Heterogeneous Computing in Charm++

Michael Robson
GPU Overview

- Programmed with CUDA
- 1,000s of threads
- 100s GB/s bandwidth
- ~16 GB of memory
- TeraFLOPS double precision performance
GPU MANAGER
GPU Manager

• **GPU task Management** library
• Register kernel for asynchronous invocation
• Automates data movement
• Overlap kernel execution and data transfer
• Pre-allocated pool of pinned memory
• Runtime profiling integration (Projections)
Using GPU Manager

• Build charm with cuda
• Enqueue work request:
  – Describe buffers
  – Callback(s)
  – Run kernel function
* The user can also get pre-pinned memory from mempool. In this case, there is no need to wait for callback.

Expensive but necessary since having too much pinned memory degrades system performance

User-defined Callback

Handles workRequests

gpuProgressFn()
Handling workRequests

- **3 streams** for data transfer & computation overlap:
  - `data_in_stream`
  - `kernel_stream`
  - `data_out_stream`
- Handled via `gpuProgressFn()`, called periodically by scheduler
Handling workRequests

State

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>Ti</td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
</tr>
</tbody>
</table>

* Q: Queued, Ti: Transfer-In, EX: Executing, TO: Transfer-Out

Not good!

Could also have started [3] cudaMemcpyAsync (if no cudaMemcpy necessary for [3])

Could have started [2] kernel, because of concurrent kernel execution
New Design

• Avoid polling via stream callbacks
  – Added new callback locations
  – Supported in CUDA 5.0 (K20)

• Enable concurrent execution
  – Spawns max streams, one per workRequest
  – Supported in compute capability 2.x (Tesla)
NVTX Concurrent Async Old
NVTX Concurrent Async New

4/18/17 Charm++ Workshop 2017
OpenAtom

• Ab-inito molecular dynamics
• Offloads forward and backwards path of pair calculator
• New GPU target
OpenAtom Performance
OpenAtom Performance

Time In Microseconds

18,440,000 18,840,000 19,240,000 19,640,000 20,040,000

PE 0
PE 1
PE 2
PE 3
PE 4
PE 5
PE 6
PE 7
PE 8
PE 9
PE 10
PE 11
PE 12
PE 13
PE 14
PE 15
OpenAtom

- Cosmological N-body simulations
- Leverages GPU Manager
- Offloads gravity kernels
- Active work in optimization
ACCEL FRAMEWORK
Motivation

• Exploit runtime info. for dynamic execution
• Runtime (RTS) can map to various platforms
• RTS can proactively move needed data
• Don’t leave hardware sitting idle
Accel Framework

• Generate code from tagged entry methods
  – Host (CPU) and device (CUDA)
  – Extend with tuning keywords
  – Annotate object data access

• Builds on GPU manager

• Batch fine grained kernel launches
Example Code

```c
entry - trigger - -- void foo()
[
  readonly : float matrix [SIZE]<implobj->matrix>,
  writeonly : float matTmp [SIZE]<implobj->matrixTmp> ] {
    // implementation here
  }
}
```

4/18/17 Charm++ Workshop 2017
Benchmarks

• stencil2d (aka jacobi)
  – weighted five point stencil
  – CPU friendly

• md
  – molecular dynamics
  – GPU friendly

• Stampede
  – 1 GPU node (K20 + Xeon E5-2680)
  – 16 cores/processing elements
Stencil 2D
Molecular Dynamics
Analysis

<table>
<thead>
<tr>
<th>Code</th>
<th>% Dev.</th>
<th>Host</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>stencil</td>
<td>30%</td>
<td>1.58x</td>
<td>3.09x</td>
</tr>
<tr>
<td>md</td>
<td>65%</td>
<td>3.02x</td>
<td>1.46x</td>
</tr>
</tbody>
</table>

- Using hardware improves performance
- Even when it’s not ideal
FUTURE WORK
Future Work

• OpenCL and other language support
• Stream priorities
• Ongoing unified memory experiments
• Heterogenous multi-node load balancing
Michael Robson
mprrobson@illinois.edu
QUESTIONS?
BACKUP SLIDES
[accel] Framework Usage

• modifiers:
  – read-only, write-only, read-write
  – shared – one copy per batch
  – persist – resident in device memory

• parameters:
  – triggered – one invocation per chare in array
  – splittable (int) – AEM does part of work
  – threadsPerBlock (int) – specify block size