Two-Phase Broadcasting

(PSC §2.4)
Optimising 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.
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

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

An $h$-relation is balanced if

\[ h = \frac{V}{p}. \]
Communication imbalance

- The **communication imbalance** of an $h$-relation is
  \[ h - \frac{V}{p}. \]

- If an $h$-relation is balanced, so that
  \[ V = \sum_{s=0}^{p-1} h_s(s) = ph, \]

  then $h_s(s) = h$ for all $s$. (Because $h_s(s) \leq h$.)
  Similarly, $h_r(s) = h$ for all $s$.

- The reverse is also true: if $h_s(s) = h$ for all $s$, then $V = ph$.

- Therefore, a **balanced** $h$-relation and a **full** $h$-relation are the same.
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: sending and receiving can have an equally overloaded processor, so that $h_s = h_r$, while the $h$-relation is still unbalanced, with $h \gg V/p$.

$h_s \neq h_r$ implies 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: $h_s \gg h_r$. Balance for senders must be improved to reduce the communication cost.
Cause of the communication imbalance

\((10a)\) if \(\phi_1(k) = t\) then for all \(i : k < i < n \land \phi_0(i) = s\) do

put \(a_{ik}\) in \(P(s, \ast)\); 

- The **sending** part of the broadcast of column \(k\) is unbalanced: only the \(\sqrt{p}\) processors in \(P(\ast, \phi_1(k))\) send.
- The senders send \(R_{k+1}(\sqrt{p} - 1) \approx n - k - 1\) elements.
- The **receiving** part is balanced: all processors receive \(R_{k+1} \approx (n - k - 1)/\sqrt{p}\) elements, except the senders.
- Total contribution of (10) to LU cost is about

\[
\sum_{k=0}^{n-1} 2(n - k - 1)g = 2g \sum_{k=0}^{n-1} k = 2g(n - 1)n/2 \approx n^2g.
\]

- This is a bottleneck vs. the computation cost \(2n^3/3p\).
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 $n$, $\text{repl}(\mathbf{x}) = P(\ast)$.

call: $\text{broadcast}(\mathbf{x}, P(0), P(\ast))$.

{ Broadcast the vector. }

(0) if $s = 0$ then for $t := 0$ to $p - 1$ do

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

put $x_i$ in $P(t)$;

Note: $\text{repl}(\mathbf{x}) = P(\ast)$ means that $\mathbf{x}$ is replicated such that each processor has a copy.
Two-phase broadcast in blocks

Phase 0

Phase 1
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, which 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.
- Idea is similar to two-phase randomised 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: \( x \) : vector of length \( n \), \( \text{repl}(x) = P(0) \).
output: \( x \) : vector of length \( n \), \( \text{repl}(x) = P(*) \).
call: \( \text{broadcast}(x, P(0), P(*)) \).

\[ b := \lceil \frac{n}{p} \rceil; \]
{ Spread the vector. }
(0) if \( s = 0 \) then for \( t := 0 \) to \( p - 1 \) do
    for \( i := tb \) to \( \min\{(t + 1)b, n\} - 1 \) do
        put \( x_i \) in \( P(t) \);

{ Broadcast the subvectors. }
(1) for \( i := sb \) to \( \min\{(s + 1)b, n\} - 1 \) do
    put \( x_i \) in \( P(*) \);
Cost analysis of two-phase broadcast

- Phase 0 costs \((n - b)g\), where \(b = \lceil n/p \rceil\) is the block size.
- Phase 1 costs \((p - 1)bg\).
- Total cost of 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

\[ \text{broadcast}((a_{ik} : k < i < n \land i \mod M = s), \]  
\[ P(s, k \mod N), P(s, *)) ; \]
\[ \text{broadcast}((a_{kj} : k < j < n \land j \mod N = t), \]  
\[ P(k \mod M, t), P(\ast, t)) ; \]

- 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).
- Less modular, but more efficient.
Optimisation: pivot value is already known

(8)  \[ \text{if } \phi_0(k) = s \land \phi_1(k) = t \text{ then put } a_{kk} \text{ in } P(\ast, t); \]

- Delete old superstep (8), because

  \[ a_{kk} \text{ (after swap)} = a_{rk} \text{ (before swap)}. \]

  Pivot value \( a_{rk} \) is already known locally.

- Divide immediately by \( a_{rk} \) in new superstep (2) of Algorithm 2.8:

(2)  \[ \text{if } k \mod N = t \text{ then} \]

  \[ s_{\text{max}} := \arg\max(|a_{rq,k}| : 0 \leq q < M); \]

  \[ r := r_{s_{\text{max}}}; \]

  \[ \text{for all } i : k \leq i < n \land i \mod M = s \land i \neq r \text{ do} \]

  \[ a_{ik} := a_{ik}/a_{rk}; \]
Optimisation: combine index and row swaps

(4) \[\text{if } k \mod M = s \text{ then} \]
\[\begin{align*}
\text{if } t = 0 \text{ then put } \pi_k \text{ as } \hat{\pi}_k \text{ in } P(r \mod M, 0); \\
\text{for all } j : 0 \leq j < n \land j \mod N = t \text{ do} \\
\quad \text{put } a_{kj} \text{ as } \hat{a}_{kj} \text{ in } P(r \mod M, t); \\
\text{if } r \mod M = s \text{ then} \\
\quad \text{if } t = 0 \text{ then put } \pi_r \text{ as } \hat{\pi}_r \text{ in } P(k \mod M, 0); \\
\quad \text{for all } j : 0 \leq j < n \land j \mod N = t \text{ do} \\
\quad \quad \text{put } a_{rj} \text{ as } \hat{a}_{rj} \text{ in } P(k \mod M, t); 
\end{align*}\]

