

# BLUE WATERS

SUSTAINED PETASCALE COMPUTING

## Toward Performance Models of MPI Implementations for Understanding Application Scaling Issues

Torsten Hoefler, William Gropp, Rajeev Thakur, and  
Jesper Larsson Träff

EuroMPI 2010, Stuttgart, Germany, Sept. 13<sup>th</sup> 2010



GREAT LAKES CONSORTIUM  
FOR PETASCALE COMPUTATION

## Imagine ...

- ... you're planning to construct a multi-million Dollar Supercomputer ...
- ... that consumes as much energy as a small [european] town ...
- ... to solve computational problems at an international scale and advance science to the next level ...
- ... with “hero-runs” of [insert verb here] scientific applications that cost \$10k and more per run ...

... and all you have (now) is ...

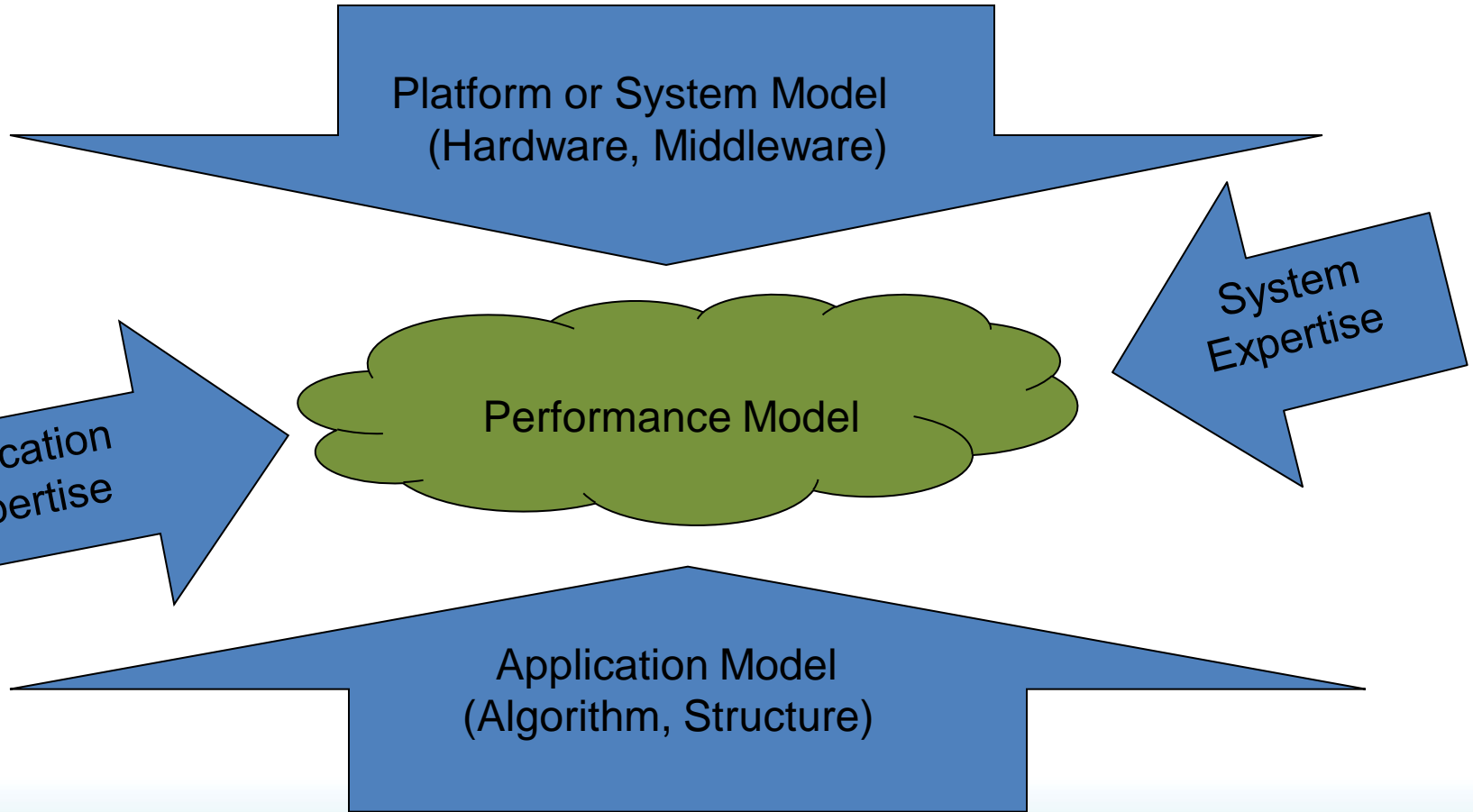


- ... then you better plan ahead! (same for Exascale)

