# CS 598: Communication Cost Analysis of Algorithms

## Lecture 1: Course motivation and overview; collective communication

Edgar Solomonik

University of Illinois at Urbana-Champaign

August 22, 2016

# A simple model for point-to-point messages

The time to send or receive a message of $s$ bytes is

$$T_{\mathrm{sr}}^{\alpha,\beta}(s) = \alpha + s \cdot \beta$$

- $\alpha$ – **latency/synchronization cost** per message
- $\beta$ – **bandwidth cost** per byte
- each processor can send and/or receive one message at a time

Let $P$ processors send a message of size $s$ in a ring,

- the **communication volume** (total amount of data sent) is $P \cdot s$
- What is the **communication cost** ($\alpha$–$\beta$-model execution time)?
  - if the messages are sent *simultaneously*,

    $$T_{\mathrm{sim-ring}}^{\alpha,\beta}(s) = T_{\mathrm{sr}}^{\alpha,\beta}(s) = \alpha + s \cdot \beta$$

  - if the messages are sent *in sequence*,

$$T_{\mathrm{seq-ring}}^{\alpha,\beta}(s, P) = P \cdot T_{\mathrm{sr}}^{\alpha,\beta}(s) = P \cdot (\alpha + s \cdot \beta)$$

# Broadcasts in the $\alpha$–$\beta$ model

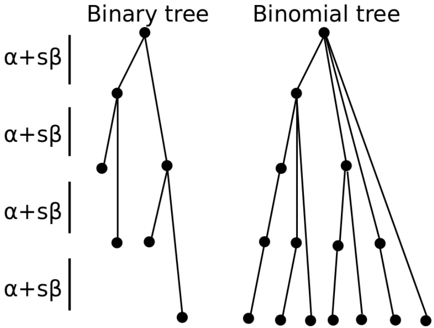
The execution time of a broadcast of a message of size $s$ to $P$ processors is

- using a **binary tree** of height $h_r = 2(\log_2(P+1)-1) \approx 2\log_2(P)$

  $$T^{\alpha,\beta}_{\mathrm{bcast-bnr}}(s,P) = h_r \cdot T^{\alpha,\beta}_{\mathrm{sr}}(s) = h_r \cdot (\alpha + s \cdot \beta)$$

- using a **binomial tree** of height $h_m = \log_2(P+1) \approx \log_2(P)$

  $$T^{\alpha,\beta}_{\mathrm{bcast-bnm}}(s,P) = h_m \cdot T^{\alpha,\beta}_{\mathrm{sr}}(s) = h_m \cdot (\alpha + s \cdot \beta)$$




Therefore, a binomial tree broadcast is $h_r/h_m \approx 2$ faster than a binary tree broadcast in the $\alpha$–$\beta$ model

# Large-message broadcasts

Lets now consider broadcasts of a message of a size $s \geq P$ bytes

- recall binomial tree broadcast cost:

$$T^{\alpha-\beta}_{\mathrm{bcast-bnm}}(s, P) = \log_2(P{+}1) \cdot (\alpha + s \cdot \beta)$$

- consider instead the following broadcast schedule
  - the root sends a different segment of the message to each processor
  - all processors exchange segments in $P - 1$ near-neighbor ring exchanges
- the cost of this broadcast schedule is

$$\begin{aligned} T^{\alpha-\beta}_{\mathrm{bcast-ring}}(s, P) = & \ (P-1)(T^{\alpha-\beta}_{\mathrm{sr}}(s/P) + T^{\alpha-\beta}_{\mathrm{sim-ring}}(s/P)) \\ & = 2(P-1)(\alpha + s/P \cdot \beta) \approx 2(P \cdot \alpha + s \cdot \beta) \end{aligned}$$

- for sufficiently large message sizes, the new schedule is faster,

$$\lim_{s \to \infty} \left( \frac{T^{\alpha-\beta}_{\mathrm{bcast-bnm}}(s, P)}{T^{\alpha-\beta}_{\mathrm{bcast-ring}}(s, P)} \right) \approx \log_2(P)/2$$

