TIMO SCHNEIDER <TIMOS@INF.ETHZ.CH>

Communication Models

Project Presentations

- Next Monday (lecture) + Thursday (recitation)
- 10 min / Team (hard limit)
- Order is randomized!!! → everyone needs to have their presentation ready on Monday
- Make sure to address comments from the 1:1 meetings
- Send your presentation by monday morning 10:00 to Timo, Subject [DPHPC Team XX Presentation]

- Report due on 12. January, 23:59, send by email to Timo, Tal and Markus on CC
- Subject [DPHPC Team XX Report]
- Page limit: 6 pages, see course website for template.
- Grade: 50% report + presentations, meeting) + 50% exam (2hrs, written, no notes)

Why model communication performance?

- Making predictions, e.g., for algorithm selection.
- Performance debugging

What is communication performance?

- Bandwidth?

- A high-performance homing pigeon flies 100 km/h and can carry 75g
- MicroSD Card: 400 GB, 0.25g
- Pidgeon has BW of 266 Gb/s from Zurich to Bern (better than HPC network)

Problems?

Model: PRAM

- All processors can work in parallel
- Communication is free
- Cost: Number of steps/instructions
- Many cool algorithms, i.e., find maximum of N numbers in O(1) steps

- Drawbacks of this model?

S. Fortune and J. Wyllie Parallelism in Random Access Machines, 1978

Model: BSP

- All processors work in parallel for some time, then synchronize (superstep)
- In each superstep, processors can exchange h messages of size m
- Cost of superstep: m*h*1/g (g=Bandwidth)

- Communication is not free any more!

- What is the cost of a Broadcast in this model?
- Drawbacks of this model?

L. G. Valiant A Bridging Model for Parallel Computation, 1990

Alpha-Beta Model

- **Processor are not synchronized automatically, they exchange messages.**
- **Time to send/receive message of size s: T(s)=alpha+s*beta**
  - *Latency modelled by alpha*
  - *Bandwidth is modelled by beta*
- **While we are sending/receiving no other operations are possible**
- **Cost of a broadcast in this model?**

Bcast in Alpha-Beta

- Root sends to P-1 others: T(s) = (P-1)(alpha+s*beta)
- Most processes do nothing!

- K-ary tree: T(s) = k*(alpha+s*beta)*log(k,P)
- Optimal k? (assume s=1)
- $0 = \frac{ln(P)\cdot k}{ln(k)}\frac{d}{dk} = \frac{ln(P)ln(k)-ln(P)}{ln^2(k)} \rightarrow k = e = 2.71...$

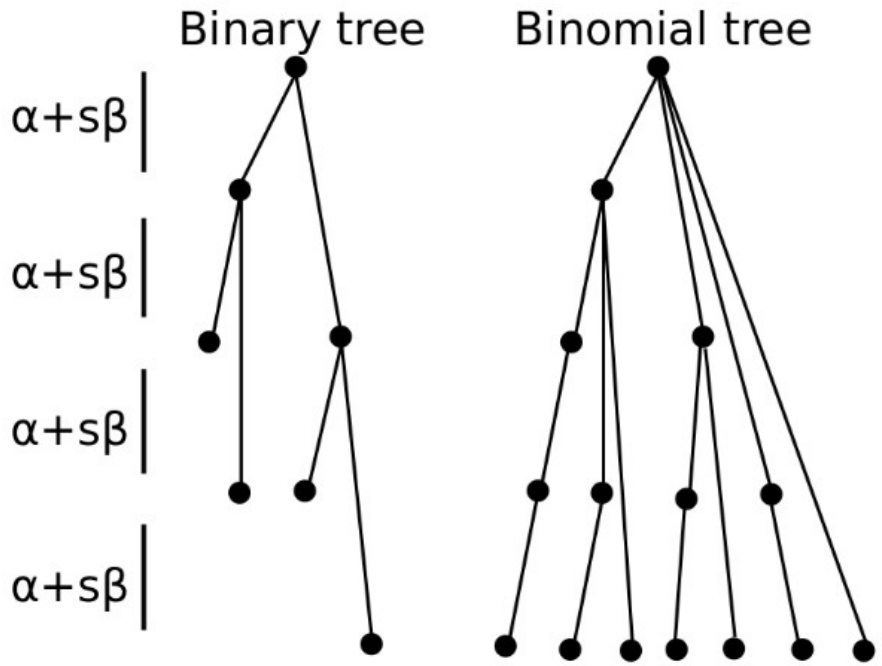
Bcast in Alpha-Beta

The cost of a **binary** tree broadcast of a message of size s is

$$2(\log_2(p+1) - 1) \cdot (\alpha + s \cdot \beta)$$

The cost of a **binomial** tree broadcast of a message of size s is

$$\log_2(p+1) \cdot (\alpha + s \cdot \beta)$$



Bcast in Alpha-Beta

- Discussion: Is this optimal?

