Enhancing the Communication Performance Models for SMPs

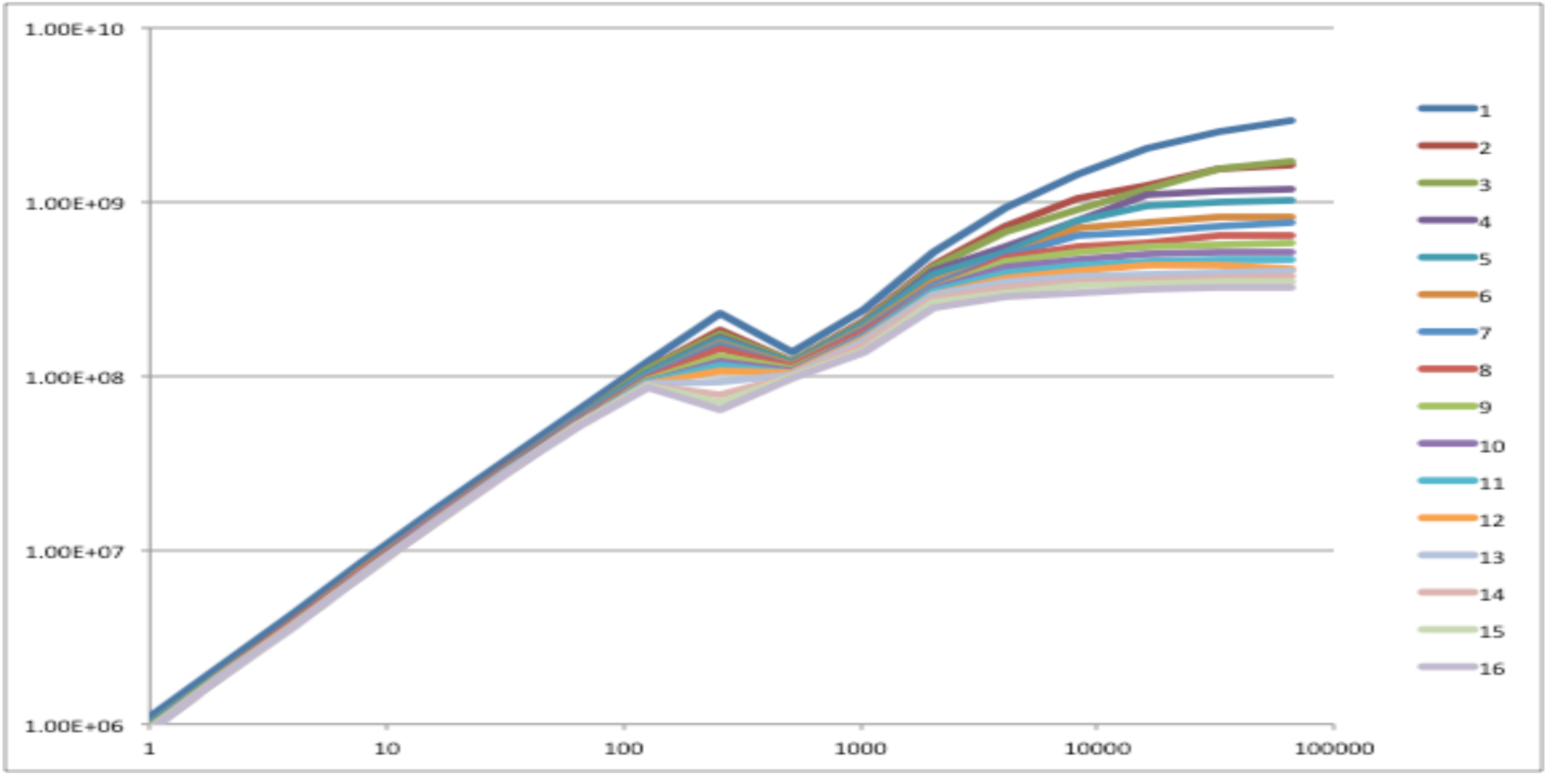
William Gropp
wgropp.cs.illinois.edu/

Classic Performance Model

- s + rn
- Model combines overhead and network latency (s) and a single communication rate 1/r
- Good fit to machines when it was introduced
- But does it match modern SMP-based machines?
  - Lets look at the the communication rate per process with processes communicating between two nodes

