Lightweight Communication Interface
A Communication Runtime for Asynchronous Task Systems

Omri Mor, Hoang-Vu Dang, Marc Snir

University of Illinois at Urbana–Champaign

April 17, 2019
Bulk Synchronous Parallel

begin
node 1
node 2
⋯
node n
sync
node 1
node 2
⋯
node n
⋯
node 1
node 2
⋯
node n
end
Asynchronous Task Systems

begin
  task 1
  task 2
  task 5
  task 3
  task 4
  task 7
  task 6
end
Asynchronous Task Systems

- Addresses major issues:
  - Load balancing
  - Execution time variability
  - Communication time variability
  - Hardware performance variability
Asynchronous Task Systems

- Addresses major issues:
  - Load balancing
  - Execution time variability
  - Communication time variability
  - Hardware performance variability

- Examples:
  - PaRSEC (UTK)
  - Legion (Stanford)
  - StarPU (INRIA)
  - Others...
Communication in Asynchronous Task Systems

- Usually MPI or GASNet
Communication in Asynchronous Task Systems

- Usually MPI or GASNet
- Problems:
  - Implicit progress
  - Complex completion
  - Multiple queuing/polling levels
Communication in Asynchronous Task Systems

- Usually MPI or GASNet
- Problems:
  - Implicit progress
  - Complex completion
  - Multiple queuing/polling levels
  - Multithreaded performance

Bandwidth & NIC message injection rate increasing—performance limited by the communication runtime on the CPU
Communication in Asynchronous Task Systems

- Usually MPI or GASNet
- Problems:
  - Implicit progress
  - Complex completion
  - Multiple queuing/polling levels
  - Multithreaded performance
- Bandwidth & NIC message injection rate increasing—performance limited by the communication runtime on the CPU
Lightweight Communication Interface (LCI)

- Low-level communication runtime
- Consumed by language runtimes, frameworks, and libraries
  - Not typically used by application programmers
- Provide middleware developers as much direct control as possible
- Expose hardware performance and functionality with low overhead
Design Goals

- Use functionality that can be directly supported by NICs
- Do not expose higher-level functionality
- Directly support communication paradigms of targeted frameworks
- Direct control of communication progress and memory allocation
- Efficient concurrent communication for massive parallelism
- Direct producer-consumer communication
- Split communication into independent streams
Concepts and Primitives

- **Endpoint**: handle local communication requests and completion of remote messages (group/communicator)
- **Completion**: simple synchronizers, counters, queues, and handlers
- **Back pressure**: higher-level runtime can deal with resource exhaustion
- **Protocol selection**: point-to-point communication with eager or rendezvous protocols, depending on data size
- **Progress**: explicit progress calls per-device
Performance: Latency

Ping-Pong on Stampede2 SKX

- **MPI**
- **psm2**
- **LCI Direct**
- **LCI Buffered**
- **LCI Immediate**

Latency ($\mu$s) vs. Bytes

- **2**
- **2^1**
- **2^2**
- **2^3**
- **2^4**
- **2^5**
- **2^6**
- **2^7**
Performance: Latency

Ping-Pong on Stampede2 SKX

Latency (µs)

Bytes

MPI
psm2
LCI Direct
LCI Buffered

Omri Mor (U. of Illinois)
Lightweight Communication Interface
Performance: Latency

Ping-Pong on Stampede2 SKX

Latency ($\mu$s)

Bytes

MPI
psm2
LCI Direct

Omri Mor (U. of Illinois) Lightweight Communication Interface 9th JLESC Workshop
Performance: Bandwidth

Ping-Pong on Stampede2 SKX, 64 Bytes

Bandwidth (Mbit/s) vs Threads

- MPI
- LCI
- LCI (2-dev)
Performance: Bandwidth

Ping-Pong on Stampede2 SKX, 4096 Bytes

Bandwidth (Gbit/s)

Threads

- MPI
- LCI
- LCI (2-dev)
Open Questions and Collaboration

- Feedback on API design
Open Questions and Collaboration

- Feedback on API design
- Minimum overhead for a communication runtime?
Open Questions and Collaboration

- Feedback on API design
- Minimum overhead for a communication runtime?
- Current users and collaborators
  - D-Galois
  - GeminiGraph
  - PaRSEC (*in progress, collaboration with UTK*)
Open Questions and Collaboration

- Feedback on API design
- Minimum overhead for a communication runtime?
- Current users and collaborators
  - D-Galois
  - GeminiGraph
  - PaRSEC (in progress, collaboration with UTK)
- We welcome further collaborations!