# Lightweight Communication Interface

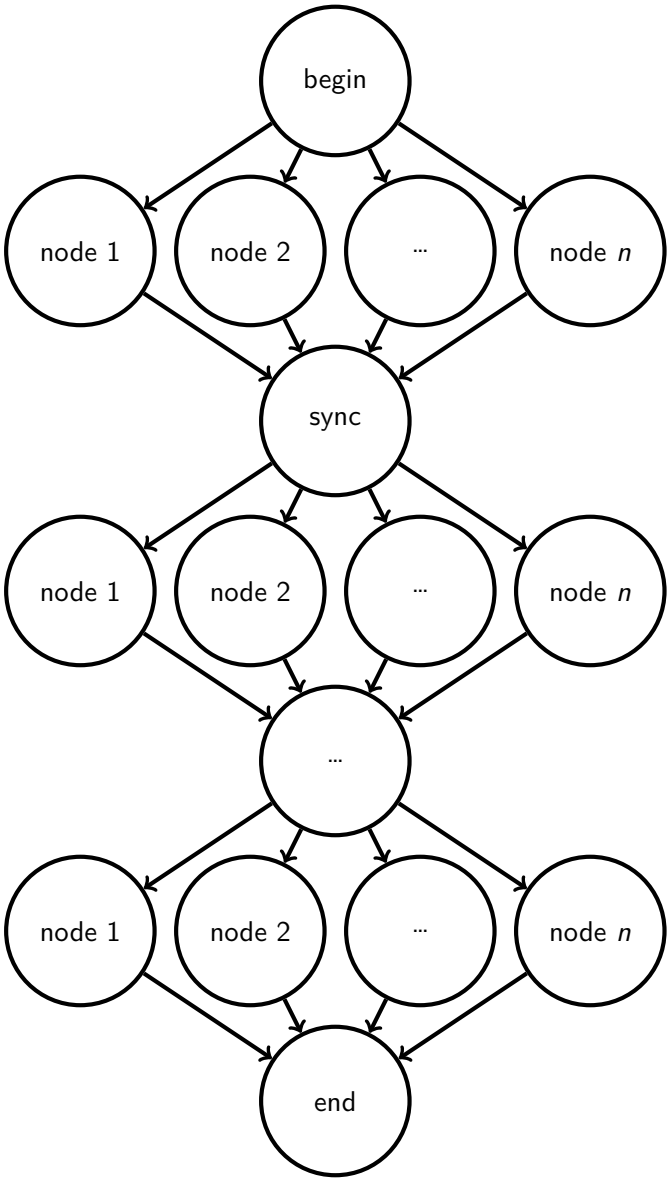
## Message Passing with Large-Scale Multithreading