Bcast in Alpha-Beta: Large s

- If s is large, our k-ary tree algorithm leaves injection bandwidth on the table.
- Idea: Pipelining – split message in segments of size z
- Processor i sends to i+1
- Runtime = T(s) = (P-2+s/z)*(alpha+z*beta)
-

- For P=4, alpha=10, beta=1, s=10^6, z=10^5 this is about 2x faster than binomial tree!
- Exercise: What is the optimal z?

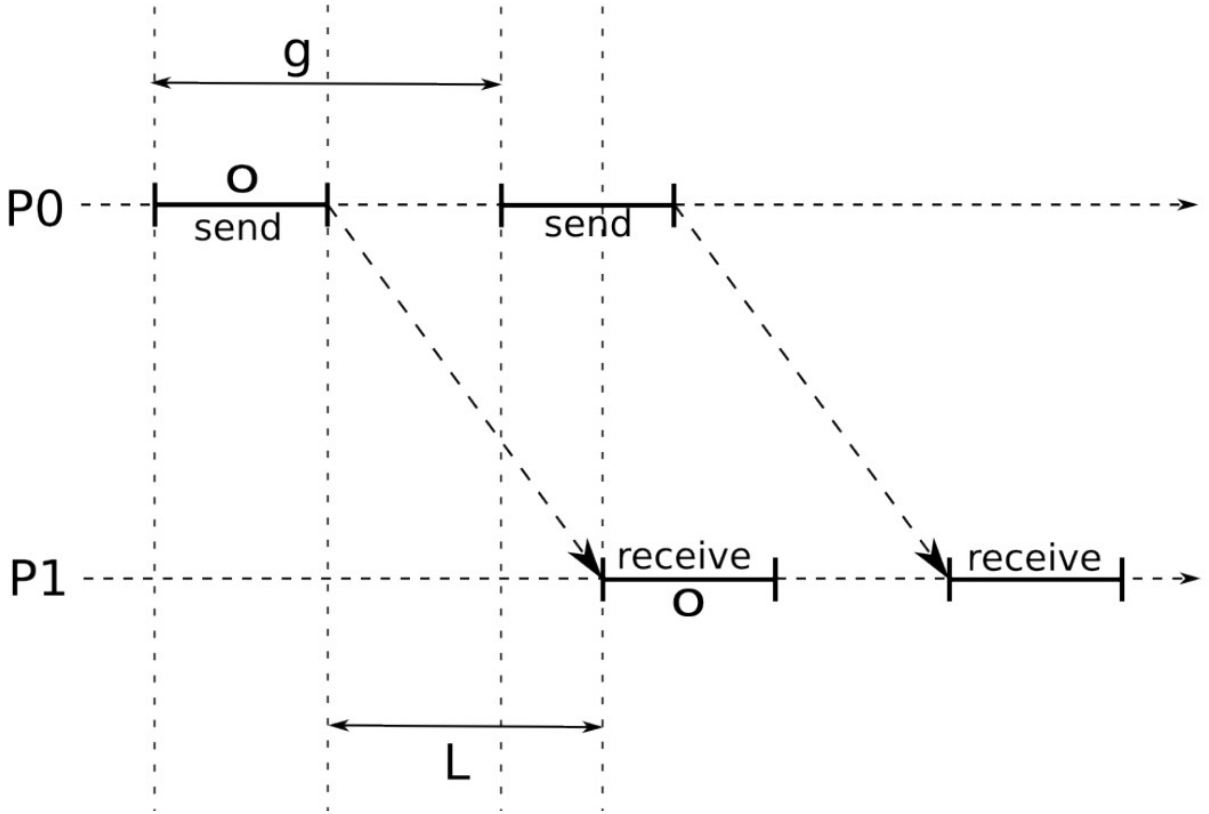
Alpha-Beta Model

- **Easy to work with, e.g., compute optimal tree radix, segment size, etc.**
- **Matches the messaging paradigm, no hidden features we don't have in practice, such as synchronization**