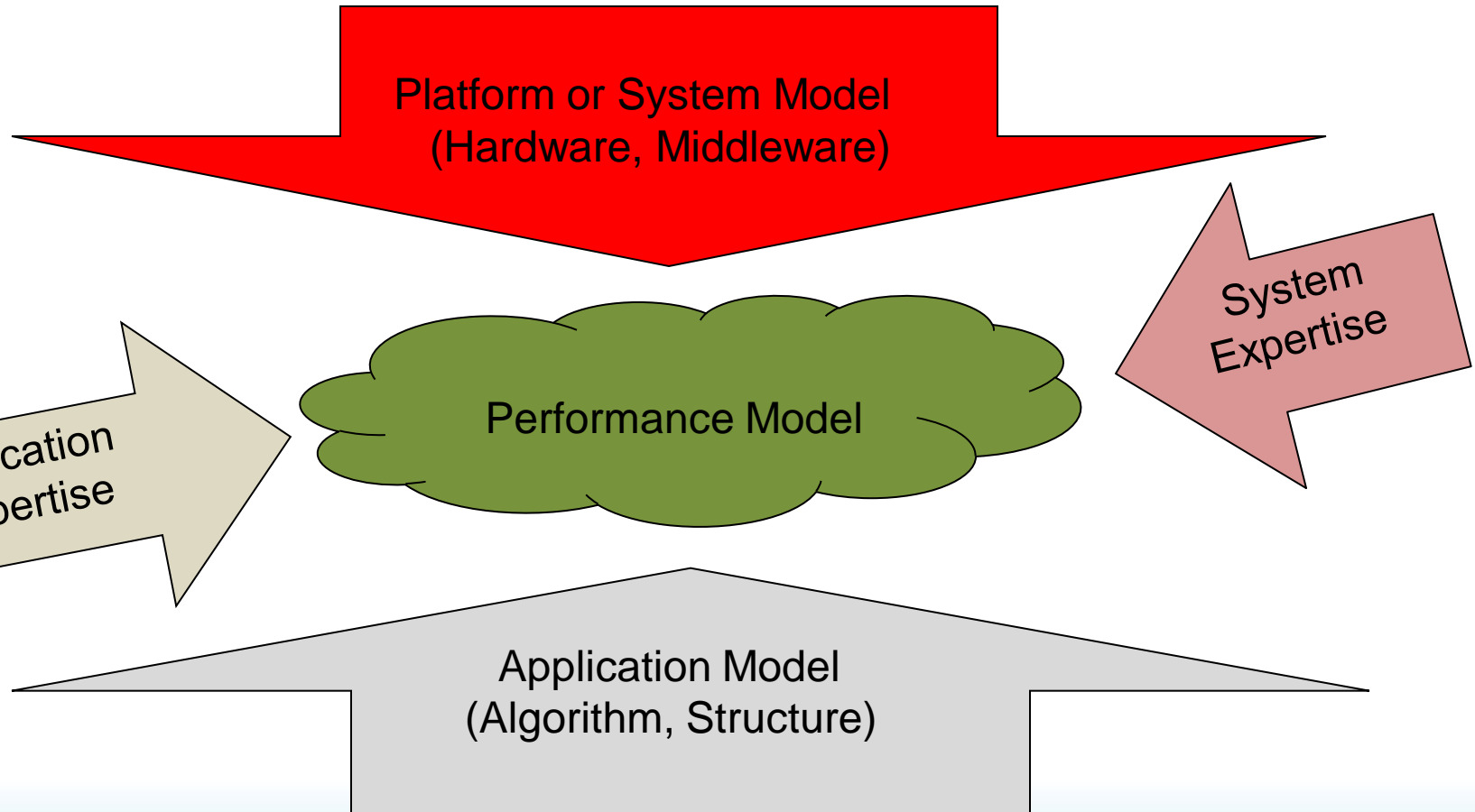
## Application Performance Modeling

- Analytic expression for expected runtime
  - Often exists as *folklore* in people's minds
- Can estimate scalability of codes and algorithms
  - Folklore: Alltoall(v) is not scalable
- Helps to make design decisions
  - E.g., estimate trade-off between single-core performance and scalability
    - General principle: reduce communication with more/redundant computation

