CS 498
Hot Topics in High Performance Computing

Networks and Fault Tolerance

6. Advanced Network Models

Intro

- What did we learn in the last lecture
  – Fast Fourier Transform in LogP
  – LogGP a first LogP extension
    - The Scatter Problem
- What will we learn today
  - The Scatter Problem
  - LogGPS a second LogP extension

LogGP Motivation: Scatter

- Simple LogGP algorithm: send all s items to each processor (assuming G is cost per item):
  - $T(s) = g(P-2) + G(P-1)(s-1) + L$
- Class Question: Can we do better than that?

LogGP Motivation: Scatter

- Simple LogGP algorithm: send all s items to each processor (assuming G is cost per item):
  - $T(s) = g(P-2) + G(P-1)(s-1) + L$
- Class Question: Can we do better than that for small s?
  - Yes: forwarding along a tree, e.g., a binomial tree
  - Root sends half of the items to one PE, reducing the problem into two half-sized problems
  - Trade network bandwidth for latency!
    - Some messages are sent log(P) times

Binomial Scatter Runtime

- Class Question: What is the runtime for the Binomial Scatter?

Binomial Scatter Runtime

- Class Question: What is the runtime for the Binomial Scatter?
  – $T(s) = \log_2(P) * (L+2o) + (P-1)sG$

- Class Question: Can we do better? If yes, how?

Binomial Scatter Runtime

- Class Question: What is the runtime for the Binomial Scatter?
  - $T(s) = \log_2(P) * (L+2o) + (P-1)sG$

- Class Question: Can we do better? If yes, how?
  - Equal halving may lead to load imbalance due to difference between L and g
  - Yes, adjust arity and number of elements!
  - Optimality is not as simple though …

Optimal 1-item Scatter

- Let t(P) be the time to scatter 1 item to P processes
- $t(0) = 0$
- $t(P) = min_{0<s<P}\{(s-1)G + max\{L + 2o + t(s), g + t(P-s)\}\}$
  - let s(P) be the optimum in the equation above
  - the source PE sends first s(P) items to another PE
  - the target PE receives those items after (s-1)G+L+2o
  - the source PE continues after (s-1)G+g recursively
  - The target PE becomes a source PE

[1]: Alexandrov et al.: “LogGP: Incorporating Long Messages into the LogP Model”

Optimal 1-item Scatter contd.

- For proof of optimality see Alexandrov et al. "LogGP: Incorporating Long Messages into the LogP Protocol"
- Binomial scatter is a special case s(P) = P/2
  - Is optimal for L+2o = g
- Optimal algorithm for k-item case?
  - The algorithm above can be generalized to be close
  - An optimal algorithm remains unknown (try it!)

[1]: Alexandrov et al.: "LogGP: Incorporating Long Messages into the LogP Model"

LogGPS – A second Extension

- A quick look at message passing protocols
  - Sender sends data and receiver determines where to put it
  - Sender might send data before the receiver is ready
- Two typical options:
  - Small messages are "eagerly" sent and buffered at the receiver ("eager protocol")
  - Large messages require the sender to wait for the receiver ("rendezvous protocol")

