THE UNIVERSITY OF WARWICK

Modern Parallel Algorithms

Lecture 2

Artur Czumaj

DIMAP and Department of Computer Science

University of Warwick

Warszawa, November 2022

Basic primitives for MPC algorithms

- Communication primitives
  - broadcasting, communication gathering, …
- Basic data primitives
  - sum computation, prefix-sums, sorting, …
- Simulations
  - PRAM simulations, BSP simulations
  - Distributed CONGESTED CLIQUE and LOCAL simulations

Broadcasting

- There is a single machine (say, M_1) having some N words of data stored in its local memory (and hence $N \leq s$)
- Goal: inform all machines in the system about this data
- If $N \cdot$ (

Broadcasting can be done in $O(\log_s M)$ rounds

- Otherwise, if $N \ll s$ then easily $O(\log_{(1+s/N)} M)$ rounds
- Otherwise, if $N \sim s$ or $N \geq s$ then $O(\log_s M)$ rounds

What does it mean when $N \geq s$?
It's when $\lceil N/s \rceil$ machines stores the input data and goal is to broadcast data to blocks of $\lceil N/s \rceil$ machines

Basic primitives for MPC algorithms

- Communication primitives
  - broadcasting, communication gathering, …
- Basic data primitives
  - sum computation, prefix-sums, sorting, …
- Simulations
  - PRAM simulations, BSP simulations
  - Distributed CONGESTED CLIQUE and LOCAL simulations

Basic data manipulation primitives

- Filtering:
  - input: N objects $x_1, \ldots, x_N$ and a predicate function f
  - output: a sequence of all objects x_i for which $f(x_i) = true$
- Predecessor:
  - input: N objects $x_1 \ldots, x_N$, each associated with a 0-1 value
  - output: a sequence $y_1 \ldots, y_N$ such that y_j is the last object x_j, with $j' < j$ which has associate value of 1
- Prefix sums:
  - input: N objects $x_1, \ldots, x_N$ and there is an associate operator $\oplus$
  - output: a sequence $y_1, \ldots, y_N$ such that $y_j = \oplus_{i=1}^{j} x_i$

Basic data manipulation primitives

- Sorting:
  - input: N objects $x_1 \ldots, x_N$ and there is a comparison function $\lesssim$ to compare any two objects (to define the total order of the input objects)
  - output: the input objects rearranged in sorted order with respect to $\lesssim$
- Duplicate removal:
  - input: N objects
  - output: set of all input objects after removing duplicates (each input object appears exactly once in the output)
- Colored summation:
  - input: a sequence of N colored numbers
  - output: for each color, sum of numbers of that color

Basic data manipulation primitives

- Sorting:

