TRAM: Improving Fine-grained Communication Performance with Topological Routing and Aggregation of Messages

Presented by Lukasz Wesolowski

11th Annual Charm++ Workshop
April 15 - 16, 2013

T opological

R outing and

A ggregation

M odule

T opological

exploits physical network topology

R outing and

A ggregation

M odule

T opological

R outing and

determines message path

A ggregation

M odule

T opological

R outing and

A ggregation

combines messages

M odule

T opological

R outing and

A ggregation

M odule

component of a larger system

Introduction

- Charm++ library
  - Prototype: Mesh Streamer
  - Originally developed for the 2011 Charm++ HPC Challenge submission
- Aggregates fine grained messages to improve communication performance

Why Aggregation?

- Sending a message involves overhead
  - Allocating buffer
  - Serializing into buffer
  - Injecting onto the network
  - Routing
  - Receiving
  - Scheduling

Communication Overhead

- Some overhead depends on data size
  – Serialization
- Some does not
  – Scheduling
- Aggregation targets the latter, constant overhead

Two Types of Constant Overhead

- Processing overhead
  - Processing time involved in sending a message
- Bandwidth overhead
  - To send some bytes on the network you must first … send some more bytes on the network
  - What does it mean to *send* a 0-byte message?
    - Answer: in Charm++, to send at least 48 bytes

Bandwidth Overhead

- Message header
  - Charm++ envelope: 48 bytes
- Network overhead
  - Routing
  - Error checking
  - Partially filled packets

Bandwidth Methodology

- Network bandwidth is tricky to deal with
- Fundamentally, it is a property of a single link, but our tendency is to try to distill it into a single value (e.g. bisection bandwidth)
- If all links are utilized equally and link bandwidth is saturated, then each link's consumption is significant
  - We can then add up each link's utilization, and concern ourselves with this aggregate bandwidth

Fine-grained Communication

- Constant communication overhead really adds up when sending large numbers of small messages
  - What about large numbers of large messages?
- Sources of fine-grained communication
  - Control messages, acknowledgments, requests, etc.
- For strong scaling, communication becomes increasingly fine-grained with increasing processor count

Why Routing?

- By routing, we mean not selection of the links along which messages travel, but instead:
  - Selection of intermediate destination nodes or processes and delivery of the message to the runtime system at the intermediate destinations
  - Analogy: bus route
- Why does a passenger bus make stops before reaching the end of the route?

Why Routing?

- Why does a bus make stops before reaching the end of the route?
  - To serve more people along its direction of travel
    - Picking up people who want to board the bus at ANY stop along the route
    - Dropping off people whose destination is ANY subsequent stop along the route
  - Stopping at n stops serves (n-1)(n-2) separate trips (source/destination pairs)
    - This is why a relatively small number of buses can serve a large area of a city

Why Topological?

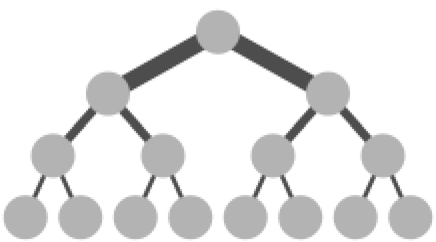
- It is infeasible to have a separate hardware network link between every pair of nodes in the system
- Consequences
  - some messages must travel through one or more intermediate nodes or switches
    - How it happens is normally invisible to the application and runtime system
  - aggregate bandwidth consumed grows linearly with every additional link along the route

Congestion

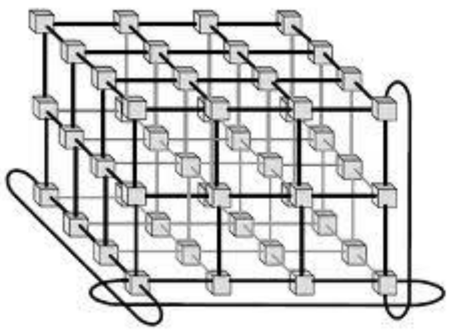
- Messages traveling concurrently along a link must split the bandwidth, leading to congestion
- Due to aggregation, TRAM messages are larger than typical, so congestion is of higher concern