- ***No overlap between sending/receiving or communication and computation***
- ***Often does not match experimental result – NICs are complex!***

LogP Model

- L=Latency
- o=host overhead on sending/receiving CPU
- g=gap between messages, imposed by NIC
- P=number of processors in the network



LogP Model

- **Sending a single message**
  - *2o+L*
- **Round-Trip**
  - *4o+2L*
- **Sending n messages**
  - *L+(n-1)max(g,o)+2o*
  - *often simplified in practice, i.e., assume o>g, drop -1, etc.*

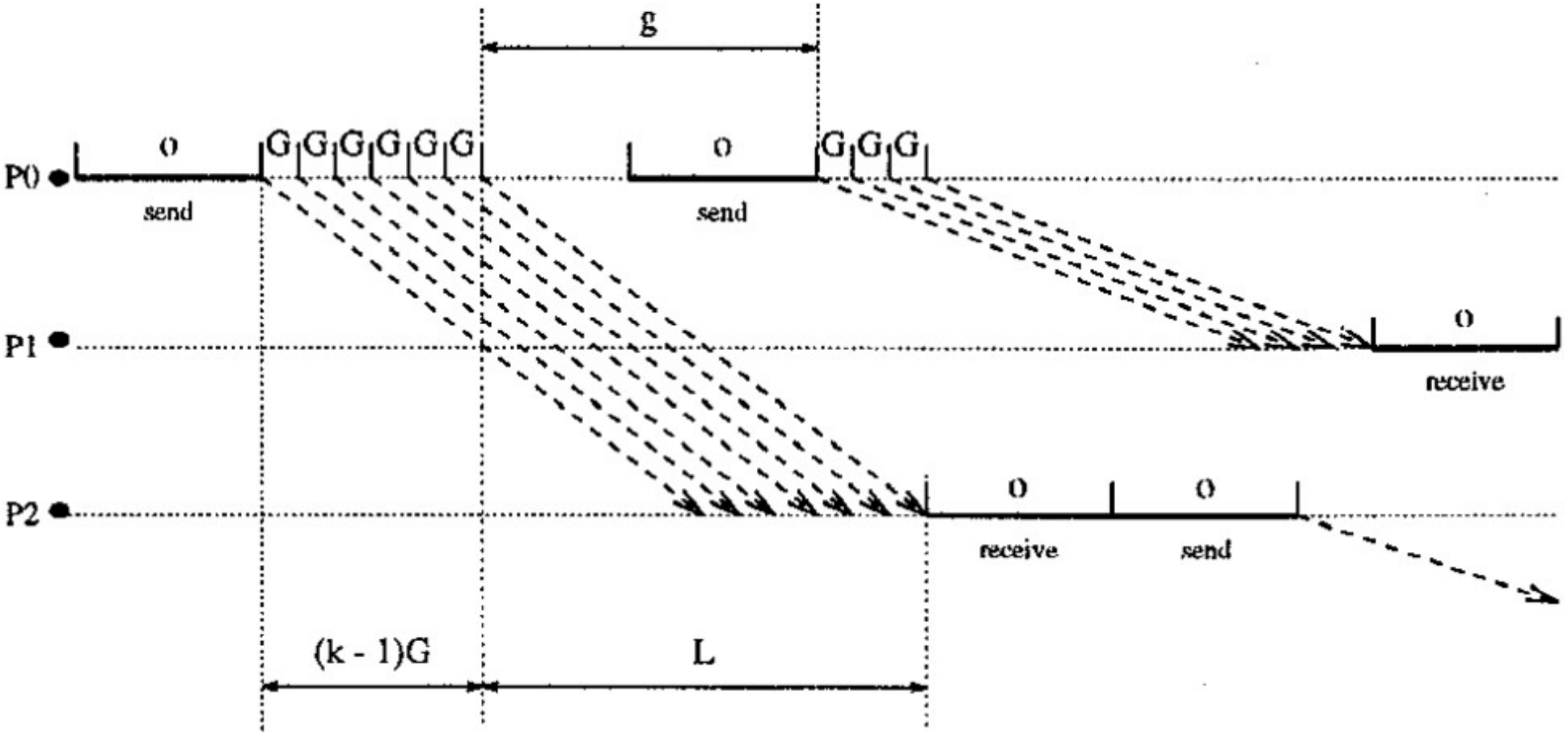
LogP Model: Realistic?