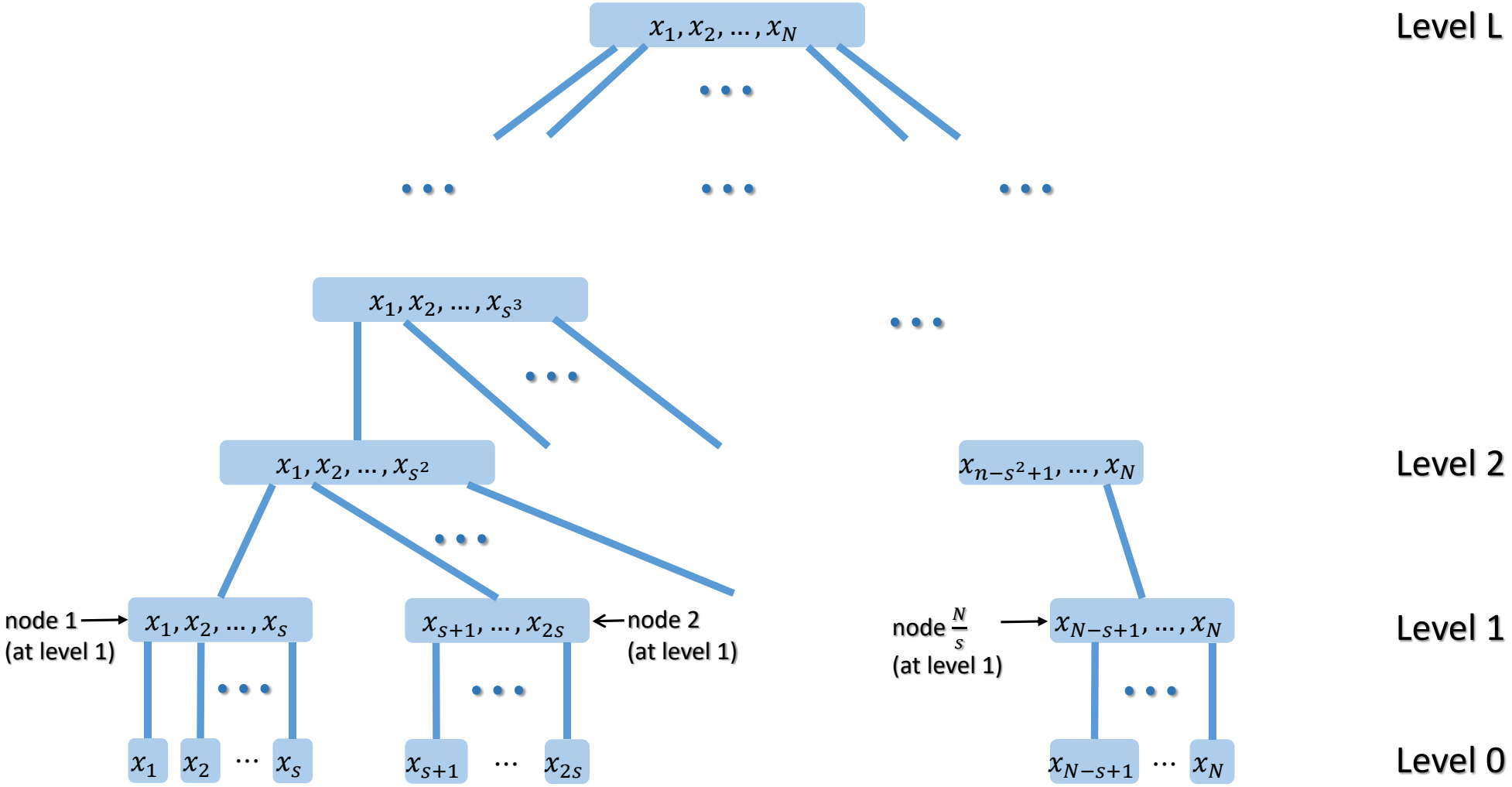
Theorem: Let $0 < \delta < 1$. The following problems can be solved deterministically in $O(1/\delta)$ rounds on an MPC with local space $s = O(N^{\delta})$ and linear global space $O(N)$:
(i) summation, (ii) filtering, (iii) predecessor, (iv) prefix sums, (v) sorting, (vi) duplicate removal, (vii) colored summation.

- Colored summation:
  - input: a sequence of N colored numbers
  - output: for each color, sum of numbers of that color

