UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

# QMPI: A Library for Multithreaded MPI Applications

Alex Brooks
Hoang-Vu Dang
Marc Snir

illinois.edu

# Outline

- Motivation
- Communication Model
- Qthreads
- QMPI
- Summary

# MOTIVATION

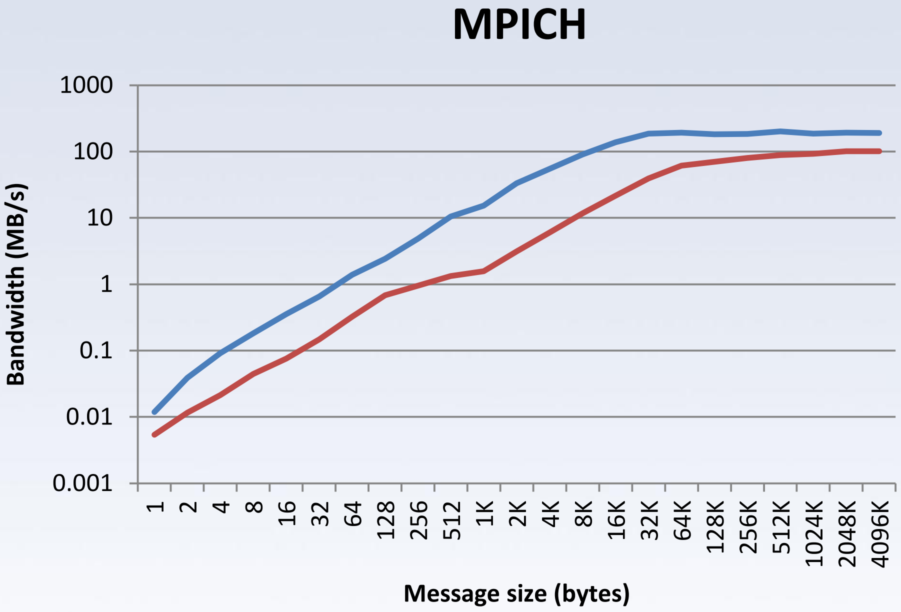
# Issue

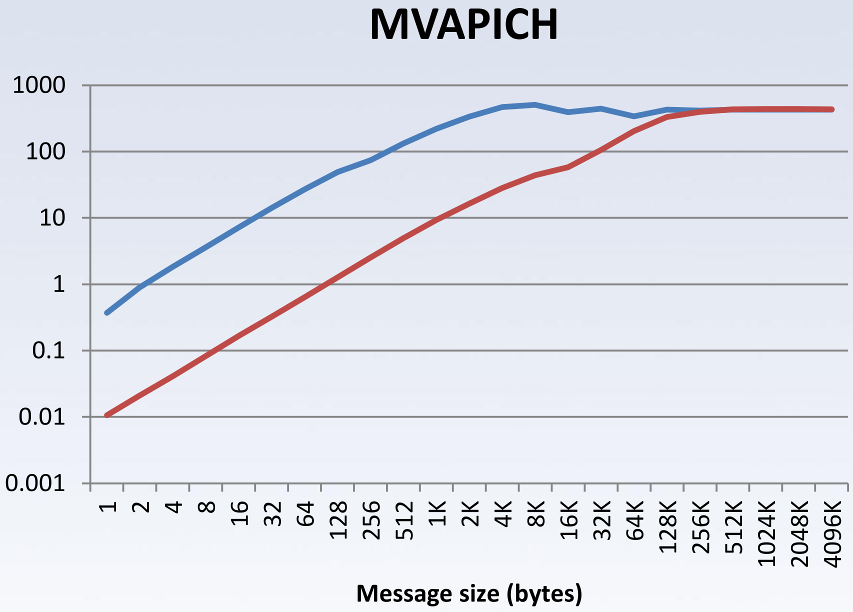
- Large numbers of threads performing communication causes problems
  - Locking
  - Polling
  - Scheduling
- As a result there are very few hybrid MPI+pthread applications

# Current MPI Design

- MPI code executed by calling thread
  - Requires coarse-grain locking – limits concurrency
  - Some implementations don’t support
- Communication completion is observed through polling
  - Separate calls to progress engine
- Scheduler is unaware of which threads have become runnable

