

# Two-phase Broadcasting

Section 2.4 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

Utrecht University



# Optimizing a parallel algorithm

- ▶ **Computation**: well-balanced, little redundancy. Hence no room for improvement.
- ▶ **Communication**: every bit of communication is one bit too much. We can always try harder.



# Communication volume

- ▶ The **communication volume** of an  $h$ -relation is the total number of data words communicated,

$$V = \sum_{s=0}^{p-1} h_S(s) = \sum_{s=0}^{p-1} h_R(s).$$

- ▶  $h_S(s)$  is the number of data words sent by processor  $P(s)$  and  $h_R(s)$  is the number received.
- ▶ Note that

$$V \leq \sum_{s=0}^{p-1} h = ph.$$



# Communication balance

- ▶ An  $h$ -relation is **balanced** if

$$h = \frac{V}{p}.$$

- ▶ A balanced  $h$ -relation satisfies

$$V = \sum_{s=0}^{p-1} h_s(s) = ph,$$

so that  $h_s(s) = h$  for all  $s$ . (Because  $h_s(s) \leq h$ .)

- ▶ The reverse is trivially true: if  $h_s(s) = h$  for all  $s$ , then  $V = ph$ .
- ▶ Similar statements hold for the  $h_r(s)$ .
- ▶ Therefore, a **balanced**  $h$ -relation and a **full**  $h$ -relation are the same.



## $h_S \neq h_R$ implies unbalanced communication

- ▶ If an  $h$ -relation is balanced, we have

$$h_S = h_R,$$

where  $h_S = \max_S h_S(s)$  and  $h_R = \max_S h_R(s)$ .

- ▶ The **reverse is not true**: some processors may be overloaded sending, others receiving, and  $h_S = h_R$  may happen while the  $h$ -relation is still unbalanced, possibly with  $h \gg V/p$ .
- ▶ If  $h_S \neq h_R$ , we are sure that the communication is unbalanced.



# Communication imbalance in LU decomposition

- ▶ **Send cost** in superstep (10), the row/column broadcast, assuming  $M = N = \sqrt{p}$ :

$$h_s = R_{k+1}(N - 1) + C_{k+1}(M - 1) = 2R_{k+1}(\sqrt{p} - 1).$$

- ▶ **Receive cost** in superstep (10):

$$h_r = R_{k+1} + C_{k+1} = 2R_{k+1}.$$

- ▶ **Large discrepancy:**  $h_s \gg h_r$ . The balance for senders must be improved to reduce the communication cost.



## Cause of the communication imbalance

**if**  $\phi_1(k) = t$  **then**

**for all**  $i : k < i < n \wedge \phi_0(i) = s$  **do**

put  $a_{ik}$  in  $P(s, *)$ ;

▷ Superstep (10a)

- ▶ The **sending** part of the broadcast of column  $k$  is **unbalanced**: only the  $\sqrt{p}$  processors in  $P(*, \phi_1(k))$  send, each sending

$$R_{k+1}(\sqrt{p} - 1) = \left\lceil \frac{n - k - 1}{\sqrt{p}} \right\rceil (\sqrt{p} - 1) \approx n - k - 1$$

data words.

- ▶ The **receiving** part is **balanced**: all processors except the senders receive

$$R_{k+1} \approx \frac{n - k - 1}{\sqrt{p}}$$

data words.



# Broadcast is a bottleneck

- ▶ The total contribution of superstep (10) to the LU cost is about

$$\sum_{k=0}^{n-1} 2(n-k-1)g = 2g \sum_{k=0}^{n-1} k = 2g \frac{(n-1)n}{2} \approx n^2 g.$$

- ▶ This is a bottleneck compared to the computation cost  $\frac{2n^3}{3p}$ .





# One-phase broadcast of a vector

*input:*  $\mathbf{x}$  : vector of length  $n$ ,  $\text{repl}(\mathbf{x}) = P(0)$ .

*output:*  $\mathbf{x}$  : vector of length  $n$ ,  $\text{repl}(\mathbf{x}) = P(*)$ .

**function** BROADCAST\_1PHASE( $\mathbf{x}$ ,  $P(0)$ ,  $P(*)$ )

**if**  $s = 0$  **then**

**for**  $t := 1$  **to**  $p - 1$  **do**

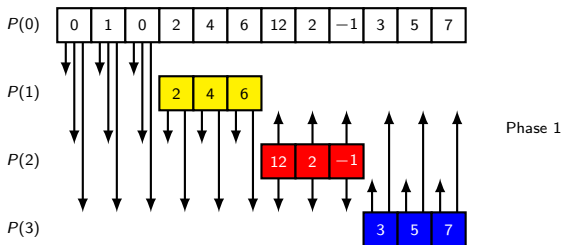
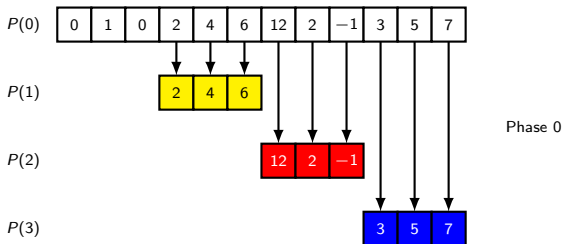
**for**  $i := 0$  **to**  $n - 1$  **do**

            put  $x_i$  in  $P(t)$ ;

- ▶  $\text{repl}(\mathbf{x}) = P(*)$  means that  $\mathbf{x}$  is replicated such that each processor has a copy.