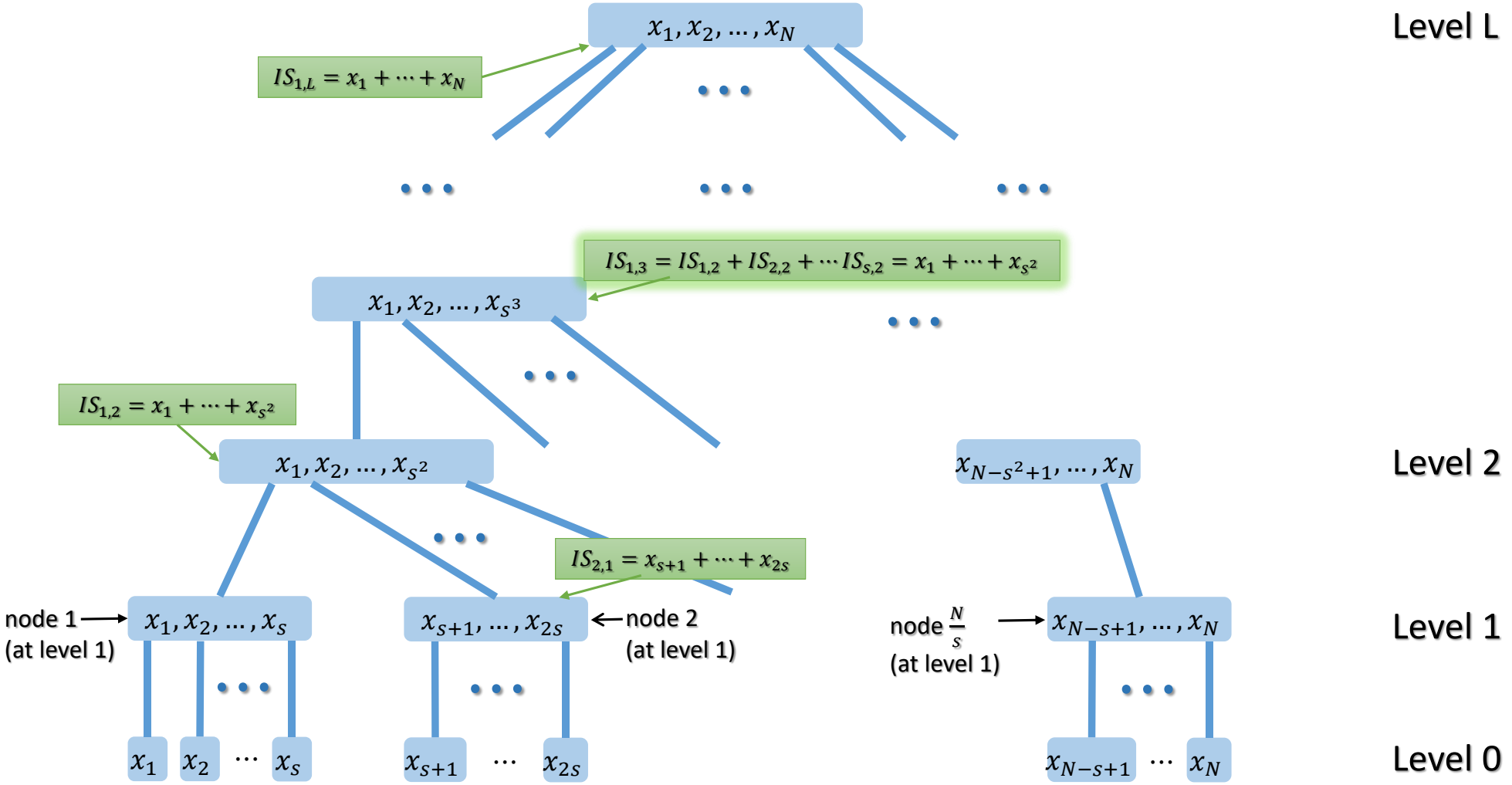
Basic data manipulation primitives

Theorem: Let $0 < \delta < 1$. The following problems can be solved deterministically in $O(1/\delta)$ rounds on an MPC with local space $s = O(N^{\delta})$ and linear global space $O(N)$:
(i) summation, (ii) filtering, (iii) predecessor, (iv) prefix sums, (v) sorting, (vi) duplicate removal, (vii) colored summation.

