Sorting and Other Distributed Set Operations

T-79.4001 Seminar on Theoretical Computer Science
Spring 2007 – Distributed Computation

Eero Häkkinen

2007-04-18

Based on sections 5.3-5.4 of
N. Santoro: Design and Analysis of Distributed Algorithms, Wiley 2007.

Definitions (1/3)

Notation (1/2)

- a *local set* D_x in an entity x
- a *distributed set* $\mathcal{D} = \bigcup_x D_x$
- a *distribution* $\mathbf{D} = \langle D_{x_1}, D_{x_2}, \ldots, D_{x_n} \rangle$ of $\mathcal{D}$ among the entities $x_1, x_2, \ldots, x_n$
- the number of data items $N = \sum_x |D_x|$
- a *topology* G of the network

Definitions (2/3)

Notation (2/2)

- a *permutation* π of the indices $\{1, 2, \ldots, n\}$
- an *i*th item $\pi(i)$ of π
  (if $\pi = \langle 2, 4, 1, 3 \rangle$, then $\pi(2) = 4$)

For Simplicity

- $id(x_i) = i$
- D_i denotes D_{x_i}

Definitions (3/3)

Sorting Condition

The distribution $\langle D_1, D_2, \ldots, D_n \rangle$ is *sorted according to* π if and only if the following sorting condition holds:

$$i < j \Rightarrow \forall d' \in D_{\pi(i)}, d'' \in D_{\pi(j)} : d' < d''$$

Some Sorting Orders

- *increasing order*: $\pi = \langle 1, 2, \ldots, n \rangle$
- *decreasing order*: $\pi = \langle n, (n-1), \ldots, 1 \rangle$

Sorting Problem

Sorting Problem

Then the initial distribution of $\mathcal{D}$ is $\mathbf{D} = \langle D_1, D_2, \ldots, D_n \rangle$, the problem is to move data items among the entities so that the final distribution of $\mathcal{D}$ is $\mathbf{D}' = \left\langle D'_1, D'_2, \ldots, D'_n \right\rangle$ and the distribution $\mathbf{D}'$ is sorted according to π.

Notes

No relation is defined between D_i and D'_i, yet. There are thus multiple variations of the problem.

Sorting and Distribution Types

Fundamental Requirements

- *invariant-sized* sorting:
  $\left|D'_i\right| = |D_i|, 1 \leq i \leq n$
- *equidistributed* sorting:
  $$\left|D'_{\pi(i)}\right| = \begin{cases} \left\lceil \frac{N}{n} \right\rceil, & \text{if } 1 \leq i < n \\ N - (n-1)\left\lceil \frac{N}{n} \right\rceil, & \text{if } i = n \end{cases}$$
- *compacted* sorting:
  $$\left|D'_{\pi(i)}\right| = \begin{cases} w, & \text{if } 1 \leq i < \frac{N}{w} \\ N - (i-1)\,w, & \text{if } i = \left\lceil \frac{N}{w} \right\rceil \leq n \\ 0, & \text{if } \left\lceil \frac{N}{w} \right\rceil < i \leq n \end{cases}, \text{ where}$$
  $w \geq \left\lceil \frac{N}{n} \right\rceil$ is the *storage capacity* of the entities

Description of OddEven-LineSort (1/2)

Restrictions

- Standard restrictions **R.**
- Ordered line: links $\left(x_{\pi(i)}, x_{\pi(i+i)}\right), 1 \leq i < n$.

Origin

- Based on the parallel algorithm odd-even-transposition sort, which is based on the serial algorithm bubble sort.

Description of OddEven-LineSort (2/2)

Technique

1. In an odd iteration, entities $x_{\pi(2i+1)}$ and $x_{\pi(2i+2)}$, $0 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor - 1$ exchange data items. The smallest items are retained by $x_{\pi(2i+1)}$ and the largest ones are retained by $x_{\pi(2i+2)}$.
2. In an even iteration, entities $x_{\pi(2i)}$ and $x_{\pi(2i+1)}$, $1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor - 1$ exchange data items. The smallest items are retained by $x_{\pi(2i)}$ and the largest ones are retained by $x_{\pi(2i+1)}$.
3. If no items change the place in an iteration other than the first one, the process stops.