# Performance


MPICH

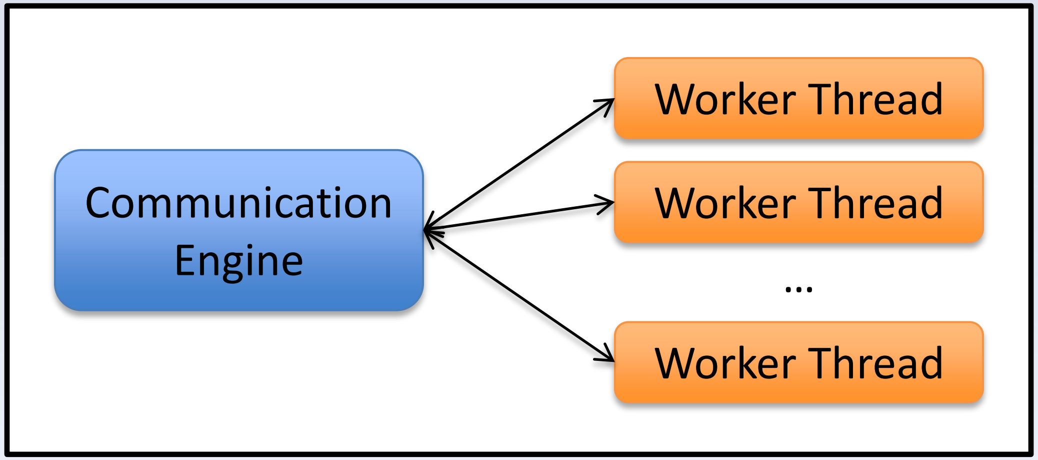

MVAPICH

# Goals

- Enable efficient use of multithreaded two-sided communication
  - Light-weight threads
  - Low-overhead scheduling upon communication completion
- Improve programmability of multithreaded MPI

# COMMUNICATION MODEL

# Main idea



- Light-weight tasks submit requests to comm. engine
- Comm. engine marks task as runnable when communication completes

# QTHREADS

# Introduction

- Tasking model which supports millions of light-weight threads
- Three main entities
  - Task - Function of execution
  - Worker - Thread executing tasks
  - Shepherd - Queue of tasks

# Synchronization

- Full/Empty bit (FEB) semantics
  - FEB determines status of data
    - 0 (empty) : data is not written
    - 1 (full) : data is written
- Read
  - Stall task until FEB is full, then read data and set as empty
- Write
  - Stall until FEB is empty, then write data and set as full

# Task Scheduler

- Each work is associated with a single shepherd
  - Tasks pulled from shepherd to execute
- Tasks can be stolen from other shepherds under certain conditions
- Tasks preempt when waiting on synchronization

# Overview

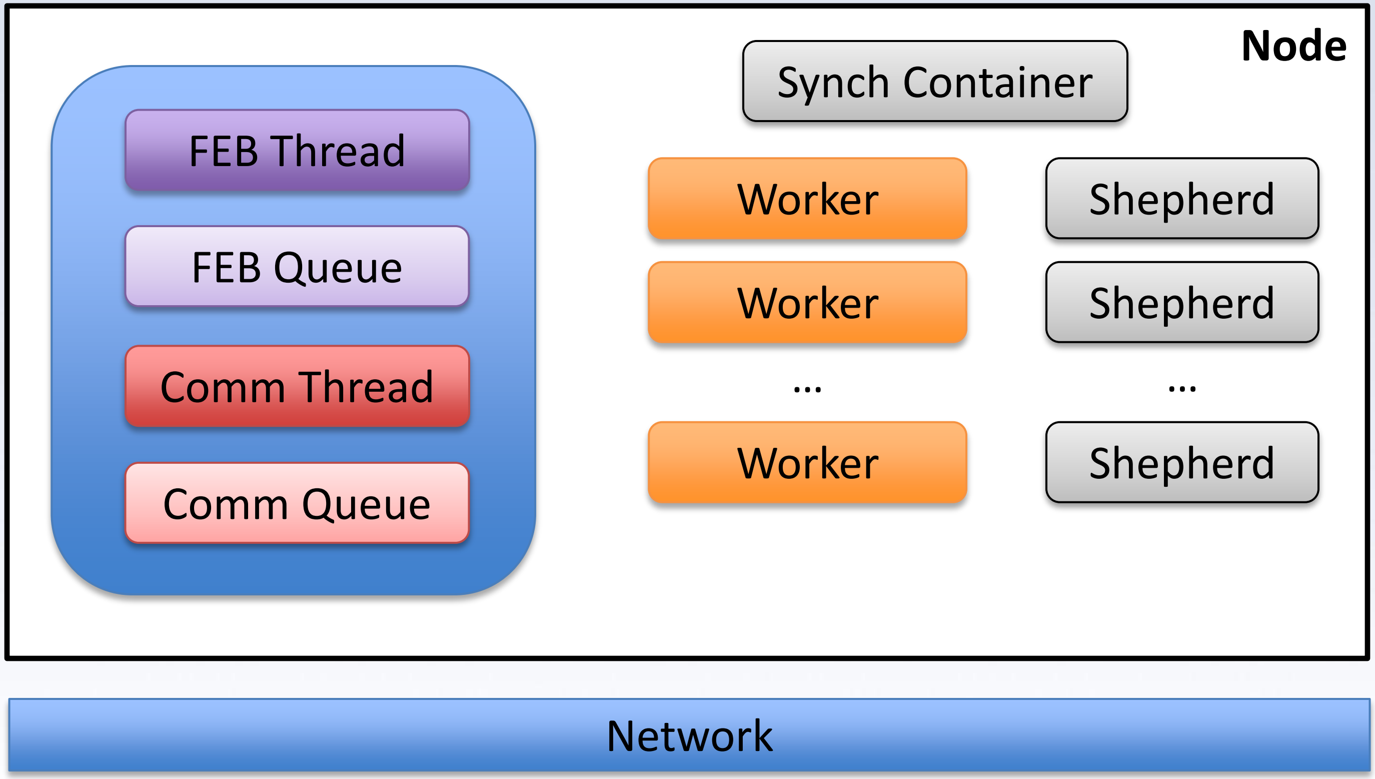
- Scalable over-subscription
  - Millions of tasks can be spawned with minimal overhead in performance
- Worker idle time is reduced through task preemption at synchronization
- “Automatic” load-balancing of tasks
- Shared-memory environment