Network Topology

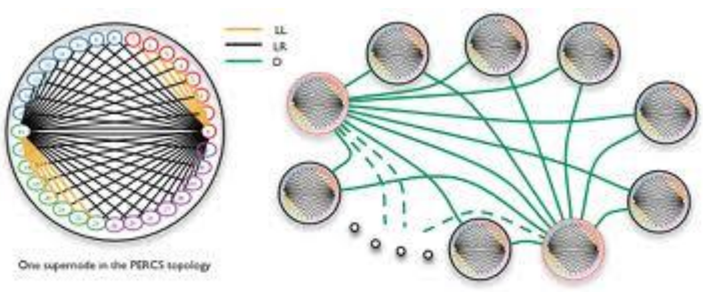
- No single network topology for supercomputers is accepted as best, so in practice several are in use



Source: en.wikipedia.org



Source: wiki.ci.uchicago.edu



Source: Bhatele et al., SC '11

Virtual Topology

- The nodes of a physical topology can be mapped onto a virtual topology
- The same virtual topology can be reused for various physical topologies
- TRAM employs a mesh virtual topology

Topological Routing

- Most messages pass across multiple links to reach the destination
- We can try combining messages, taking advantage of intermediate destinations analogously to bus stops
- But hardware-level routing is transparent to the runtime system
  - Solution: lift routing into software, at the level of the runtime system
  - Possible pitfall: routing will still happen independently in hardware

Minimal Routing

- Routing is minimal if every message sent travels over the minimum number of links possible to reach its destination
- Our goal with TRAM is to preserve minimal routing if possible
  - Reason: non-minimal routing consumes additional aggregate bandwidth

Virtual to Physical Topology Mapping

- Simplest and often best: make virtual topology identical to physical
  - Using Charm++ Topology Manager
- For high dimensional meshes, tori
  - Reduce number of dimensions while preserving minimal routing
- Fat trees
  - 2D within/across nodes

Data Item

- Unit of fine-grained communication to be sent by TRAM
- Sent for a particular destination
- Submitted using a local library call instead of the regular Charm++ syntax for a message send

