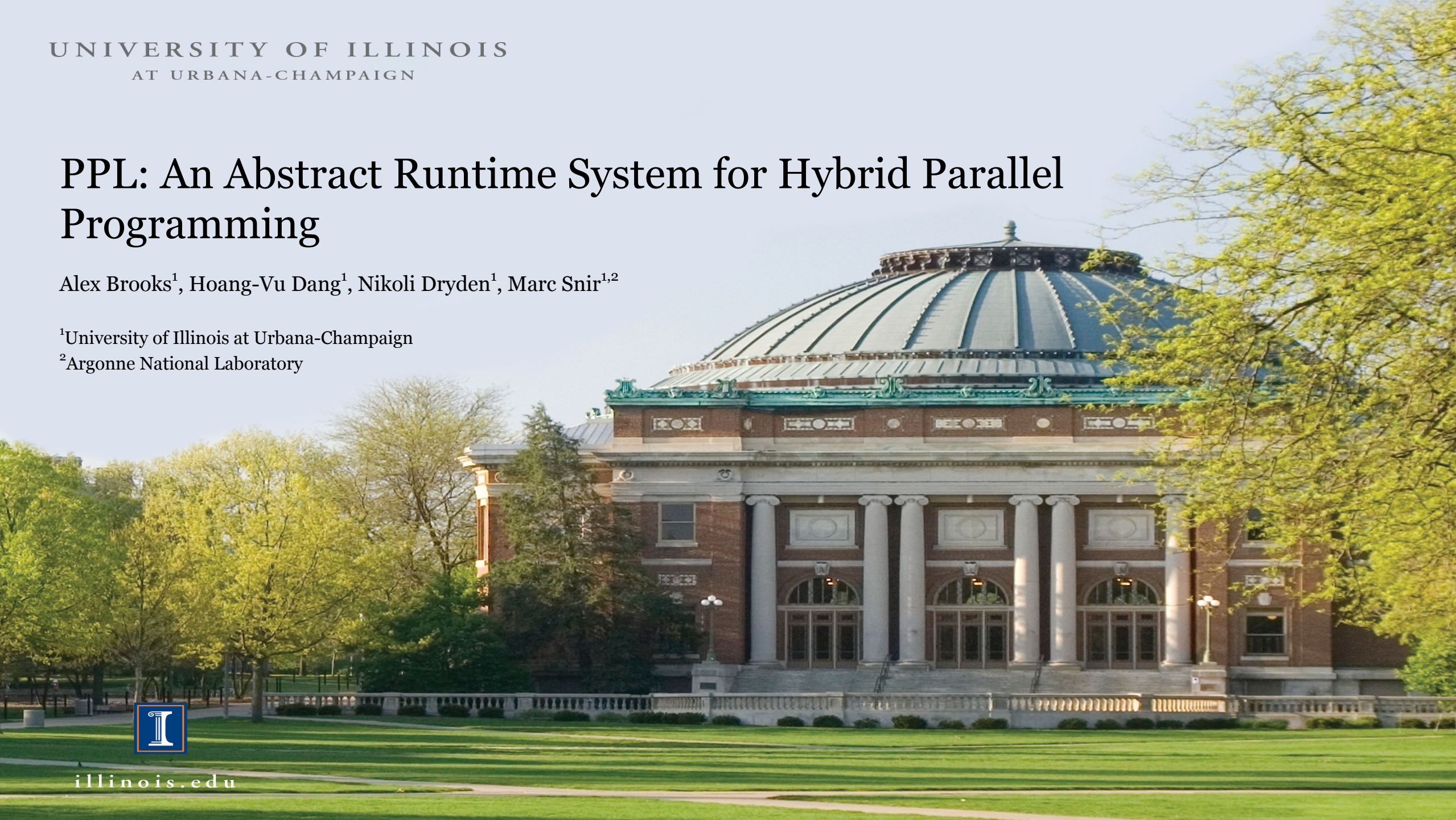
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