# Two-phase broadcast in blocks



# The two-phase idea

- ▶ First **spread** the data, then **broadcast** them. This lets every processor participate.
- ▶ This method is also used in the **BitTorrent protocol** for peer-to-peer file sharing.
- ▶ BitTorrent splits a file to be distributed into small pieces and spreads these pieces among downloaders, who in turn make the pieces available for further distribution.
- ▶ The idea is similar to **two-phase randomized routing** (Valiant 1982): first send data to a randomly chosen intermediate location, then route them to their final destination. This avoids congestion.
- ▶ We don't need randomness here: in our regular problem, we can choose the intermediate location optimally and deterministically.



## Two-phase broadcast of a vector

*input:*  $\mathbf{x}$  : vector of length  $n$ ,  $\text{repl}(\mathbf{x}) = P(0)$ .

*output:*  $\mathbf{x}$  : vector of length  $n$ ,  $\text{repl}(\mathbf{x}) = P(*)$ .

**function** BROADCAST\_2PHASE( $\mathbf{x}$ ,  $P(0)$ ,  $P(*)$ )

$b := \lceil n/p \rceil$ ;

{ Spread the vector }

**if**  $s = 0$  **then**

▷ Superstep (0)

**for**  $t := 1$  **to**  $p - 1$  **do**

**for**  $i := tb$  **to**  $\min\{(t + 1)b, n\} - 1$  **do**

            put  $x_i$  in  $P(t)$ ;

{ Broadcast the subvectors }

**for**  $t := 1$  **to**  $p - 1$  **do**

▷ Superstep (1)

**for**  $i := sb$  **to**  $\min\{(s + 1)b, n\} - 1$  **do**

        put  $x_i$  in  $P(t)$ ;



## Cost analysis of two-phase broadcast

- ▶ Phase 0 costs  $(n - b)g$ , where  $b = \lceil \frac{n}{p} \rceil$  is the block size.
- ▶ Phase 1 costs  $(p - 1)bg$ .
- ▶ The total cost of a two-phase broadcast of a vector of length  $n$  to  $p$  processors is

$$T_{\text{broadcast}} = \left( n + (p - 2) \left\lceil \frac{n}{p} \right\rceil \right) g + 2l \approx 2ng + 2l.$$

- ▶ Much less than the cost  $(p - 1)ng + l$  of a one-phase broadcast, except for large  $l$ .



# Two-phase broadcast in LU decomposition

▷ Superstep (6)–(7)

Broadcast( $(a_{ik} : k < i < n \wedge i \bmod M = s)$ ,  
 $P(s, k \bmod N), P(s, *)$ );

Broadcast( $(a_{kj} : k < j < n \wedge j \bmod N = t)$ ,  
 $P(k \bmod M, t), P(*, t)$ );

- ▶ Phase 0 of the row broadcast and phase 0 of the column broadcast are **done together** in superstep (6).
- ▶ Phases 1 are done together in (7).
- ▶ This is **less modular, but more efficient**.



## Optimization: use already known pivot value

**if**  $\phi_0(k) = s \wedge \phi_1(k) = t$  **then** ▷ Superstep (8)  
    put  $a_{kk}$  in  $P(*, t)$ ;

- ▶ Delete superstep (8) from the basic Algorithm 2.4, because

$$a_{kk} \text{ (after swap)} = a_{rk} \text{ (before swap)}$$

and the pivot value  $a_{rk}$  is already known locally.

- ▶ Instead, **divide immediately** by  $a_{rk}$  in superstep (2) of the improved Algorithm 2.8:

**if**  $k \bmod N = t$  **then** ▷ Superstep (2)  
     $s_{\max} := \operatorname{argmax}(|a_{r_q, k}| : 0 \leq q < M)$ ;  
     $r := r_{s_{\max}}$ ;  
    **for all**  $i : k \leq i < n \wedge i \bmod M = s \wedge i \neq r$  **do**  
         $a_{ik} := a_{ik}/a_{rk}$ ;



## Optimization: combine index and row swaps

**if**  $k \bmod M = s$  **then**

put  $\pi_k$  as  $\hat{\pi}_k$  in  $P(r \bmod M, t)$ ;

**for all**  $j : 0 \leq j < n \wedge j \bmod N = t$  **do**

put  $a_{kj}$  as  $\hat{a}_{kj}$  in  $P(r \bmod M, t)$ ;

**if**  $r \bmod M = s$  **then**

put  $\pi_r$  as  $\hat{\pi}_r$  in  $P(k \bmod M, t)$ ;

**for all**  $j : 0 \leq j < n \wedge j \bmod N = t$  **do**

put  $a_{rj}$  as  $\hat{a}_{rj}$  in  $P(k \bmod M, t)$ ;

▷ Superstep (4)

- ▶ Combining communication supersteps **saves synchronizations**.