- We can only send messages of a single size!
- The idea was: networks use packets – but packets can vary in size, HW usually offloads packetizing

- Idea: Add message size back into LogP!

LogGP

- L,o,g,P are the same as in LogP
- G is the inverse of bandwith (Cost per Byte)



LogGP

- **Sending a single message of size s**
  - *2o+L+(s-1)G*
- **Round trip of a single message of size s**
  - *4o+2L+2(s-1)G*
- **Sending n messages of size s**
  - *L+(n-1)max(g,o)+n(s-1)G+2o*

LogGP: Simple Observations

- Assuming max(o,g) > G, sending large messages is good!
- Splitting messages only helps when pipelining (cf. LogP)
-

LogGP: Scatter

- Now instead of looking at Broadcast, let's look at Scatter
- Scatter: Single root, different data (size s) for each of the P processors
- Simple idea: send a message to each P-1 processors

- Runtime: $T(s) = g(P-2) + G(P-1)(s-1) + L$
- Can we do better?

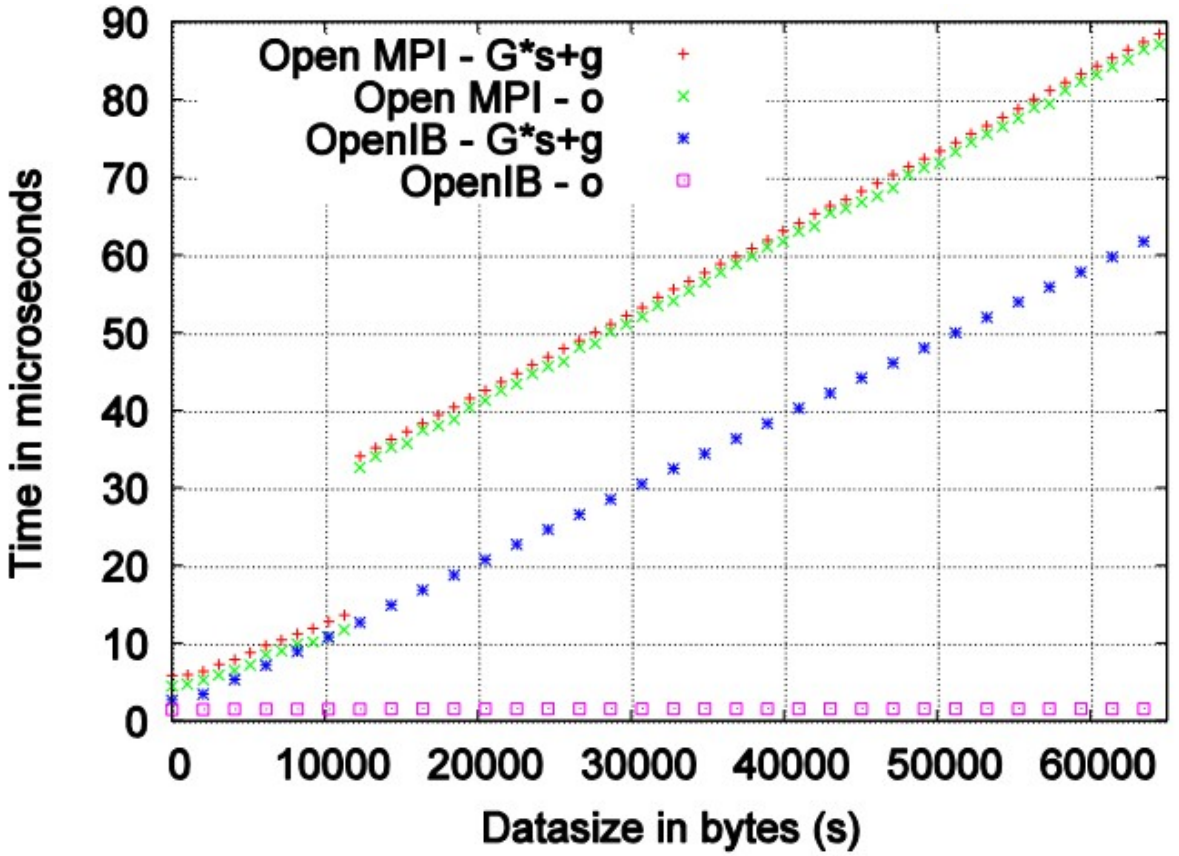
LogGP: Scatter

- Root sends half the data to p1, other half to p2
- p1 and p2 become new roots - problem is reduced to half the size