Properties of OddEven-LineSort

Complexity

- Sorting an equidistributed distribution requires at most $n - 1$ iterations if the required sorting is invariant-sized, equidistributed or compacted.
- Invariant-sized sorting requires at most $N - 1$ iterations.
- $\mathbf{T}\left[OddEven{-}LineSort_{\text{invariant}}\right] = O\left(nN\right)$
- $\mathbf{M}\left[OddEven{-}LineSort_{\text{invariant}}\right] = O\left(nN\right)$

Description of OddEven-Merge (1/2)

Restrictions

- Standard restrictions **R**.
- Complete graph.

Initially

- Sorted partial distributions $\left\langle A_1, A_2, \ldots, D_{\frac{p}{2}} \right\rangle$ and $\left\langle A_{\frac{p}{2}+1}, A_{\frac{p}{2}+2}, \ldots A_p \right\rangle$.
- For simplicity, p is a power of 2.

Description of OddEven-Merge (2/2)

Technique

1. If $p = 2$, there are two entities y_1 and y_2 containing sets A_1 and A_2. Entities y_1 and y_2 exchange data items. The smallest items are retained by y_1 and the largest ones are retained by y_2. This is a *merge*.
2. If $p > 2$,
    1. Recursively OddEven-Merge partial distributions $\left\langle A_1, A_3, \ldots, A_{\frac{p}{2}-1} \right\rangle$ and $\left\langle A_{\frac{p}{2}+1}, A_{\frac{p}{2}+3}, \ldots, A_{p-1} \right\rangle$.
    2. Recursively OddEven-Merge partial distributions $\left\langle A_2, A_4, \ldots, A_{\frac{p}{2}} \right\rangle$ and $\left\langle A_{\frac{p}{2}+2}, A_{\frac{p}{2}+4}, \ldots, A_p \right\rangle$.
    3. Merge A_{2i} and A_{2i+1}, $1 \leq i \leq \frac{p}{2} - 1$.

Description of OddEven-MergeSort

Technique

1. Recursively OddEven-MergeSort the partial distribution $\left\langle D_1, D_2, \ldots, D_{\frac{n}{2}} \right\rangle$.
2. Recursively OddEven-MergeSort the partial distribution $\left\langle D_{\frac{n}{2}+1}, D_{\frac{n}{2}+2}, \ldots, D_n \right\rangle$.
3. OddEven-Merge partial distributions $\left\langle D_1, D_2, \ldots, D_{\frac{n}{2}} \right\rangle$ and $\left\langle D_{\frac{n}{2}+1}, D_{\frac{n}{2}+2}, \ldots, D_n \right\rangle$.

Properties of OddEven-MergeSort

Complexity

- Sorting requires at most $1 + \log n$ iterations.
- $\mathbf{M}\left[OddEven{-}MergeSort\right] = O\left(N \log n\right)$

Correctness

Does it work? Not always.

Lower Bounds – Analysis

Sorting Problem (recapitulation)

Then the initial distribution of $\mathcal{D}$ is $\mathbf{D} = \langle D_1, D_2, \ldots, D_n \rangle$, the problem is to move data items among the entities so that the final distribution of $\mathcal{D}$ is $\mathbf{D}' = \left\langle D'_1, D'_2, \ldots, D'_n \right\rangle$ and the distribution $\mathbf{D}'$ is sorted according to π.

Messages

- $\left| D_i \cap D'_j \right|$ items to be moved from x_i to x_j.
- At least $d_G\left(x_i, x_j\right)$ messages for each item to be moved.
- The total cost at least $\mathbf{C}\left(\mathbf{D}, G, \pi\right) = \sum_{i \neq j} \left| D_i \cap D'_j \right| d_G\left(x_i, x_j\right)$.

Lower Bounds – Values

Ordered Line

- $\mathbf{C}(\mathbf{D}, G, \pi) = \sum_{i \neq j} \left| D_i \cap D'_j \right| d_G(x_i, x_j) = \Omega(nN)$
- OddEven-LineSort has $O(nN)$. The same!

