Design of Parallel and High-Performance Computing

Fall 2017

Recitation Session: distributed memory

Motivational video: [*https://www.youtube.com/watch?v=PuCx50FdSic*](https://www.youtube.com/watch?v=PuCx50FdSic)

Instructor: Torsten Hoefler & Markus Püschel

TA: Salvatore Di Girolamo

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Administrivia

- **Final project presentation: Monday 12/18 (next week)**
  - **Send presentation to me by Sunday 12/17 11:59pm**

    Presentation filename: dphpc_XX.{pdf,pptx} where XX is your team ID.
  - Should have (pretty much) final results
  - Show us how great your project is
  - Some more ideas what to talk about:

    Which architecture(s) did you test on?

    How did you verify correctness of the parallelization?

    Use bounds models for comparisons [1]!

    (Somewhat) realistic use-cases and input sets?

    Emphasize on the key concepts (may relate to theory of lecture)!

    What are remaining issues/limitations?
- **Report will be due in January!**
  - Still, starting to write early is very helpful --- write – rewrite – rewrite (no joke!)

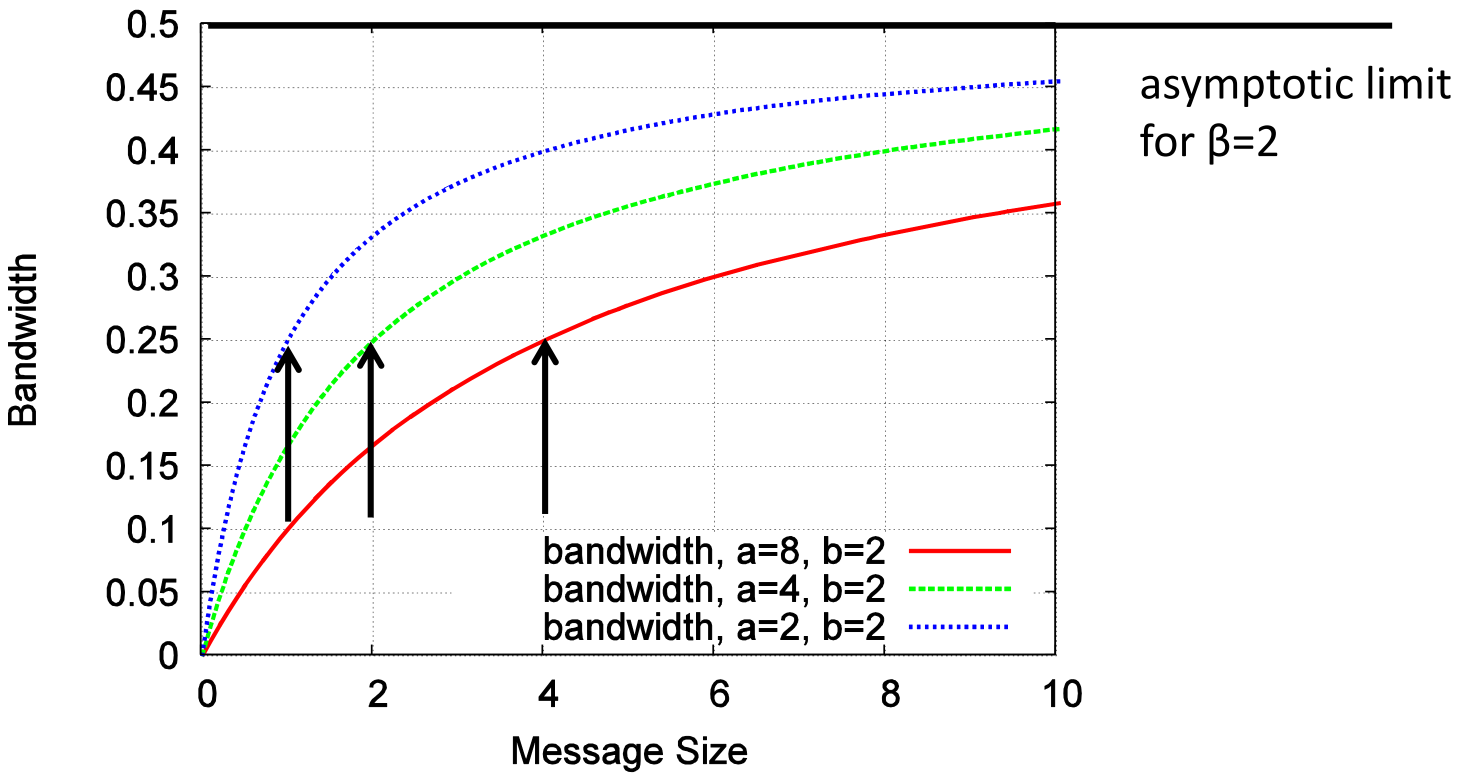
[1]: T. Hoefler, R. Belli: Scientific Benchmarking of Parallel Computing Systems, IEEE/ACM SC15