Combining communication supersteps saves synchronisations.
Optimisation: combine first and last superstep

\[
\text{for } k := 0 \text{ to } n - 1 \text{ do }
\]

\[(0) \quad \text{if } k \mod N = t \text{ then }
\]
\[
rs := \arg\max(|a_{ik}| : k \leq i < n \land i \mod M = s);
\]

\[
\ldots
\]

\[(0') \quad \text{for all } i : k < i < n \land i \mod M = s \text{ do }
\]
\[
\text{for all } j : k < j < n \land j \mod N = t \text{ do }
\]
\[
a_{ij} := a_{ij} - a_{ik}a_{kj};
\]

- Combining the first and last superstep of the loop saves a synchronisation.
- In an implementation: no unnecessary \texttt{bsp\_sync} at the end of the main loop.
Optimal aspect ratio $M/N$

- Two-phase broadcast reduces cost. Is $M = N$ still optimal?
- The cost of $(6)/(7)$ is about $2(R_{k+1} + C_{k+1})g$. A bound is

$$R_{k+1} + C_{k+1} < \left( \frac{n - k - 1}{M} + 1 \right) + \left( \frac{n - k - 1}{N} + 1 \right)$$

$$= (n - k - 1) \frac{M + N}{p} + 2,$$

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

- The row and index swap in superstep (4) costs $(C_0 + 1)g$, where $C_0 = \lceil n/N \rceil$, so that larger values $N$ are preferred. Swap cost for $M = N$ is of same order as broadcast cost.

- Overall: $M = N$ close to optimal, but slight preference for $M < N$. 

Lecture 2.4 Two-phase broadcasting
Exact cost analysis

We need to compute sums of the form

\[
\sum_{k=0}^{n-1} R_k = \sum_{k=0}^{n-1} \left\lfloor \frac{n-k}{\sqrt{p}} \right\rfloor = \sum_{k=1}^{n} \left\lfloor \frac{k}{\sqrt{p}} \right\rfloor.
\]

Lemma 2.9. Let \( n, q \geq 1 \) be integers with \( n \mod q = 0 \). Then

\[
\sum_{k=0}^{n} \left\lfloor \frac{k}{q} \right\rfloor = \frac{n(n+q)}{2q}, \quad \sum_{k=0}^{n} \left\lfloor \frac{k}{q} \right\rfloor^2 = \frac{n(n+q)(2n+q)}{6q^2}.
\]
Proof Lemma 2.9 (first part)

\[
\sum_{k=0}^{n} \left\lceil \frac{k}{q} \right\rceil = \left\lceil \frac{0}{q} \right\rceil + \left( \left\lceil \frac{1}{q} \right\rceil + \cdots + \left\lceil \frac{q}{q} \right\rceil \right) + \cdots + \left( \left\lceil \frac{n - q + 1}{q} \right\rceil + \cdots + \left\lceil \frac{n}{q} \right\rceil \right)
\]

\[
= q \cdot 1 + q \cdot 2 + \cdots + q \cdot \frac{n}{q}
\]

\[
= q \sum_{k=1}^{n/q} k
\]

\[
= q \frac{n}{2q} \left( \frac{n}{q} + 1 \right)
\]

\[
= \frac{n(n + q)}{2q}.
\]
Total cost of LU decomposition

\[ T_{LU} = \frac{2n^3}{3p} + \left( \frac{3}{2\sqrt{p}} - \frac{2}{p} \right) n^2 + \frac{5n}{6} \]
\[ + \left( \left( \frac{3}{\sqrt{p}} - \frac{2}{p} \right) n^2 + \left( 4\sqrt{p} - \frac{4}{\sqrt{p}} + \frac{4}{p} - 3 \right) n \right) g \]
\[ + 8nl \]
\[ \approx \frac{2n^3}{3p} + \frac{3n^2}{2\sqrt{p}} + \frac{3n^2g}{\sqrt{p}} + 8nl. \]
Summary

- We have optimised 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;
  - reorganising the algorithm to **combine supersteps**, thus saving synchronisations.

- Cost analysis gives a diagnosis, such as \( h_s \gg h_r \).

- The resulting LU decomposition is efficient if

  \[
  \frac{2n^3}{3p} \geq \frac{3n^2g}{\sqrt{p}} \quad \text{and} \quad \frac{2n^3}{3p} \geq 8nl.
  \]

- Equivalent to \( n \geq \max\{4.5g, 2\sqrt{3l}\} \cdot \sqrt{p} \).