Cray XE6

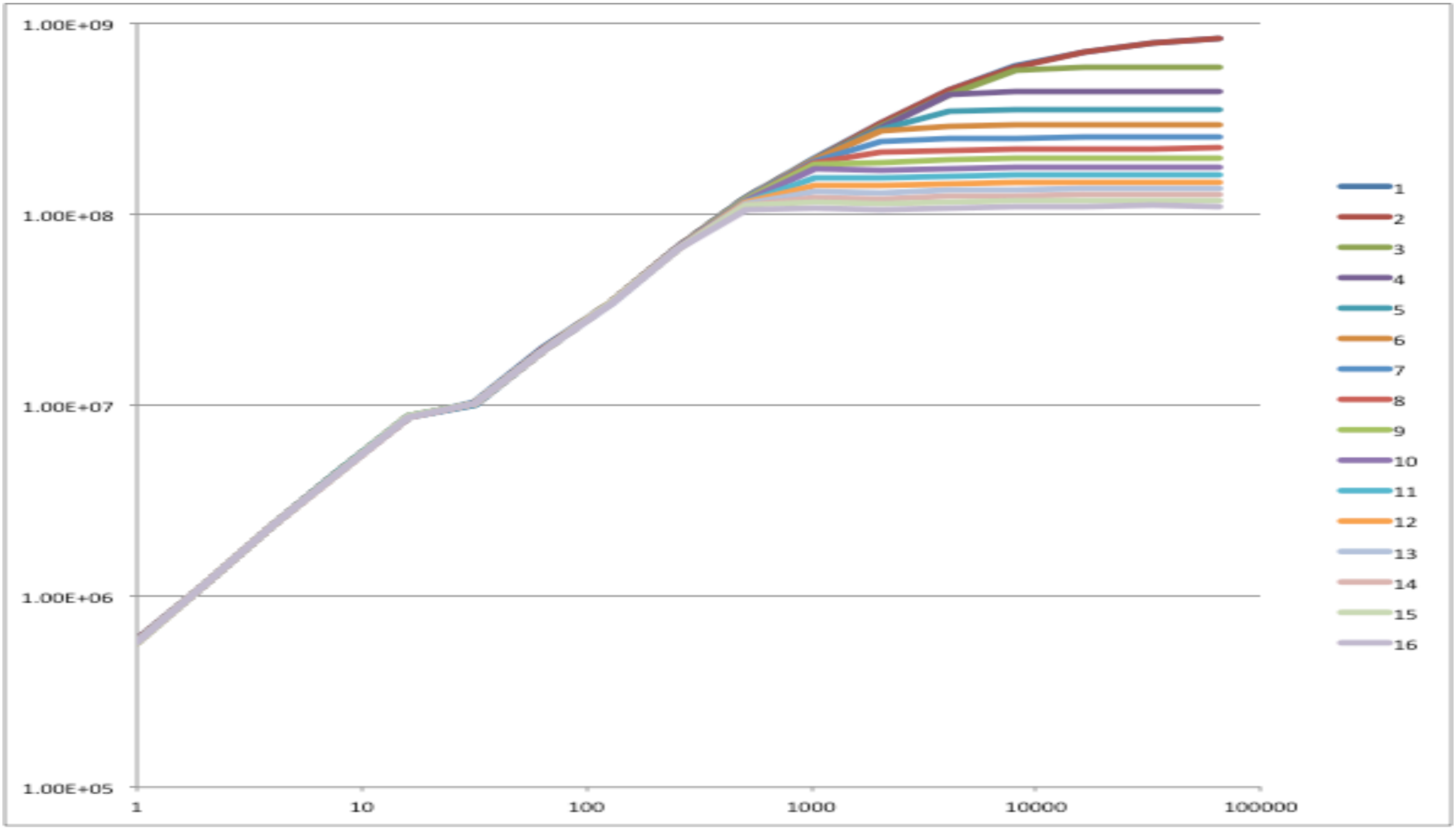
- Rate per MPI process



PARALLEL@ILLINOIS

Blue Gene/Q

- Rate per MPI process

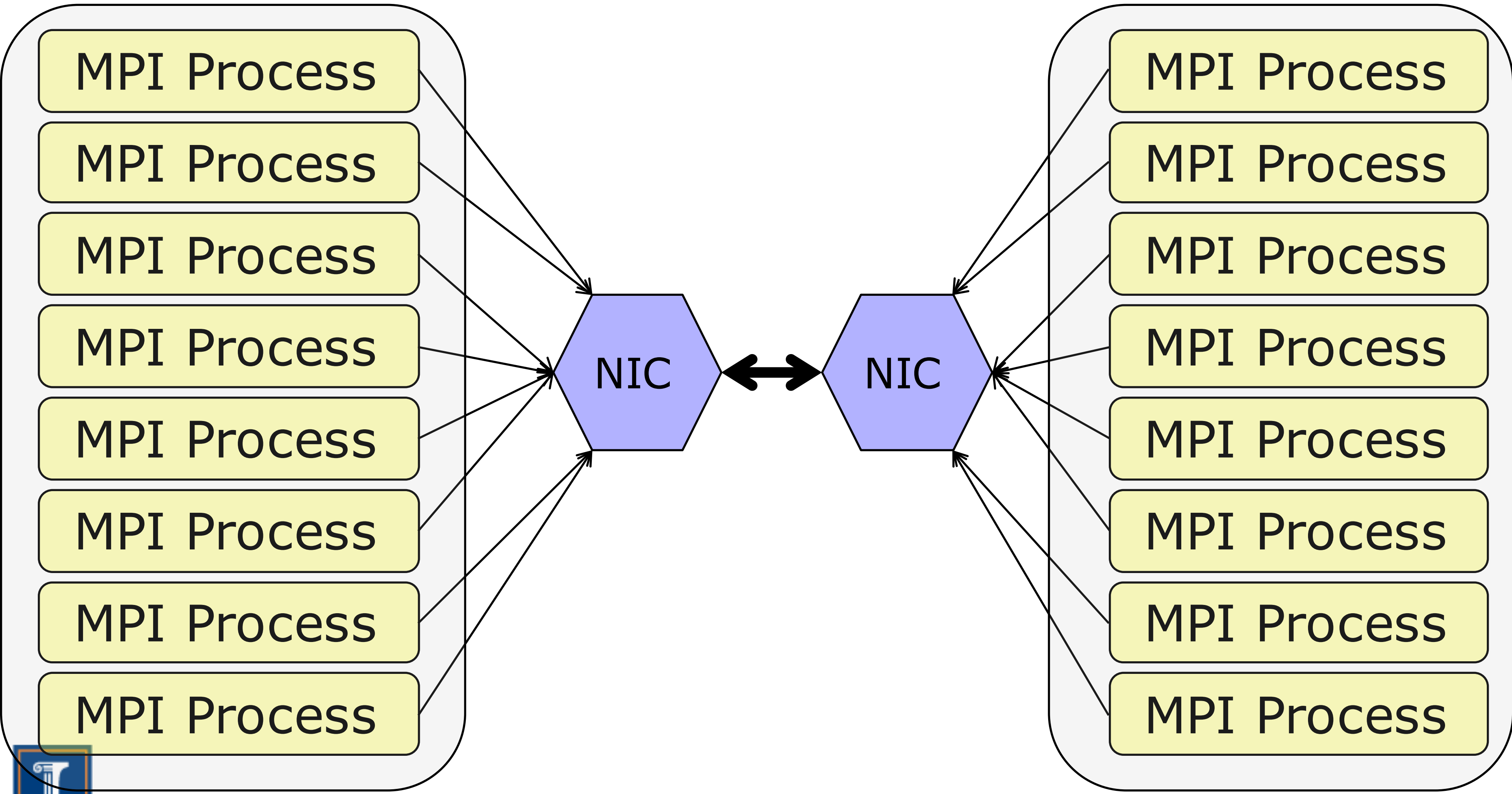


PARALLEL@ILLINOIS

Why this Behavior?

- The $T = s + rn$ model predicts the *same* performance independent of the number of communicating processes
  - What is going on?
  - How should we model the time for communication?

SMP Nodes: One Model



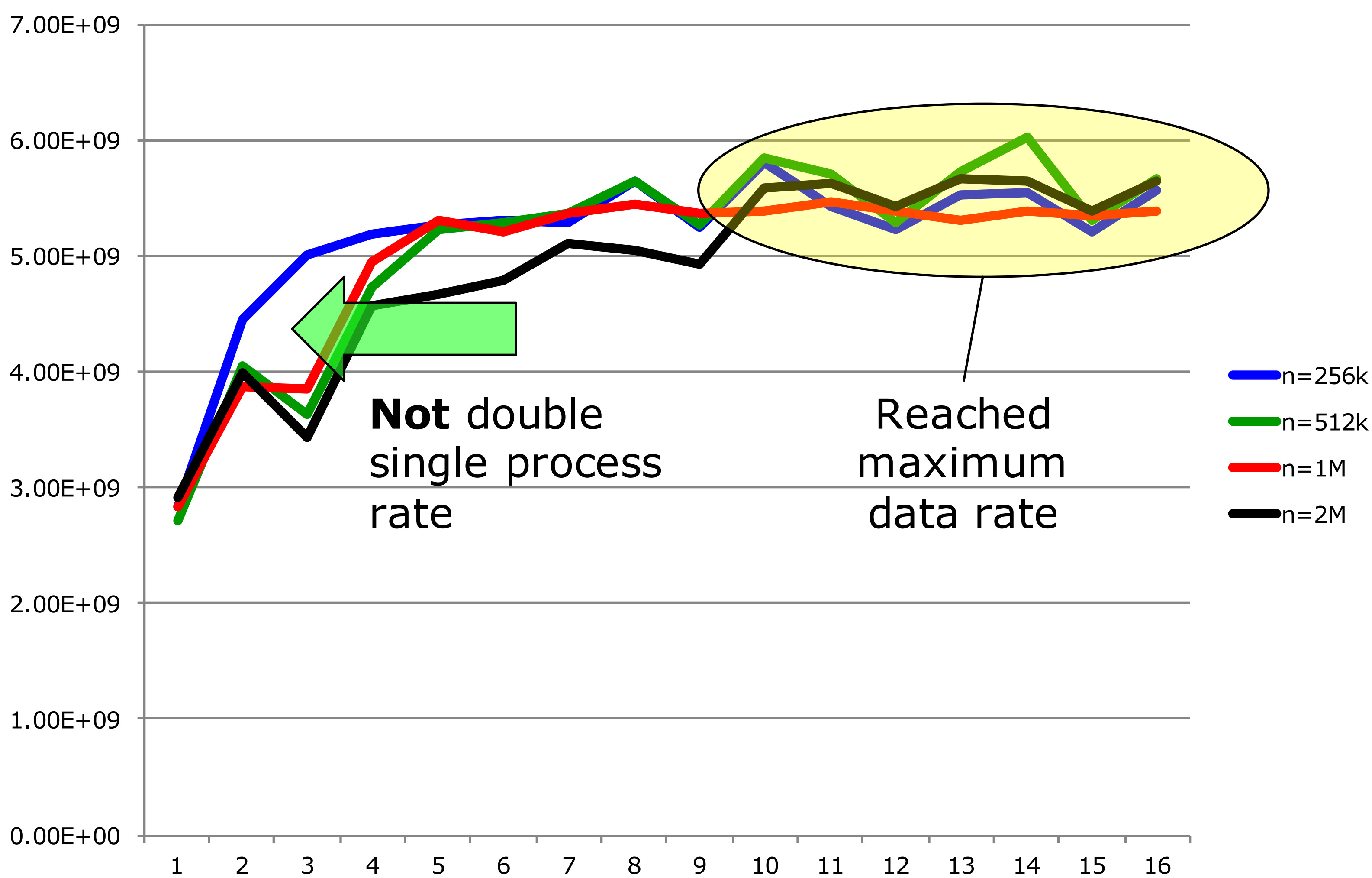
Modeling the Communication

- Each link can support a rate r_L of data
- Data is pipelined (Logp model)
  - ♦ Store and forward analysis is different
- Overhead is completely parallel
  - ♦ k processes sending one short message each takes the same time as one process sending one short message