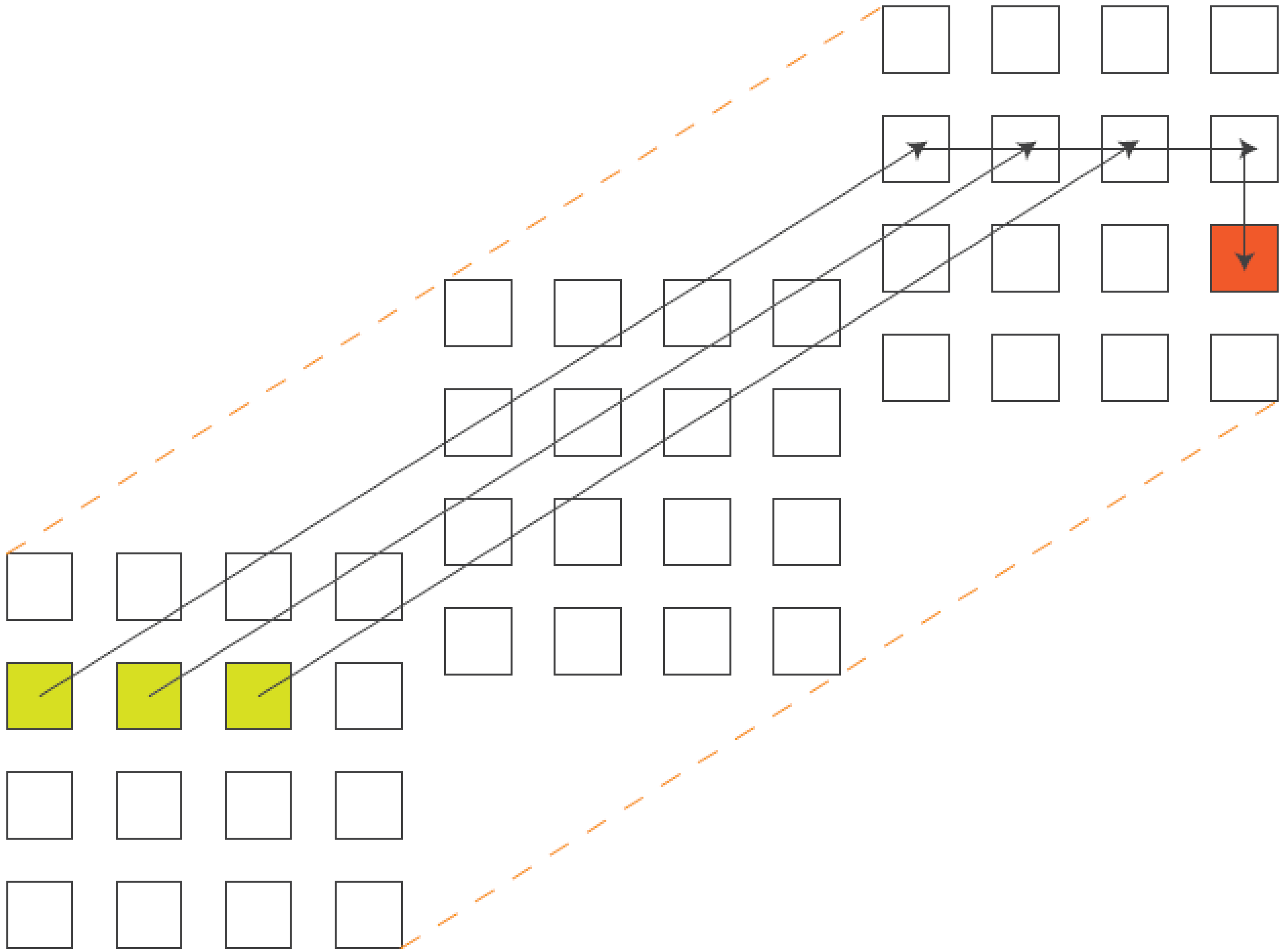
TRAM Peers

- In the context of TRAM, a process is allowed to communicate only with its peers
  - peers are all the processes that can be reached from it by moving arbitrarily far strictly along a single dimension

Mesh Routing Algorithm

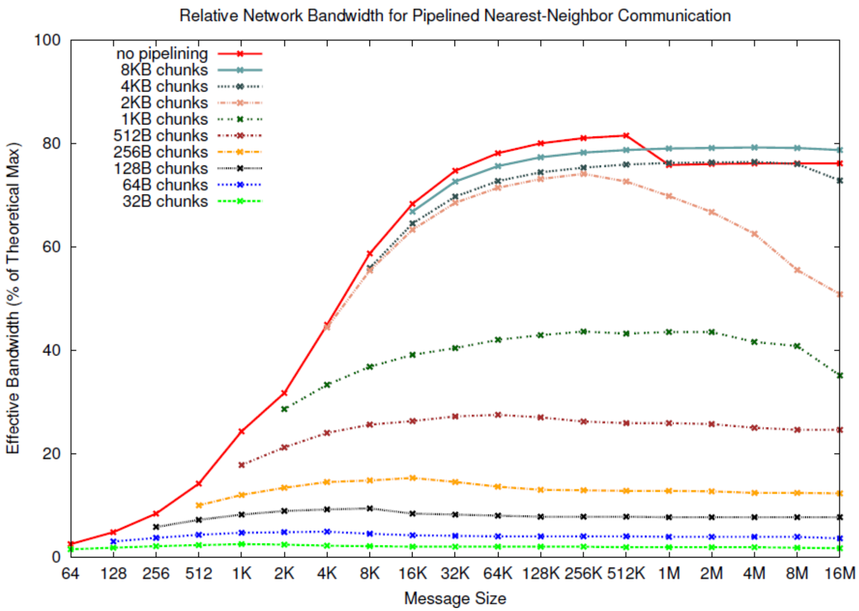
- Order the N dimensions in the virtual topology
- According to the order, send data items along the highest dimension whose index does not match the destination’s
  - to the peer whose index does match the final destination’s index along that dimension
- Aggregate at the source and each intermediate destination

Mesh Routing and Aggregation



Aggregation Buffer Size

- Buffers should be large enough to give good bandwidth utilization, but no larger
  - Buffering time should be relatively low
- On Blue Gene/P Buffers of size 4 KB or more are sufficient to almost saturate the bandwidth