Complete Graph

- $\mathbf{C}(\mathbf{D}, G, \pi) = \sum_{i \neq j} \left| D_i \cap D'_j \right| \underbrace{d_G(x_i, x_j)}_{=1} = \Omega(N)$
- OddEven-MergeSort has $O(N \log n)$.

Description of SelectSort

Technique

1. Entity $x_{\pi(j)}$ broadcasts the number of items k_j it must end up with.
2. The entities find the k_jth smallest item b_j still under consideration using a distributed selection algorithm.
3. The item b_j is broadcasted.
4. Each entity assigns items which are still under consideration and smaller or equal to b_j to be sent to $x_{\pi(j)}$.

After $n - 1$ iterations, items are sent to their destinations using the shortest paths.

Properties of SelectSort

Properties

- Generic regarding topology.
- Correct if the distributed selection algorithm is correct.
- Additional cost of iterations is
$$\sum_{1 \le i \le n-1} M\left[k_i, N - K_{i-1}\right] =$$
$$M\left[Rank\right] \sum_{1 \le i \le n-1} \log\left(\min\left\{k_i, N - K_i + 1\right\}\right) + l.o.t.$$
- If $N \gg n$ (for instance $N \ge n^2 \log n$) in a complete graph, the additional cost is $o\left(N\right)$ and the total cost is $O\left(N\right)$.

Description of DynamicSelectSort

Protocol

begin
  for $j = i, \ldots, n-1$ **do**
    Collectively determine $b_j = \mathbf{D}[k_j]$ using distributed selection;
    $D_{i,j} := d \in D_i : b_{j-1} < d \leq b_j$;
    $n_i(j) := |D_{i,j}|$;
  end
  $D_{i,n} := d \in D_i : b_{n-1} < d$;
  $n_i(n) := |D_{i,n}|$;
  if $x_i \neq \overline{x}$ **then**
    send $\langle n_i(1), \ldots, n_i(n) \rangle$ **to** $\overline{x}$;
  else
    wait until receive information from all entities;
    determine $\overline{\pi}$ and notify all entities;
  end
  send $D_i(j)$ **to** $x_{\pi(j)}, i \leq j \leq n$;
end

Properties of DynamicSelectSort

Properties

- Selects a permutation which results the least amount of items to be moved.
- Sorts according to the selected permutation.
- Does not move items if already sorted.
- Additional cost is $\sum_{x} \left(|N(x)| + 2n\right) d_G(x, \overline{x})$.

Operations on Distributed Sets (1/2)

Notation

- sets $A, B, C, \ldots$
- an entity $x(A), x(B), x(C), \ldots$ owning the corresponding set
- an entity x making a query
- a strategy S_i to find the result of a query

Query Expression Example

$$A \cup ((B \cap C) \setminus (B \cap D))$$

Operations on Distributed Sets (2/2)

Costs of Some Strategies

- $Vol(S_1) = \underbrace{|A|}_{x(A)\to x} + \underbrace{|B|}_{x(B)\to x} + \underbrace{|C|}_{x(C)\to x} + \underbrace{|D|}_{x(D)\to x}$
- $Vol(S_2) = \underbrace{|B|}_{x(B)\to x(C)} + \underbrace{|B \cap C|}_{x(C)\to x(D)} + \underbrace{|(B \cap C) \setminus D|}_{x(D)\to x(A)} + \underbrace{|A \cup ((B \cap C) \setminus D)|}_{x(A)\to x}$
- $Vol(S_3) = \underbrace{|C|}_{x(C)\to x(D)} + \underbrace{|C \setminus D|}_{x(D)\to x(B)} + \underbrace{|A|}_{x(A)\to x(B)} + \underbrace{|A \cup (B \cap (C \setminus D))|}_{x(B)\to x}$

Description of Intersection Difference Partitioning

Motivation

Some queries can be evaluated locally.