Sending One Message From Each Process

- How do we model each process sending one message to another process on another node?
  - Classic "postal" model:
  - $T = s + r\,n$
  - Each process has no impact on the time that another process takes

Observed Rates for Large Messages



A Slightly Better Model

- Assume that the sustained communication rate is limited by
  - ♦ The maximum rate along any shared link
    - The link between NICs
  - ♦ The aggregate rate along parallel links
    - Each of the “links” from an MPI process to/from the NIC

A Slightly Better Model

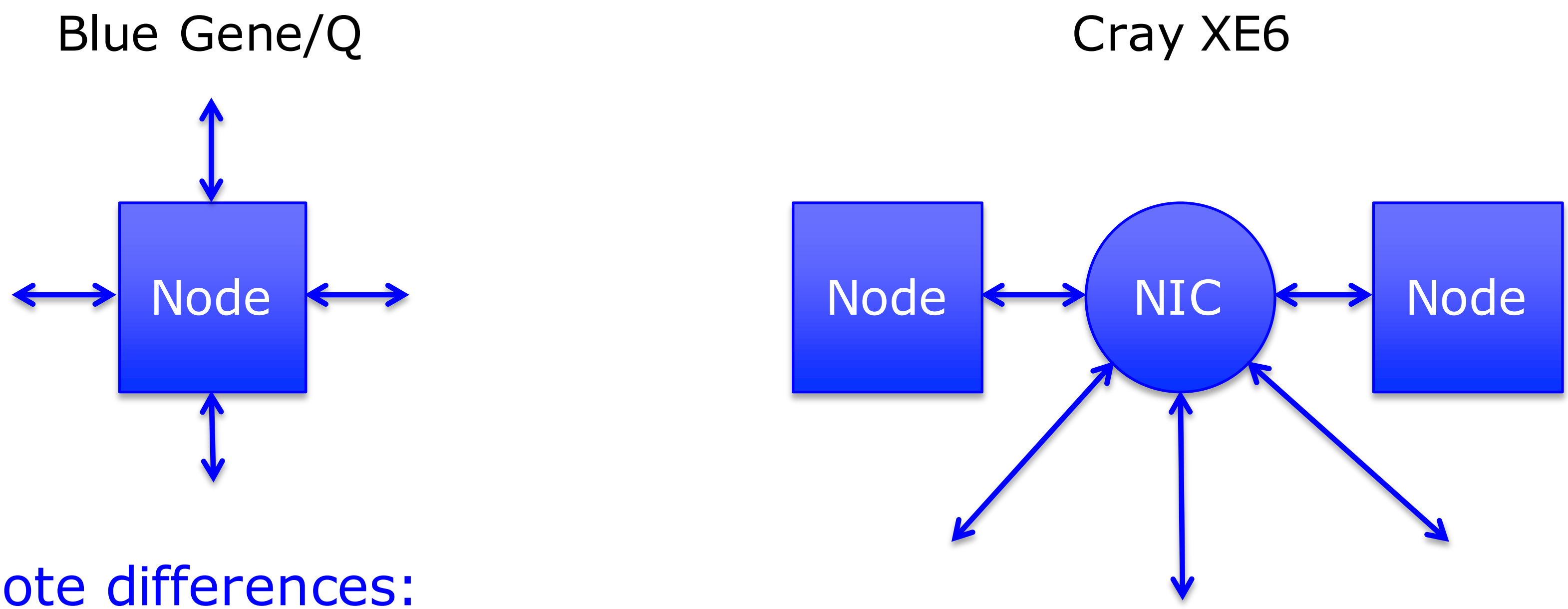
- For k processes sending messages, the sustained rate is
  - ♦ $\min(R_{NIC-NIC}, kR_{CORE-NIC})$
- Thus
  - ♦ $T = s + kn/Min(R_{NIC-NIC}, kR_{CORE-NIC})$
- Note if $R_{NIC-NIC}$ is very large (very fast network), this reduces to
  - ♦ $T = s + kn/(kR_{CORE-NIC}) = s + n/R_{CORE-NIC}$

A Slight Refinement

- Once there is more than one process communicating, the MPI implementation needs to do more work. This *might* result in a different incremental rate.
- We can model this as
  - $T = s + kn/\min(R_N, R_{Cb} + (k-1)R_{Ci})$

Two Examples

- Two simplified examples:



- Note differences:
  - BG/Q : Multiple paths into the network
  - Cray XE6: Single path to NIC (shared by 2 nodes)
  - Multiple processes on a node sending can exceed the available bandwidth of the single path

The Test

- Nodecomm uses routines to discover the underlying physical topology
- Performs point-to-point communication (ping-pong) using 1 to # cores per node to another node (or another chip if a node has multiple chips)
- Outputs communication time for 1-num_cores along a single communication channel
  - Note that hardware may route some communication along a longer path to avoid contention. This test can't control that