- $T(s) = \log_2(P)(L+2o) + (P-1)sG$

Message Passing - Implementations

- **On MPI Send**
  - *Either send data immediately – and buffer at receiver (EAGER Protocol)*
  - *Or wait until receiver is ready (RENDEZVOUS Protocol)*

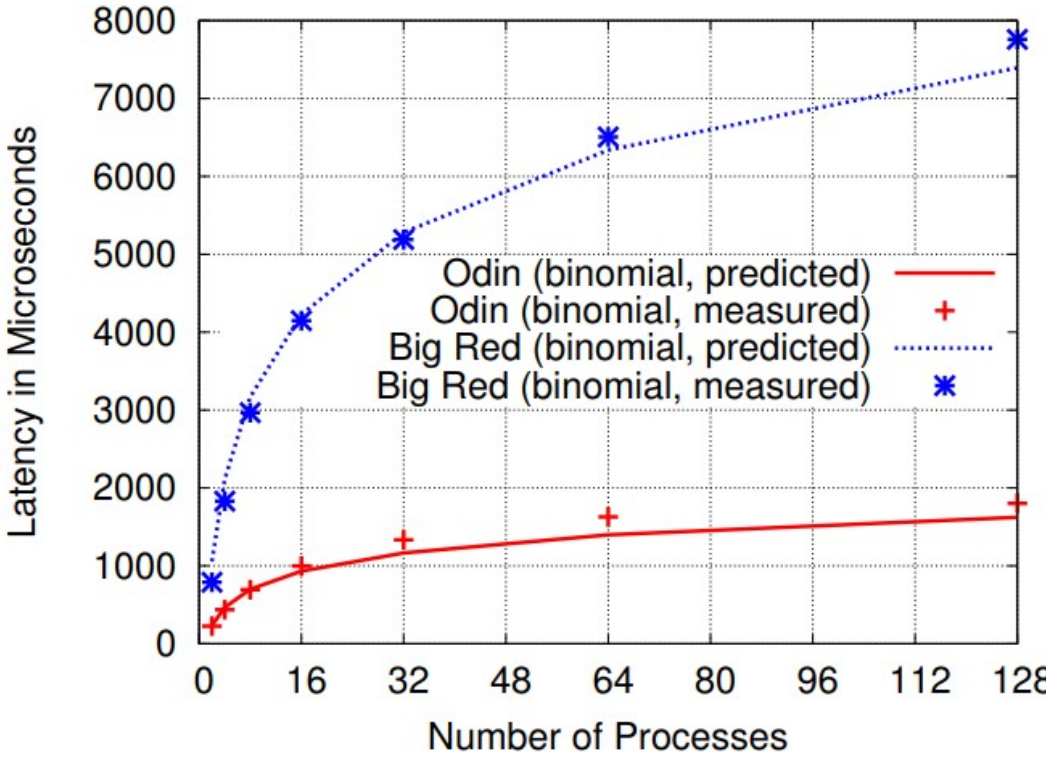


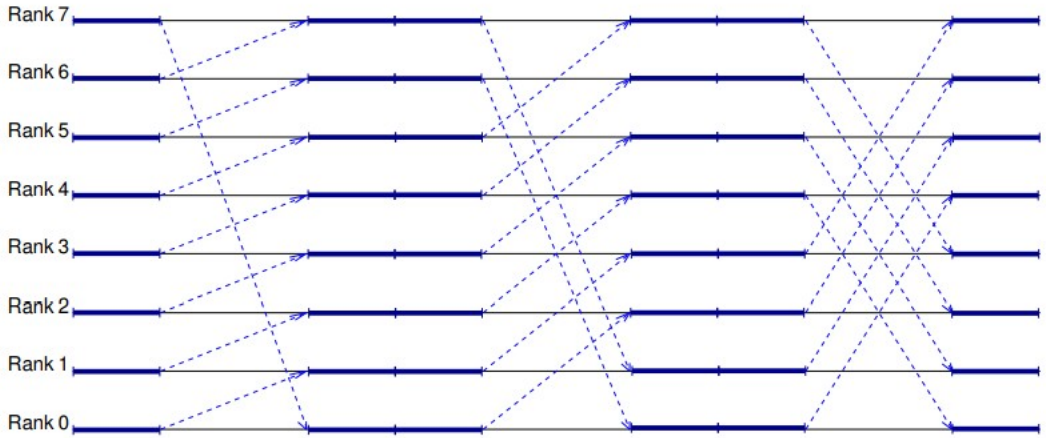
LogGPS

- L,o,g,G,P same as in LogGP
