added int64 functions to match existing int32**

<table>
<thead>
<tr>
<th>Atom Op</th>
<th>int32</th>
<th>int64</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>cas</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>exch</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>min/max</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>and/or/xor</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- **2 – 10x performance gains**
  - Shorter processing pipeline
  - More atomic processors
  - Slowest 10x faster
  - Fastest 2x faster

- **Atomics are now fast enough to use within inner loop**
Texture Cache Unlocked

- Added a new path for compute
  - Avoids the texture unit
  - Allows a global address to be fetched and cached
  - Eliminates texture setup
- Why use it?
  - Separate pipeline from shared/L1
  - Highest miss bandwidth
  - Flexible, e.g. unaligned accesses
  - Managed automatically by compiler
    - “const __restrict” indicates eligibility
Grid Management Unit

**Stream Queue Mgmt**

- C
- B
- A
- R
- Q
- P
- Z
- Y
- X

**Work Distributor**

- 16 active grids

**Fermi**

- SM
- SM
- SM
- SM

**CUDA Generated Work**

- SMX
- SMX
- SMX
- SMX

**Grid Management Unit**

- Pending & Suspended Grids
  - 1000s of pending grids

**CUDA Kepler GK110**

- SMX
- SMX
- SMX
- SMX

**Work Distributor**

- 32 active grids