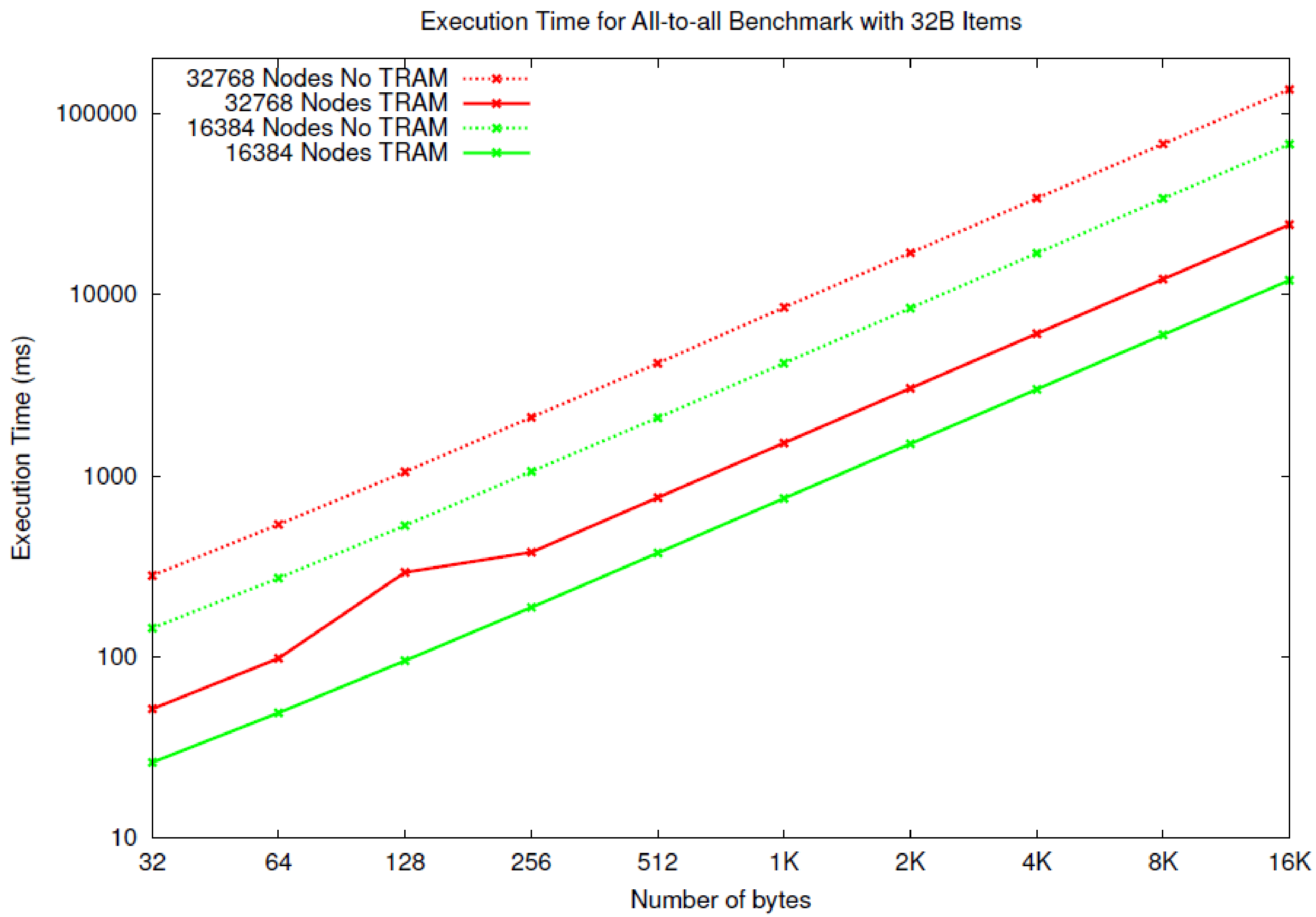
TRAM Memory Footprint

- Number of peers is typically a small fraction of all the processes in the run
  – For example, 32 x 32 x 32 topology
    - 32768 processes
    - 93 peers
- This allows TRAM's memory footprint to remain relatively small
  – Small enough for lower level cache