# Pipelined binary tree broadcast

Send a packet of size $k$ to left child then to right child

- as before, total message size $s$, tree height $h \approx \log_2(P)$
- each message costs $\alpha + k \cdot \beta$
- root sends $2s/k$ messages
- last packet takes $2h$ sends to reach rightmost tree leaf
- therefore, the total cost expression is

$$\begin{aligned} T_{\mathrm{PBT}}^{\alpha,\beta}(s,P,k) &\approx 2(h + s/k)(\alpha + k \cdot \beta) \\ &= 2(h \cdot \alpha + s \cdot \beta + (s/k) \cdot \alpha + hk \cdot \beta) \end{aligned}$$

- we can now derive the optimal message size

$$k_{\mathrm{opt}}^{\alpha,\beta}(s,P) = \underset{k}{\mathrm{argmin}}(T_{\mathrm{PBT}}^{\alpha,\beta}(s,P,k)) = \sqrt{\frac{s \cdot \alpha}{h \cdot \beta}}$$

- furthermore, $T_{\mathrm{PBT}}^{\alpha,\beta}(s,P,k_{\mathrm{opt}}^{\alpha,\beta}(s,P)) \approx 2(\sqrt{h \cdot \alpha} + \sqrt{s \cdot \beta})^2$, a factor of 2 more expensive than the Träff and Ripke protocol

# Double Tree

**Observation:** *the leaves of a binary tree, $(P-1)/2$ processors, send nothing, while the internal nodes do all the work.*

**Double Pipelined Binary Tree Broadcast**

- define two pipelined binary trees with a shared root
- non-root processors act as a leaf in one and as an internal node in the second
- send half of the message down each tree, alternating directions with packets of size $k$

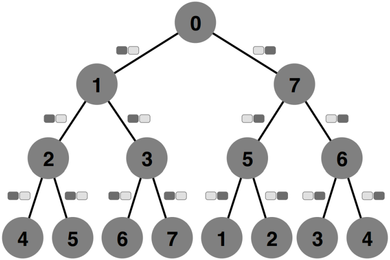



Diagram taken from: Hoefler, Torsten, and Dmitry Moor. "Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations."

# Double pipelined binary tree

The cost of the double pipelined binary tree is essentially the same as the cost of a single pipelined binary tree with half the message size,

$$T^{\alpha,\beta}_{\mathrm{DPBT}}(s,P) \approx 2h\cdot\alpha + 2\sqrt{2s\cdot h}\cdot\sqrt{\alpha\cdot\beta} + s\cdot\beta$$

for a sufficiently large message size ($s$) this is twice as fast as a single pipelined binary tree.

# Other types of collective communication

We can classify collectives into four categories

- **One-to-All**: Broadcast, Scatter
- **All-to-One**: Reduce, Gather
- **All-to-One + One-to-All**: Allreduce (Reduce+Broadcast), Allgather (Gather+Broadcast), Reduce-Scatter (Reduce+Scatter), Scan
- **All-to-All**: All-to-all

MPI (Message-Passing Interface) provides all of these as well as variable size versions (e.g. (All)Gatherv, All-to-allv), see online for specification of each routine.

We now present protocols for these and their cost in the $\alpha - \beta$ model, with

$$s = \begin{cases} \text{input size} & : \text{one-to-all collectives} \\ \text{output size} & : \text{all-to-one collectives} \\ \text{per-processor input/output size} & : \text{all-to-all collectives} \end{cases}$$

# Tree collectives

We have demonstrated how (double/pipelined) binary trees and binomial trees can be used for broadcasts

- *A reduction may be done via any broadcast tree with the same communication cost, with reverse data flow*

$$T_{\text{reduce}} = T_{\text{broadcast}} + \text{cost of local reduction work}$$

Scatter is strictly easier than broadcast, pipeline half message to each child in a binary tree

$$T^{\alpha,\beta}_{\text{scatter}}(s, P) \approx 2\log_2(P)\cdot\alpha + s\cdot\beta$$

- *A gather may be done via the reverse of any scatter protocol*:

$$T_{\text{gather}} = T_{\text{scatter}}$$

**All-to-One + One-to-All** collectives can be done via two trees, but is this most efficient? What about **All-to-All** collectives?