Main paradigm: s-ary tree



Summing N elements



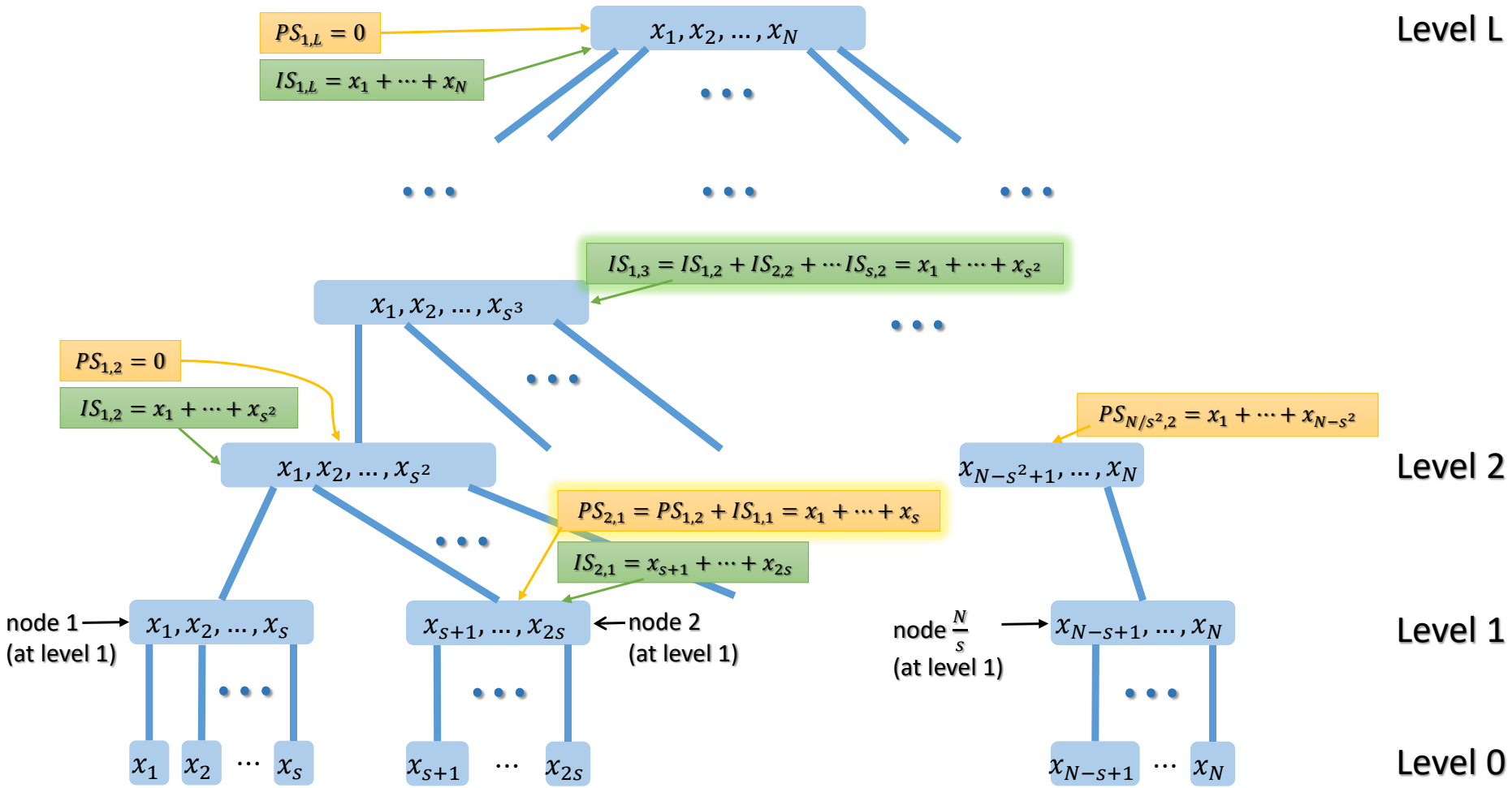
Summing N elements

Input: numbers $x_1, \ldots, x_N$; machine $M_{i,1}$ stores $x_{(i-1)\cdot s+1}, \ldots, x_{\min\{i\cdot s, N\}}$