- S is the eager-threshold, if s>S add 2L+4o
- Hard to use in algorithm design and lower-bound proofs
- Good for simulations

LogGOPSim

- Use a LogGPS model
- Reads MPI traces, can extrapolate, inject noise, change parameters, etc.

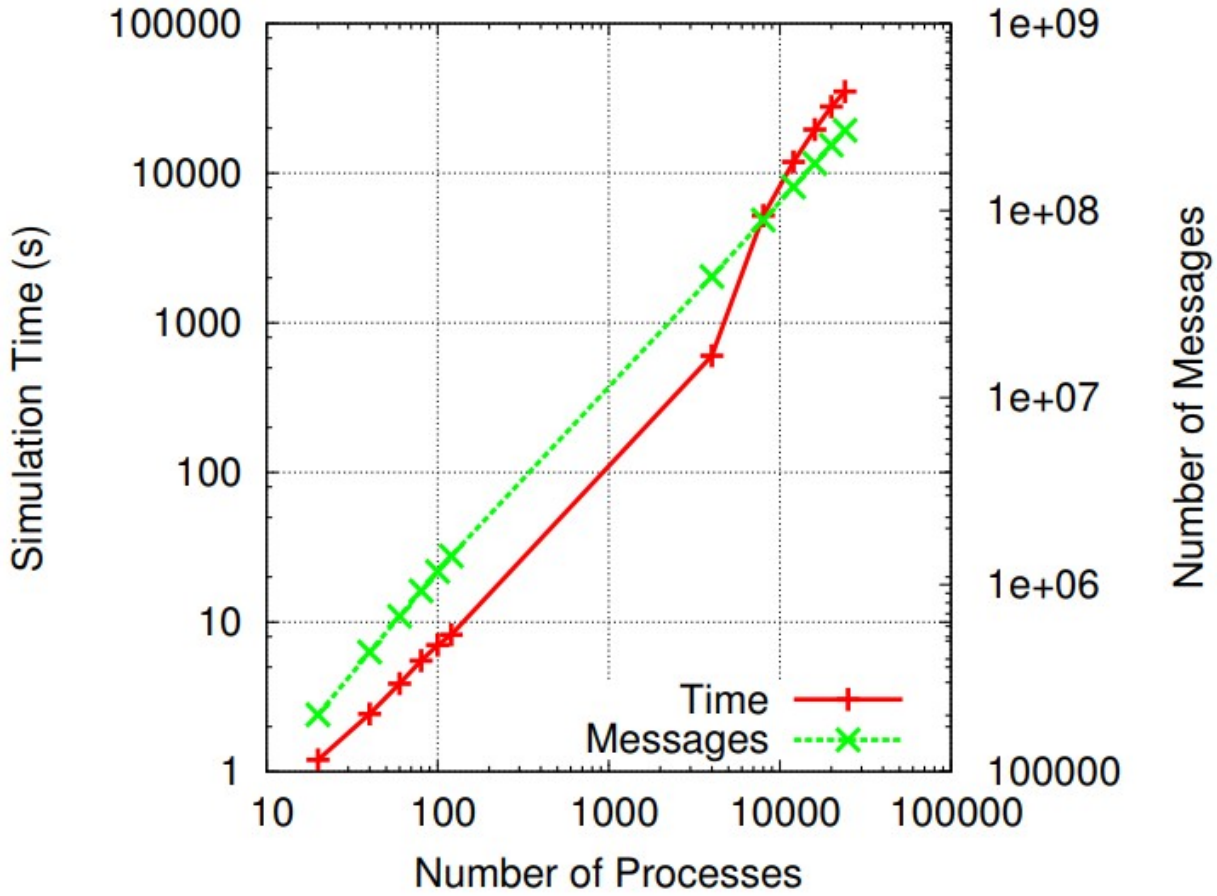




Hoefler, Schneider, Lumsdaine: LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model

