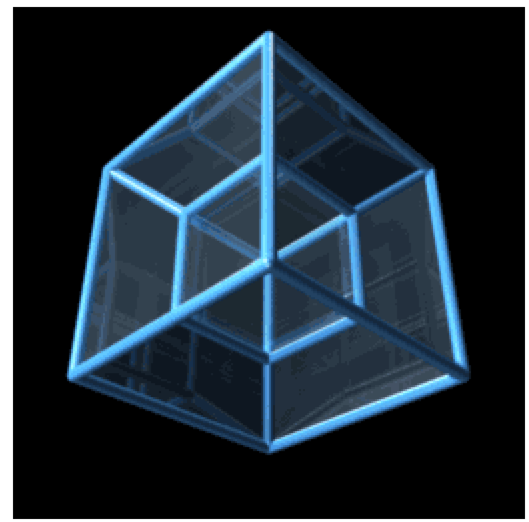
Fast Merging and Sorting on a Partitioned Optical Passive Stars Network

Amitava Datta and Subbiah Soundaralakshmi

Anders Ellvin and Tobias Pulls

Partitioned Optical Passive Stars (POPS) network

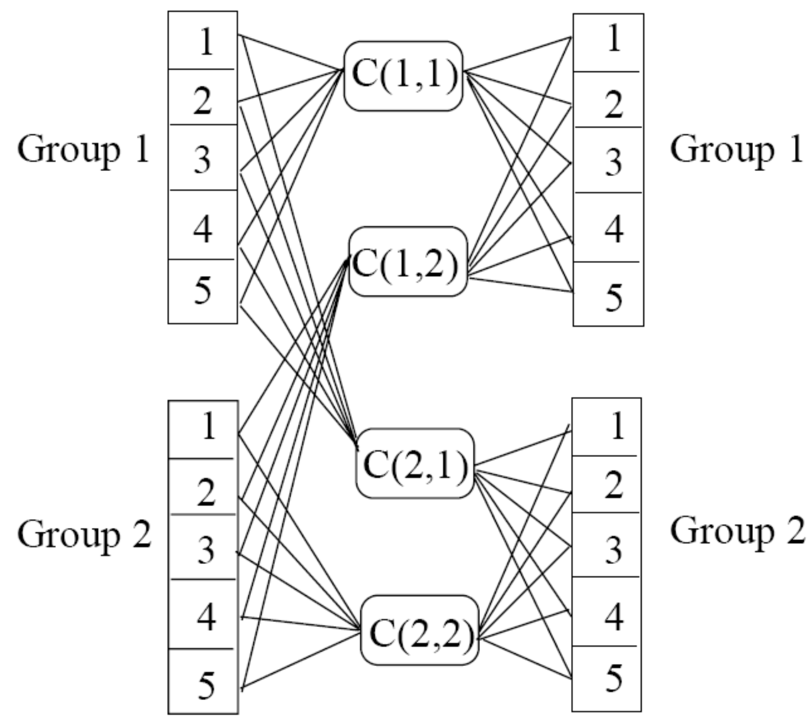
- An optical interconnection network for a multiprocessor system
- Uses multiple optical passive star (OPS) couplers
- Can simulate SIMD hypercubes



made by w:user:JasonHise
http://en.wikipedia.org/wiki/Hypercube

Terminology

- POPS(d, g)
  - d, number of processors in a group
  - g, number of groups
- g^2 couplers are needed
- For coupler c(i, j), j is the source processor group, i is the destination processor group
- A slot is the time for a coupler to send and receive data