PPL: An Abstract Runtime System for Hybrid Parallel Programming

Alex Brooks[1], Hoang-Vu Dang[1], Nikoli Dryden[1], Marc Snir[1,2]

[1]University of Illinois at Urbana-Champaign
[2]Argonne National Laboratory



illinois.edu

Motivation

- Efficiency requires exploiting inter- and intra-node parallelism
  - Load balancing, latency hiding, communication management
- Requirements:
  - Lightweight threads (tasks)
  - RDMA
  - Support large numbers of communicating tasks
    - Close integration of threading and communication
  - Fine-grained remote synchronization
    - Efficient event-driven task scheduling

Outline

1. **Design and Implementation**
2. Runtime Evaluation
3. Application Evaluation
4. Future Work

PPL Design

- Modularized elements:
  - Communication Layer
  - Memory Management System
  - Threading Layer
- Implemented in C++11
- Provide necessary interaction between communication events and scheduling
- Modules are interoperable for minimizing runtime overhead
- **Result:** Flexible tool for analyzing & developing future runtime systems

Communication

- One-sided communication & active messages
  - Performed on global objects
- Two implementations
  - GASNet – general purpose
  - RDMAX – custom-built with InfiniBand Verbs API
- Dedicated communication threads
  - Offload work from communicating threads
  - 1 thread for communication, 1 for thread re-scheduling

Memory Management

- Partitioned Global Address Space (PGAS) Environment
  - Global and local heaps
- Global objects (`gptr`, `gvar`, and `gvec`)
  - Allocated on global heap, cached objects on local heap
- Software caching
  - Synergistic caching
  - No implicit coherence
  - Policies: `nocache`, `noevict`, `evict`
  - Different get operations specify interaction with the cache
    - `lget`, `rget`, `get`

Threading

- Support threads/futures which can yield
- Goals:
  - Flexible enough to use any underlying threading library
  - Support any threading model (tasks vs. kernel threads)
- Two implementations:
  - Argobots and Qthreads
- Primary synchronization primitive modeled as full/empty-bit semantics
  - Native in Qthreads
  - Custom-built atop Argobots

Outline

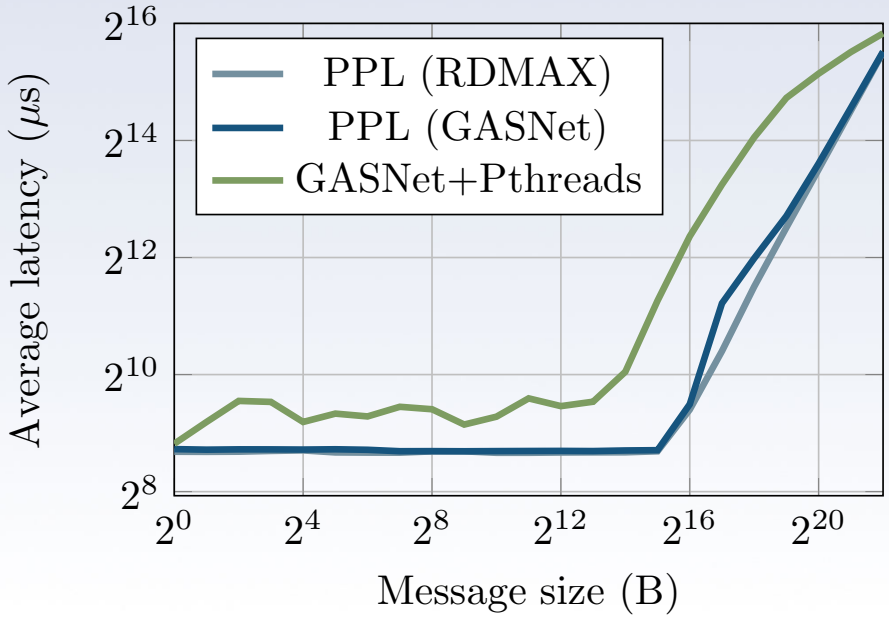
1. Design and Implementation
2. **Runtime Evaluation**
3. Application Evaluation
4. Future Work

