# Enhancing the Communication Performance Models for SMPs

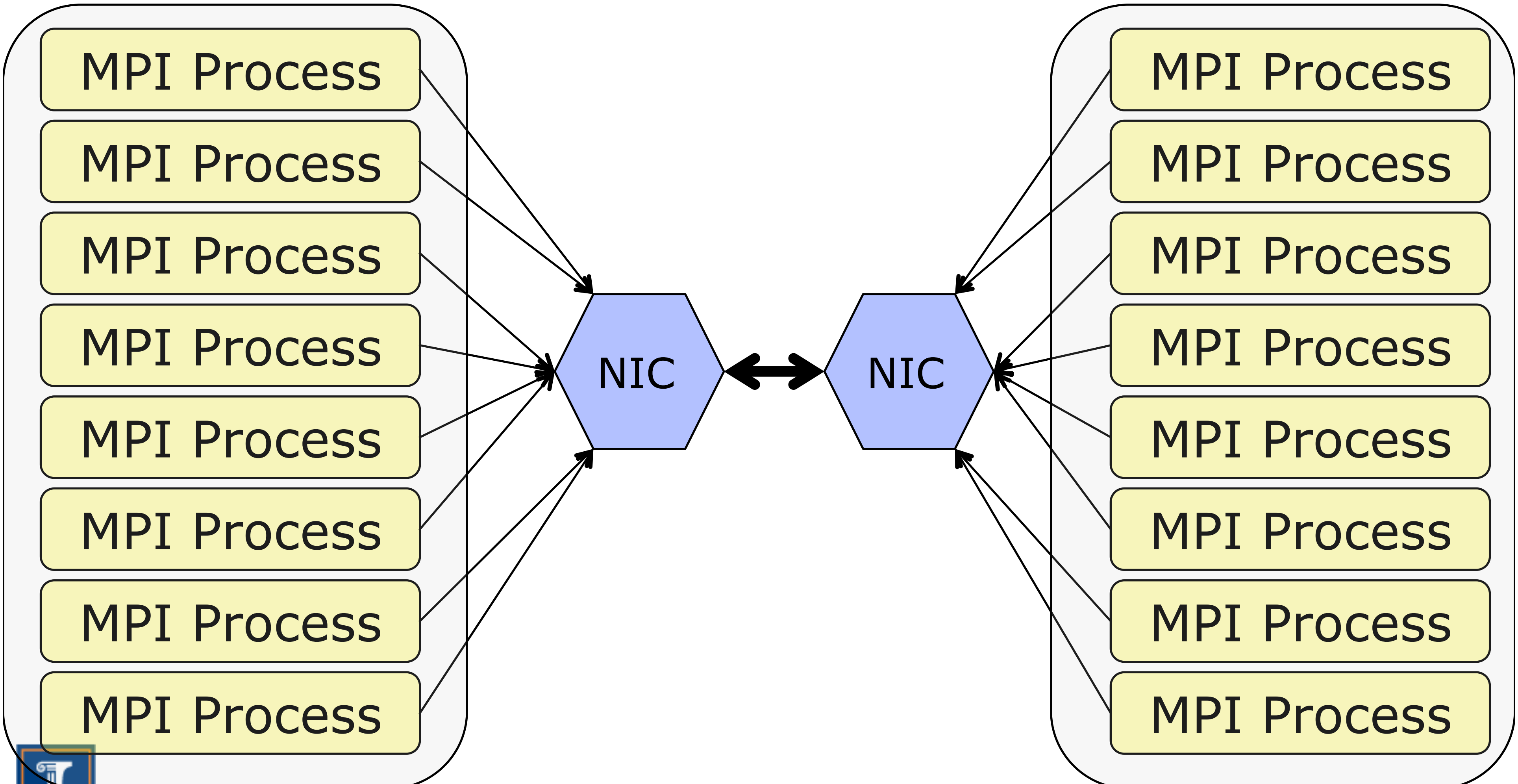

William Gropp

www.cs.illinois.edu/~wgropp

# 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?

# 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

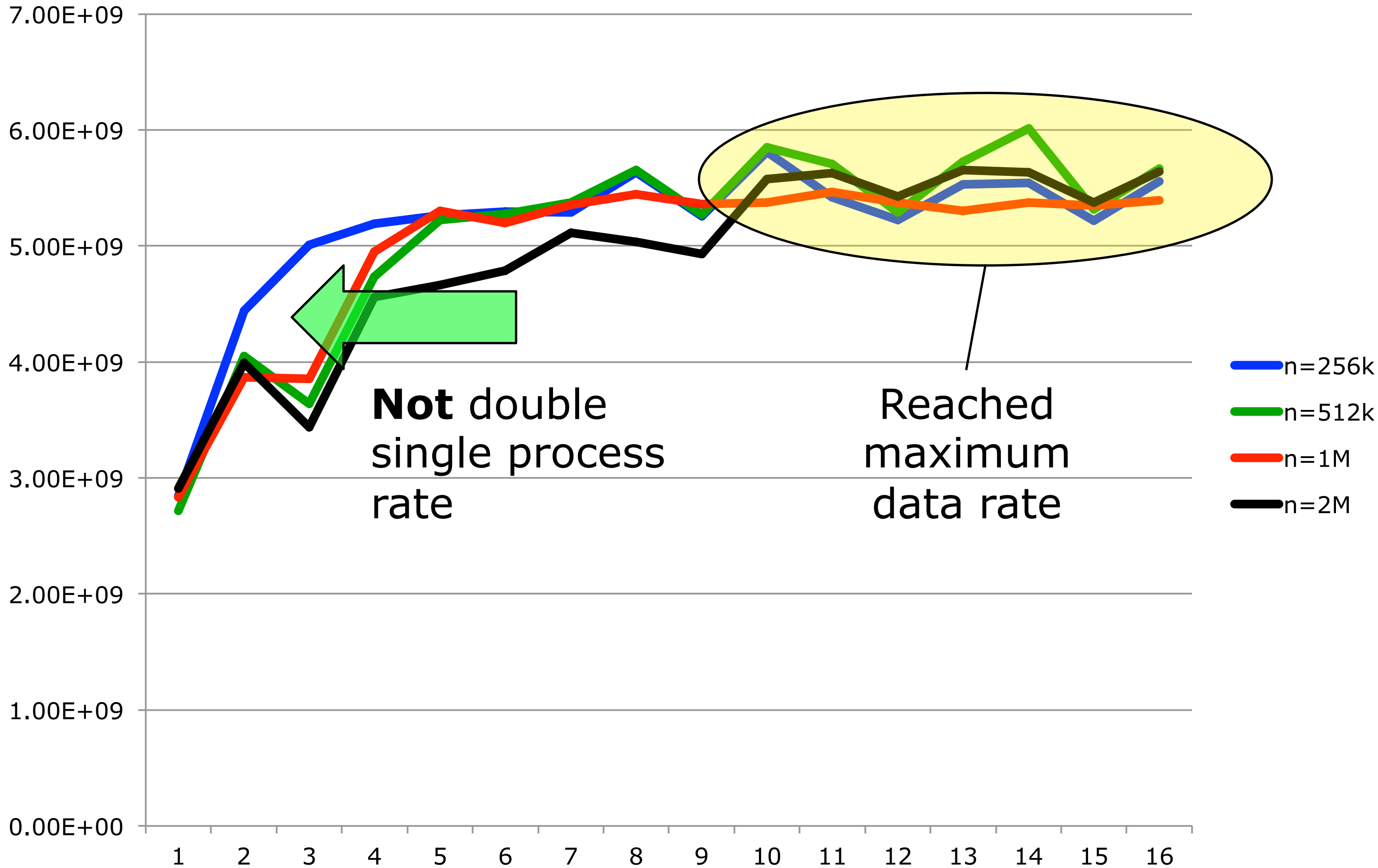
# A Slightly Better Model

- 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
  - $\min(R_{NIC\text{-}NIC}, kR_{CORE\text{-}NIC})$
- Thus
  - $T = s + kn/\mathrm{Min}(R_{NIC\text{-}NIC}, kR_{CORE\text{-}NIC})$
- Note if $R_{NIC\text{-}NIC}$ is very large (very fast network), this reduces to
  - $T = s + kn/(kR_{CORE\text{-}NIC}) = s + n/R_{CORE\text{-}NIC}$