POPS(5, 2) network

Theorems

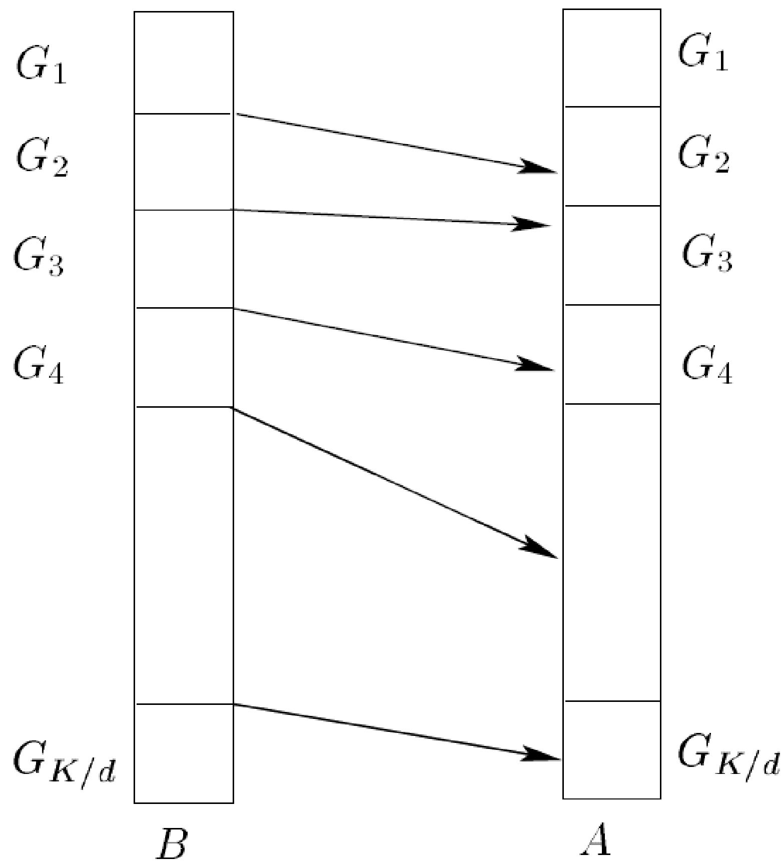
- Theorem 1. *An n processor POPS(d,g) can simulate every move of an n processor SIMD hypercube using one slot when d = 1 and using 2[d/g] slots when d > 1.*
- Bitonic sorting algorithm $O(2[d/g] \log^2 n)$ on a hypercube
- A fast algorithm for merging and sorting sequences

Merging Using a POPS network

- Two sorted sequences A (a_1, a_2, …, a_k) and B (b_1, b_2, …, b_k) of length K
- One element per processor, dg = 2K
- $\text{rank}(b_i : A \cup B) = \text{rank}(b_i : B) + \text{rank}(b_i : A)$
- Two phase binary search

First phase

- Starts with the last element in the middle group
- 2 subproblems
- Complexity: 2*(log g)
- Second phase – depends on g and d

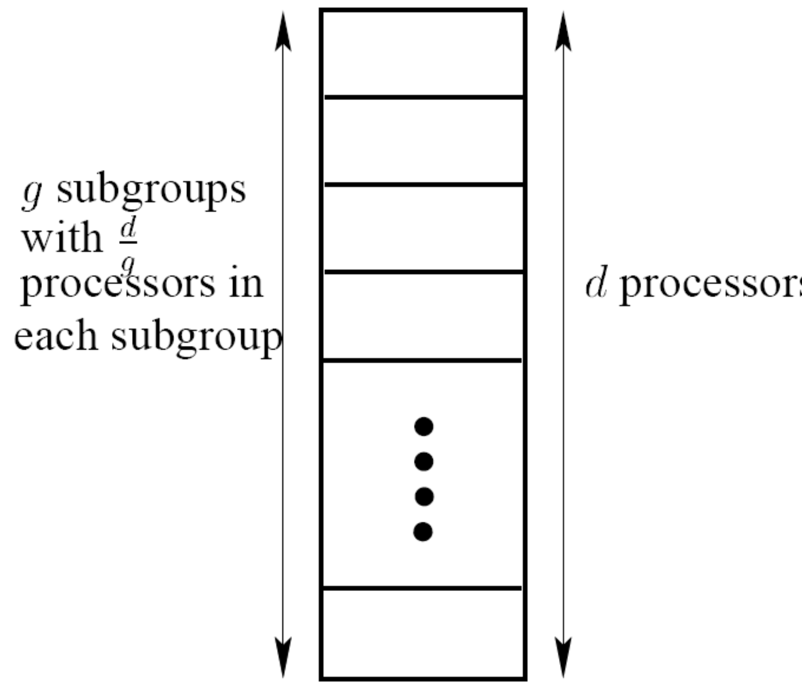


Second phase (g ≥ d)