LogGPS Synchronization Modeling

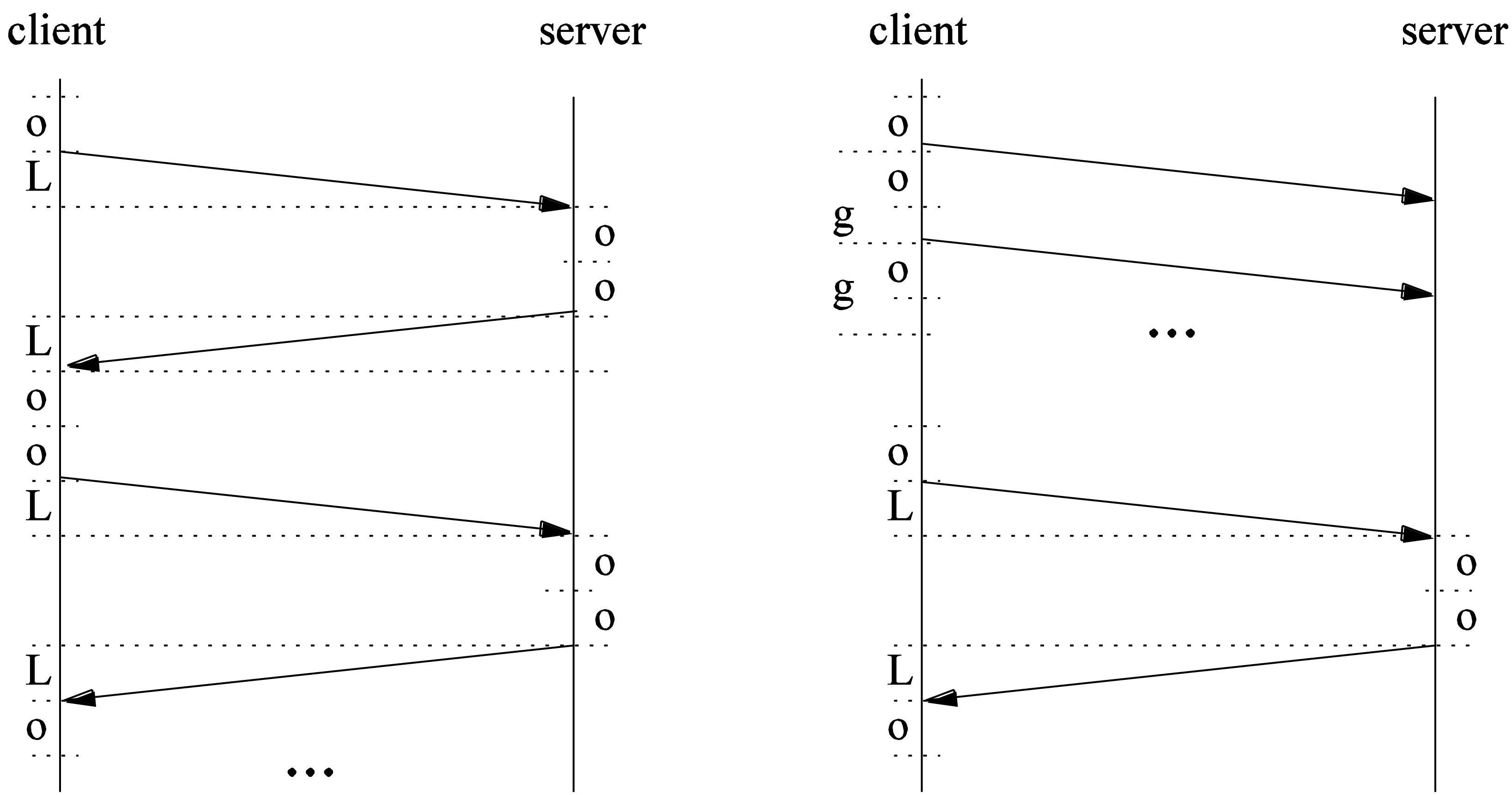
- Sender waits for receiver if s>S
  - Charges additional 2L+4o for the synch messages
  - Often leads to very complex equations
  - More useful in simulation studies
- Hard to use in algorithm design and lower bound proofs
  - Often simplified LogGP models
  - Ignoring it can have unwanted effects though

Measuring LogGPS Parameters

- But how do we get those LogGPS parameters for my favorite network?
  - Documentation (rarely)
  - Measurements (very hard)
- Measurement methodology:
  - Should be accurate (ignore single outliers)
  - Should not flood/congest the network (enables online measurements)

Challenges in Distributed Measurement

- Usually no synchronized time-source
  - Measurement on one host only, two techniques



Method 1: Culler et al./Iannello et al.

- differentiates between o_s and o_r
- o_s: issue small number (n) of sends and divide by n
- o_r: delay between messages, larger as RTT, subtract o_s
- g: flood network
- L: RTT/2 - o_r - o_s (errors propagate)

Method 2: Kielmann et al.

- changes the model to pLogP
- o_s: time for a single send
- o_r: time to copy the message from the receive buffer
- g: flood network (if accurate)
- L: (RTT(0)-2g(0))/2 (higher order errors)

Method 3: Bell et al.