# Performance Modeling from 10,000 Feet



# MPI is part of the Platform Model



## MPI Performance Models are Critical

- Folklore exists, e.g.,
  - Transmitting a message of size  $S = \Theta(S)$
  - MPI\_Alltoall on  $P$  processes =  $\mathcal{O}(P)$
- Model inaccuracies often insignificant for small  $S, P!$ 
  - Huge impact for large  $S, P$
- E.g., application models assume MPI\_Bcast or MPI\_Allreduce =  $\Theta(S \log(P)) \rightarrow$  **wrong**
  - A good implementation =  $\Theta(S + \log(P))$

## An Approach to Standardized Models

- Giving accurate estimates is a hard task
  - Users might see it as a contract
  - Vendors are hesitant to agree to or define contracts
- Many variables
  - CPU parameters (memory, speed, architecture, ...)
  - Network parameters (latency, bw, topology, ...)
  - Protocols (eager, rendezvous, ...)
- We propose hierarchical modeling
  - Various levels of accuracy



# The Four Levels of Accuracy

## 1. Asymptotic

- Asymptotic scaling only, e.g.,  $\Theta(S + \log(P))$

## 2. Dominant term exact

- Significant terms, e.g.,  $\beta S + \mathcal{O}(\log(P)) < 2\beta S + \mathcal{O}(\log(P))$