# QMPI

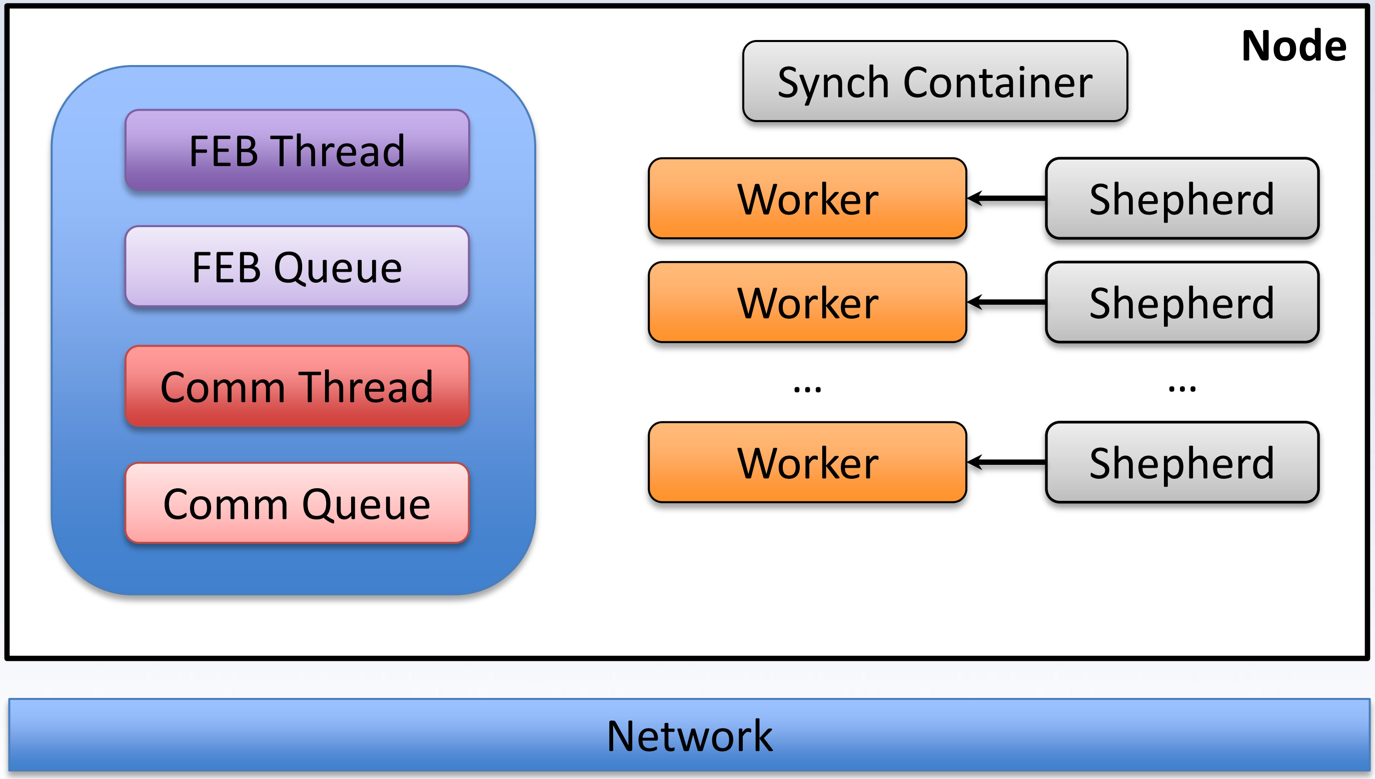
# Overview

- Qthreads+MPI
  - Qthreads light-weight task model with communication through MPI
- Two threads dedicated for communication engine
  - One for communication, one for FEB management