Intersection Difference Partitioning (IDP)

- $Z^i_{0,1} = D_i$
- $S_1, S_2, \ldots$ are sets $D_1, D_2, \ldots$ excluding D_i
- $Z^i_{l+1,2j-1} = Z^i_{l,j} \cap S_{l+1}$
- $Z^i_{l+1,2j} = Z^i_{l,j} \setminus S_{l+1}$
- $Z^i_{n-1,j} = Z^i_j$
- $Z_i = \left\langle Z^i_1, Z^i_2, \ldots, Z^i_{2^{n-1}} \right\rangle$ is a partition of D_i and denotes it.

Example of IDP

Partitioning

$$D_1 = Z^1_{0,1} = \{a, b, e, f, g, m, n, q\}$$
$$Z^1_{1,1} = \{a, e, f, g\} \qquad Z^1_{1,2} = \{b, m, n, q\}$$
$$Z^1_{2,1} = \{e, f\} \quad Z^1_{2,2} = \{a, g\} \quad Z^1_{2,3} = \{m, q\} \quad Z^1_{2,4} = \{b, n\}$$

$$D_2 = Z^2_{0,1} = \{a, e, f, g, o, p, r, u, v\}$$
$$Z^2_{1,1} = \{a, e, f, g\} \qquad Z^2_{1,2} = \{o, p, r, u, v\}$$
$$Z^2_{2,1} = \{e, f\} \quad Z^2_{2,2} = \{a, g\} \quad Z^2_{2,3} = \{p, r, v\} \quad Z^2_{2,4} = \{o, u\}$$

$$D_3 = Z^3_{0,1} = \{e, f, m, p, q, r, v\}$$
$$Z^3_{1,1} = \{e, f, m, q\} \qquad Z^3_{1,2} = \{p, r, v\}$$
$$Z^3_{2,1} = \{e, f\} \quad Z^3_{2,2} = \{m, q\} \quad Z^3_{2,3} = \{p, r, v\} \quad Z^3_{2,4} = \{\}$$

Properties of IDP

Expressions

$$
\begin{aligned}
Z^i_{l,j} &= \bigcup_{1 \le k \le 2^{n-1-l}} Z^i_{k+(j-1)2^{n-1-l}} \\
D_i &= \bigcup_{1 \le j \le 2^l} Z^i_{l,j} &&= \bigcup_{1 \le j \le 2^{n-1}} Z^i_j \\
D_i \cap S_l &= \bigcup_{1 \le j \le 2^{l-1}} Z^i_{l,2j-1} &&= \bigcup_{\substack{1 \le j \le 2^{l-1} \\ 1 \le k \le 2^{n-1-l}}} Z^i_{k+(j-1)2^{n-l}} \\
D_i \setminus S_l &= \bigcup_{1 \le j \le 2^{l-1}} Z^i_{l,2j} &&= \bigcup_{\substack{1 \le j \le 2^{l-1} \\ 1 \le k \le 2^{n-1-l}}} Z^i_{k+(2j-1)2^{n-l-1}}
\end{aligned}
$$

Local Evaluation Using IDP

Local Queries

- If an expression E can be evaluated locally and an expression E' is an arbitrary local expression, then $E \cap E'$ can be evaluated locally.
- The same is true for $E \setminus E'$.
- If an expressions E_1 and E_2 can be evaluated locally, then $E_1 \cup E_2$ can be evaluated locally.

Properties

- No messages are sent.

Global Evaluation (1/2)

Technique

1. x decomposes a query Q into sub-queries $Q_1, Q_2, \ldots, Q_k$ which satisfy the following properties:
   - $\forall Q_j : \exists y_j : Q_j \in E(y_j)$, where $E(y_j)$ is the sets of expressions y_j can evaluate locally.
   - $\forall i \neq j : Q_i \cap Q_j = \varnothing$
   - $Q = \bigcup_{1 \leq j \leq k} Q_j$
2. x sends Q_js to y_js.
3. y_j evaluates Q_j locally and sends the result to x.
4. x computes the union of all received results.

Global Evaluation (2/2)

Properties

- Each result item is sent only once.
- Data transfer optimal.