## 3. Bounded (parameterized)

- Specify bounds, e.g.,  $\beta S + \log(P) < T < 2\beta S + \log(P)$

## 4. Exact

- Exact model (if possible), e.g.,  $T_{BAR} = 0.95\mu s$

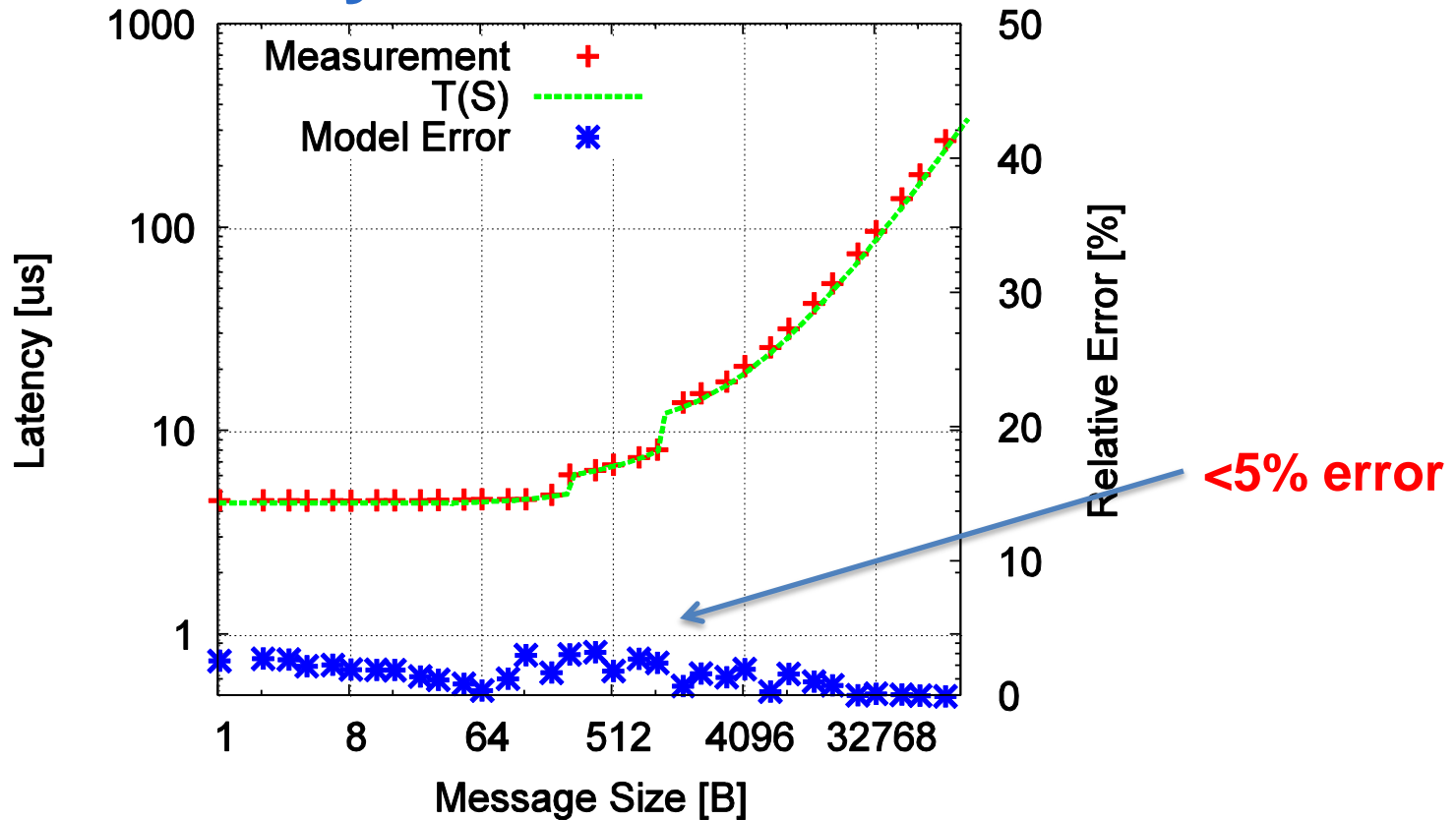
## Current Point-to-Point Models

- Asymptotic (trivial):  $\Theta(S)$
- Latency-bandwidth models:  $T = \alpha + S\beta$
- Need to consider different protocol ranges
- Exact model for BG/P:

$$T(S) = \begin{cases} 4.5\mu s + 2.67ns/B \cdot S & : S \leq 256B \\ 5.7\mu s + 2.67ns/B \cdot S & : 256B < S \leq 1024B \\ 9.8\mu s + 2.67ns/B \cdot S & : 1024B < S \end{cases}$$

- Used Netgauge/logp benchmark
- Three ranges: small, eager, rendezvous

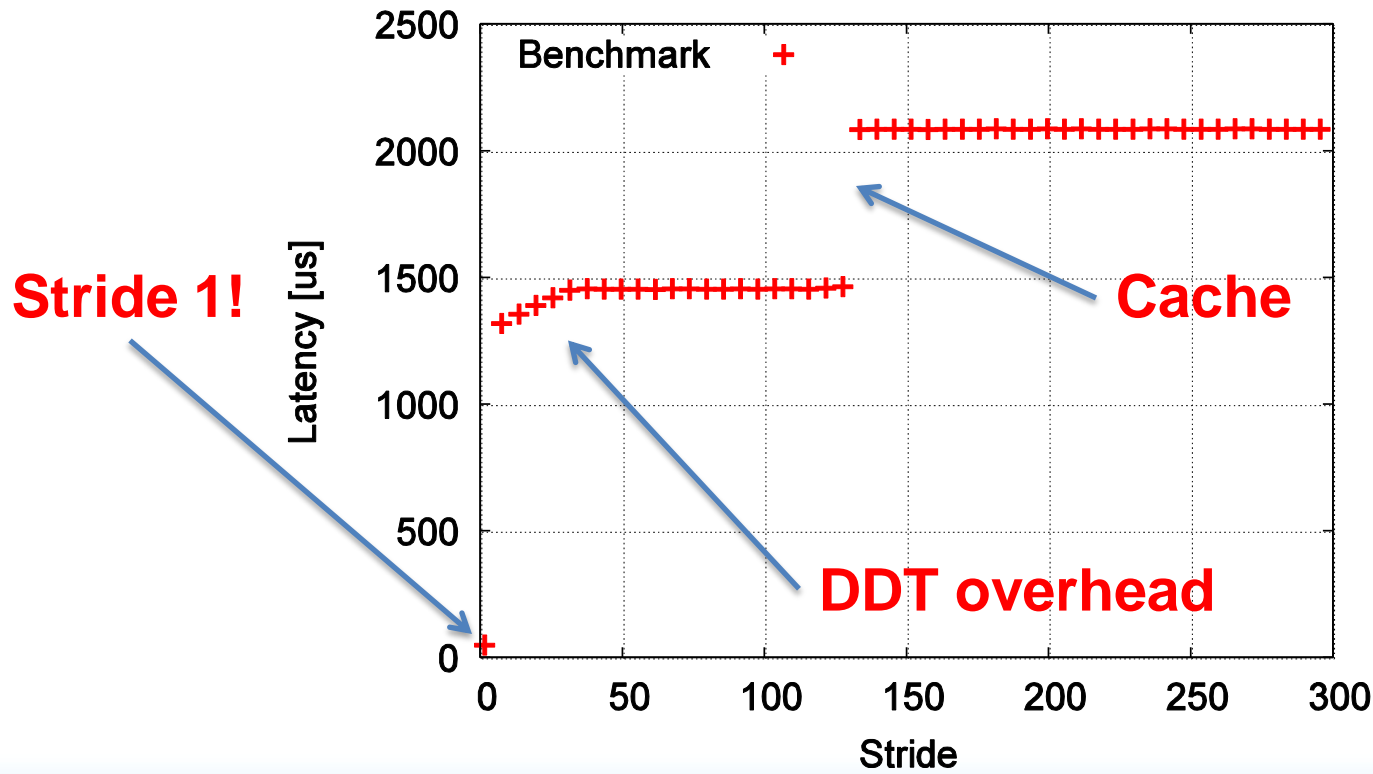
# Model Accuracy



- Looks good, but there are problems!