Output: machine $M_{1,L}$ knows $\sum_{i=1}^{N} x_i$

1 $L := \lceil \log_s N \rceil$
2 **for** $\ell = 1$ **to** L **do**
3 **for** $i = 1$ **to** $\lceil \frac{N}{s^\ell} \rceil$ **do in parallel**
4 // Invariant: Machine $M_{i,\ell}$ knows $IS_{j,\ell-1}$ for $(i-1)s+1 \le j \le \min\{is, \lceil \frac{N}{s^{\ell-1}} \rceil\}$
5 Machine $M_{i,\ell}$ computes $IS_{i,\ell} := \sum_{j=(i-1)s+1}^{\min\{is, \lceil \frac{N}{s^{\ell-1}} \rceil\}} IS_{j,\ell-1}$
6 **if** $\ell < L$ **then** machine $M_{i,\ell}$ sends $IS_{i,\ell}$ to machine $M_{\lceil \frac{i}{s} \rceil, \ell+1}$
7 **end**
8 **end**

All-prefix sum



All-prefix sum

Input: numbers $x_1, \ldots, x_N$; machine $M_{i,1}$ stores $x_{(i-1)\cdot s+1}, \ldots, x_{\min\{i\cdot s, N\}}$
Output: $S_1, \ldots, S_N$ with $S_i = \sum_{j=1}^{i} x_j$; machine $M_{i,1}$ stores $S_{(i-1)\cdot s+1}, \ldots, S_{\min\{i\cdot s, N\}}$

$L := \lceil \log_s N \rceil$
for $\ell = 1$ to L do
  for $i = 1$ to $\lceil \frac{N}{s^\ell} \rceil$ do in parallel
    // Invariant: Machine $M_{i,\ell}$ knows $IS_{j,\ell-1}$ for $(i-1)s+1 \le j \le \min\{is, \lceil \frac{N}{s^{\ell-1}} \rceil\}$
    Machine $M_{i,\ell}$ computes $IS_{i,\ell} := \sum_{j=(i-1)s+1}^{\min\{is, \lceil \frac{N}{s^{\ell-1}} \rceil\}} IS_{j,\ell-1}$
    if $\ell < L$ then machine $M_{i,\ell}$ sends $IS_{i,\ell}$ to machine $M_{\lceil \frac{i}{s} \rceil, \ell+1}$
  end
end
$PS_{1,L} := 0$
for $\ell = L$ downto 1 do
  for $i = 1$ to $\lceil \frac{n}{s^\ell} \rceil$ do in parallel
    // Invariant: Machine $M_{i,\ell}$ knows $PS_{i,\ell}$
    for $j = (i-1)s + 1$ to $\min\{is, \lceil \frac{N}{s^{\ell-1}} \rceil\}$ do
      Machine $M_{i,\ell}$ computes $PS_{j,\ell-1} := PS_{i,\ell} + \sum_{b=(i-1)s+1}^{j-1} IS_{b,\ell-1}$
      if $\ell > 1$ then machine $M_{i,\ell}$ sends $PS_{j,\ell-1}$ to machine $M_{j,\ell-1}$
    end
  end
end
for $i = 1$ to $\lceil \frac{N}{s} \rceil$ do in parallel
  for $j = (i-1)s + 1$ to $\min\{is, N\}$ do
    Machine $M_{i,1}$ computes $S_j := PS_{i,1} + \sum_{h=(i-1)s+1}^{j} x_h$
  end
end

Sorting

- This is a FUNDAMENTAL primitive
- Sorting of N numbers on a PRAM with $poly(N)$ processors requires $\Omega(\log N)$ rounds.
- On MPC we can do it in constant number of rounds (if $s = N^{\Theta(1)}$)
- or in $O(\log_s N)$ rounds in general
- We assume, without loss of generality, that all input numbers are distinct

Sorting

Let's start with the basics:

Theorem: One can deterministically sort N numbers in $O(\log_s N)$ rounds on an MPC with local space s with total space $\mathrm{O}(N^2)$.

Can we achieve the same bound if total space is linear?