# Observed Rates for Large Messages






PARALLEL@ILLINOIS

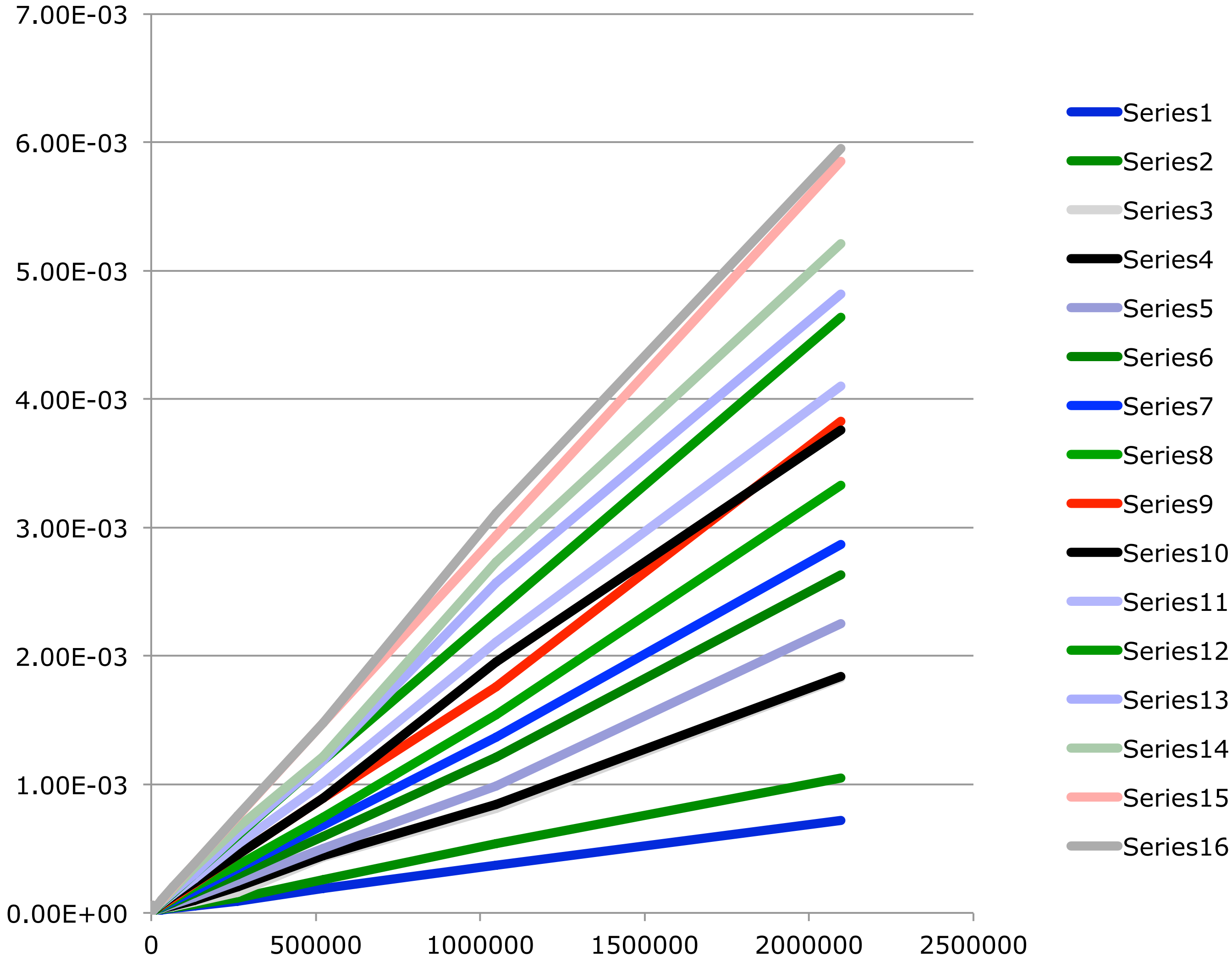
# A Slight Refinement

- Assume that handling more than one communication in the NIC requires a little extra overhead
  - This is pretty arbitrary but we'll see it sometimes matches the data
  - $T = s + kn/\mathrm{Min}(R_{NIC\text{-}NIC}, R_{CoreBase} + (k-1)R_{CoreIncr})$
  - If $R_{CoreBase} = R_{CoreIncr}$, reduces to the previous forumula