# The not-so-ideal (but realistic) Case I

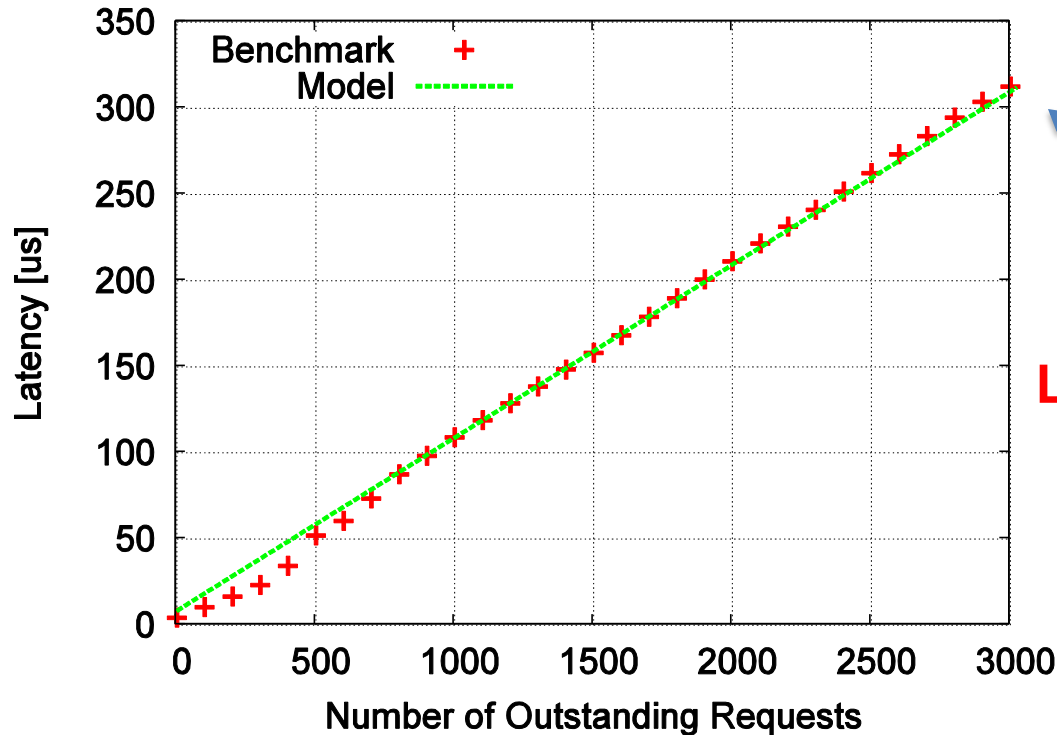
- Strided data-access (model assumed stride-1)



- Benchmark: Netgauge: one\_one\_dtype

## The not-so-ideal (but realistic) Case II

- Matching queue overheads (very common)