Sorting

Definition (r-pivot):

Let I be any set of distinct reals. For any integer r, a subset P of I is called an r-pivot if after ordering the elements from P as $y_1 < \cdots < y_h$, and setting $y_0 = -\infty$ and $y_{h+1} = +\infty$, then $|\{x \in I : y_i < x \leq y_{i+1}\}| \leq r$ for all $0 \leq i \leq h$.

Definition (Pivot-net)

Let I be any set of distinct reals. For any integer r and any r-pivot P $y_1 < y_2 < \cdots < y_h$, a pivot-net is a partition of I into $h + 1$ sets $\Pi_0, \Pi_1 \ldots, \Pi_h$ such that $\Pi_i = \{x \in I : y_i < x \leq y_{i+1}\}$, where we set $y_0 = -\infty$ and $y_{h+1} = +\infty$.

Sorting

Skeleton of a parallel random sampling based sorting

1. Randomly sample κ input elements (pivots) $a_1, \ldots, a_\kappa$ and send them to all machines
2. On each machine, locally split all input elements from that machines into $\kappa + 1$ intervals, with the ith interval containing all elements between the $(i-1)$-st smallest sampled element and the i-th smallest sampled element
3. On each machine, send all elements from its ith interval to the ith machine
4. For every $1 \leq i \leq \kappa + 1$: on the ith machine locally sort all elements received in the previous step

Sorting

High-level description of pivot-based sorting
1 Find an r-pivot (for an appropriate value of r)
2 Find a corresponding pivot-net $\Pi_0, \Pi_1, \ldots, \Pi_h$
3 Recursively sort each $\Pi_0, \Pi_1, \ldots, \Pi_h$

Sorting

Random sampling based sorting

i. How large the pivot set P (obtained via random sampling) should be?

ii. With the pivot set P at hand, how to compute the corresponding pivot-net on MPC?

iii. How to rearrange the pivot-net (elements from each Π_i) on MPC machines for efficient MPC implementation in the recursive calls?

Sorting

Lemma. Let $0 < p < 1$ and $\alpha > 0$ be arbitrary, and let N be sufficiently large. For any set of N distinct reals, let us choose the pivot set P by selecting (independently at random) each of the N input elements to P with probability p.

- If $p \geq \frac{3\,\alpha \ln N}{N}$, then $|P| \leq 2pN$ with probability at least $1 - N^{-\alpha}$.
- With probability at least $1 - N^{-\alpha}$ set P is an r-pivot with $r = \left\lceil \frac{(\alpha+1)\ln N}{p} \right\rceil$

One should read Lemma that if P is a random sample choosing each element with probability $\frac{h}{N}$, then $|P| = O(h)$ and P is an r-pivot set with $r = O(\frac{N \log N}{h})$, with high probability (assuming h is not too small)

Sorting

First attempt: choose $h := O(N^{1/3} \log^{2/3} N)$

1. Choose a random pivot set P (with prob. $\frac{h}{N}$)
2. Move P to a single machine (say, M_1) and then broadcast it to all machines
3. Each machine M_i determines its own pivot-net $\Pi_0^{<i>}, \dots, \Pi_h^{<i>}$ w.r.t. P
4. Each machine M_i sends its set $\Pi_j^{<i>}$ to machine M_j
5. Each machine M_j sorts the elements from $\Pi_j := \bigcup_i \Pi_j^{<i>}$
6. Combine sorted sets $\Pi_0, \Pi_1, \dots, \Pi_h$ into a single sorted sequence

Sorting

First attempt: choose $h := O(N^{1/3} \log^{2/3} N)$

Second attempt: choose a better h and revise algorithm

1. Choose a random pivot set P (with prob. $\frac{h}{N}$)
2. Move P to a single machine (say, M_1) and then broadcast it to all machines
3. Each machine M_i determines its own pivot-net $\Pi_0^{<i>}, \dots, \Pi_h^{<i>}$ w.r.t. P
4. Each machine M_i sends its set $\Pi_j^{<i>}$ to machine M_j
5. Each machine M_j sorts the elements from $\Pi_j := \bigcup_i \Pi_j^{<i>}$
6. Combine sorted sets $\Pi_0, \Pi_1, \dots, \Pi_h$ into a single sorted sequence