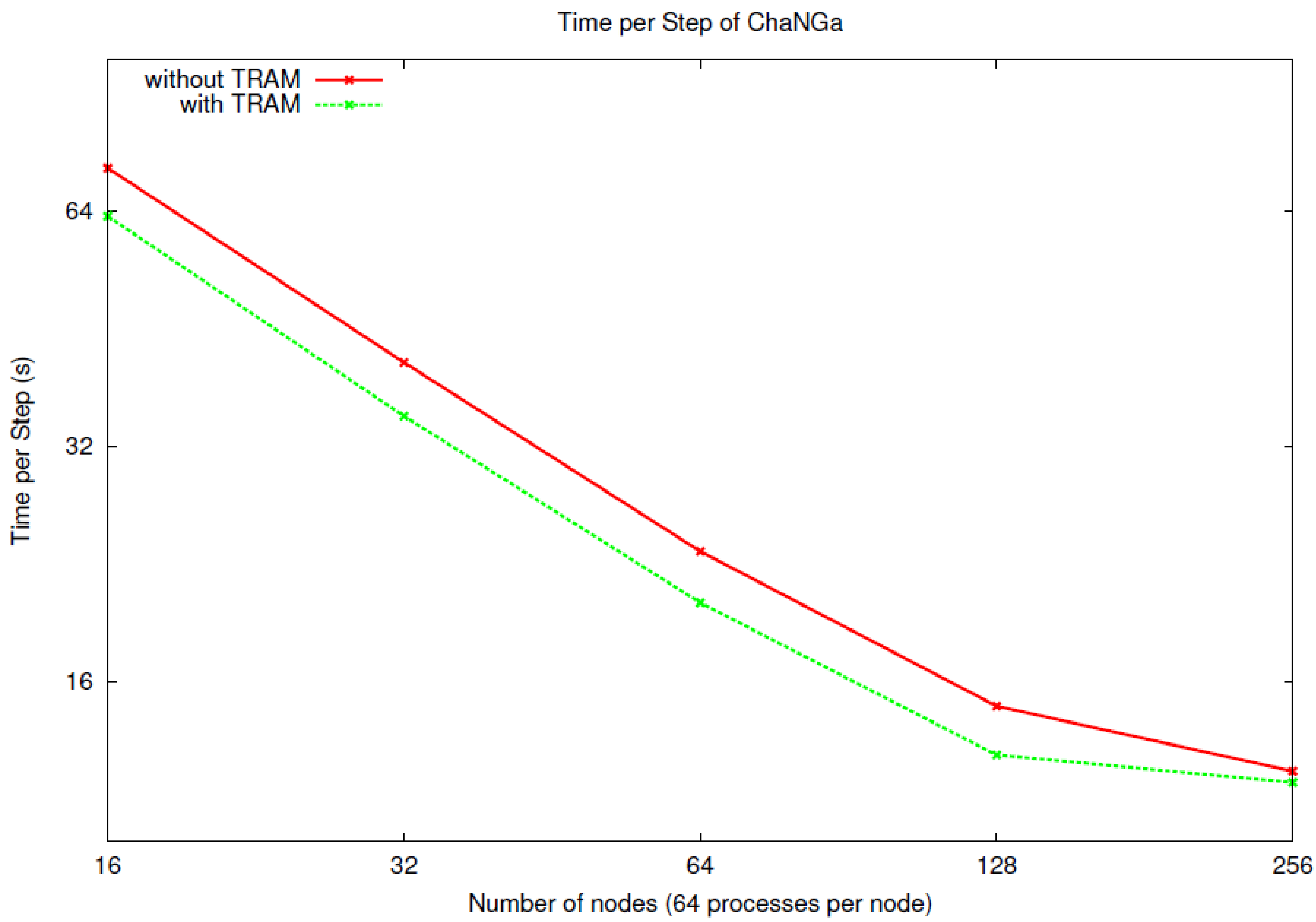
TRAM Usage Pattern

- Start-up
- Initialization
- Sending and receiving
- Termination
- Re-initialization