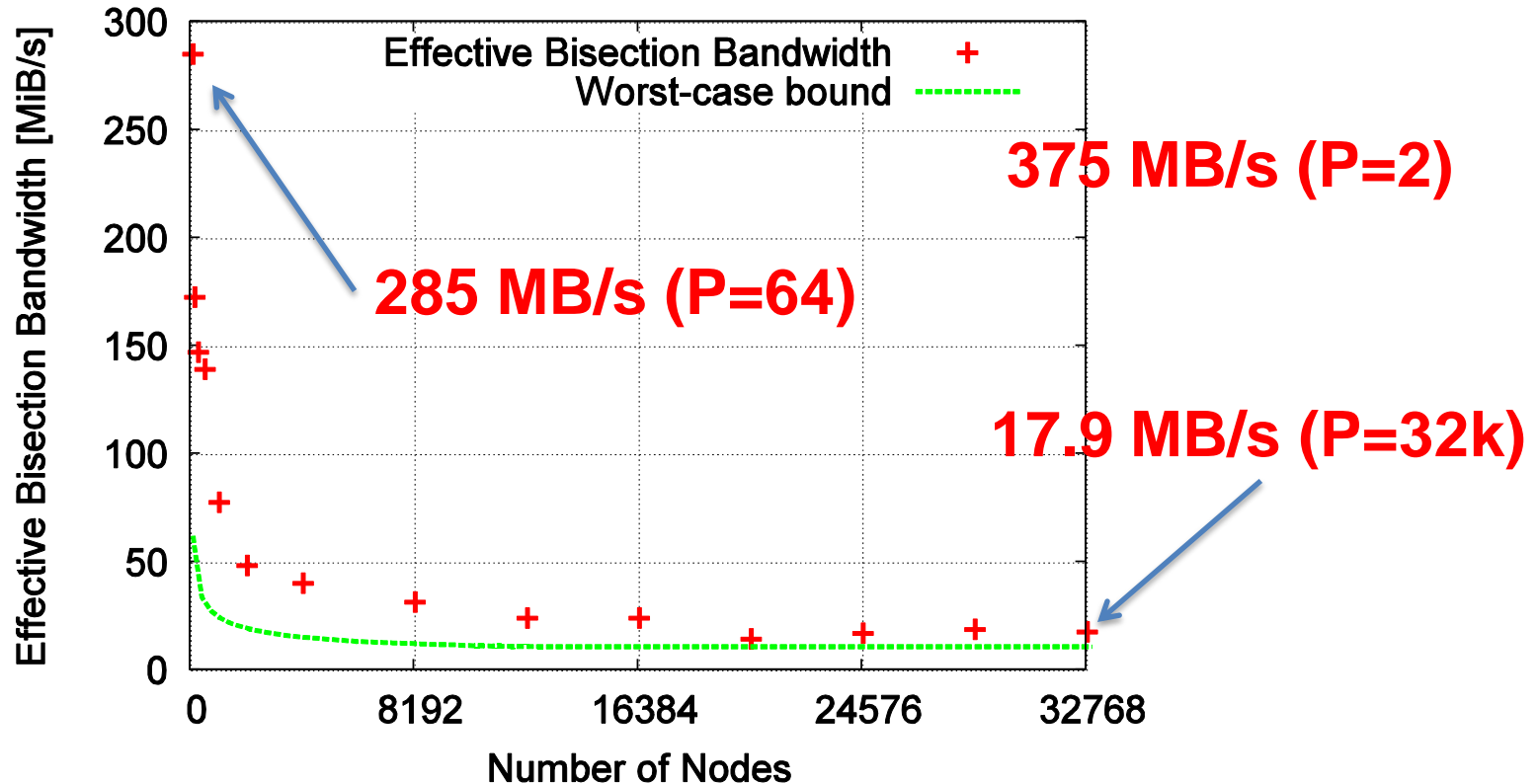
Latency factor of 35!

- R requests:  $T_{match}(R) \leq 100ns \cdot R$ ;  $T(13) \leq 1.3\mu s$ 
  - Benchmark: Netgauge/one\_one\_req\_queue

## The not-so-ideal (but realistic) Case III

- Congestion is often ignored
  - Very hard to determine but worst-case can be calculated
  - effective Bisection Bandwidth
    - Average bandwidth of a random perfect matching
- Upper bound is congestion-less (see model)
- Lower bound assumes worst-case mapping
  - Assume ideal adaptive routing (BG/P)
  - Congestion of  $\mathcal{O}(\sqrt[3]{P})$  per link

# Worst-case vs. Average Case Congestion



- Average seems to converge to worst-case (large P)
  - Benchmark: Netgauge/ebb

## Bounded Model with Congestion

$$9.8\mu s + 2.67^{ns/B} \cdot S \leq T(S, P) < 9.8\mu s + 2.67^{ns/B} \cdot S \cdot 3/2 \sqrt[3]{P}$$

- Only considers congestion
  - Needs to add datatypes, matching queue, ...
- Some parameters might be ignored
  - Typically application-specific
  - E.g., application only uses stride-1 access



## Collective Communication

- Crucial for estimating application scalability
- Often simpler to use than p2p models
  - Matching queue, synchronization and cong. hidden
- Simple latency-bandwidth:  $T = \alpha(P) + S \cdot \beta(P)$ 
  - Empirical parameterization, might be hard to use
- Build upon point-to-point models
  - E.g., LogGOPS model
  - More complex to derive (and often less precise)