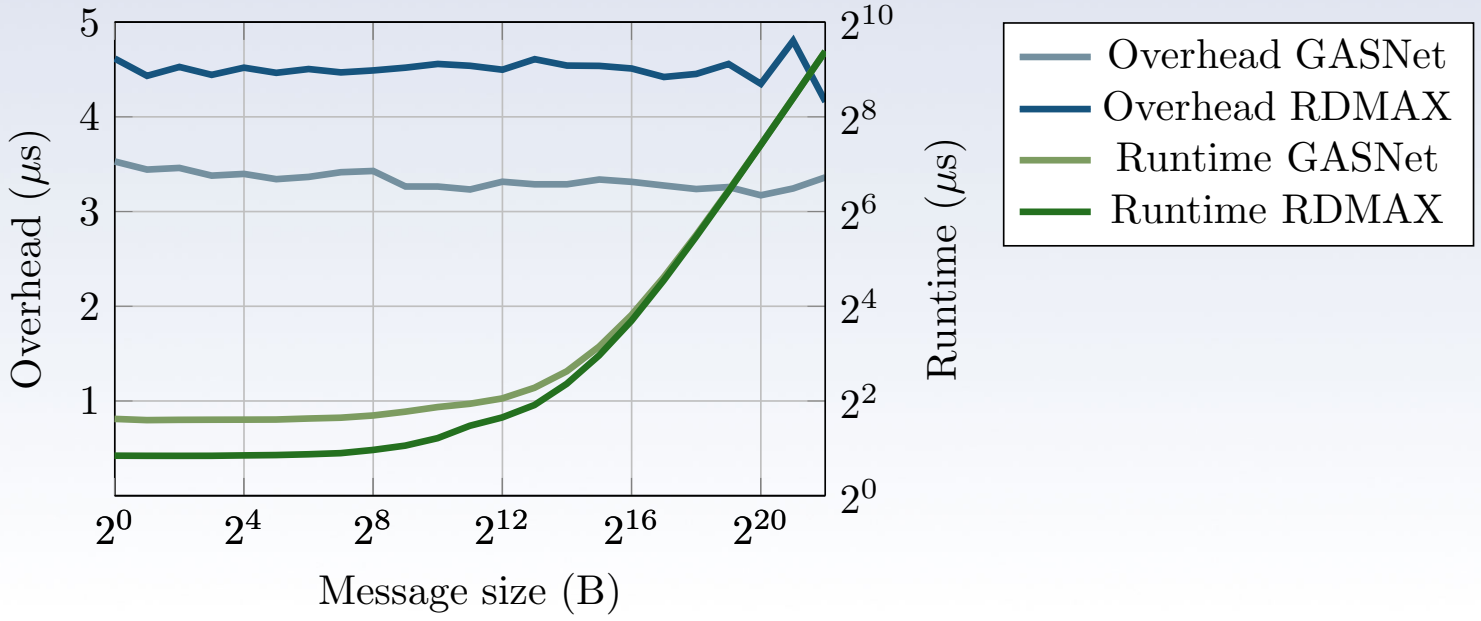
Experiment Details

- Tested on Stampede Supercomputer at TACC
  - Mellanox FDR Infiniband
  - 2 Intel Xeon E5-2680 (8 C)
  - 6,400 Nodes
- Use the following module implementations unless otherwise noted:
  - RDMAX, Qthreads, `noevict` cache

Communication Overhead

- Perform `gptr` get operations with 64 threads (left graph) & single thread (right graph)