# An Example From Blue Waters

- Experiment:
  - 2 nodes, 1 MPI process per core-module
  - Ping-pong test, with k processes on one node sending to k processes on an adjacent node

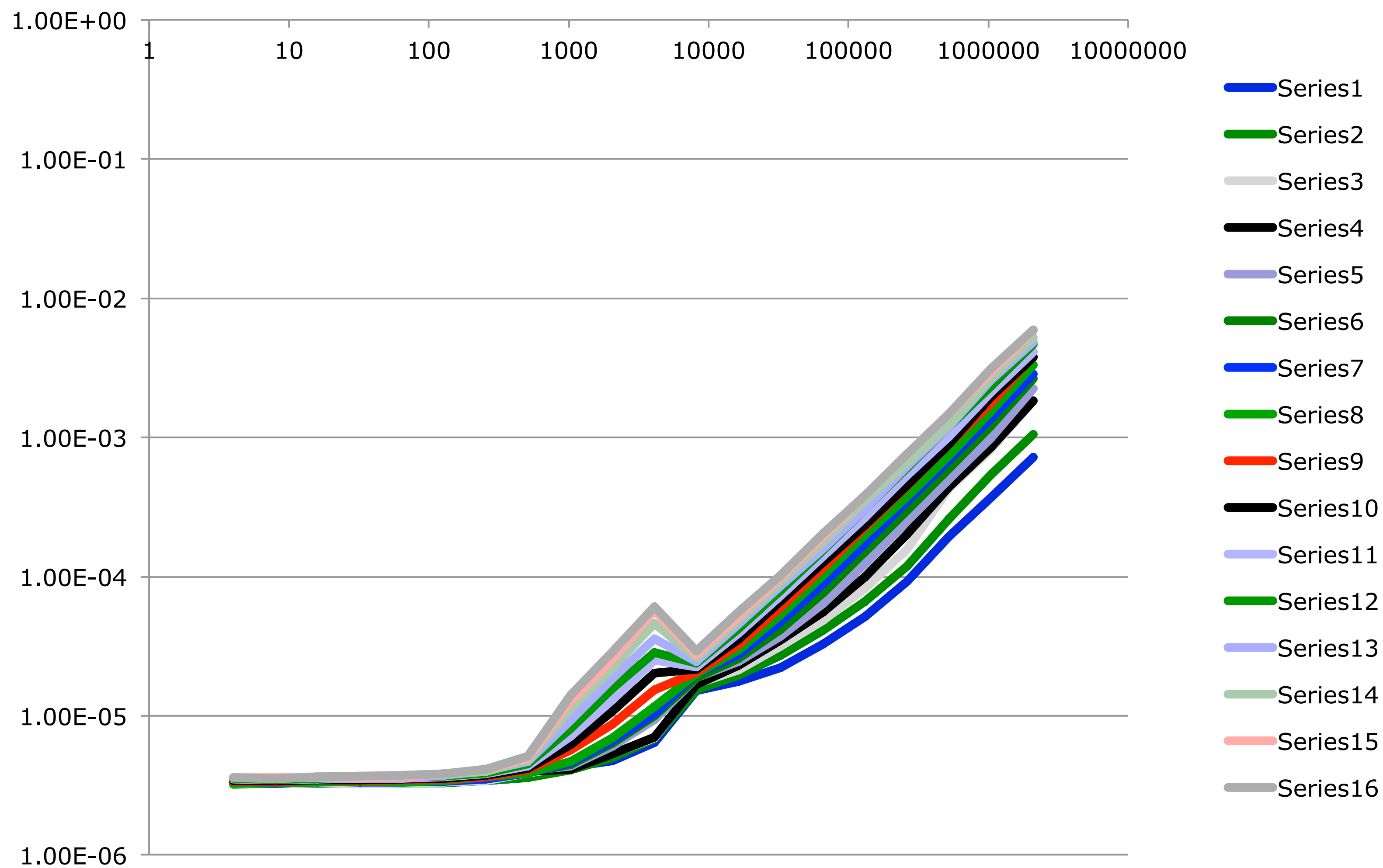
# Time for PingPong with k Processes






PARALLEL@ILLINOIS