Remember: A Simple Model for Communication

- **Transfer time $T(s) = \alpha+\beta s$**
  - α = startup time (latency)
  - β = cost per byte (bandwidth=$1/\beta$)
- **As s increases, bandwidth approaches $1/\beta$ asymptotically**
  - Convergence rate depends on α
  - $s_{1/2} = \alpha/\beta$
- **Assuming no pipelining (new messages can only be issued from a process after all arrived)**

Bandwidth vs. Latency

- **$s_{1/2} = \alpha/\beta$ often used to distinguish bandwidth- and latency-bound messages**
  - $s_{1/2}$ is in the order of kilobytes on real systems



Quick Example

- **Simplest linear broadcast**
  - One process has a data item to be distributed to all processes
- **Broadcasting s bytes among P processes:**
  - T(s) = (P-1) * (α+βs) = $\mathcal{O}(P)$
- **Class question: Do you know a faster method to accomplish the same?**

k-ary Tree Broadcast

- **Origin process is the root of the tree, passes messages to k neighbors which pass them on**
  - k=2 -> binary tree
- **Class Question: What is the broadcast time in the simple latency/bandwidth model?**
  - $T(s) \approx \lceil log_k(P) \rceil \cdot k \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$ (for fixed k)
- **Class Question: What is the optimal k?**
  - $0 = \frac{ln(P) \cdot k}{ln(k)} \frac{d}{dk} = \frac{ln(P) ln(k) - ln(P)}{ln^2(k)} \rightarrow k = e = 2.71...$
  - Independent of P, α, βs? Really?

Faster Trees?

- **Class Question: Can we broadcast faster than in a ternary tree?**
  - Yes because each respective root is idle after sending three messages!
  - Those roots could keep sending!
  - Result is a k-nomial tree

    For k=2, it's a binomial tree
- **Class Question: What about the runtime?**
  - $$T(s) = \lceil log_k(P) \rceil \cdot (k-1) \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$$
- **Class Question: What is the optimal k here?**
  - T(s) d/dk is monotonically increasing for k>1, thus k_{opt}=2
- **Class Question: Can we broadcast faster than in a k-nomial tree?**
  - $\mathcal{O}(log(P))$ is asymptotically optimal for s=1!
  - But what about large s?

Open Problems

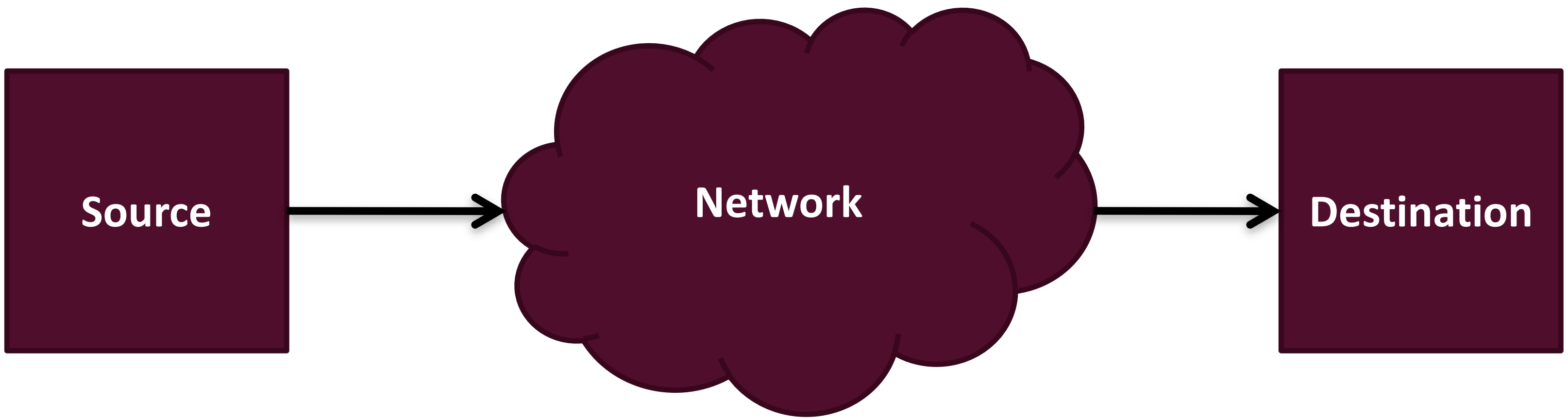
- **Look for optimal parallel algorithms (even in simple models!)**
  - And then check the more realistic models
  - Useful optimization targets are MPI collective operations