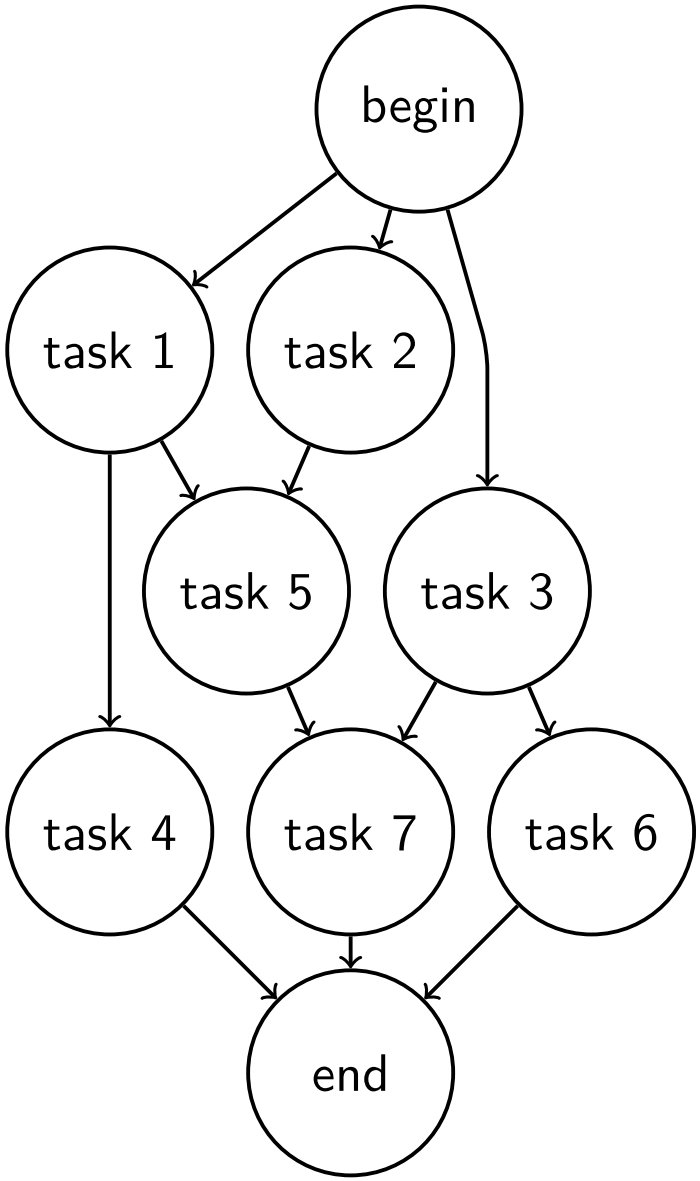
Omri Mor, Marc Snir

University of Illinois at Urbana–Champaign

September 8–10, 2020

# BSP versus Asynchronous Task Systems



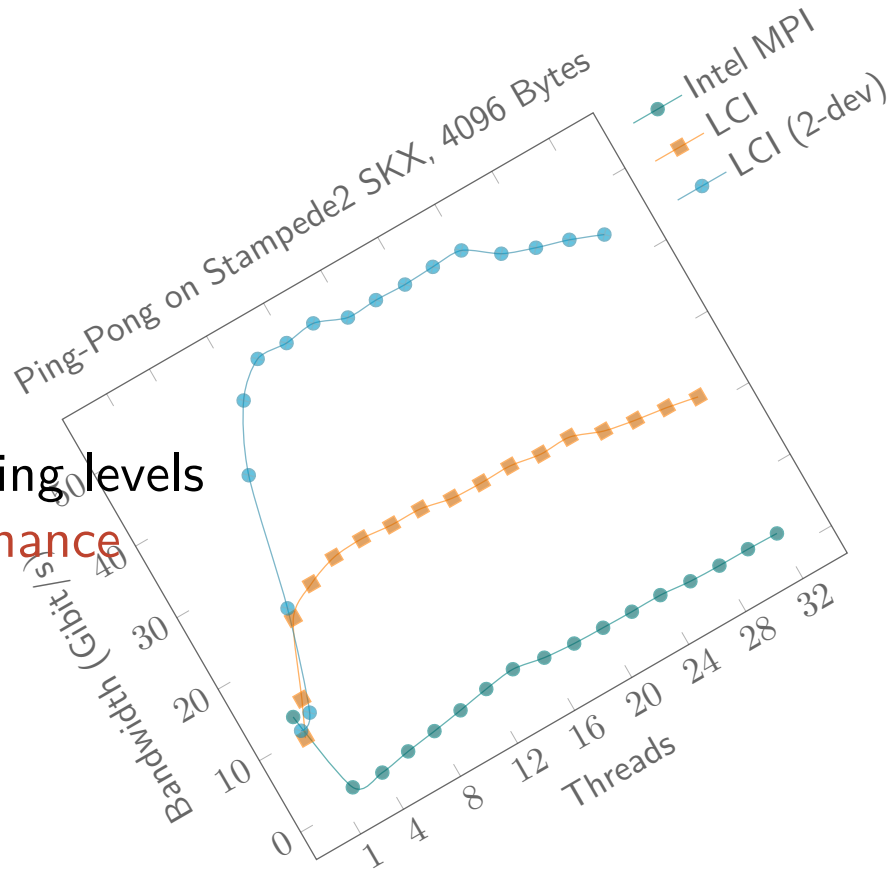



# Communication in Asynchronous Task Systems

- Usually MPI or GASNet

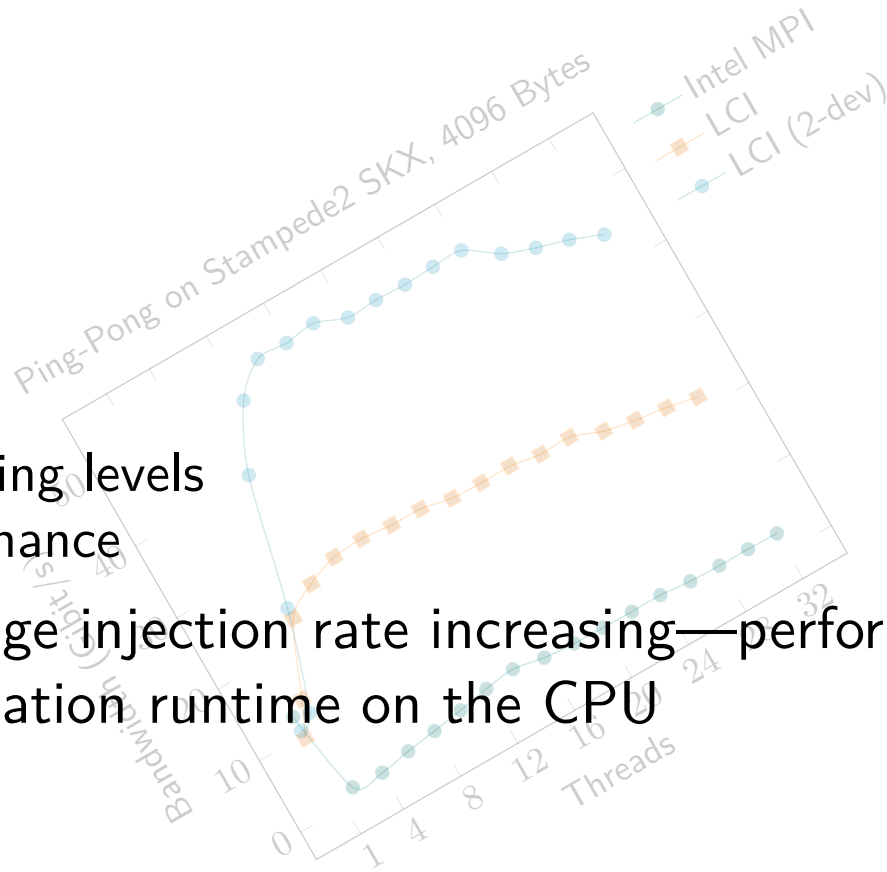
# Communication in Asynchronous Task Systems

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



# 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
- Allow for deep task scheduler integration

# PaRSEC

- Task-based runtime for distributed architectures
- Several DSLs & programming models
  - Parameterized Task Graph
  - Dynamic Task Discovery
  - Templated Task Graph
- New "Communication Engine" interface
  - Replaces hard dependency on MPI
  - One-sided active message paradigm (e.g. put & notify)
  - Communication thread for comm progress & callbacks
  - Message ordering unnecessary—scheduled by runtime

# PaRSEC/LCI Overview

- Separate progress thread ensures asynchronous progress
- "Progress" function executes callbacks from queue
- Handler completion for low-latency packet return to pool
  - Handler pushes metadata to callback queue if on progress thread
  - If not on progress thread, executes callback directly
  - Completion queue packet return semantics problematic

# PaRSEC/LCI Details

- Active Message
  - Send medium/buffered message with dynamic allocation at destination