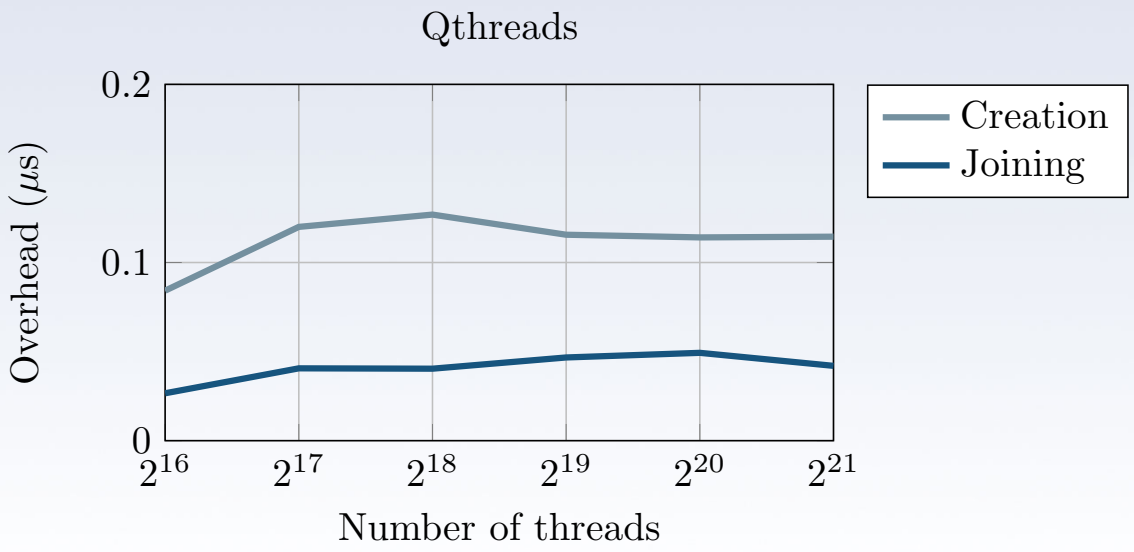
Caching Overhead

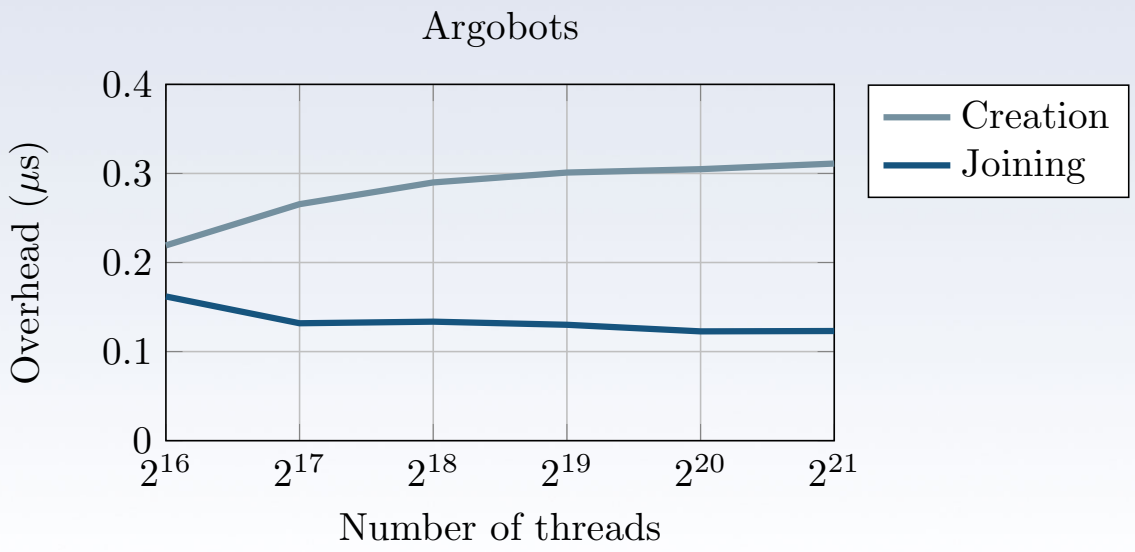
- Perform get operations with each of PPL's caching policies
  - Compared against strictly performing get operations

	nocache	noevict	evict
1 B	7.4 %	15.3 %	17.7 %
1 KB	8.3 %	6.6 %	14.8 %
1 MB	0.8 %	0.2 %	0.6 %