Examples from Current Systems

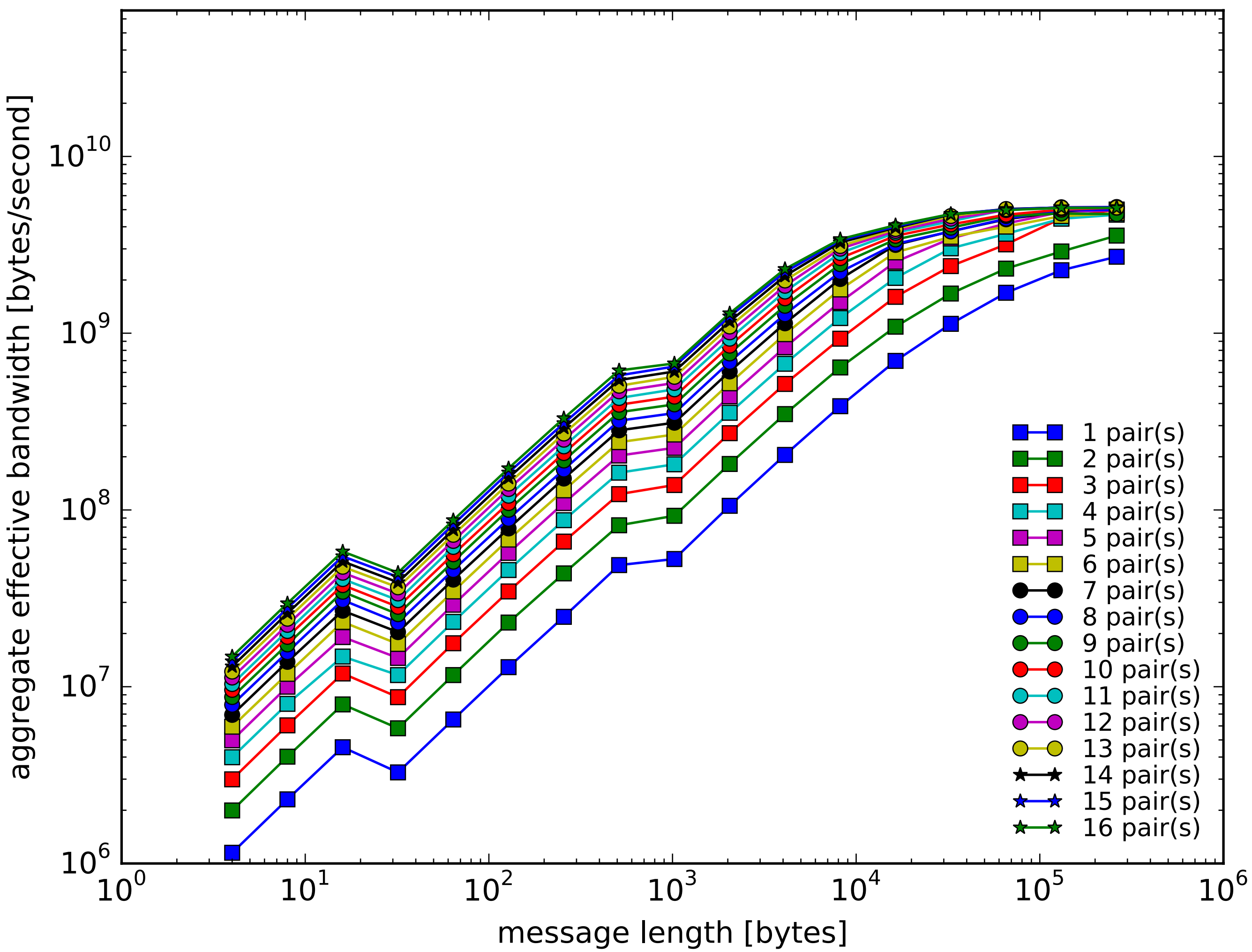
- The following results are taken from
  - ♦ Modeling MPI Communication Performance on SMP Nodes: Is it Time to Retire the Ping Pong Test
    - W Gropp, L Olson, P Samfass
    - Proceedings of EuroMPI 16 (to appear)
  - ♦ Code available at
    - https://bitbucket.org/williamgropp/baseenv

PARALLEL@ILLINOIS

New Model
(Full PingPong Time, 4 parameter model)

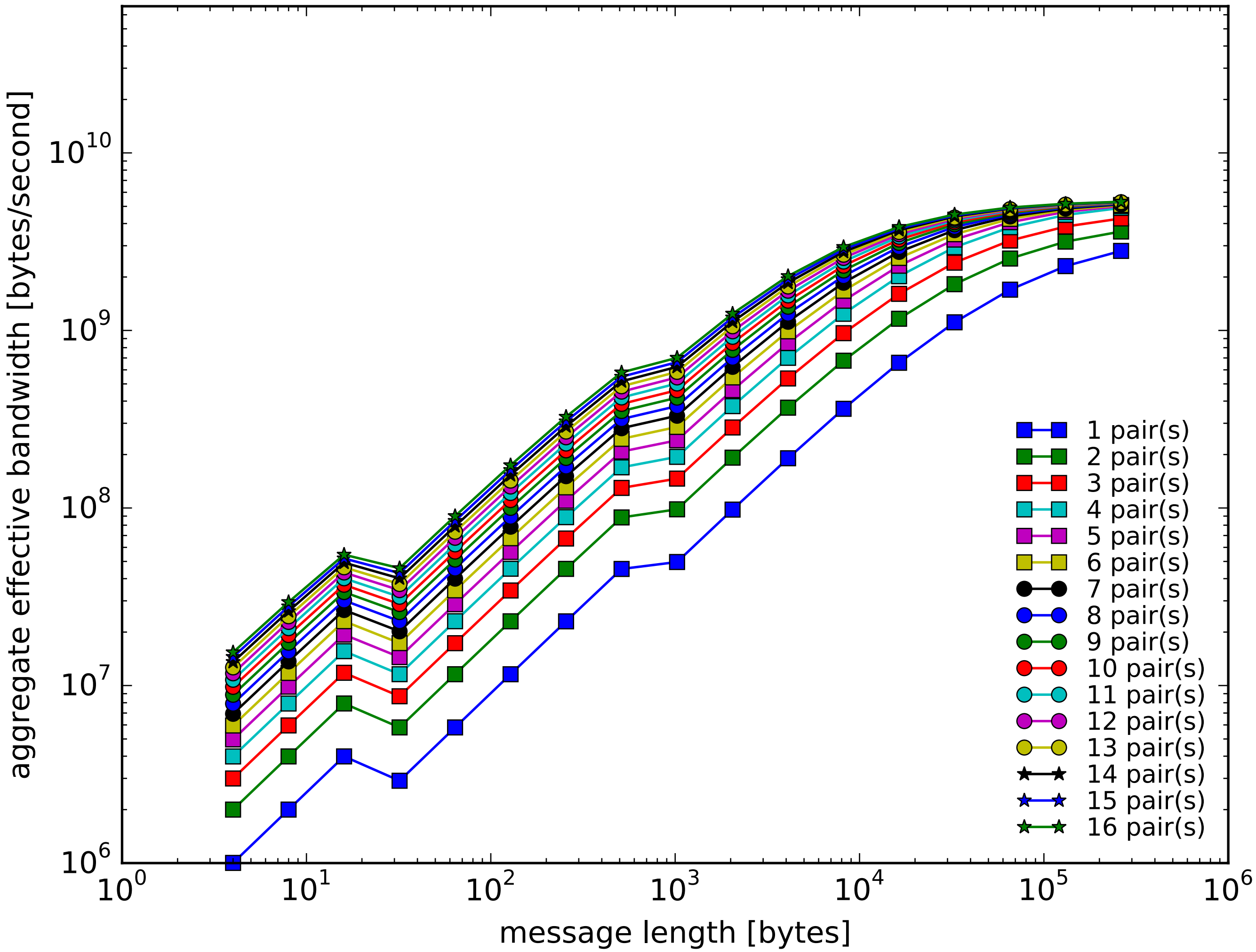
- $R_N = R_{NIC}$; $R_C = R_{CORE-NIC}$
- Short regime
  - ♦ s = 4 usec, R_C = 0.63 GB/s, R_{Ci}=-0.18GB/s, $R_N=\infty$
- Eager regime
  - ♦ s = 11 usec, R_{Cb} = 1.7GB/s, R_{Ci} = 0.062GB/s, $R_N=\infty$
- Rendezvous regime
  - ♦ s = 20 usec, R_{Cb} = 3.6 GB/s, R_{CI}=0.61GB/s, R_N = 5.5 GB/s