- differentiates between o_s and o_r
- o_s: uses delay between message sends (adjust delay until
- $d + o = g + (s - 1)G$ (multiple measurements) $\Rightarrow o_s = g + (s - 1)G - d$ (second order errors)
- o_r: similar to Culler et al.
- g: flood network (similar to Kielmann et al.)
- L: not measured (network effects)
- EEL: end-to-end latency (RTT)

An Improved Technique

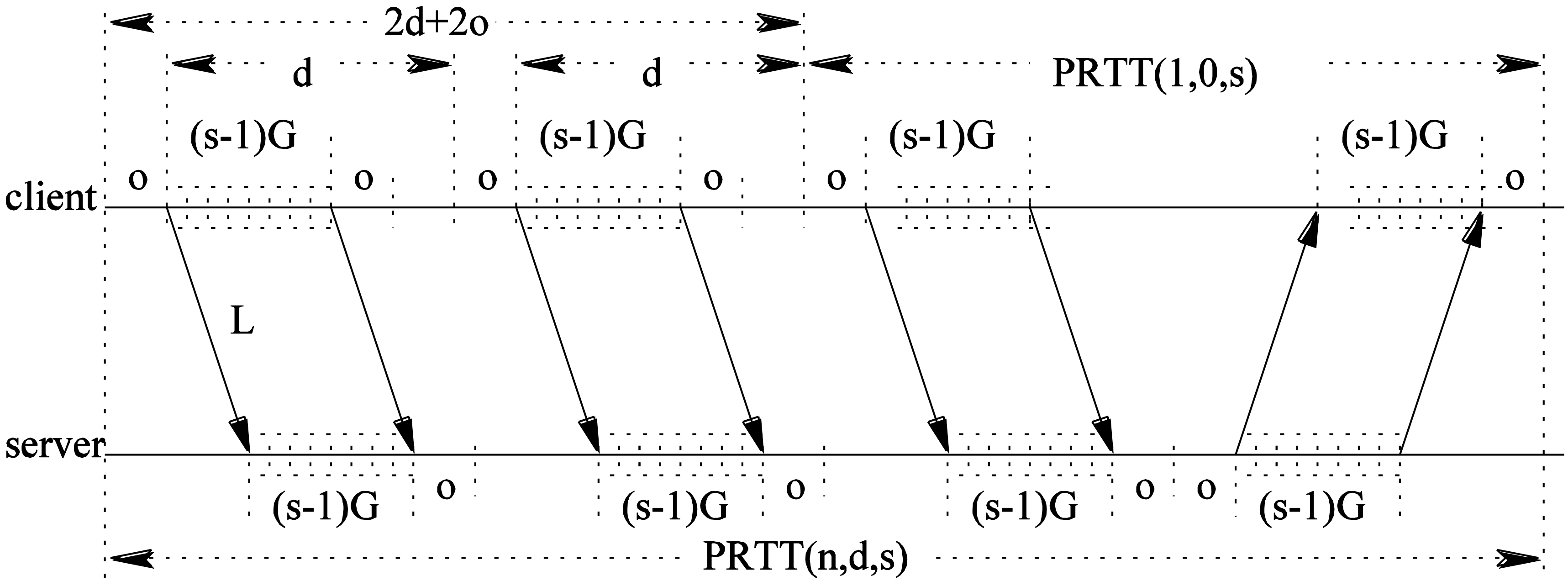
- A bit more complex
- Introducing parameterized RTT PRTT(n,d,s)
  - n - number of successive messages
  - d - delay between messages
  - s - message size
- Incorporates ping-pong and ping-ping benchmarks together with delays

PRTT(n,d,s) and LogGP

- $PRTT(1, 0, s) = 2 \cdot (L + 2o + (s - 1)G)$
- Set $G_{all} = g + (s - 1)G$
- $PRTT(n, 0, s) = 2 \cdot (L + 2o + (s - 1)G) + (n - 1) \cdot G_{all}$
- $PRTT(n, 0, s) = PRTT(1, 0, s) + (n - 1) \cdot G_{all}$
- $PRTT(n, d, s) = PRTT(1, 0, s) + (n - 1) \cdot \max\{o + d, Gall\}$

Measuring o

- $\frac{PRTT(n,d,s)-PRTT(1,0,s)}{n-1} = max\{o+d, G_{all}\}$
- we choose $d>G_{all}$
- $\frac{PRTT(n,d,s)-PRTT(1,0,s)}{n-1} = o+d$
- chose d = PRTT(1,0,s)