# PaRSEC/LCI Details

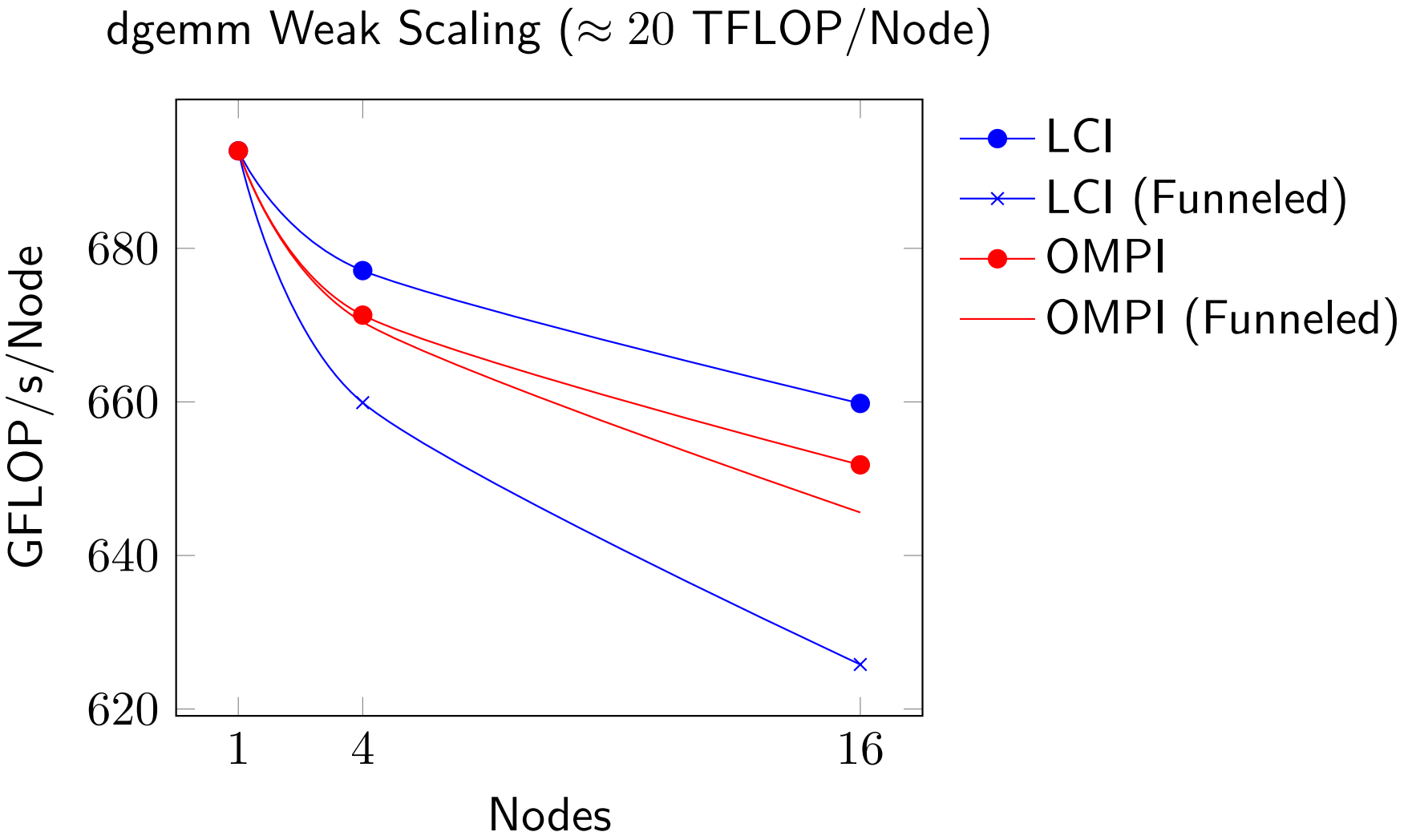
- Active Message
  - Send medium/buffered message with dynamic allocation at destination
- Put
  - Must send two buffers: data and metadata (including callback data)
  - Ready-to-send message with metadata & callback, like active message
  - RDMA with signal used to send data

# PaRSEC/LCI Details

- Active Message
  - Send medium/buffered message with dynamic allocation at destination
- Put
  - Must send two buffers: data and metadata (including callback data)
  - Ready-to-send message with metadata & callback, like active message
  - RDMA with signal used to send data
- Progress Thread
  1. Call `lc_progress` in loop until no progress done
  2. Push all callbacks from thread-private queue to shared queue
  3. Sleep (yield), if sharing hardware resources
  - Can support multiple NICs/devices with multiple threads

# PaRSEC/LCI Performance

## DPLASMA on Comet@SDSC



# Open Questions and Collaborations

- API design
  - Works well on multi/manycore CPUs
  - What design for GPU-initated communication?

# Open Questions and Collaborations

- API design
  - Works well on multi/manycore CPUs
  - What design for GPU-initated communication?
- Minimum overhead for a communication runtime?

# Open Questions and Collaborations

- API design
  - Works well on multi/manycore CPUs
  - What design for GPU-initated communication?
- Minimum overhead for a communication runtime?
- Where else is LCI useful?
  - Graph Analytics:
    - D-Galois
    - GeminiGraph
  - Asynchronous Task Systems:
    - PaRSEC (UTK)
  - Future work:
    - STAPL (U. of Illinois)
    - FMSolvr (Jülich)
  - Open to collaborations!