Threading Overhead

- Perform thread creation and joining
  - Overhead added by PPL to Qthreads & Argobots



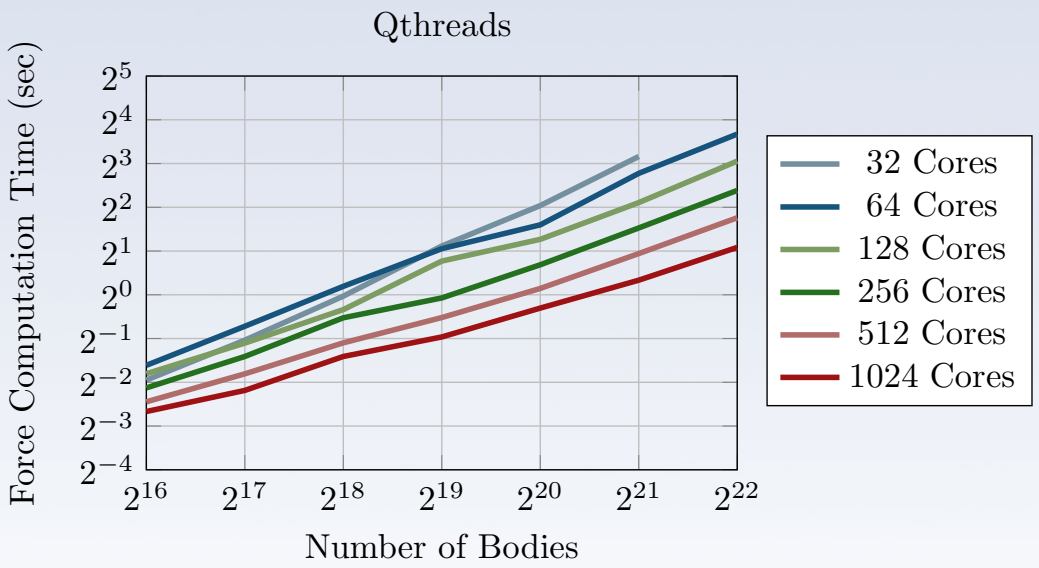


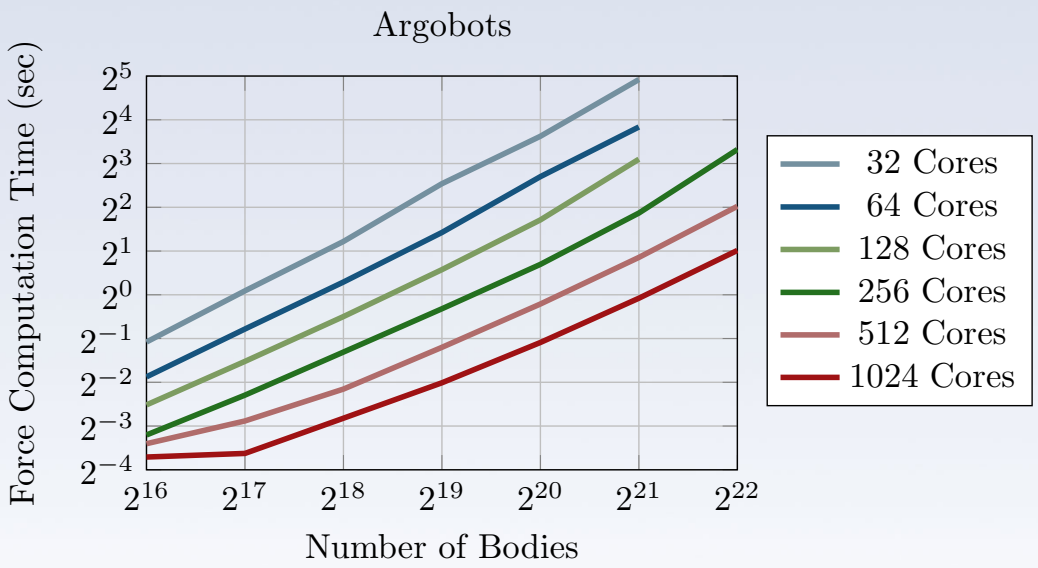
Outline

1. Design and Implementation
2. Runtime Evaluation
3. **Application Evaluation**
4. Future Work

Barnes-Hut Results

- PPL implementation based off previous work[2]