LogGOPSim: Use cases

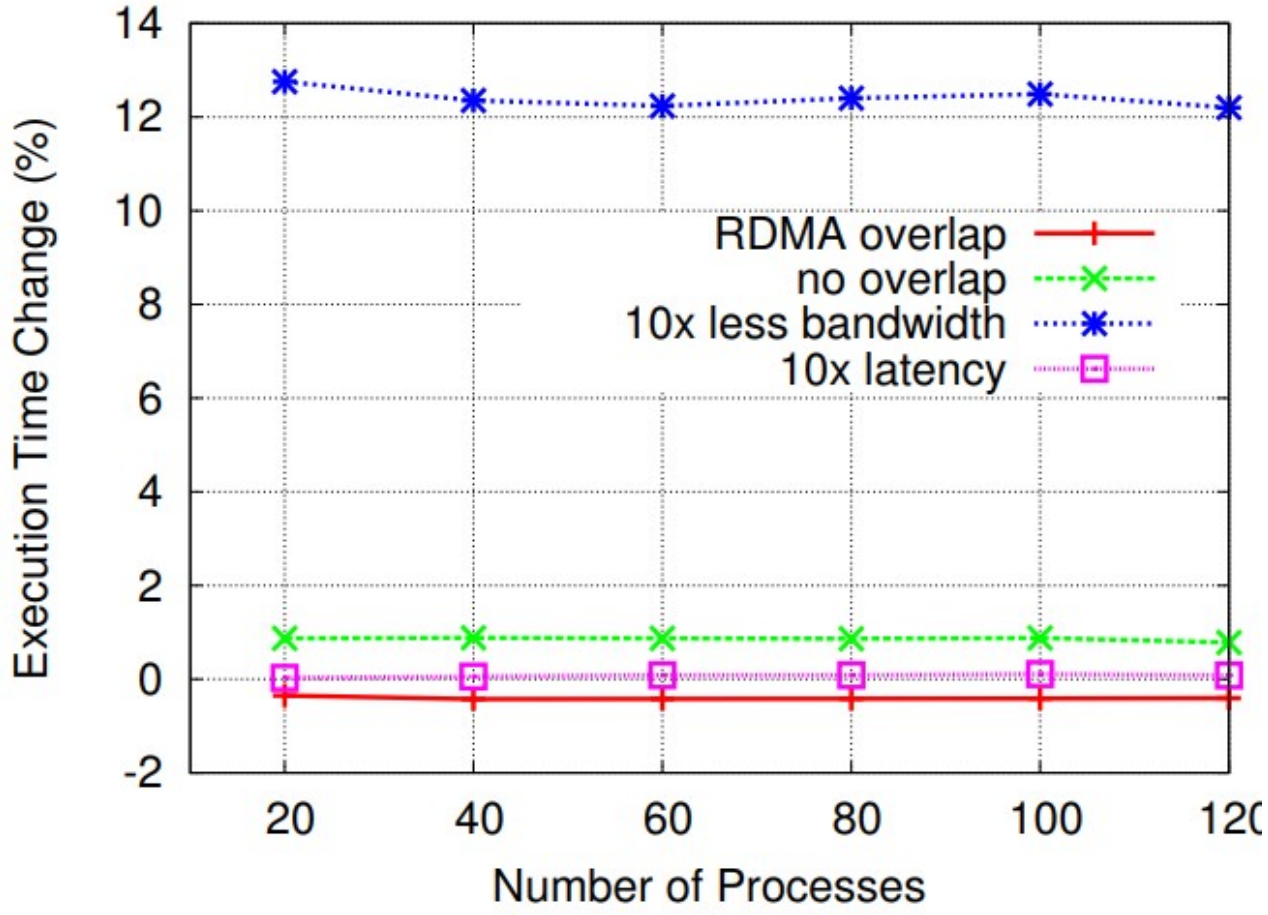
- Can simulate 100,000 processes of a real application in a day on my laptop



Hoefler, Schneider, Lumsdaine: LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model

LogGOPSim: Use cases

- What happens to my application if…



Hoefler, Schneider, Lumsdaine: LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model

Modeling: What did we miss?