## Optimization: combine first and last superstep

**for**  $k := 0$  **to**  $n - 1$  **do**

**if**  $k \bmod N = t$  **then** ▷ Superstep (0)

$r_s := \operatorname{argmax}(|a_{ik}| : k \leq i < n \wedge i \bmod M = s);$

...

**for all**  $i : k < i < n \wedge i \bmod M = s$  **do** ▷ Superstep (0')

**for all**  $j : k < j < n \wedge j \bmod N = t$  **do**

$a_{ij} := a_{ij} - a_{ik}a_{kj};$

- ▶ Combining the first and last superstep of the loop saves a synchronization.
- ▶ In an implementation: **no unnecessary bsp\_sync** at the end of the main loop.



## Is the choice $M = N$ still optimal?

- ▶ The two-phase approach reduces the cost of the row and column broadcasts to about  $2(R_{k+1} + C_{k+1})g$ .
- ▶ A bound on the cost is obtained by

$$\begin{aligned}R_{k+1} + C_{k+1} &< \left( \frac{n-k-1}{M} + 1 \right) + \left( \frac{n-k-1}{N} + 1 \right) \\ &= (n-k-1) \frac{M+N}{p} + 2,\end{aligned}$$

which is indeed minimal for  $M = N = \sqrt{p}$ .



## Choice of $M, N$ for row swaps

- ▶ The row and index swap in superstep (4) costs  $(C_0 + 1)g$ , where  $C_0 = \lceil n/N \rceil$ , which is minimal for  $M = 1$  and  $N = p$ .
- ▶ The swap cost for  $M = N$ , however, is of the same order as the broadcast cost, so we cannot gain much by decreasing  $M$ .
- ▶ Conclusion:  $M$  should be close to  $N$ , but slightly below.



# Exact cost analysis

We need to compute sums of the form

$$\sum_{k=0}^{n-1} R_k = \sum_{k=0}^{n-1} \left\lceil \frac{n-k}{\sqrt{p}} \right\rceil = \sum_{k=1}^n \left\lceil \frac{k}{\sqrt{p}} \right\rceil.$$

**Lemma 2.10.** Let  $n, q \geq 1$  be integers with  $n \bmod q = 0$ . Then

$$\sum_{k=0}^n \left\lceil \frac{k}{q} \right\rceil = \frac{n(n+q)}{2q}, \quad \sum_{k=0}^n \left\lceil \frac{k}{q} \right\rceil^2 = \frac{n(n+q)(2n+q)}{6q^2}.$$



## Proof of first part of Lemma 2.10

$$\begin{aligned}\sum_{k=0}^n \left\lceil \frac{k}{q} \right\rceil &= \left\lceil \frac{0}{q} \right\rceil + \left( \left\lceil \frac{1}{q} \right\rceil + \cdots + \left\lceil \frac{q}{q} \right\rceil \right) + \cdots \\ &\quad + \left( \left\lceil \frac{n-q+1}{q} \right\rceil + \cdots + \left\lceil \frac{n}{q} \right\rceil \right) \\ &= q \cdot 1 + q \cdot 2 + \cdots + q \cdot \frac{n}{q} \\ &= q \sum_{k=1}^{n/q} k \\ &= q \frac{n}{2q} \left( \frac{n}{q} + 1 \right) \\ &= \frac{n(n+q)}{2q}.\end{aligned}$$



# Total BSP cost of LU decomposition

$$\begin{aligned} T_{\text{LU}} &= \frac{2n^3}{3p} + \left( \frac{3}{2\sqrt{p}} - \frac{2}{p} \right) n^2 + \frac{5n}{6} \\ &+ \left( \left( \frac{3}{\sqrt{p}} - \frac{2}{p} \right) n^2 + \left( 4\sqrt{p} - \frac{4}{\sqrt{p}} + \frac{4}{p} - 3 \right) n \right) g \\ &+ 8nl \\ &\approx \frac{2n^3}{3p} + \frac{3n^2}{2\sqrt{p}} + \frac{3n^2g}{\sqrt{p}} + 8nl. \end{aligned}$$



# Summary

- ▶ Cost analysis gives a **diagnosis** of underperformance, such as an imbalance  $h_s \gg h_r$ .
- ▶ We have optimized our basic parallel LU decomposition algorithm by
  - ▶ performing **two-phase broadcasting** to spread the communication load evenly;
  - ▶ exploiting **local information** on the pivot value to avoid unnecessary communication;
  - ▶ reorganizing the algorithm to **combine supersteps**, thus saving synchronizations.
- ▶ The resulting LU decomposition is **efficient** if

$$\frac{2n^3}{3p} \geq \frac{3n^2g}{\sqrt{p}} \quad \text{and} \quad \frac{2n^3}{3p} \geq 8nl,$$

i.e.,

$$n \geq \max\{4.5g, 2\sqrt{3l}\} \cdot \sqrt{p}.$$