    Broadcast/Reduce, Scatter/Gather, Alltoall, Allreduce, Allgather, Scan/Exscan, …
  - Implementations of those (check current MPI libraries ☺)
  - Useful also in scientific computations

    Barnes Hut, linear algebra, FFT, …
- **Lots of work to do!**
  - Contact me for thesis ideas (or check SPCL) if you like this topic
  - Usually involve optimization (ILP/LP) and clever algorithms (algebra) combined with practical experiments on large-scale machines (10,000+ processors)

HPC Networking Basics

- **Familiar (non-HPC) network: Internet TCP/IP**
  - Common model:

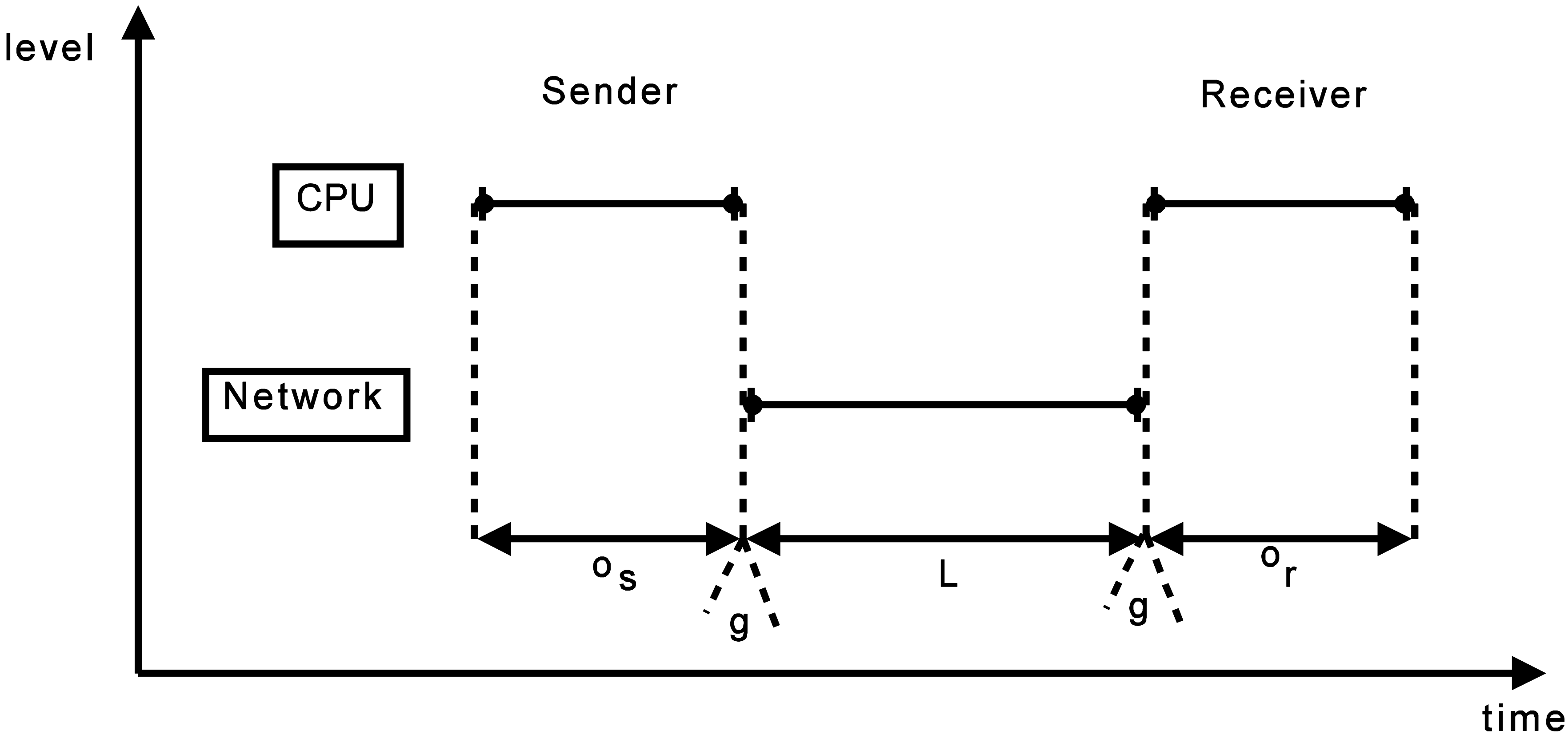


- **Class Question: What parameters are needed to model the performance (including pipelining)?**
  - Latency, Bandwidth, Injection Rate, Host Overhead

The LogP Model

- **Defined by four parameters:**
  - L: an upper bound on the latency, or delay, incurred in communicating a message containing a word (or small number of words) from its source module to its target module.
  - o: the overhead, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.
  - g: the gap, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor. The reciprocal of g corresponds to the available per-processor communication bandwidth.
  - P: the number of processor/memory modules. We assume unit time for local operations and call it a cycle.

The LogP Model



Simple Examples

- **Sending a single message**
  - $T = 2o+L$
- **Ping-Pong Round-Trip**
  - $T_{RTT} = 4o+2L$
- **Transmitting n messages**
  - $T(n) = L+(n-1)*\max(g, o) + 2o$

Simplifications

- **o is bigger than g on some machines**