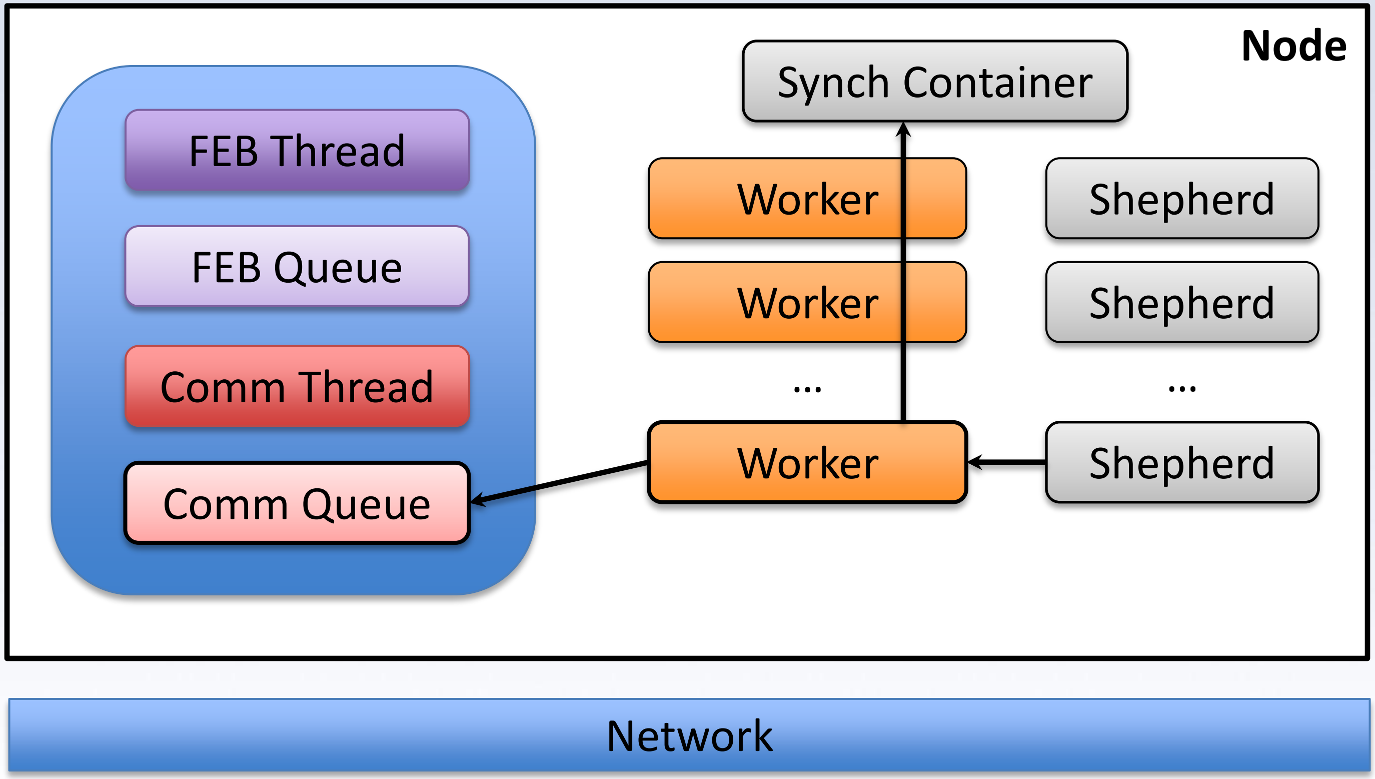
# Communication Model



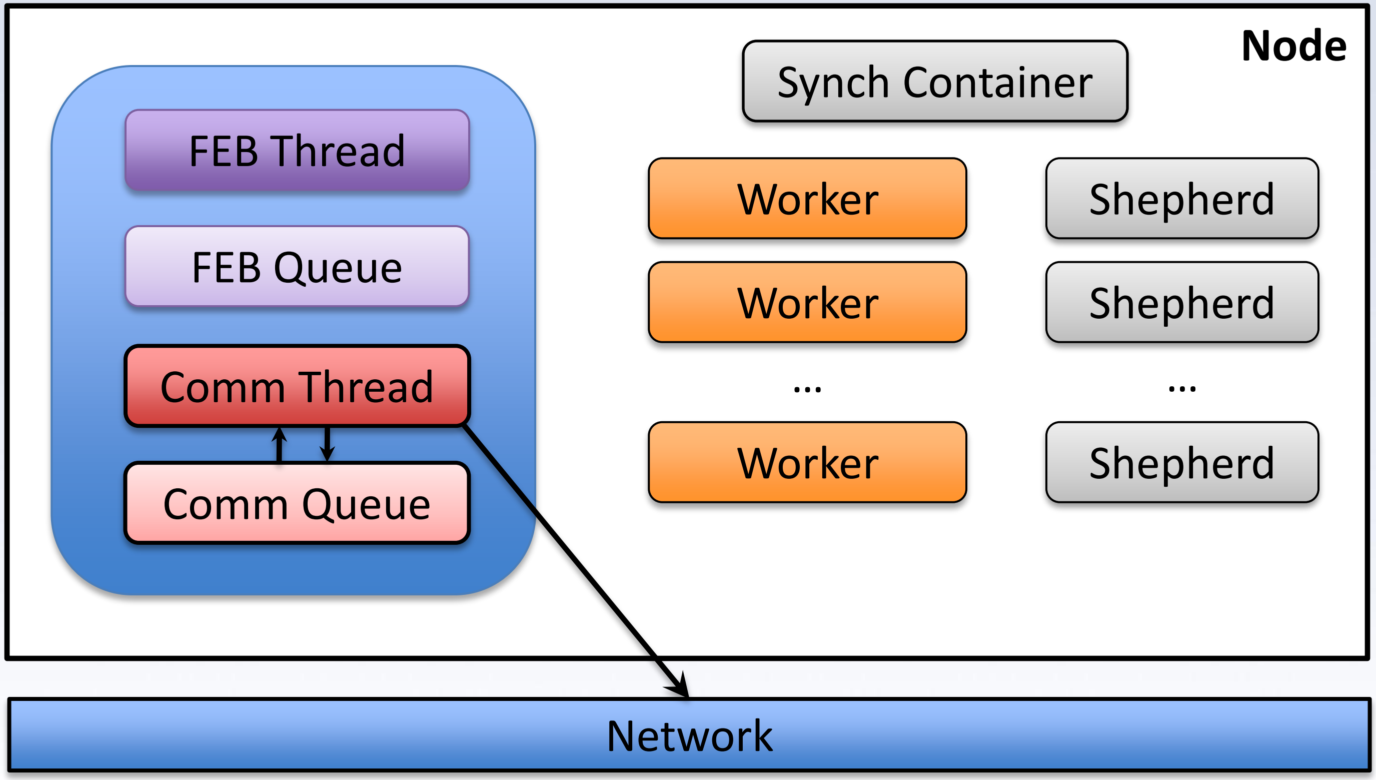
# Communication Model



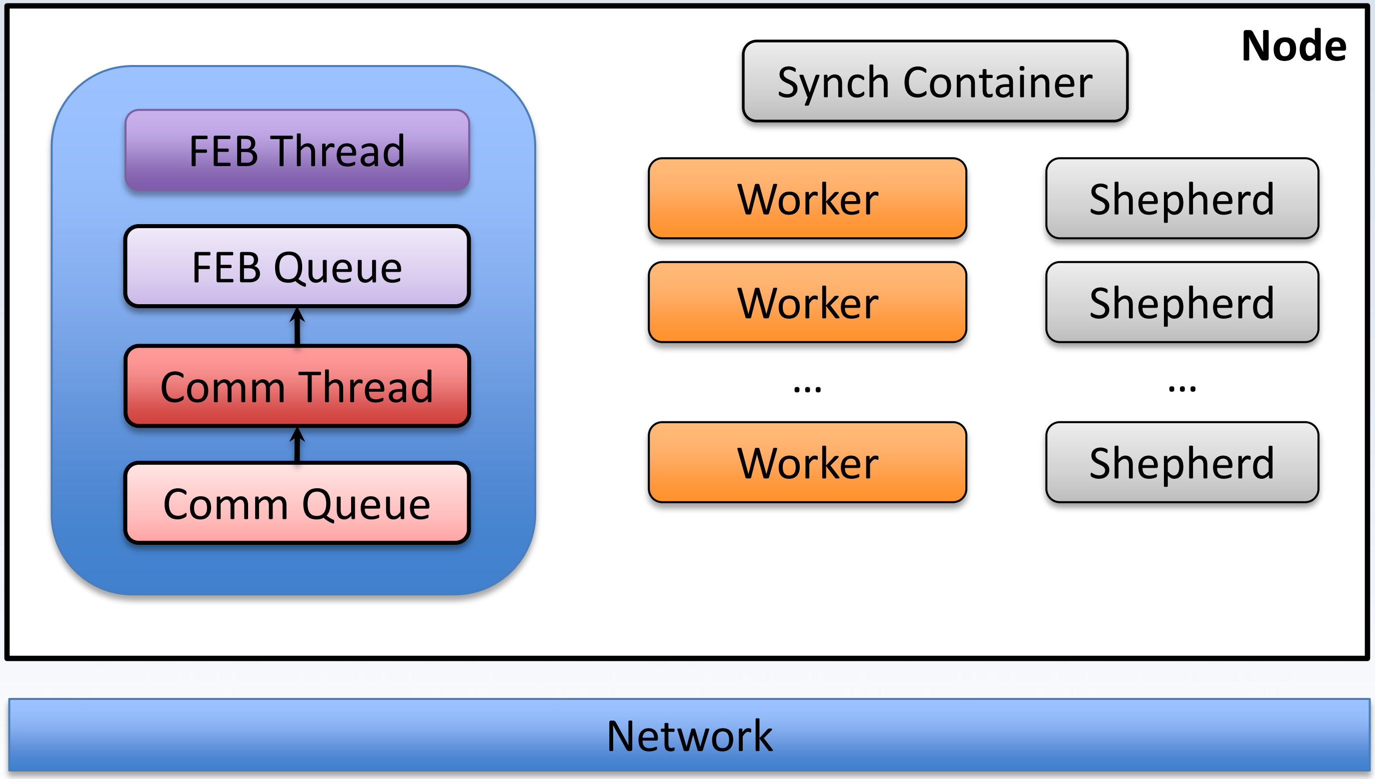
# Communication Model



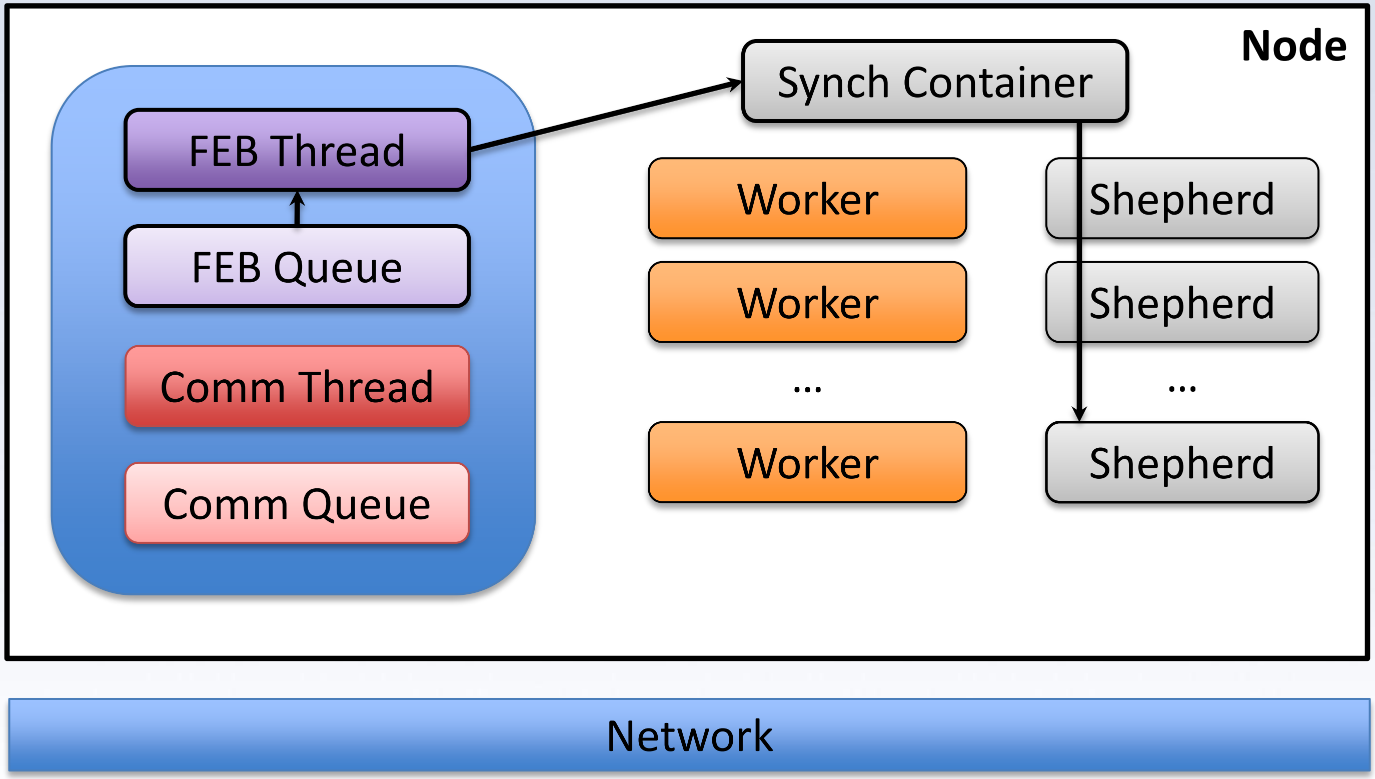
# Communication Model



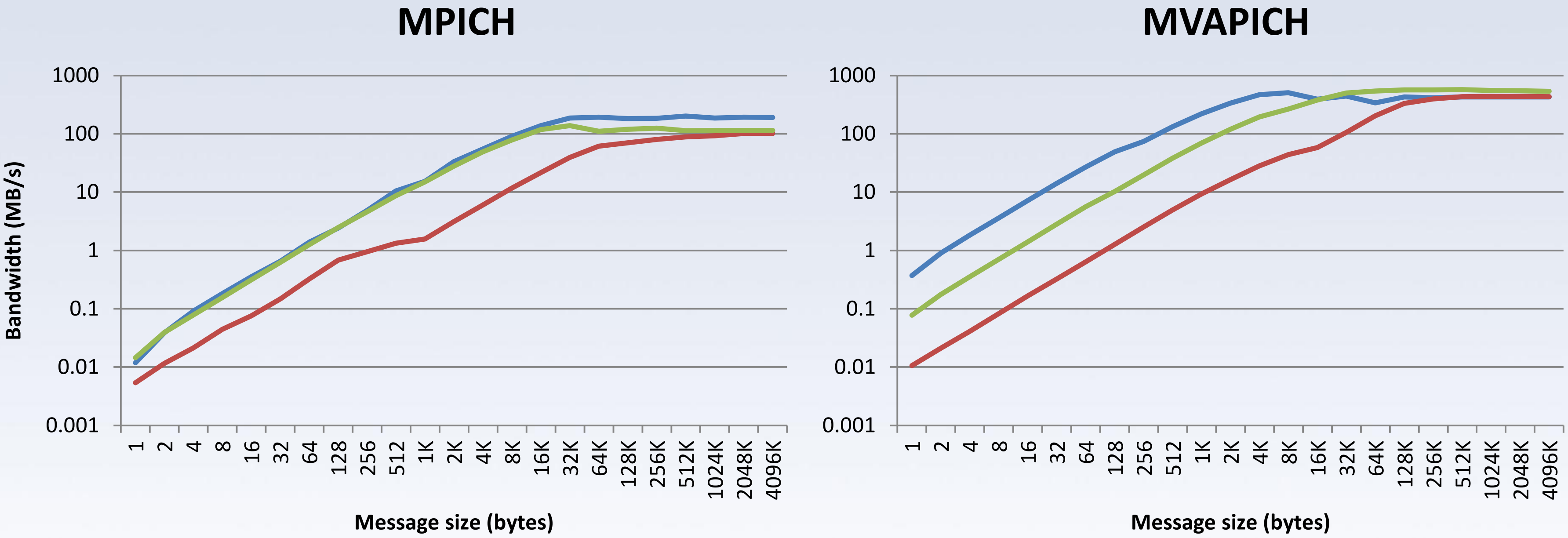