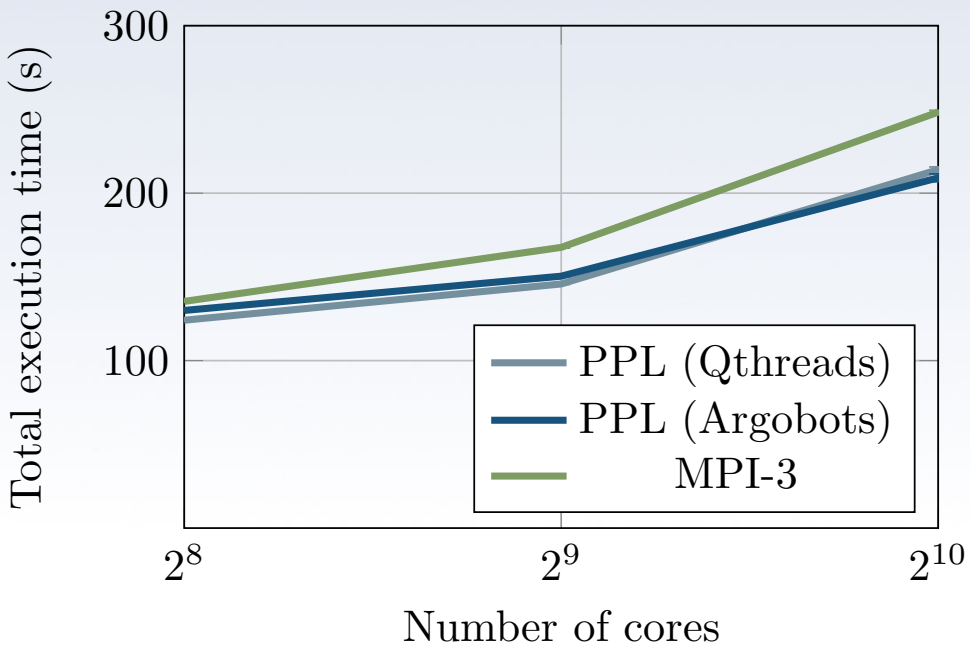
Number of Bodies	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
Qthreads	1.3	2.9	4.9	6.5	5.1
Argobots	2.6	8.0	13.0	13.4	8.8

[2] . Zhang, B. Behzad, and M. Snir. Design of a multithreaded Barnes-Hut algorithm for multicore clusters. IEEE Transactions on Parallel and Distributed Systems, 26(7):31–36, 2015.