  - g can be ignored (eliminates max() terms)
  - be careful with multicore!
- **Offloading networks might have very low o**
  - Can be ignored (not yet but hopefully soon)
- **L might be ignored for long message streams**
  - If they are pipelined
- **Account g also for the first message**
  - Eliminates “-1”

Benefits over Latency/Bandwidth Model

- **Models pipelining**
  - L/g messages can be “in flight”
  - Captures state of the art (cf. TCP windows)
- **Models computation/communication overlap**
  - Asynchronous algorithms
- **Models endpoint congestion/overload**
  - Benefits balanced algorithms

Example: Broadcasts

- **Class Question: What is the LogP running time for a linear broadcast of a single packet?**
  - $T_{lin} = L + (P-2) * \max(o,g) + 2o$
- **Class Question: Approximate the LogP runtime for a binary-tree broadcast of a single packet?**
  - $T_{bin} \le \log_2 P * (L + \max(o,g) + 2o)$
- **Class Question: Approximate the LogP runtime for an k-ary-tree broadcast of a single packet?**
  - $T_{k\text{-}n} \le \log_k P * (L + (k-1)\max(o,g) + 2o)$

Example: Broadcasts

- **Class Question: Approximate the LogP runtime for a binomial tree broadcast of a single packet (assume L > g!)?**
  - $T_{bin} \leq \log_2 P * (L + 2o)$
- **Class Question: Approximate the LogP runtime for a k-nomial tree broadcast of a single packet?**
  - $T_{k\text{-}n} \leq \log_k P * (L + (k-2)\max(o,g) + 2o)$
- **Class Question: What is the optimal k (assume o>g)?**
  - Derive by k: $0 = o * \ln(k_{opt}) - L/k_{opt} + o$ (solve numerically)