# Communication Model



# Communication Model



# Performance



illinois.edu

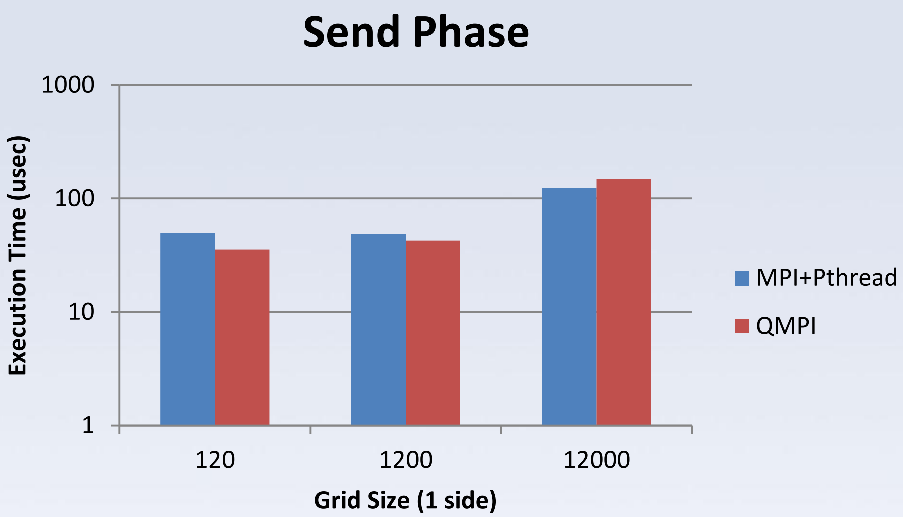
# Target Applications

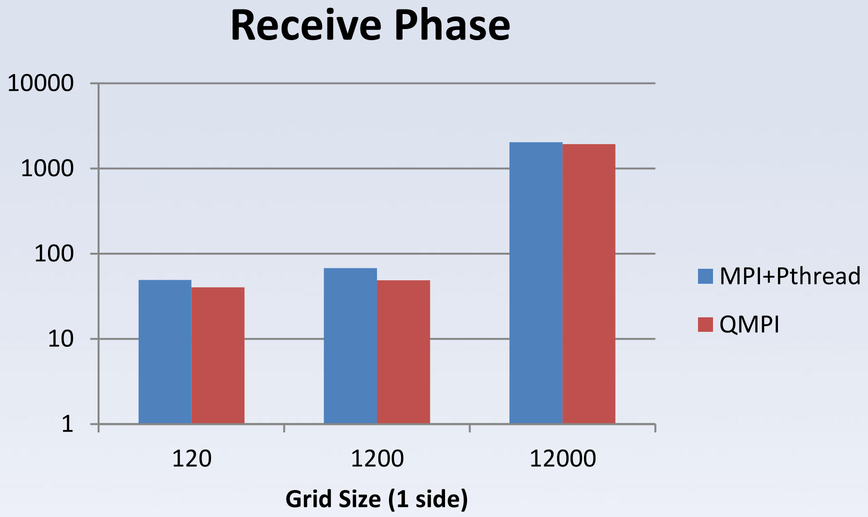
- Not beneficial for all problems
  - Little overlap in multithreaded communication → increase runtime
- Bulk-synchronous communication
- Oversubscription