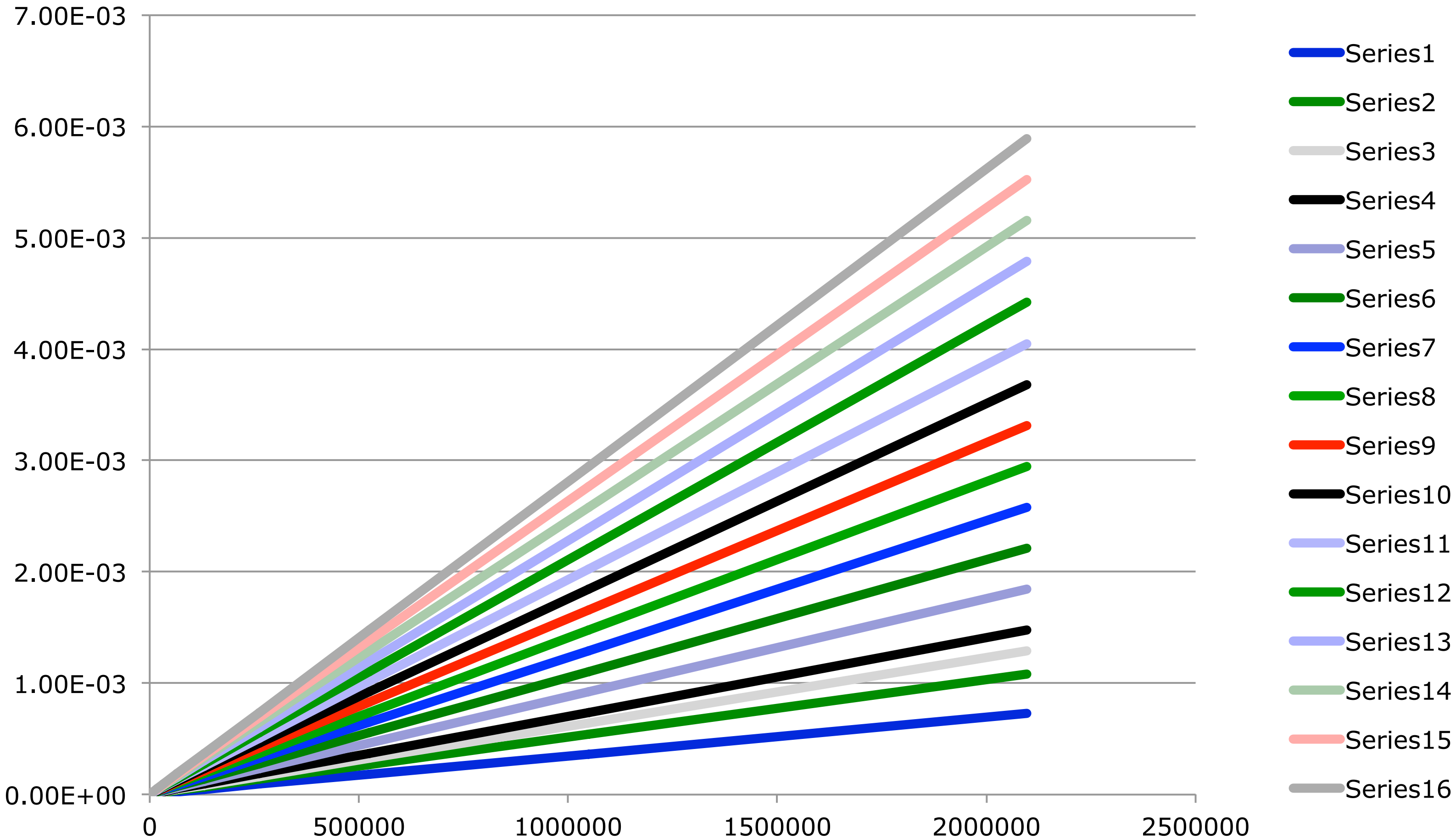
# Time for PingPong with k Processes



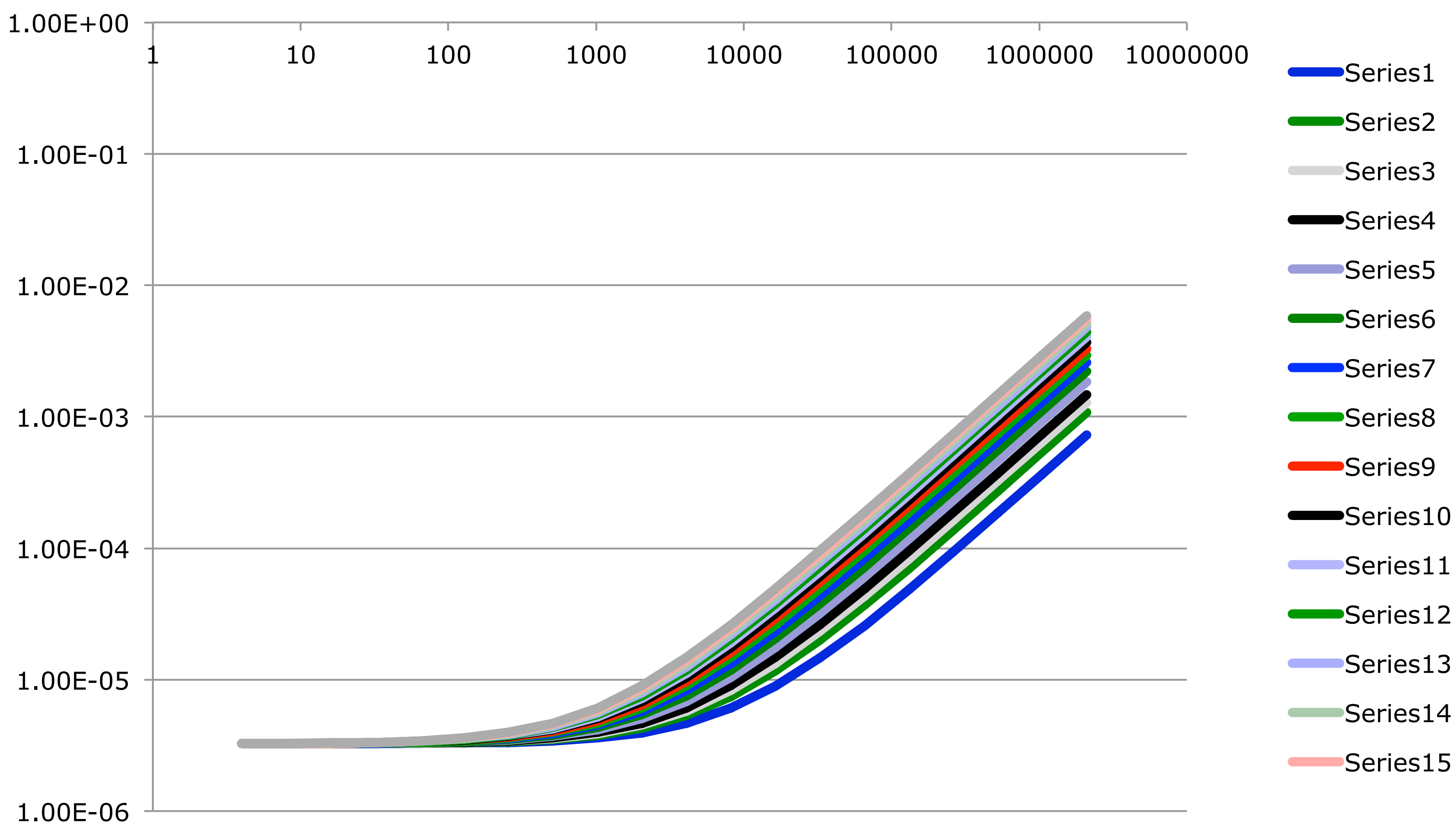
# New Model (Full PingPong Time)

