High-Performance Scientific Computing
Lecture 5: More OpenCL, MPI

MATH-GA 2011 / CSCI-GA 2945 · October 3, 2012
Today

Tool of the day: Git

OpenCL: Device Language

OpenCL: Synchronization

Intro to MPI
Bits and pieces

- HW1 grades sent
- HW2 graded soon
- HW3 due
- HW4 out tomorrow
- Cuda cluster accounts
- Mailing list messages

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
Outline

Tool of the day: Git

OpenCL: Device Language

OpenCL: Synchronization

Intro to MPI
Demo time
Tool of the day: Git

OpenCL: Device Language

OpenCL: Synchronization

Intro to MPI
Moar OpenCL!

Demo time
OpenCL device language is C99, with these differences:

- Index getters
- Memory space qualifiers
- Vector data types
- Many generic ('overloaded') math functions
- Synchronization
- Recursion
- Fine-grained `malloc()`
- Function pointers
Address Space Qualifiers

<table>
<thead>
<tr>
<th>Type</th>
<th>Per</th>
<th>“Speed”</th>
</tr>
</thead>
<tbody>
<tr>
<td>private*)</td>
<td>work item</td>
<td>super-fast</td>
</tr>
<tr>
<td>local</td>
<td>group</td>
<td>fast</td>
</tr>
<tr>
<td>global</td>
<td>grid</td>
<td>kinda slow</td>
</tr>
</tbody>
</table>

*) default, so optional

Should really discuss “speed” in terms of latency/bandwidth. Both decrease with distance from the point of execution.

Tool of the day: Git  
OpenCL: Device Language  
OpenCL: Synchronization  
Intro to MPI
# Address Space Qualifiers

<table>
<thead>
<tr>
<th>Type</th>
<th>Per</th>
<th>“Speed”</th>
</tr>
</thead>
<tbody>
<tr>
<td>private*)</td>
<td>work item</td>
<td>super-fast</td>
</tr>
<tr>
<td>local</td>
<td>group</td>
<td>fast</td>
</tr>
<tr>
<td>global</td>
<td>grid</td>
<td>kinda slow</td>
</tr>
</tbody>
</table>

*) default, so optional

Should really discuss “speed” in terms of latency/bandwidth.

*Both* decrease with distance from the point of execution.

---

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
Tool of the day: Git

OpenCL: Device Language

OpenCL: Synchronization

Intro to MPI
Concurrency and Synchronization

GPUs have layers of concurrency.
Each layer has its synchronization primitives.
GPUs have layers of concurrency.
Each layer has its synchronization primitives.

- **Intra-group:**
  - `barrier(...)`,
  - `mem_fence(...)`

- **Inter-group:**
  - Kernel launch

- **CPU-GPU:**
  - Command queues
Synchronization

What is a Barrier?
What is a Barrier?
What is a Barrier?
What is a Barrier?
Synchronization

What is a Barrier?

![Diagram of a barrier](image-url)
What is a Barrier?
Synchronization

What is a Barrier?
Synchronization

What is a Memory Fence?
Synchronization

What is a Memory Fence?

write 18

17
What is a Memory Fence?
What is a Memory Fence?
What is a Memory Fence?

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
Synchronization

What is a Memory Fence?

write 18

18
Synchronization

What is a Memory Fence?
What is a Memory Fence? An ordering restriction for memory access.
What is a Memory Fence? An ordering restriction for memory access.
What is a Memory Fence? An ordering restriction for memory access.
What is a Memory Fence? An ordering restriction for memory access.
What is a Memory Fence? An ordering restriction for memory access.
Synchronisation

What is a Memory Fence? An ordering restriction for memory access.
What is a Memory Fence? An ordering restriction for memory access.
What is a Memory Fence? An ordering restriction for memory access.
Golden Rule:
Results of the algorithm must be independent of the order in which work groups are executed.
Synchronization between Groups

Golden Rule:
Results of the algorithm must be independent of the order in which work groups are executed.