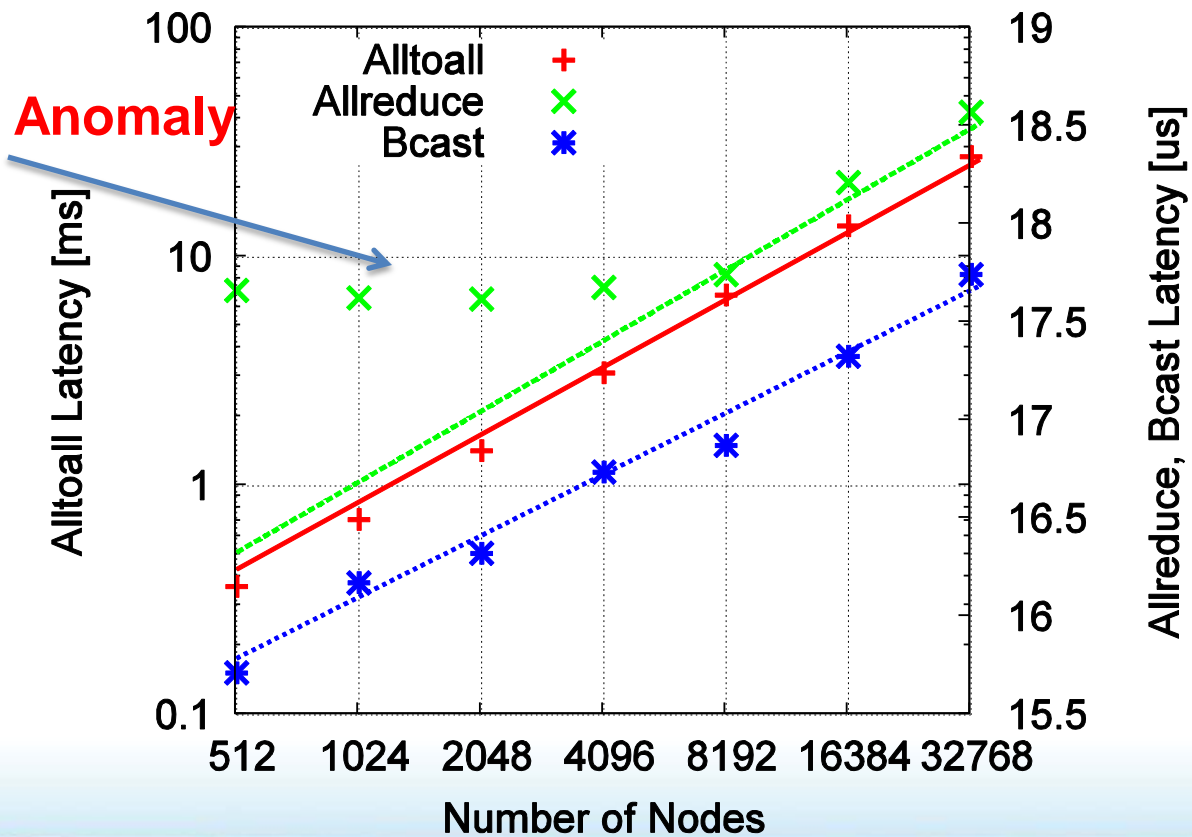
## Example: Small MPI\_Bcast, MPI\_Allreduce

- Handled by Collective (Tree) network in BG/P
- MPI\_Bcast:  $T_{BC}(P, 8) = \alpha_T + \beta_T^{BC} \log_2(P)$ 
  - $\alpha_T$  = startup overhead
  - $\beta_T$  = cost per stage
  - Empirically determined:  $\alpha_T = 13\mu s$ ,  $\beta_T^{BC} = 0.31\mu s$
- MPI\_Allreduce:  $T_{ARE}(P, 8) = \alpha_T^{SUM} + \beta_T^{SUM} \log_2(P)$ 
  - Empirically determined:  $\beta_T^{SUM} = 0.37\mu s$

