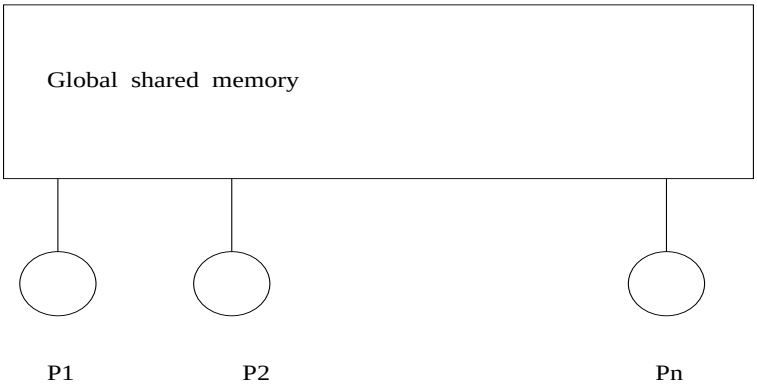
A Parallel Random Access Machine (PRAM) consists of

1. Multiple independent processors. Each processor has its own local control and local memory.
2. Shared global memory. Can be accessed by all the processors in each step. At each step each processor can perform the following steps
   (a) Read from or write to arbitrary location in global memory.
   (b) Perform some computation according to its local control and its local memory.



Several variations of the model have been defined based on whether or not multiple processors are allowed to access the same memory location in the same step.

- **EREW (Exclusive Read Exclusive Write)**
  All processors must access distinct memory locations at each step.
- **CREW (Concurrent Read Exclusive Write)**
  At each step several processors may read the same memory location. However, no two processors may write to the same location.

- **CRCW (Concurrent Read Concurrent Write)**
  Concurrent reading is allowed. Concurrent writing is also allowed; there are different variations of the model which determine how this can be done:
  - **Common**: Several processors may are allowed to write same location if they write same value.
  - **Priority** : The highest priority processor succeed in writing a memory location when more than one processors tries to write same memory location.
  - **Combining** : Value written by the processor is combined by using some commutative and associative operator.

Theorem 1 *Any step of N processor PRAM can be simulated in $O(N)$ steps on a uniprocessor.*

Proof: The uniprocessor keeps track of the programme counters of all PRAM processors. In $O(1)$ time the execution of a single instruction on any of the PRAM processors can be simulated, and hence in $O(N)$ time we can simulate the execution of all the N processors. ■

1 Simulation of PRAMS on Networks

Let m denote the size of the global memory of the PRAM and N the number of processors.

We can simulate such a PRAM on a network of processors. Each location of the PRAM is mapped to some location in the memory of some processor in the network model. Then reading or writing a memory location in PRAM is accomplished using packet routing. For example write "x" to y can be processed as send "x" to processor that is holding location y (upon receipt of this message the processor writes x in the required location in its memory).
Example: PRAM address x is mapped to (x mod N) memory location of processor (x div N). But in this case if i^{th} processor reads/writes memory $m + i$ then all the requests will be send to processor m and it will be a bottleneck.

Better Solution (Randomized): Use random hash function to decide where PRAM address x is to be stored. By this kind of mapping above mentioned problem can be solved. But a processor can be bottleneck if all the processors reads/writes same memory location. To overcome this, we can use idea similar to the random rank/ghost messages scheme for packet routing algorithm in level directed network.

In this case if all the processor chose rank of their packet (containing read/write request) such that, if processor i and j wants to read/write same location then, they chose same rank. Now intermediate processors can easily identify read/write request for same memory location, for read it can only send one request and it can also process multiple write request depending on the CW model.

Summary: Any variety of N processor PRAM can be simulated in $O(\log N)$ steps on an N node Butterfly, or $O(\sqrt{N})$ steps on an N node 2 dimensional mesh, and similarly for other networks.

2 Comparison of CREW and EREW

Consider a problem, Input: rooted pseudo forest, find the number of trees in the forest. The input is given as a list of pointers to parents of all the node in forest, in shared memory.

So any algorithm to solve this problem should read its parent's id from the list. In case, pseudo forest contains star, all the node will try to read common parent and so ER model cannot solve this problem efficiently.

CREW model can solve the problem in $\log h$ time by pointer doubling as in Wyllie's algorithm for list ranking, where h is the height of any of the trees. EREW PRAM will require $O(\log N)$ time.

3 Comparison of EW and CW

Consider a problem, Input: array of bits, compute OR of all the bits. The answer is to be computed in some location R in shared memory.

CW model can do this in $O(1)$ time as follows. First processor all write a 0 in location R. Then in parallel for all i, every processor i reads the ith input bit, and writes a 1 into R if the input bit read was a 1.

Clearly, after execution, R contains the desired result. Further, the common variety (weakest) CRCW PRAM is needed for this.

It is possible to prove that $O(1)$ time is not sufficient for solving this problem on the CREW model.

Exercises

1. Show that any step of a N processor CRCW PRAM can be simulated by an N processor EREW PRAM in $O(\log N)$ time. You may assume that an N processor EREW PRAM can sort N keys in $O(\log N)$ time.

2. Show how an N processor CRCW PRAM can compute the AND of N bits in $O(1)$ time.

3. Show how an N^2 processor CRCW PRAM can compute the maximum of N numbers in $O(1)$ time. (Hint: use the ideas from the previous problem.)

4. Show how an N processor CRCW PRAM can compute the maximum of N numbers in $O(\log \log N)$ time. (Hint: use the ideas from the previous problem.)