- s = 3.26 usec
  - For a single send/receive, use half of this
- $R_{NIC\text{-}NIC}$ = 5.7 GB/sec
- $R_{CoreBase}$ = 2.9 GB/sec
- $R_{CoreIncr}$ = 1GB/sec
- Note that these are rough numbers for illustration only
  - Not numerical fit to the data – "eyeball norm" only

# Model Time Estimate



# Model Time Estimate

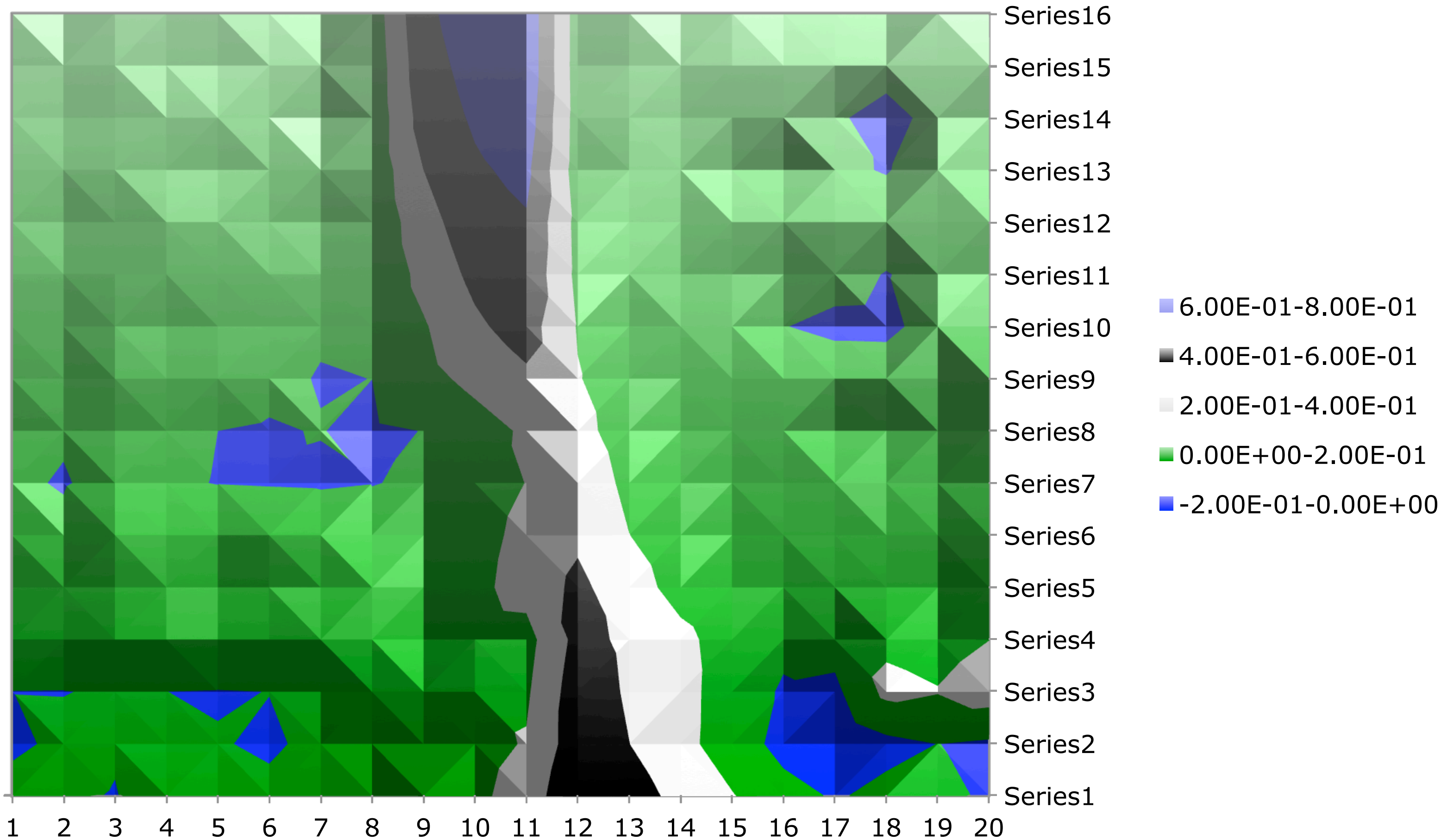



PARALLEL@ILLINOIS

# Notes on Model

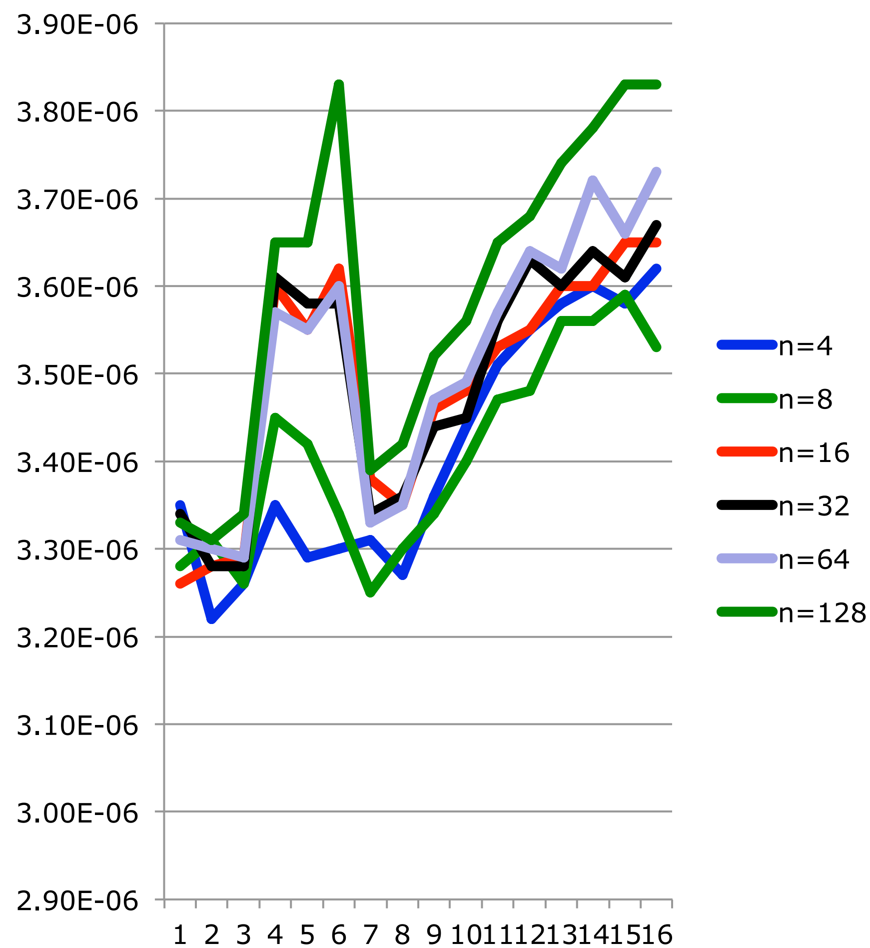
- This model ignores the transition between eager and rendezvous
  - Like logGP model, different method for moving large methods may have different rate
- Maximum in formula complicates fit
  - No longer simple linear least squares problem
- Blue Waters nodes have two chips
  - The one chip is closer to the NIC than the other
- Another constraint is maximum memory bandwidth
  - Assumed higher than link rates

# Relative Error in Model



# Notes on Relative Error

- Typically less than 10%
- Highest error in region where eager to rendezvous occurs
  - As expected
- Model has no term for impact on latency (s)
  - Graph on left shows time for small messages vs. number of processes
  - Suggests similar term for latency (max(s0,s1+k*s2))



# 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?!