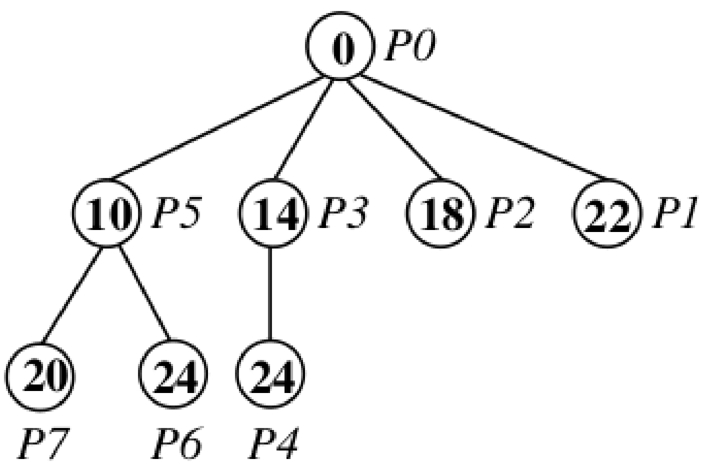
    For larger L, k grows and for larger o, k shrinks
  - Models pipelining capability better than simple model!

Example: Broadcasts

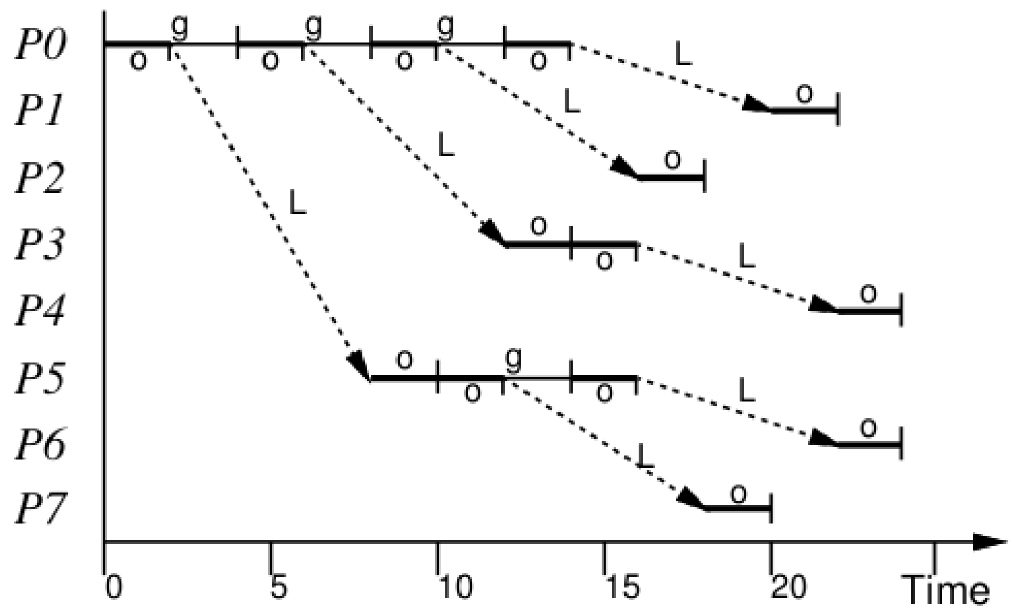
- **Class Question: Can we do better than k_{opt}-ary binomial broadcast?**
  - Problem: fixed k in all stages might not be optimal
  - We can construct a schedule for the optimal broadcast in practical settings
  - First proposed by Karp et al. in "Optimal Broadcast and Summation in the LogP Model"

Example: Optimal Broadcast

- **Broadcast to P-1 processes**
  - Each process who received the value sends it on; each process receives exactly once