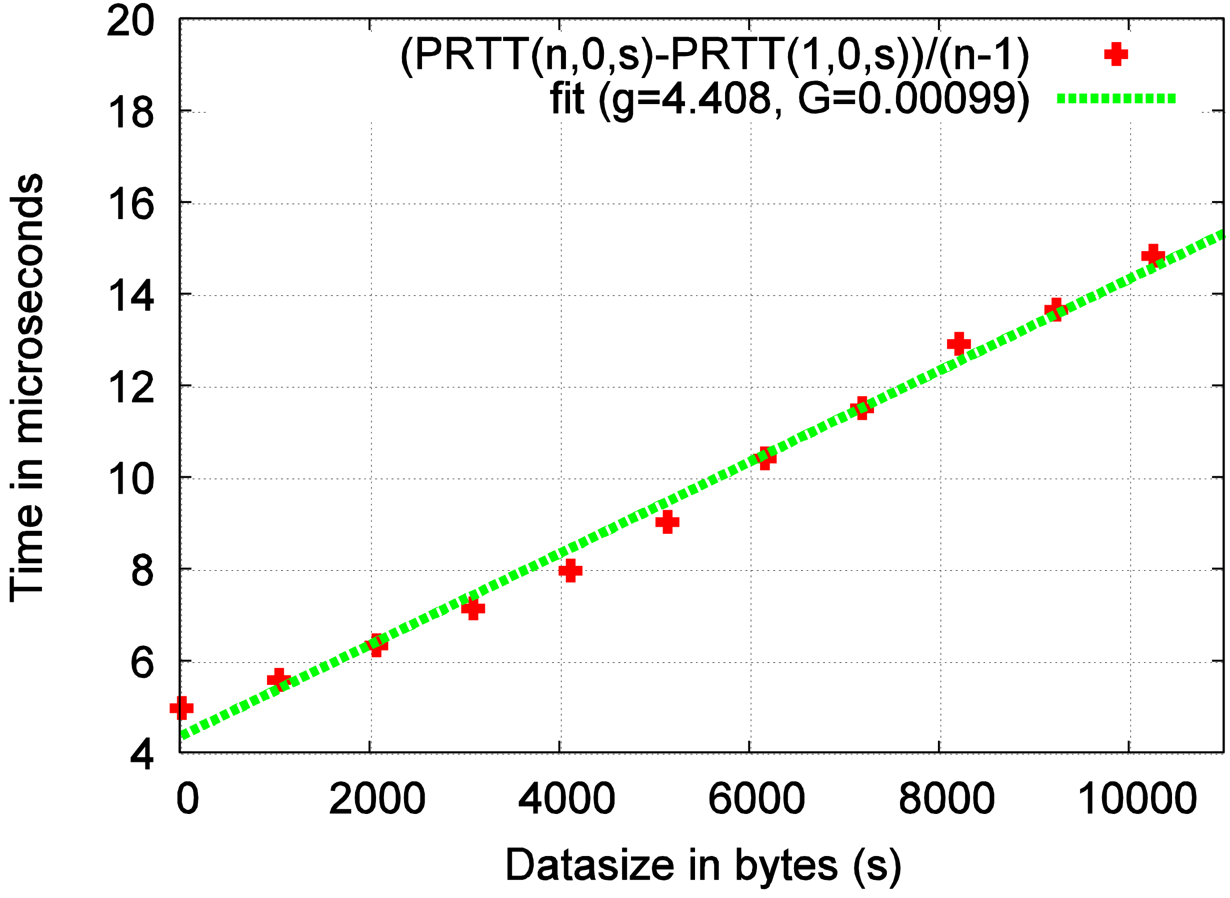


Measuring g and G

- $G(s-1)+g=\frac{PRTT(n,0,s)-PRTT(1,0,s)}{n-1}$
- Expresses a linear function
  - Measure PRTT(n,0,s) and PRTT(1,0,s) for varying s
- Least squares linear fit (a+bx)
  - b (slope of the curve) is G
  - a (value for x=0) is g

