Active Global Address Space (AGAS): Global Virtual Memory for Dynamic Adaptive Many-Tasking (AMT) Runtimes

Dissertation Research Showcase Presentation

Abhishek Kulkarni
Center for Research in Extreme Scale Technologies
Indiana University
November 19, 2015
HPC Programming / Execution models

- **Message Passing**
  - BSP: Compute-Communicate paradigm
  - Data divided statically among available processors

- **Asynchronous Many-tasking (AMT)**
  - Lightweight tasks concurrently operating on global data
  - Well-suited for dynamic and adaptive execution
Asynchronous Many-Tasking (AMT) models

- Static/Dynamic task and dataflow graph
- Lightweight threads
- Inter-thread synchronization
- Active Messages
- Global Address Space
ParalleX Execution Model

- **Lightweight multi-threading**
  - Divides work into smaller tasks
  - Increases concurrency
- **Message-driven computation**
  - Move work to data
  - Keeps work local, stops blocking
- **Constraint-based synchronization**
  - Declarative criteria for work
  - Event driven
  - Eliminates global barriers
- **Data-directed execution**
  - Merger of flow control and data structure
- **Shared name space**
  - Global address space
  - Simplifies random gathers
The HPX runtime system reifies the ParalleX execution model:
- Localities
- Lightweight Threads
- Processes
- Local Control Objects (LCOs)
- Parcels (Active Messages)
- Active Global Address Space (AGAS)

Fine-grained concurrency
- Tasks, Interrupts, Lightweight Threads

Lightweight, globally-addressable synchronization objects
- Eg: Futures (IVars), Monotonic counters (Gate), Reductions

http://hpx.crest.iu.edu/
Global Address Space in HPX-5

- Global memory is globally addressable
- Supports local, cyclic, blocked and user-defined distributions
- Two modes of operations:
  - Static (PGAS)
  - Active (AGAS)
- Asynchronous access to global memory
- Memory must be pinned locally prior to access
- GAS attributes, atomics, collectives
<table>
<thead>
<tr>
<th>Partitioned Global Address Space</th>
<th>Active Global Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Global Physical Memory</td>
<td>• Global Virtual Memory</td>
</tr>
<tr>
<td>• Physical location encoded in the address</td>
<td>• Physical location maintained separately</td>
</tr>
<tr>
<td>• Memory space same as address space</td>
<td>• Memory space distinct from address space</td>
</tr>
<tr>
<td>• Faster address translation</td>
<td>• Address translation potentially expensive</td>
</tr>
<tr>
<td>• Traditional PGAS supports put/get/atomics</td>
<td>• Support arbitrary AMs</td>
</tr>
<tr>
<td>• Static</td>
<td>• Dynamic (allows migration)</td>
</tr>
</tbody>
</table>
Load and Data Imbalance in Graph Algorithms

Move the smallest partition to:
- Successor (agas-succ)
- Predecessor (agas-pred)
- Random (agas-random)
- Least Loaded (agas-ll)
AGAS Features

• User-defined data distributions
  – Existing PGAS models support cyclic, block-cyclic
  – Similar to Chapel, ZPL
  – Higher-order and composable

• Data Migration
  – Explicit moving of individual chunks
  – Rebalancing of global data
  – Implicit runtime-managed remapping based on introspection

• Co-data Actions
  – Relational dependencies between tasks and data
  – Runtime chooses between a) moving data to computation or b) moving computation to data

• Caching, Data stealing, speculative migration
Software AGAS

- Flat, byte-addressable global address space
- Chunk size limited to 4TB, Nodes limited to 64K
- Scalable memory allocators handle local/cyclic segments
  - jemalloc, TBBMalloc
- Chunk table maps LVA → GVA
- CTT (Chunk Translation Table) maps GVA → LVA
- CTT entry:
  
<table>
<thead>
<tr>
<th>count</th>
<th>owner</th>
<th>lva</th>
<th>blocks</th>
<th>on_unpin</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>192</td>
<td>256</td>
<td>288</td>
<td>0110100100101001010</td>
<td>101011110000001010</td>
</tr>
</tbody>
</table>

HPX-5 AGAS Allocator

OS Virtual Memory

Local Chunk Allocator

Chunk Table

CTT

Scalable Global Memory Allocator (Local)

Scalable Global Memory Allocator (Cyclic)

Cyclic Chunk Allocator

Local

Memory Allocator

Scalable Global

Memory Allocator

(Cyclic)

Source

Figure 5.5: A CTT entry.

Figure 5.6: AGAS Move
Cyclic Allocation

- Symmetric cyclic segment/heap
- Can be segmented to speed up cyclic allocation
  - At the expense of limiting allocation size
Move

- When a chunk moves, its owner is updated
- Move protocol
  - Overlaps routing updates and data transfer
- Home always “knows” where a chunk is
- Current implementation uses a two-sided transport
  - Relies on being able to “resend” a parcel
- Complex interactions between move, unpin and delete
Hardware AGAS

- Network-managed global address space
- CTT implemented by TCAM in hardware
- The network “knows” the location of chunks
- Pros:
  - Message optimal in terms of hops required for routing
- Cons:
  - Centralized directory
  - Existing hardware limitations
Hardware AGAS

- Proof of concept based on the GASNet runtime system
- Uses SDN for routing table management
- Uses IB RD multicast
- IB multicast GIDs are mapped to multicast Ethernet MAC addresses
- Put performance:
  - High switching latency
  - Photon conduit overheads

---

### Time taken (us)

<table>
<thead>
<tr>
<th>Message Size (bytes)</th>
<th>software (IBV)</th>
<th>software (Photon)</th>
<th>hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

1. Ranks 0, 1 and 2 locally attach to multicast addresses based on their page id:
   - 0 → ff0e::ffff::1
   - 1 → ff0e::ffff::2
   - 2 → ff0e::ffff::3

2. Ranks 0, 1, 2 push OpenFlow mods:
   - 33:33:00:00:00:00:01 → port1
   - 33:33:00:00:00:00:02 → port2
   - 33:33:00:00:00:00:03 → port3

3. Endpoints attach multicast addresses for all block-ids (ff0e::ffff:<bid>). The switch does forwarding based on the destination MAC address associated with the block id (33:33:<bid>).
Comparing Hardware and Software AGAS

• Bounded software cache
  – Cache-replacement policies (random, LRU)
  – Evictions cause cache misses and degrade GAS performance

• Thread contention in concurrent cache
  – Concurrent cache is faster than a single-threaded cache but 25% slower than the hardware implementation
Remap performance in HW AGAS

- **GUPS microbenchmark**
  - table size 2048 words, 512 words/page, 16 nodes (192 cores)
  - 4 million random updates were performed

- As page movement frequency increases, SW becomes more expensive than HW
Conclusions

• The HPX-5 runtime system provides dynamic, adaptive features necessary for execution of large-scale irregular applications.

• Global virtual memory using an active global address space (AGAS) enables the runtime to manage both locality and load-imbalance concerns.

• Network-managed AGAS shows promise at smaller scales but requires support from vendors and supercomputing centers to be viable at larger scales.
Questions?