Sorting

Randomized sorting of N numbers on an MPC with $N \leq$

Input: distinct numbers $x_1, \ldots, x_N$; machine M_i stores $x_{(i-1)\cdot s+1}, \ldots, x_{i\cdot s}$, $1 \leq i \leq \mathfrak{m}$
Output: machine M_i stores $x_{\pi((i-1)\cdot s+1)}, \ldots, x_{\pi(i\cdot s)}$, $1 \leq i \leq \mathfrak{m}$, where π is an n-permutation such that $x_{\pi(1)} < x_{\pi(2)} < \cdots \leq x_{\pi(N)}$

1 Set $h = \Theta(\mathfrak{m} \cdot \log N)$
2 Partition the set $Q := \{0, 1, \ldots, 2h\}$ into $\mathfrak{m}$ sets $Q_1, \ldots, Q_{\mathfrak{m}}$ with $|Q_k| \leq \lceil \frac{2h+1}{\mathfrak{m}} \rceil$ for each k
3 Each machine goes through all its input elements and locally marks each as being in pivot set $\mathcal{P}$ independently at random with probability $\frac{h}{N}$
4 Each machine sends all its pivot elements to the first machine M_1
5 Machine M_1 broadcasts the entire set $\mathcal{P}$ to all machines
6 Each machine M_j, for every $0 \leq i \leq |\mathcal{P}|$, locally determines set $\Pi_i^{\langle j \rangle}$ of the elements from machine M_j that are in set Π_i
7 Each machine M_j, for every $0 \leq i \leq |\mathcal{P}|$, sends $|\Pi_i^{\langle j \rangle}|$ to M_k, where $k \in Q_j$
8 Each machine M_j locally compute $|\Pi_k| := \sum_{j=1}^{\mathfrak{m}} |\Pi_k^{\langle j \rangle}|$ for all $k \in Q_j$
9 Each machine M_j sends all its values $|\Pi_k|$ with $k \in Q_j$ to machine M_1
10 Machine M_1 locally assigns the sets $\Pi_0, \Pi_1, \ldots, \Pi_{|\mathcal{P}|}$ to $O(\frac{N}{s})$ machines so that (i) all objects from the same set Π_i, $0 \leq i \leq |\mathcal{P}|$, are assigned to the same machine $M_{\tau(i)}$, (ii) $\tau(0) \leq \tau(1) \leq \cdots \leq \tau(|\mathcal{P}|)$, and (iii) each machine receives at most s objects
11 Machine M_1 broadcasts numbers $\tau(0), \ldots, \tau(|\mathcal{P}|)$ and sizes $|\Pi_0|, \ldots, |\Pi_{|\mathcal{P}|}|$ to all machines
12 Each machine M_j sends all elements from each set $\Pi_i^{\langle j \rangle}$, $0 \leq i \leq |\mathcal{P}|$, to machine $M_{\tau(i)}$
13 Each machine M_i:
14 ▶ locally computes $\mathit{shift}(i) := \sum_{j:\tau(j)<i} |\Pi_j|$
15 ▶ for any input element x on M_i, locally computes its rank $\mathit{rank}_i(x)$ on machine M_i
16 ▶ for any input element x on M_i, sets its global rank $\mathit{rank}(x) := \mathit{rank}_i(x) + \mathit{shift}(i)$
17 ▶ sends x (and its rank) to machine $M_{\lfloor \mathit{rank}(x)/s \rfloor}$

$\mathfrak{m}$ – number of machines used
$\mathfrak{m} = O(N/s)$

Sorting

$\mathfrak{m}$ – number of machines used
$\mathfrak{m} = O(N/\mathfrak{s})$

Randomized sorting of N numbers on an MPC with $N \leq O(\frac{\mathfrak{s}^2}{\log \mathfrak{s}})$

