A Data Motion Algorithm

by

Allan Gottlieb
Clyde Kruskal †

Ultracomputer Research Laboratory
Courant Institute of Mathematical Sciences
715 Broadway, 10th Floor
New York, NY 10003

Ultracomputer Note #7
January, 1980

† This work was supported by DOE grant DE-AC02-76ER03077 and by NSF grant NSF-MCS79-21258.

ABSTRACT

We desire to permute N items $w_0 ..., w_{N-1}$, in an ultracomputer containing P processing elements (PEs), $PE_0, ... PE_{P-1}$. Under the assumption that N+P and that $w_i \varepsilon PE_i$, Schwartz gives the following worst case analyses: The static permutation algorithm requires 4 log P - 3 data communication steps. It is easily seen that for both algorithms the average case behavior closely approximates the worst case.

Here we present a data motion algorithm oriented toward average case rather than worst case performance, and supply an argument suggesting that the following average number of data communication steps required is approximately 3 log P.

1. Introduction

[UC] introduced the idea of an ultracomputer and reviewed algorithms for two permutation problems: The ''static permutation problem'', in which an algorithm is tailored to each specific permutation π (given in advance), and the ''dynamic permutation problem'', in which one algorithm must effect all permutations (i.e., the permutation is part of the data).

We desire to permute N items $w_0,...,w_{N-1}$, in an ultracomputer containing P processing elements (PEs), $PE_0,...,PE_{P-1}$. Under the assumption that N=P and that $w_i \varepsilon PE_i$, [UC] gives the following worst case analyses: The static permutation algorithm requires 4 log P - 3 data communication steps, and the dynamic permutation problem requires $\theta((\log P)^2)$ data communications steps. It is easily seen that for both algorithms the average case behavior closely approximates the worst case.

Here we present a data motion algorithm oriented toward average case rather than worst case performance, and supply an argument suggesting that the average number of data communication steps required is approximately 3 log P.

Note that [UCN11] considers ultracomputers containing L items in each PE (thus N = LP) and presents algorithms plus worst case analyses for both permutation problems defined above. The static permutation problem essentially scales up linearly, requiring 4L logP - L data communication steps; whereas the dynamic permutation algorithm improves in efficency requiring $\theta(L \log P)$ data communications steps (providing that L is large enough). Again the average case behavior closely approximates the worst case.

2. The Algorithm

Let us assume that N = P, $w_i \varepsilon PE_i$, and the permutation $i \rightarrow \pi(i)$ is available. Note that π may either be specified in advance or be part of the data. Our algorithm is derived from the packing algorithm presented in [UC]: The idea is for each w_i to make progress towards $PE_{\pi(i)}$ one bit at a time.

Each PE_i maintains a queue Q_i of ''data, destination, count'' triples. Let x be the triple currently at the front of Q_i, let d be the second (destination) component of x, and let k be the third (count) component. PE_i is assumed to be idle during any step in which Q_i is empty. If $i_D ... i_1$ is the binary representation of i,

we define unshuffle(i) = $i_1 i_D i_{D-1} \ldots i_2$, and partner (i) = $i_D \cdots i_2 \bar{i}_1$ where $\bar{i}_1 = 1\text{-}i_1$.

The algorithm proceeds as follows: Each PE_i initializes Q_i to (w_i, π (i), 1) and then iterates through the following 3 steps until, for all i, $w_i \in PE_{\pi(i)}$.

(1) If d_k, the k^{th} bit of d, and i have different parities, then send x to the rear of $Q_{\text{partner (i)}}$.

(2) If d = i, remove x (since the w contained therein is $w_{\pi^{-1}(i)}$).

(3) Set k = k+1 and send x to the rear of $Q_{\text{unshuffle(i)}}$.

3. A Crude Analysis of the Average Case

We make the fundamental assumptions that the permutation has been selected at random and that at each step the triples are randomly distributed among the processors. The latter assumption should be reasonable except near the beginning and end of the algorithm's execution.

Using the fundamental assumptions, it is easy to see that the average number of non-empty queues is

$$P\ (1-(1-1/P)^P\ ) \simeq P\ (1 - 1/e).$$

Since each of the P pairs must execute steps (1) and (3) at most log P times and at each iteration of steps (1) and (3), P(1-1/e) pairs are processed, it follows that (log P) / (1 - 1/e) $\simeq$ 1.58log P iterations are required. Thus the number of data communications steps is approximately 3log P.

4. Extensions and Remarks

We can use an analogous algorithm when there are L items contained in each PE. Then the expected number of non-empty queues is

$$P(1 - (1 - 1/P)^N) \simeq\ P(l - l/e)^L).$$

and the expected number of data communication steps $\simeq$ 2L logP / (l - $(l/e)^L$).

On average only half the PEs that execute step (1) actually send a pair to their partners. Thus, if we allow asynchronous operation and ignore the issue of data contention, the expected number of data communication steps $\simeq$ 1.5L logP / (1 - $(1/e)^L$).

The following table indicates that even for modest L both the original and asynchronous algorithms approach their asymptotic limits rapidly. Note that, since one can easily find a permutation that routes L$\sqrt{P}$ items through PE_0, the algorithms that we have proposed exhibit markedly different worst case and average case behavior.

L	Original	Asynchronous
1	3.16	2.37
2	2.31	1.73
3	2.10	1.58
4	2.04	1.53
5	2.01	1.51

Expected number of data communication steps divided by L logP.

5. Acknowledgements

We wish to thank Malcolm Harrison for suggesting the idea behind these algorithms and Jack Schwartz for reading a preliminary version of this note.

6. References

[UC] Jacob Schwartz, ‘‘Ultracomputers’’, to appear in *TOPLAS*.

[UCN11] Allan Gottlieb and Clyde Kruskal, *Supersaturated Ultracomputer Algorithms*, Ultracomputer Note #11, Comp. Sci. Dept., New York University, N.Y., 1980.

Table of Contents

1. Introduction 1
2. The Algorithm 1
3. A Crude Analysis of the Average Case 2
4. Extensions and Remarks 2
5. Acknowledgements 3
6. References 3