Cray: Measured Data



PARALLEL@ILLINOIS

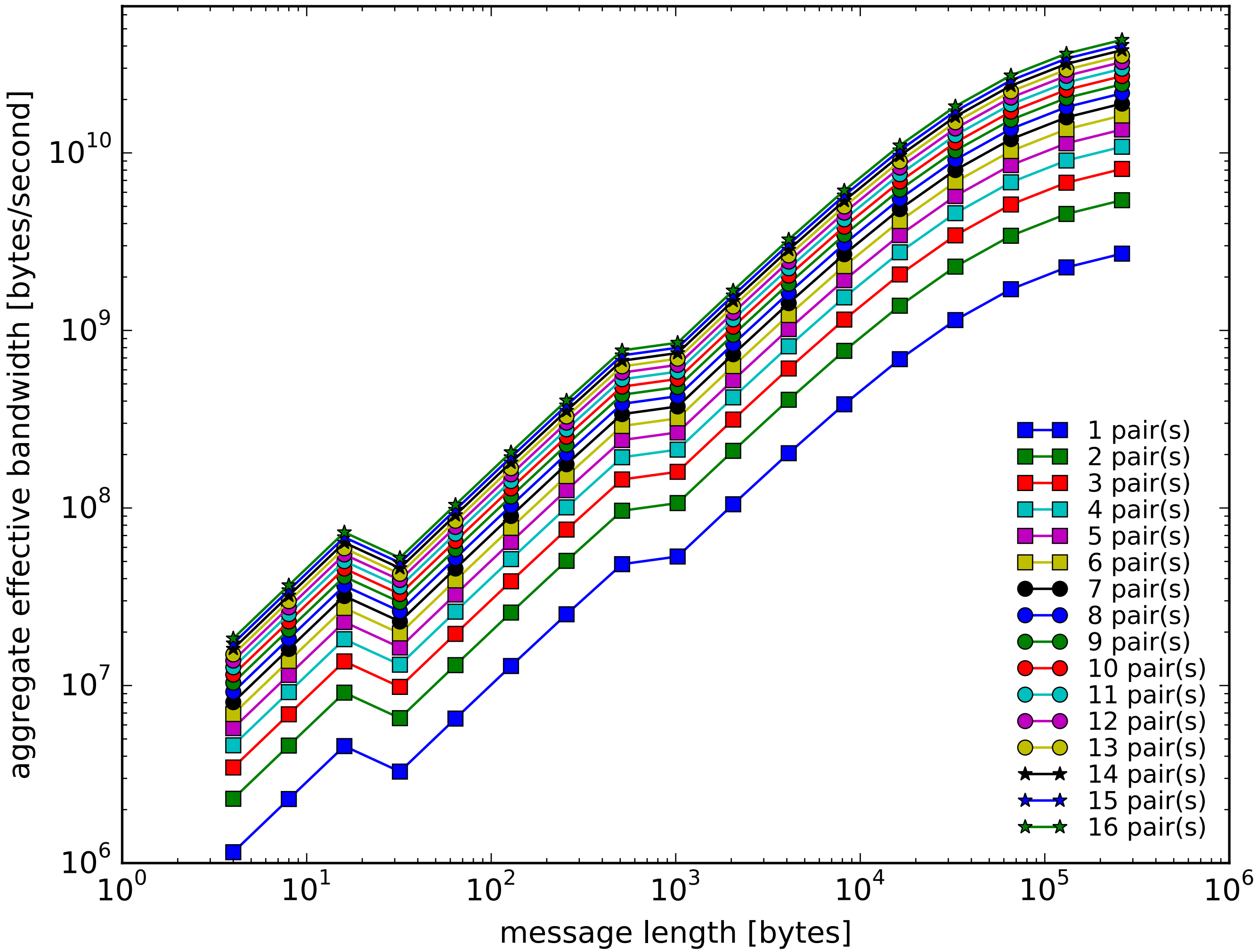
Cray: 3 parameter model



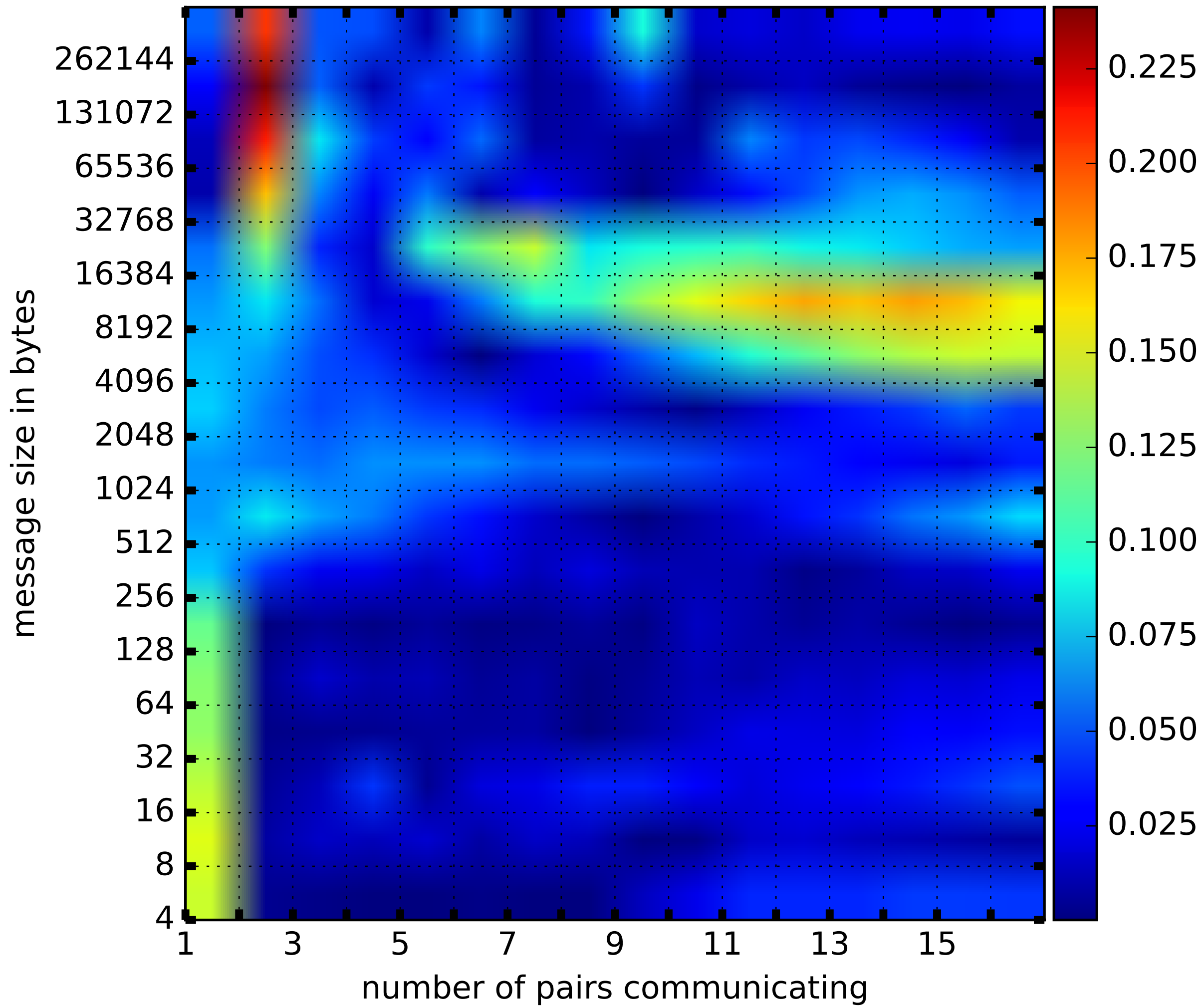
1867

PARALLEL@ILLINOIS

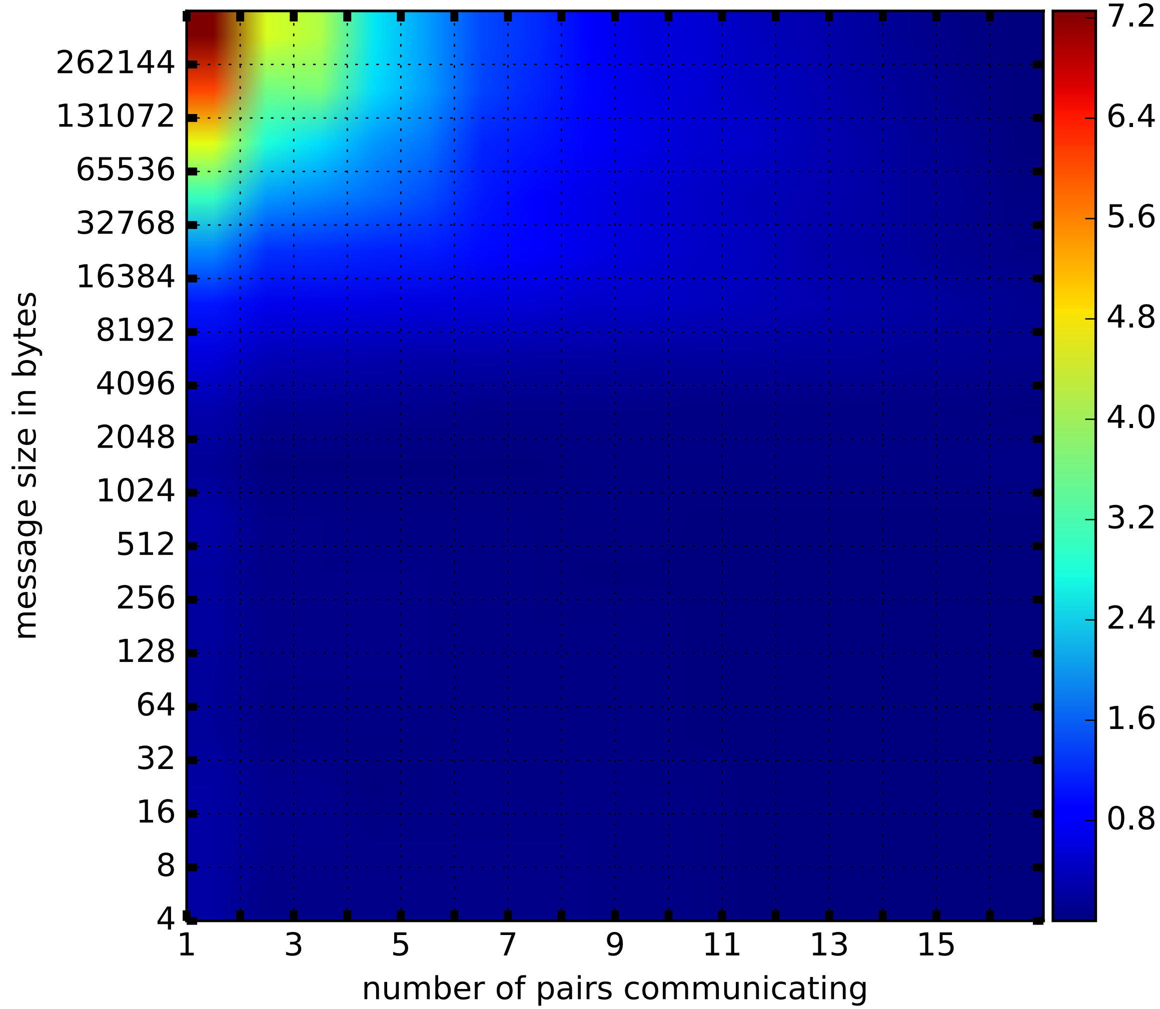
Cray: 2 parameter model



Cray: 3 parameter relative error

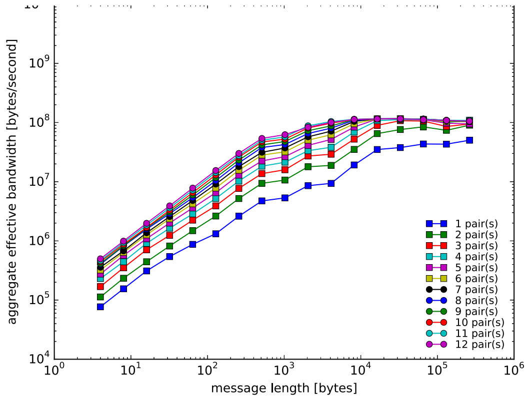


Cray: 2 parameter relative error