  - Benefit directly from Qthreads

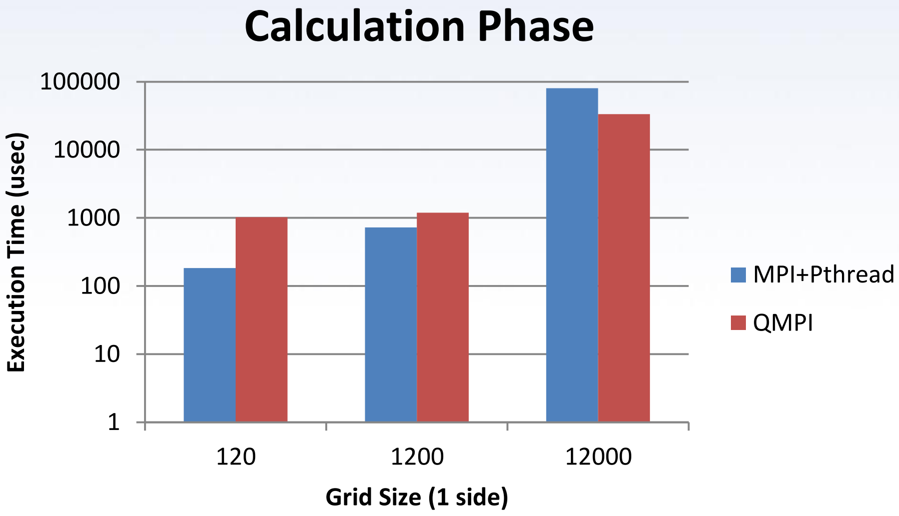
# Simple Experiment

- 5-point stencil computation
  - Send edge values to neighbors
  - Recv edge values from neighbors
  - Compute new values

# Results


Send Phase
Execution Time (usec)
1000
100
10
1
120
1200
12000
Grid Size (1 side)
MPI+Pthread
QMPI


Receive Phase
10000
1000
100
10
1
120
1200
12000
Grid Size (1 side)
MPI+Pthread
QMPI


Calculation Phase
Execution Time (usec)
100000
10000
1000
100
10
1
120
1200
12000
Grid Size (1 side)
MPI+Pthread
QMPI

# SUMMARY

# Conclusion

- Large numbers of threads performing communication causes problems
- QMPI uses a communication model to decrease communication overhead
- QMPI performs much better than traditional MPI+pthreads in many situations

# On-going/Future Work

- Test QMPI with real applications
  - MiniGhost, Lulesh, UTS, etc.
- Message Aggregation
- Push QMPI model to an internal feature of MPI

# QUESTIONS