Input: distinct numbers $x_1, \ldots, x_N$; machine M_i stores $x_{(i-1)\cdot\mathfrak{s}+1}, \ldots, x_{i\cdot\mathfrak{s}}$, $1 \leq i \leq \mathfrak{m}$
Output: machine M_i stores $x_{\pi((i-1)\cdot\mathfrak{s}+1)}, \ldots, x_{\pi(i\cdot\mathfrak{s})}$, $1 \leq i \leq \mathfrak{m}$, where π is an n-permutation such that $x_{\pi(1)} < x_{\pi(2)} < \cdots \leq x_{\pi(N)}$

1 Set $h = \Theta(\mathfrak{m} \cdot \log N)$
2 Partition the set $Q := \{0, 1, \ldots, 2h\}$ into $\mathfrak{m}$ sets $Q_1, \ldots, Q_{\mathfrak{m}}$ with $|Q_k| \leq \lceil \frac{2h+1}{\mathfrak{m}} \rceil$ for each k
3 Each machine goes through all its input elements and locally marks each as being in pivot set $\mathcal{P}$ independently at random with probability $\frac{h}{N}$
4 Each machine sends all its pivot elements to the first machine M_1
5 Machine M_1 broadcasts the entire set $\mathcal{P}$ to all machines
6 Each machine M_j, for every $0 \leq i \leq |\mathcal{P}|$, locally determines set $\Pi_i^{\langle j \rangle}$ of the elements from machine M_j that are in set Π_i
7 Each machine M_j, for every $0 \leq i \leq |\mathcal{P}|$, sends $|\Pi_i^{\langle j \rangle}|$ to M_k, where $k \in Q_j$
8 Each machine M_j locally compute $|\Pi_k| := \sum_{j=1}^{\mathfrak{m}} |\Pi_k^{\langle j \rangle}|$ for all $k \in Q_j$
9 Each machine M_j sends all its values $|\Pi_k|$ with $k \in Q_j$ to machine M_1
10 Machine M_1 locally assigns the sets $\Pi_0, \Pi_1, \ldots, \Pi_{|\mathcal{P}|}$ to $O(\frac{N}{\mathfrak{s}})$ machines so that (i) all objects from the same set Π_i, $0 \leq i \leq |\mathcal{P}|$, are assigned to the same machine $M_{\tau(i)}$, (ii) $\tau(0) \leq \tau(1) \leq \cdots \leq \tau(|\mathcal{P}|)$, and (iii) each machine receives at most $\mathfrak{s}$ objects
11 Machine M_1 broadcasts numbers $\tau(0), \ldots, \tau(|\mathcal{P}|)$ and sizes $|\Pi_0|, \ldots, |\Pi_{|\mathcal{P}|}|$ to all machines
12 Each machine M_j sends all elements from each set $\Pi_i^{\langle j \rangle}$, $0 \leq i \leq |\mathcal{P}|$, to machine $M_{\tau(i)}$
13 Each machine M_i:
14 ▶ locally computes $\mathit{shift}(i) := \sum_{j:\tau(j)<i} |\Pi_j|$
15 ▶ for any input element x on M_i, locally computes its rank $\mathit{rank}_i(x)$ on machine M_i
16 ▶ for any input element x on M_i, sets its global rank $\mathit{rank}(x) := \mathit{rank}_i(x) + \mathit{shift}(i)$
17 ▶ sends x (and its rank) to machine $M_{\lfloor \mathit{rank}(x)/\mathfrak{s} \rfloor}$

Sorting

- One can do "similarly" for arbitrary N and s: to sort in $O(\log_s N)$ rounds
- This can be done even deterministically!

Corollary: Sorting of N numbers on an MPC with local space $s = N^{\delta}$ and linear total space $O(N)$ can be done deterministically in $O(1)$ rounds

PRAM: sorting lower bound of $\Omega(\log N)$ time (with any $\mathrm{poly}(N)$ number of processors)