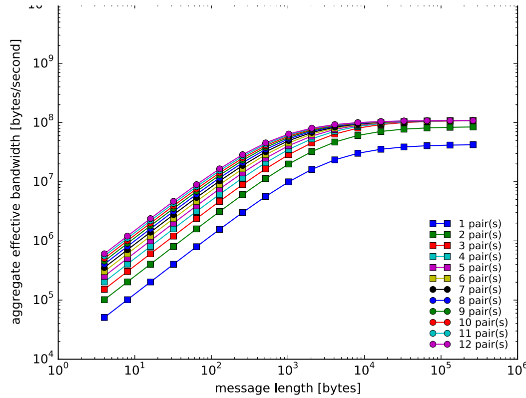


PARALLEL@ILLINOIS

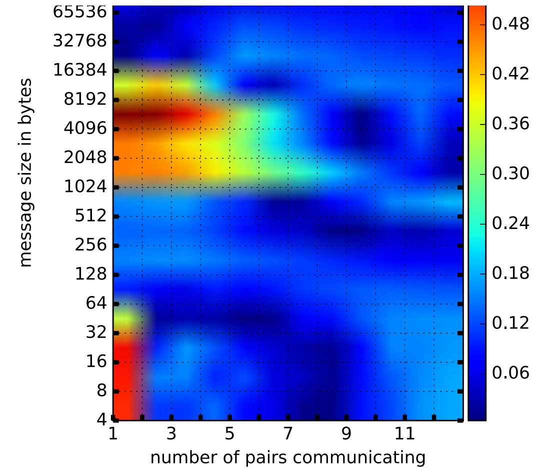
InfiniBand connected cluster



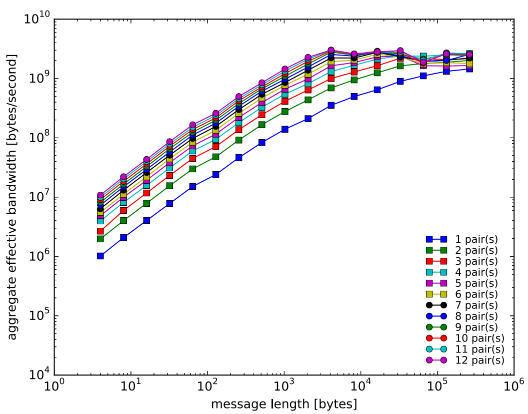
(a) Measured data (TCP).

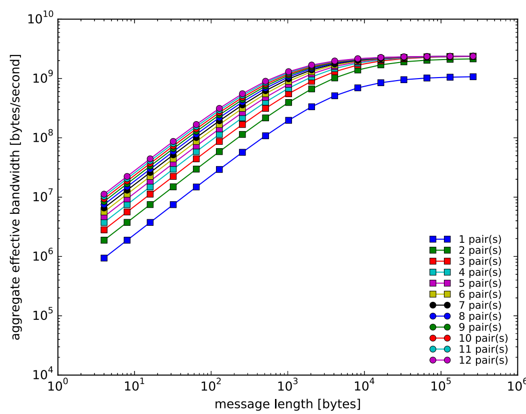


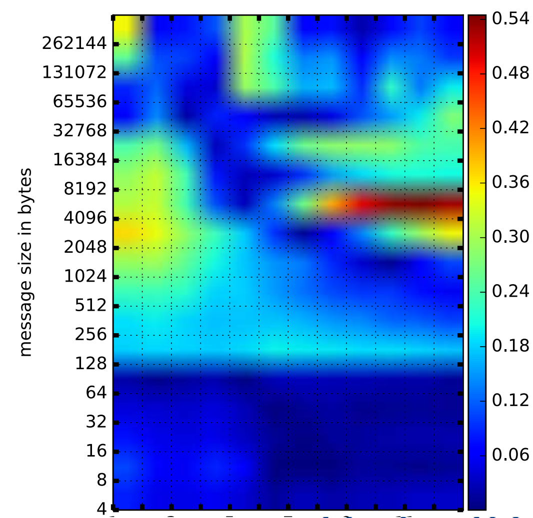
(b) Max-rate, three-parameter model (TCP).