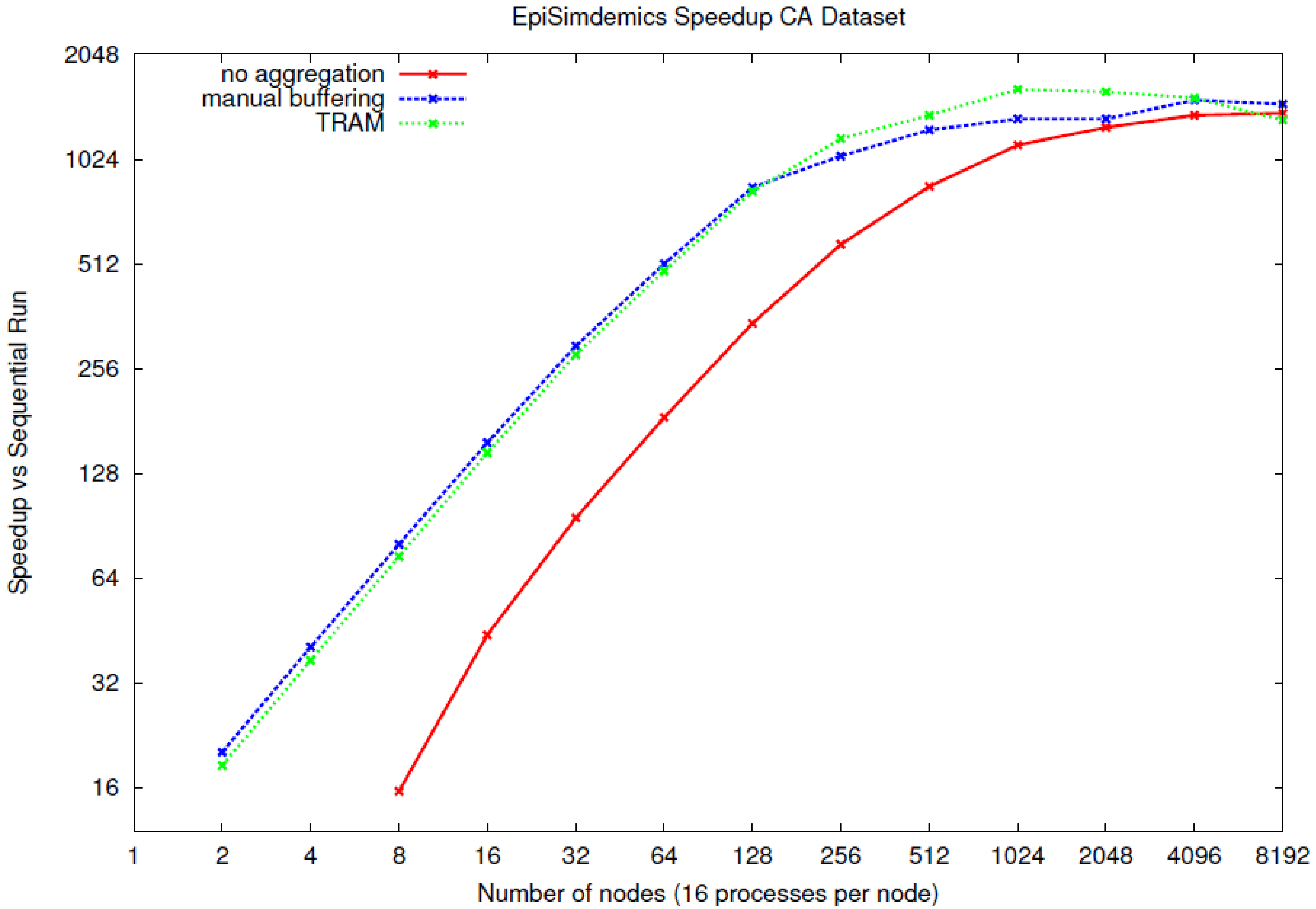
Alltoall Performance on Blue Gene/P



ChaNGa on Blue Gene/Q



EpiSimdemics on Blue Waters



Future Plans

- Develop alternative virtual topologies for non-mesh networks
- Generalize
  - First within Charm++
  - Then to other communication models
- Automate
  - Library parameter selection
  - Virtual topology dimensions
  - Choice of which messages to aggregate

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

EpiSimdemics results courtesy of Jae-Seung Yeom and the EpiSimdemics team.

Thank You