- Can you think of anything important which we did not model?
- We always assume uniform bandwidth/latency across the network
- No congestion!
- Reality: Many different topologies to choose from (why?)
- Reality: Adaptive Routing is hard (why?)

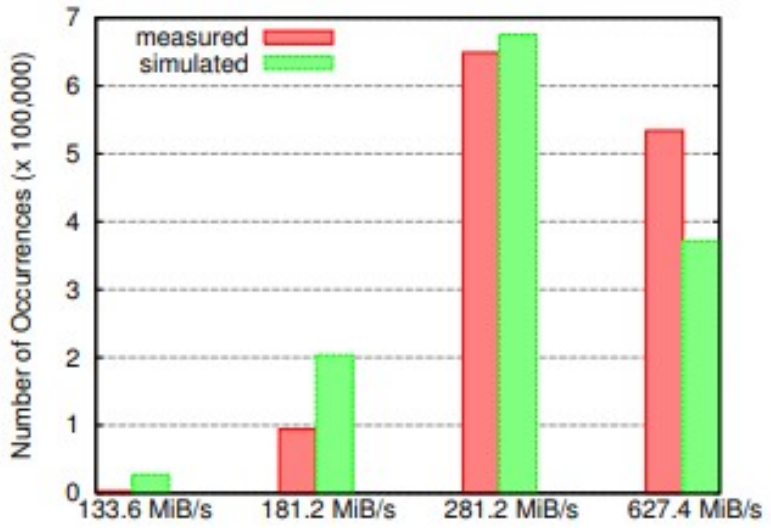


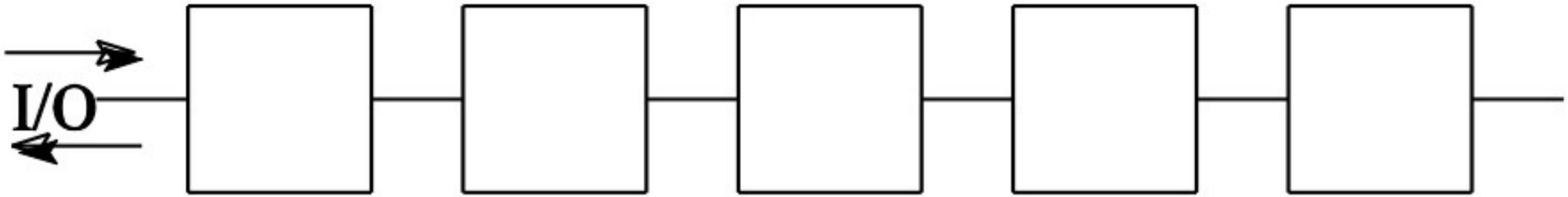
Figure 4. Simulation and Benchmark Results for a 512 node bisect Pattern with 1MiB messages in the CHiC system

Hoefler, Schneider, Lumsdaine:
Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks

Can we get more accurate models than LogGOPSim?

- Sure, next step is simulating packets/switch buffers
- Useless for algorithm design, ok for simulations (LogGOPSim is faster than reality this is much slower)
- What if we want to design switches / routing algorithms – good option
- Many tools available: OMNET++, Booksim, NS3, SST, etc.
- But is there a step in between?

Topologies: Linear Array



- Each PE has a small local storage
- How do we sort on this?
- What is the best sequential sorting algorithm (comparison based)?
- What is the parallel speedup using this approach?

Recap

- PRAM: Communication is free (CRCW-PRAM) or modelled with unit costs
- BSP: Supersteps, too coarse for, e.g., collectives
- Alpha-Beta: Easy to prove things in, but no overlap of sending with anything else (recv or compute)
- LogP: Overlap, simple packetization, does not reflect higher bandwidth for bigger messages
- LogGP: Realistic packetization, much harder to prove optimality (many open problems), ignores eager/rdvz
- LogGPS: Eager/Rdvz accounted for, even harder to use for analytical models, good for simulations (LogGOPSim), but ignores congestion and topology
- Simulation of packets on wires buffers (OMNET++, Booksim, NS3, SST) – great detail, low speed
- Treating topology as graph: Can build specific models with analytical bounds, metrics such as bisection ignore routing