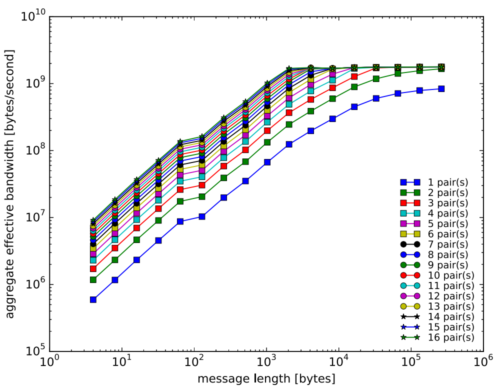
(c) Relative error (TCP).



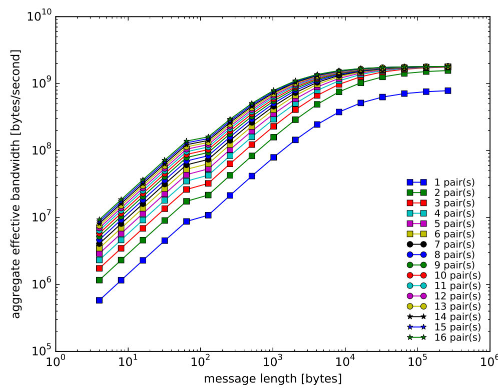




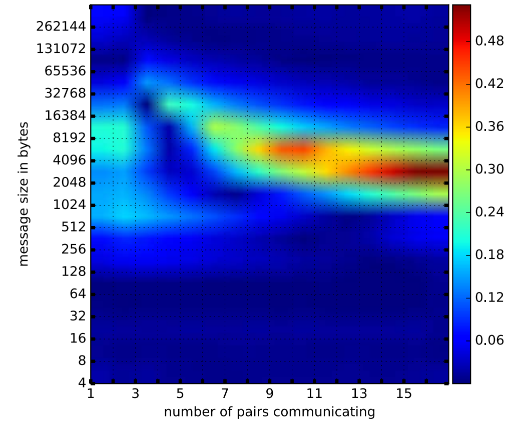
IBM BlueGene/Q



(a) Measured data.

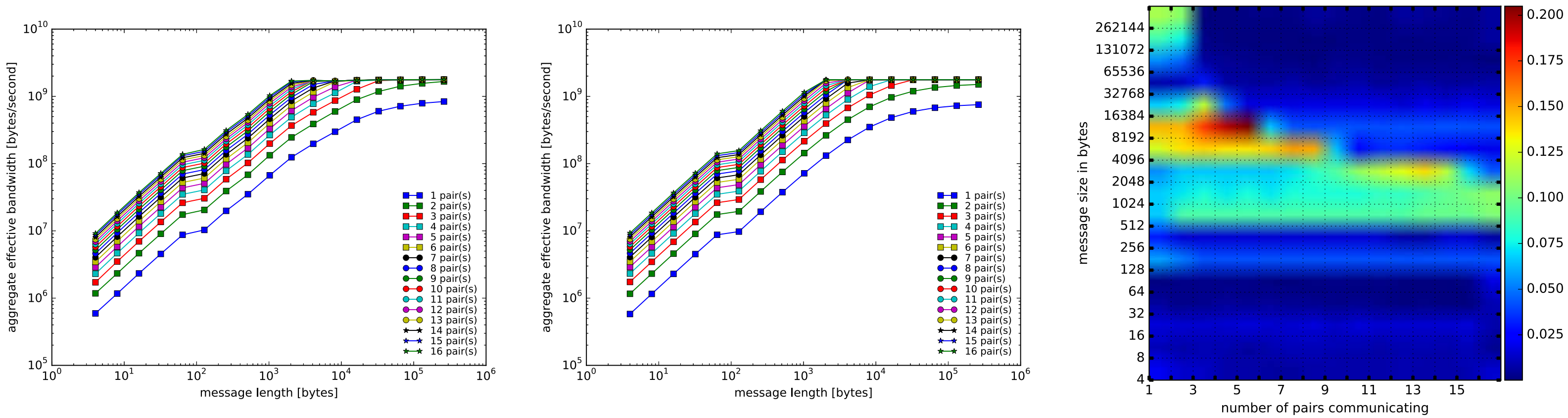


(b) Max-rate, three-parameter model.



(c) Relative error.

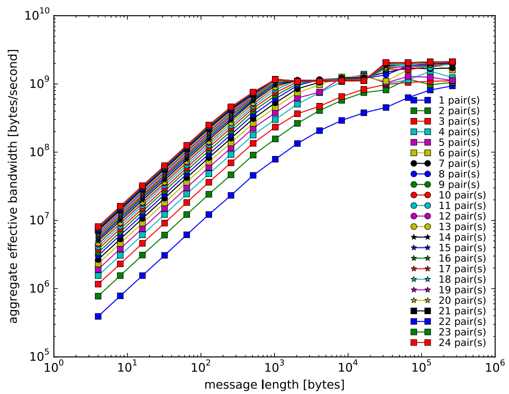
IBM BlueGene/Q (alternate model)



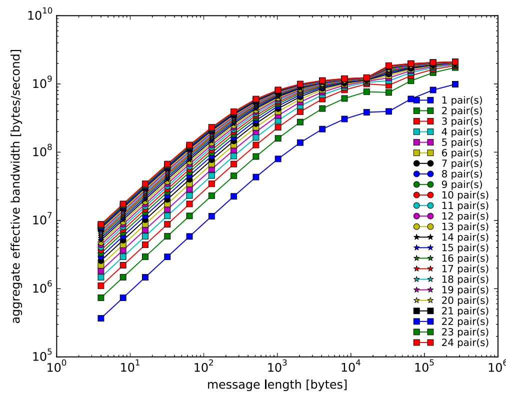
(a) Measured data. (b) Modified max-rate model. (c) Relative error of the model fit.

PARALLEL@ILLINOIS

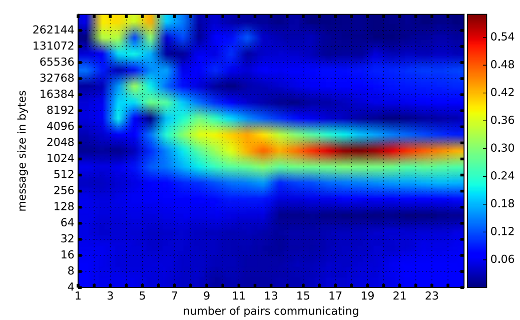
Cisco cluster (alternate network)



(a) Measured data.



(b) Max-rate, three-parameter model.