Consequences:

• Work groups may read the same information from global memory.
• But: Two work groups may not validly write different things to the same global memory.
• Kernel launch serves as
  • Global barrier
  • Global memory fence
Atomic Operations

Collaborative (inter-block) Global Memory Update:

Read → Increment → Write

Interruptible!

Atomic Global Memory Update:

Read → Increment → Write

Protected

How?

atomic

{add, inc, cmpxchg, ...} (int *global, int value);
Atomic Operations

Collaborative (inter-block) Global Memory Update:

Read → Increment → Write

Interruptible!

```
atomic {
    add, inc, cmpxchg, ... 
}(int *global, int value);
```

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
Atomic Operations

Collaborative (inter-block) Global Memory Update:

![Diagram showing the process of collaborative global memory update: Read → Increment → Write with interruptible steps]

Interruptible!

How?

atomic {
    add, inc, cmpxchg, ... 
}(int *global, int value);
Atomic Operations

Collaborative (inter-block) Global Memory Update:

Atomic Global Memory Update:

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
Atomic Operations

Collaborative (inter-block) Global Memory Update:

Read → Increment → Write
Interruptible! → Interruptible!

Atomic Global Memory Update:

Read → Increment → Write
Protected
Atomic Operations

Collaborative (inter-block) Global Memory Update:

Atomic Global Memory Update:
Atomic Operations

Collaborative (inter-block) Global Memory Update:

Read → Increment → Write

Interruptible! → Interruptible!

Atomic Global Memory Update:

Read → Increment → Write

Protected → Protected

How?

atomic_{add,inc,cmpxchg,...}(int *global, int value);

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
**Atomic: Compare-and-swap**

```c
int atomic_cmpxchg(__global int *p, int cmp, int val)
int atomic_cmpxchg(__local int *p, int cmp, int val)
```

**Does:**

- Read the 32-bit value (referred to as old) stored at location pointed by `p`.
- Compute `(old == cmp) ? val : old`.
- Store result at location pointed by `p`.
- Returns `old`.
Atomic: Compare-and-swap

```c
int atomic_cmpxchg (__global int *p, int cmp, int val)
int atomic_cmpxchg (__local int *p, int cmp, int val)
```

Does:

- Read the 32-bit value (referred to as old) stored at location pointed by p.
- Compute \((\text{old} == \text{cmp}) \text{ ? val : old}\).
- Store result at location pointed by p.
- Returns old.

Implement atomic \texttt{float add}?
Outline

Tool of the day: Git

OpenCL: Device Language

OpenCL: Synchronization

Intro to MPI
Message Passing Interface:
MPI

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
MPI

Key questions:
• Who can I send mail to?
• How much mail can go through the system (bandwidth)?
• How fast does mail arrive (latency)?
• Should I wait for the return receipt?

Tool of the day: Git  
OpenCL: Device Language  
OpenCL: Synchronization  
Intro to MPI
Not enough throughput? Just buy more computers*
Not enough throughput? Just buy more computers*
Not enough throughput? Just buy more computers*

Key questions:
- Who can I send mail to?
- How much mail can go through the system (bandwidth)?
- How fast does mail arrive (latency)?
- Should I wait for the return receipt?

Tool of the day: Git  OpenCL: Device Language  OpenCL: Synchronization  Intro to MPI
Not enough throughput? Just buy more computers*

Key questions:
- Who can I send mail to?
- How much mail can go through the system (bandwidth)?
- How fast does mail arrive (latency)?
- Should I wait for the return receipt?
- Why haven’t I heard from the other guys yet?
MPI 3.0
Born September 21, 2012

MPI 1.0: June 1994
Demo time
Questions?
Image Credits

- Onions: flickr.com/darwinbell cc
- Server: sxc.hu/Kolobsek
- Envelope: sxc.hu/ilco
- Gift box: sxc.hu/iprole