# Small MPI\_Alltoall

- Direct sends (**not optimal!**):  $T_{A2A}(P, 8) = \alpha + g(P - 1)$

Performance Anomaly



## Large MPI\_Bcast

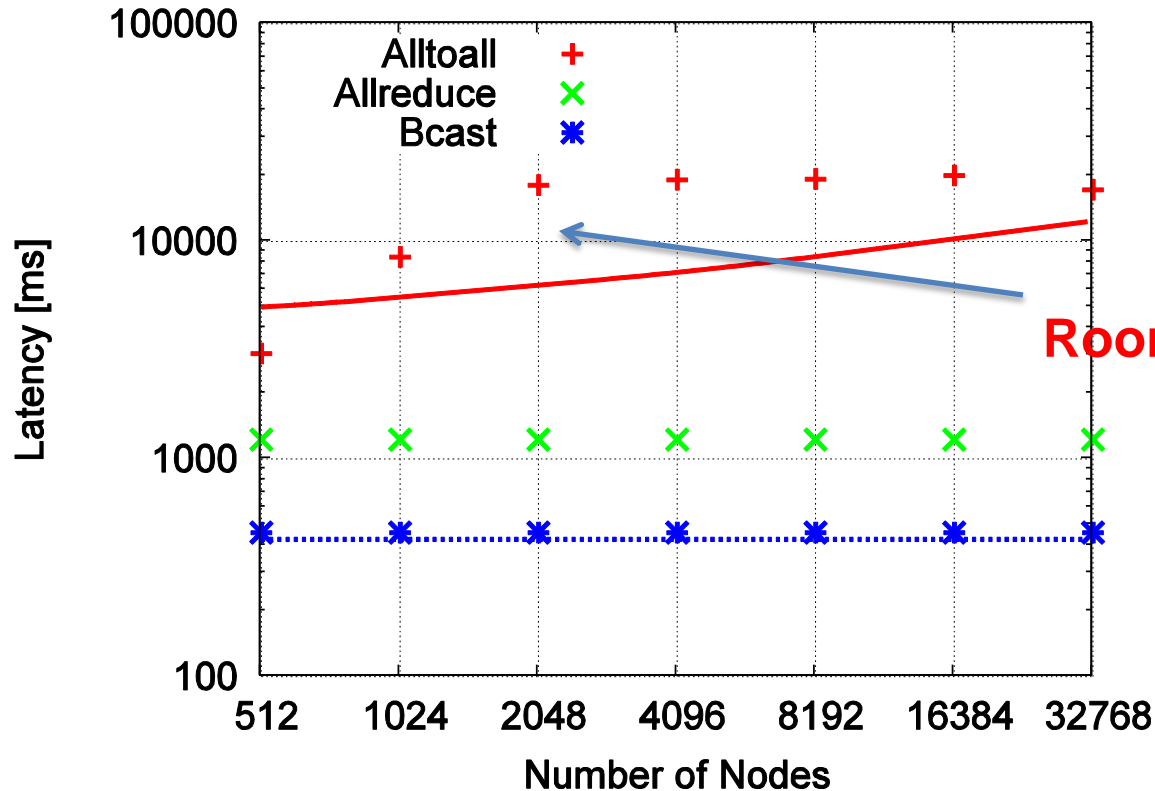
- Exact model for small MPI\_Bcast
- Large bcast needs  $T_{BC}(P, S)$  to reach all processes
  - Extend with bandwidth term
- Assume using all six torus links (2.67ns/B/link)
  - $T_{BC}(P, S) = \alpha_T + \beta_T^{BC} \log_2(P) + \frac{2.67}{6} ns/B \cdot S$
- Based on first-principles (documentation+LogGP)
  - No fit necessary
  - Still accurate for large S (see next slides)
    - Middle-ground can be covered by less accurate models

## Large MPI\_Alltoall

- Algorithm sends to all targets (random permutation)
  - Uses all six links, hits bisection bandwidth
  - Model:  $T_{A2A} = (P - 1)g + SG \cdot \max\{\frac{P-1}{6}, C(P)\}$
- Simple counting argument for  $C(P)$ 
  - Assume  $k^3$  processor grid, 1-d distance  $d = \frac{k-1}{2}$
  - Total number of occupied links:  

$$N(k) \leq k^3 \cdot 2 \sum_{x=0}^d 2 \sum_{y=0}^d 2 \sum_{z=0}^d (x + y + z) = k^3 12d (d + 1)^3 = \mathcal{O}(k^7)$$
  - Total number of links:  $6k^3$
  - Congestion per link:  $C(P) \leq \sqrt[3]{P}(\sqrt[3]{P}/2 + 1)^3 = \mathcal{O}(P\sqrt[3]{P})$ 
    - Model:  $T_{A2A} \geq g(P - 1) + SG\sqrt[3]{P}(\sqrt[3]{P}/2 + 1)^3$

# Results for Large Collectives



Room for improvement?

- No model for MPI\_Allreduce (constant)
- Lower-bound for MPI\_Alltoall holds

## Takeaways, Questions & Discussion

- **MPI Libraries should provide performance models!**
  - Different levels allow accuracy/effort trade-off
    - Even black-box models work
  - Models greatly benefit application design
  - Models are useful to check performance consistency
  - Reasoning about large-scale
    - Is the current MPI performance sufficient?
  - Model design is a community effort!
    - More research needed!