P=8, L=6, g=4, o=2



Optimal Broadcast Runtime

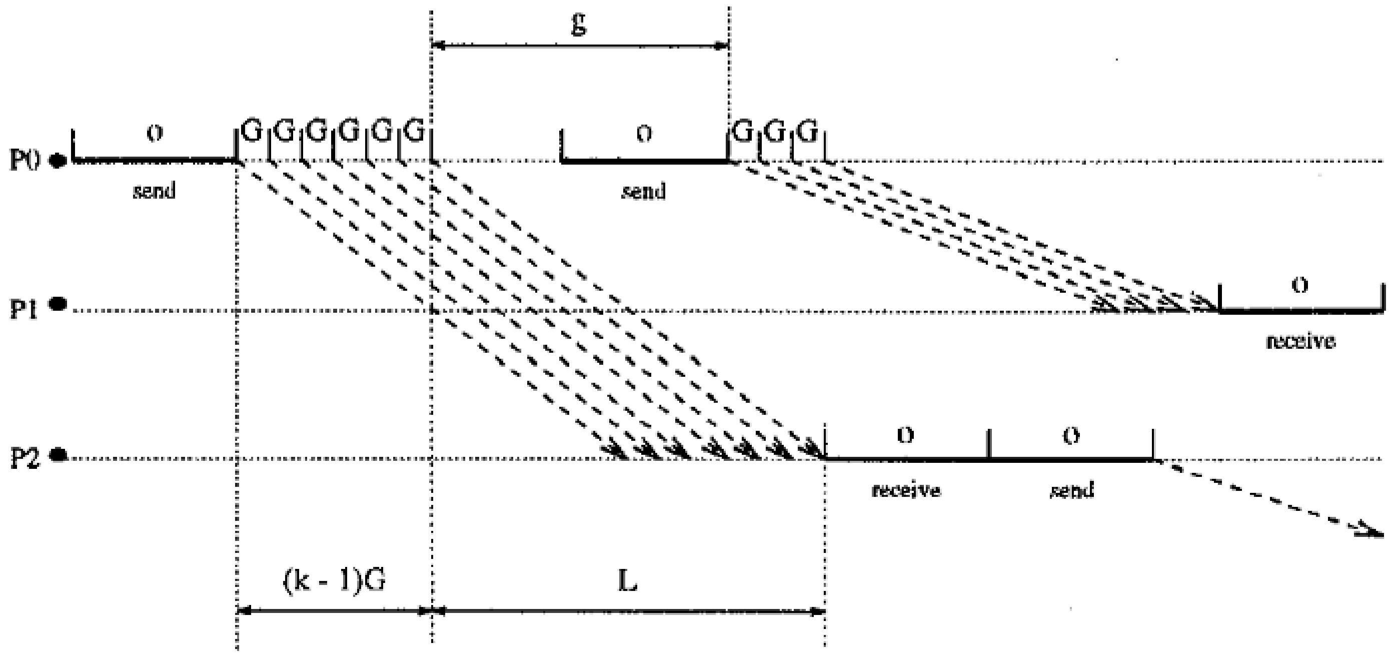
- **This determines the maximum number of PEs (P(t)) that can be reached in time t**
- **P(t) can be computed with a generalized Fibonacci recurrence (assuming o>g):**

$$P(t) = \begin{cases} 1 : & t < 2o + L \\ P(t-o) + P(t-L-2o) : & \text{otherwise.} \end{cases} \quad (1)$$

- **Which can be bounded by (see [1]):** $2^{\left\lfloor \frac{t}{L+2o} \right\rfloor} \leq P(t) \leq 2^{\left\lfloor \frac{t}{o} \right\rfloor}$
  - A closed solution is an interesting open problem!

[1]: TH et al.: "Scalable Communication Protocols for Dynamic Sparse Data Exchange" (Lemma 1)

LogGP



Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. 1995. LogGP: incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation.

Scatter

- **Single item: Distribute P-1 items from the source processor to their respective destinations.**
  - k-item: multiple items for each processor
- **Parameter simplifications:**
  - G normalized to 1, other parameters scaled.
  - G is the gap per item
  - o not considered (only communication)
- **LogP and k-item scatter: source sends k(P-1) to all processors**
  - No message size considered.
- **Simple for LogGP:**
  - Group messages to the same processor
- **Binomial tree**

Algorithm	Time complexity
Short-Message	$((P-1)k-1)g+L$
Simple Long-Msg.	$(P-2)g+(P-1)(k-1)+L$
Binomial Tree	$\max\{L,g\}\log_2 P+(P-1)k-\log_2 P$

Optimal Scatter

- **Problem: the sending processor may be ready to send before the receiving processor can send the first message…**

- **Optimal scatter runtime:**

$$
\begin{aligned}
t(1) &= 0 \\
t(P) &= \min_{0<s<P} \{(s-1) + \max\{L + t(s), g + t(P-s)\}\}
\end{aligned}
$$

  - Assume P_i has data items for P processors
  - P_i splits the data in two groups:

    one of size s=S(P), send to P_j

    the other of size P-s, kept for itself