Monte Carlo Particle Transport

- Approximate the diffusion of particles through a structure
  - Involve frequent lookups of large, static cross-section data (100GB+)
- Energy Band Memory Server (EBMS) algorithm[3]

Number of cores	2^8	2^9	2^{10}
PPL (Qthreads)	1.09	1.15	1.16
PPL (Argobots)	1.04	1.11	1.19

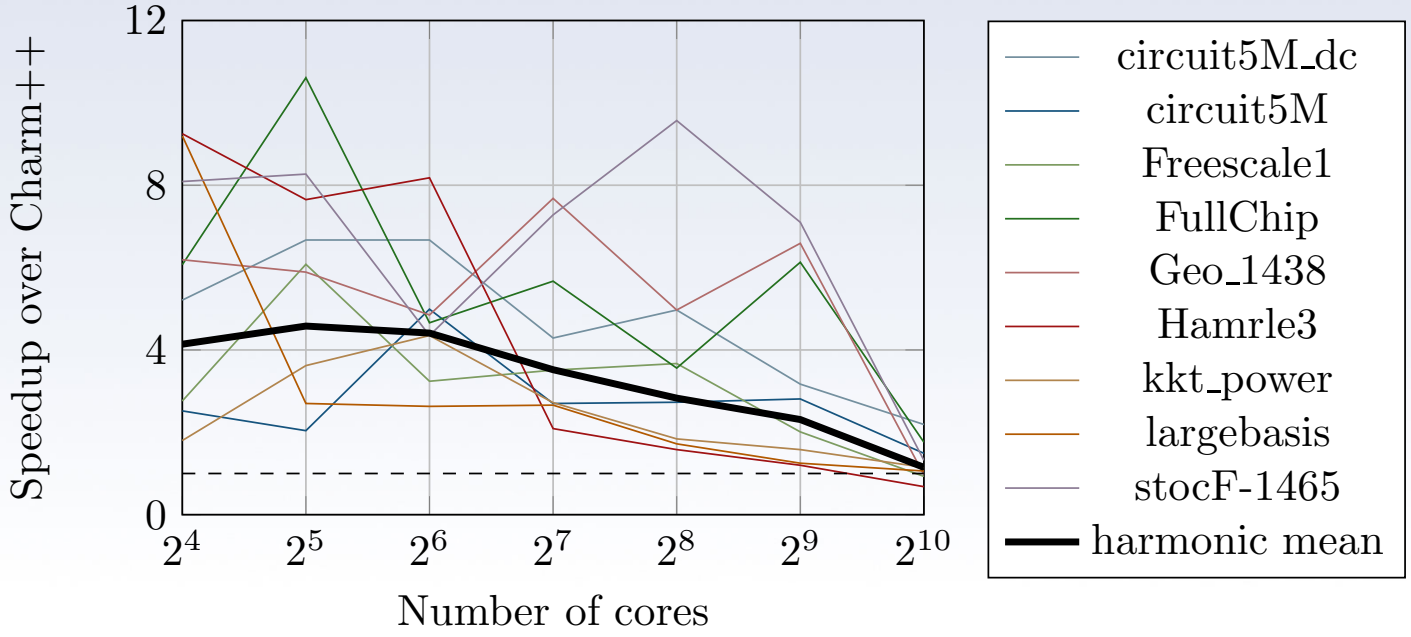


illinois.edu

[3] K. G. Felker, A. R. Siegel, K. S. Smith, P. K. Romano, and B. Forget. The Energy Band Memory Server Algorithm for Parallel Monte Carlo Transport Calculations. In Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo, 2013.

Sparse Triangular Linear System Solver

- Implementation based off Charm++ implementation[4]
  - PPL uses active messages to provide data-driven execution model



[4] E. Totoni, M. T. Heath, and L. V. Kale. Structure-adaptive parallel solution of sparse triangular linear systems. Parallel Computing, 40(9):454–470, 2014.

Outline

1. Design and Implementation
2. Runtime Evaluation
3. Application Evaluation
4. **Future Work**

Future Work

- Reduce overhead of threading module
  - Expect native implementation to improve
- Improve interactions between communication threads and communicating tasks
- Develop better performance models that focus on close communication and threading interaction
- Continued development

THANK YOU!

Contact info:
brooks8@illinois.edu | hdang8@illinois.edu | dryden2@illinois.edu | snir@illinois.edu

QUESTIONS?