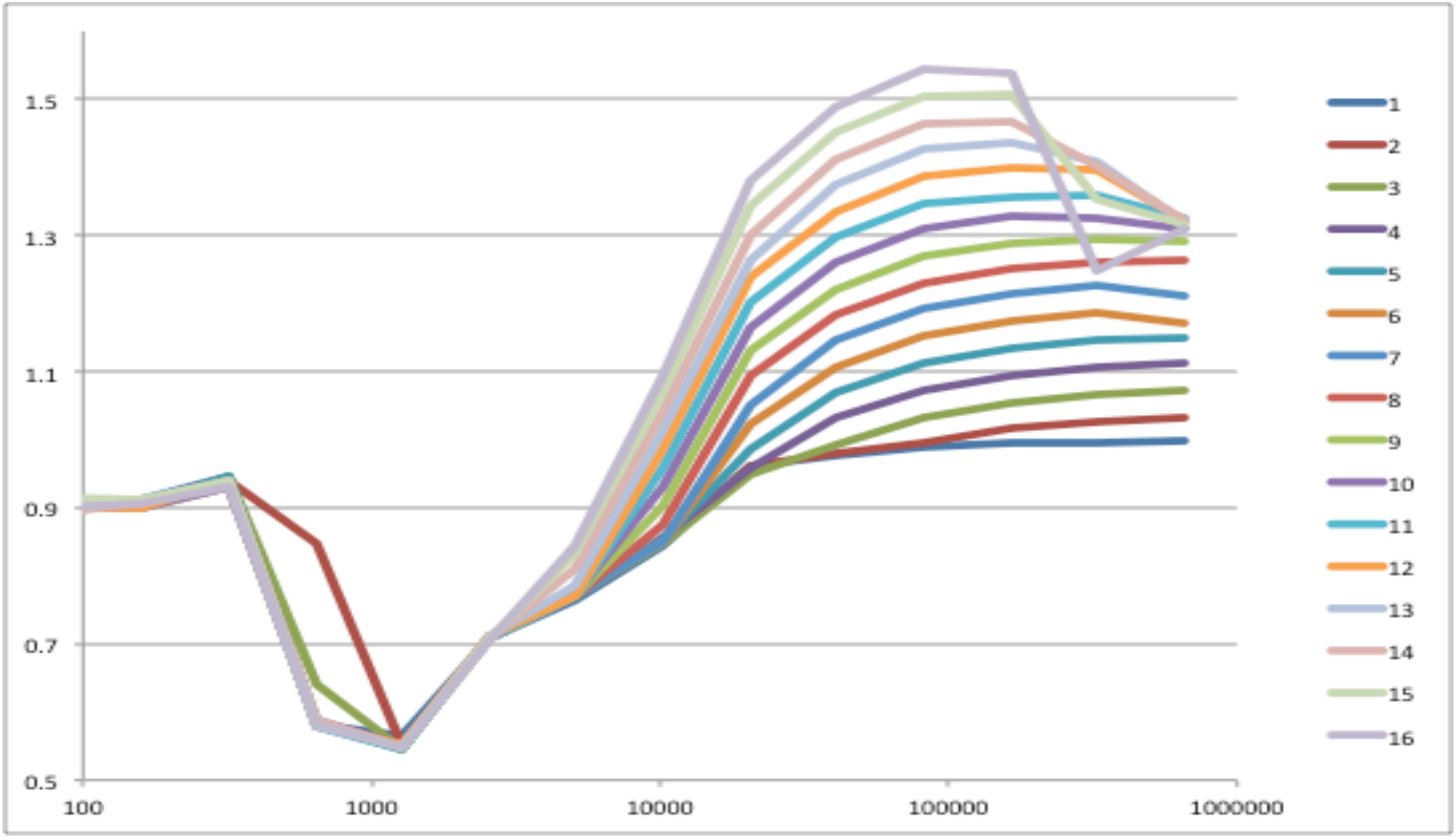
(c) Relative error.

Notes

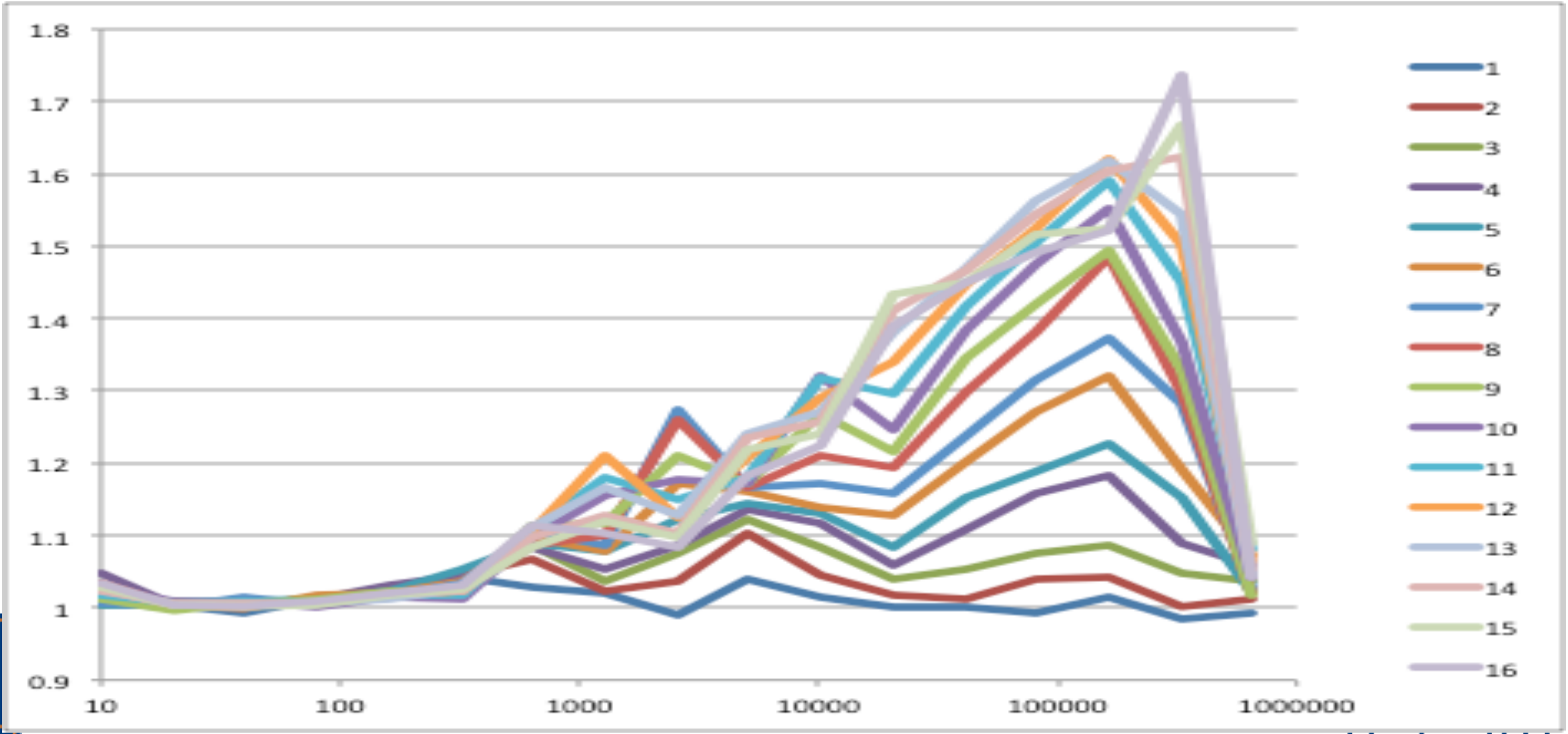
- Both Cray and BG/Q have inadequate bandwidth to support each core sending data along the same link
  - But BG/Q has more independent links, so it is able to sustain a higher effective "halo exchange"

Does Communication Overlap Help? (BG/Q)

- Graph show performance advantage to using overlap as a function of work size (message size = 1/10 work)



Does Communication Overlap Help? (Cray XE6)



PARALLEL@ILLINOIS

Some Notes on Performance Modeling

- Form an abstract machine model
  - ♦ This is the "execution model"
- Give it a simple performance model
  - ♦ Try to minimize the number of parameters – two is often enough
- *Test your assumptions*
  - ♦ Refine your model but keep it simple
- You can't predict everything
  - ♦ What is that weird behavior for small messages and 4-6 processes?!

Modeling Communication

- For k processes sending messages concurrently from the same node, the correct (more precisely, a much better) time model is
  - $T = s + kn/Min(R_{NIC\text{-}NIC}, kR_{CORE\text{-}NIC})$
- Further terms improve this model, but this one is sufficient for many uses