Example for g, G

- OpenMPI over InfiniBand

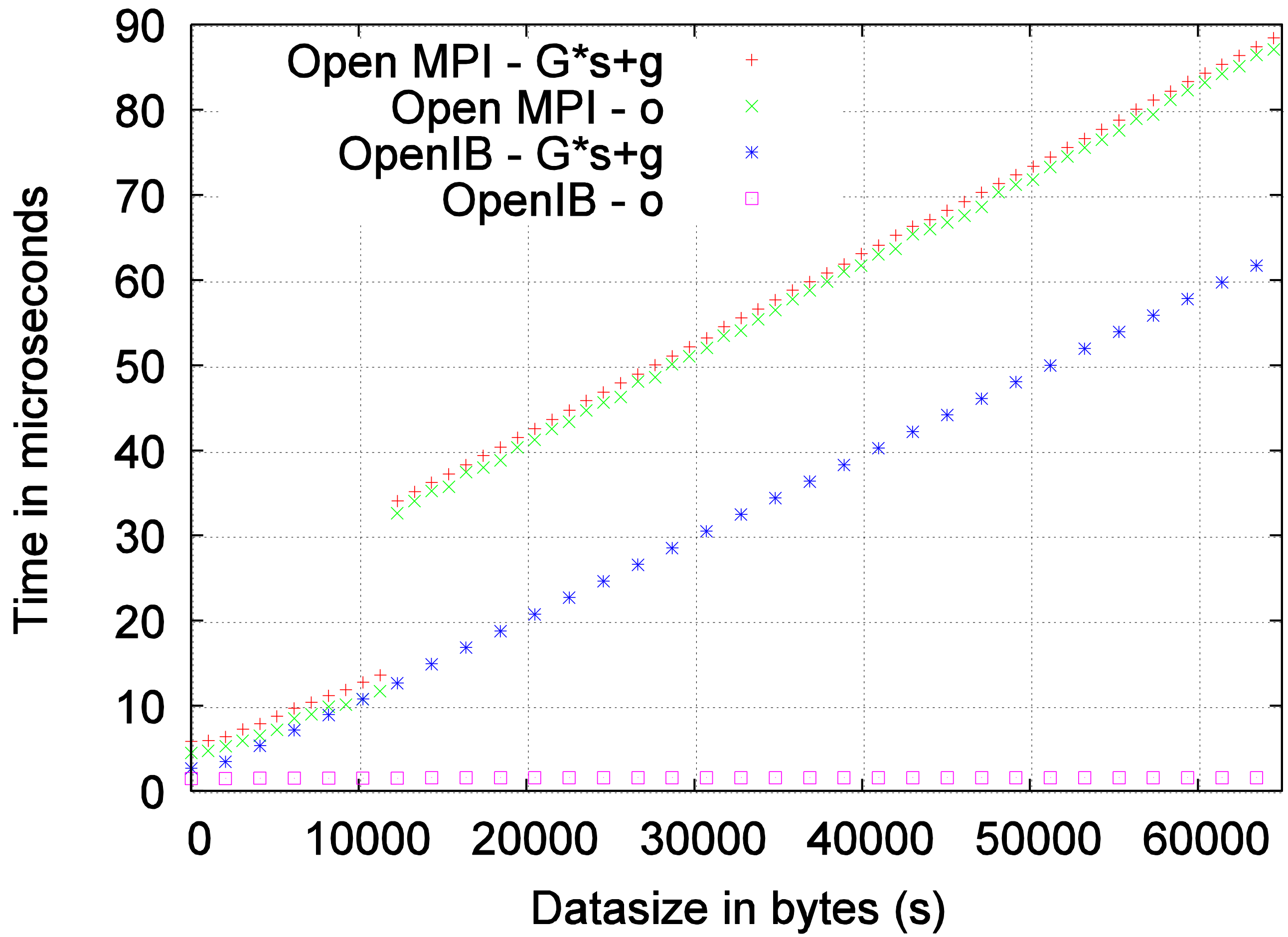


Measuring S

- comm. subsystems use data-size dependent protocols (eager/rendezvous)
- different parameters
- auto-detection possible
- changes in the mean least squares deviation
  - changes in g and G

Example Measurement

- OpenMPI over IB vs. OFED directly



Open Problems

- Measure nonblocking communication
  - This is most important for assessing o accurately
  - Would be a good student project
- Measure o_r ab-initio
  - o_r uses scheme by Culler (uses o_s)
- Measure L
  - We can’t measure L ☹
  - Use End-to-End Latency like Bell et al.