- G_i, $1 \le i \le K/d$; $G_{i,j}$ $1 \le j \le d$
- We know rank($G_{k,d}$: A) & rank(G_{k+1},d : A)
- We start with $G_{k+1,d/2}$
- Broadcast boundaries to $G_{k+1,d/2}$ (2 slots)
- We get rank($G_{k+1,d/2}$: A) by another 2 slots
- 2 subproblems, which can be done simultaneously
- Complexity: 4*(log d)

Second phase (g < d)

- Previous approach doesn't work
- Divide each group into subgroups
- Rank the first element of each subgroup (2logg slots)
- Sequential ranking of the remaining elements (4d/g slots)



Conclusion for merge

- Lemma 1. Two sorted sequences of length K each can be merged into a single sorted sequence on a POPS(d,g), K = dg with the following complexities:
  - In 2logg + 8logd + 1 slots if g ≥ d
  - In 6logg + 9d/g slots if g < d

Sorting

- Based on the merge-sort algorithm
- n numbers to be sorted using a POPS(d,g) network where dg = n
- Couple restrictions
- Previous merge algorithm
- Sort elements in each group – depends on n and d

Case 1 ($d \leq \sqrt{n}$)

- We begin with $d = g = \sqrt{n}$
  - g^2 couplers, hence g couplers for each group and we have one coupler for every d

- G_i, $1 \leq i \leq g$; $G_{i,j}$ $1 \leq j \leq d$

- Two sequences, P and Q, with k elements each ($1 \leq 2k \leq d$)

Case 1 ($d \leq \sqrt{n}$) contd.

- rank(P : Q)
- Start: rank($P_{k/2}$: Q) where $G_{i,j}$ holds $P_{k/2}$
- Elements of Q in the processors $G_{i,r}$,.., $G_{i,s}$
- $G_{i,j}$ -> c(j, i) -> $G_{j,i}$ -> c(i, j) -> $G_{i,k}$, k = r…s
- We now have 2 subproblems! Takes 4logk slots

Case 1 ($d \leq \sqrt{n}$) conclusion

- Pairwise merging adjacent elements
  - 2 -> 4 ->8 etc ..

- Logd stages at each stage that requires 8logd + 1 slots

- $8\log^2 d + \log d$ slots

- Works for when $d < \sqrt{n}$ as well

Case 2 ($d > \sqrt{n}$)

- Less couplers connected to each group then processors

- Divide each group into g subgroups denoting them K_1, K_2, ..., K_g

- For each group:
  - Send out K_i to G_i
  - Gives us POPS(d/g, g) -> theorem 2

- 2d + 2d/g * logd slots

Case 2 ($d > \sqrt{n}$) - improvements

- We simulate the algorithm on each group without distributing the elements

- When sorting the elements for G_1:
  - Take the first subgroup from each group and simulate the hypercube algorithm
  - Place result in G_1 in sorted order

- $d + 2d/g * \log d$ slots

Case 2 ($d > \sqrt{n}$) – further improvements

- After we finish sorting a group we don't move the elements into G_1 but keep them in their original groups

- After g rounds of sorting:
  - Subgroup1 in every group is destined for G1, subgroup 2 in every group is destined for G2 etc
  - Every group has g couples so we transfer g elements in parallel from each group.

- $d/g + 2d/g * \log d$ slots

Sorting within groups – Conclusion

- Elements in each group in a POPS(d,g) can be sorted within the following complexities:
  - In $8\log^2 d + \log d$ slots if $g \geq d$
  - In $d/g + 2d/g * \log d$ slots if $g < d$

Sorting – conclusion

- First we sort the elements in each group
- Then we pairwise merge groups using the algorithm described earlier logd times
- Complexity is thus:
  - $8\log n + 8\log^2 d + 2\log^2 g + \log d + \log g$ slots if $g \geq \sqrt{n}$
  - $2d/g * \log n + 7d/g * \log g + d/g + 6\log^2 g$ slots if $g < \sqrt{n}$

Conclusion

- One algorithm for sorting and one for merging on a POPS network
- More efficient when compared to simulated hypercube algorithm when $d > g$