PRAM Models

In general, parallel computing machines are considered as PRAM (Parallel Random Access Machine).

PRAM machines have been classified into four models on the basis of their read/write capability. These models are distinguished in order to avoid the shared memory conflicts.

Benefits

It is suitable for the design of a parallel program targeted to a real machine

The base algorithm can help establish tight lower and upper complexity bounds for practical implementations

Assumptions made in PRAM model for ideal parallelism help architecture designers to develop innovative designs

It is suitable for modern day architectures, e.g., GPUs

The models are mentioned as below:

1. EREW(Exclusive Read, Exclusive Write)
2. ERCW(Exclusive Read, Concurrent Write)
3. CREW(Concurrent Read, Exclusive Write)
4. CRCW(Concurrent Read, Concurrent Write)

EREW (Exclusive Read, Exclusive Write)

This model works on traditional approach of execution where at a time only one instruction can get executed because either one processor can access the memory for read or write. So this approach is considered as similar as sequential approach of execution.

In this mode of computation no two processors are allowed to access similar shared memory cell for read/write at the same time.

For example:-

A binary search problem using this model may execute in following steps:-

a) Any value ‘k’ which is needed to be searched will be broadcasted by a processor p0 to other processers in Logp times.

b) Each processor will search ‘k’ sequentially in n/p times for n/p items.

c) Finally all the processor will set their flag value as True/False after checking their sub problem.

Hence the total time taken by the algorithm using this model will be “logn+n/p”.

ERCW (Exclusive Read, Concurrent Write)

This model allows the concurrent write operation at a time at similar memory location but exclusive read is also applicable. Due to this restriction multiple processors will write similar value at any location of memory. So practically this model was not implemented as it was considered as a useless model for parallel computing.

This model increases cost and time complexity of any parallel algorithm more than a sequential approach.

CREW (Concurrent Read, Exclusive Write)

It is a widely used and adapted model where concurrent read is allowed by multiple processors with restriction of exclusive write.

In this approach multiple processing elements are allowed to read a common memory location simultaneously with exclusive write.

For example:-

Solving binary search using this model will result the same as EREW because multiple processors can read *'k'* simultaneously in *O(1)* time. But the final reduction takes *O(log p)* time using p processors.

Hence the total time taken by the algorithm using this model will be "logp".

CRCW (Concurrent Read, Concurrent Write)

In this approach both simultaneous reads and both simultaneous writes of the same memory cell are allowed.

Concurrent Read has a clear semantics, whereas Concurrent Write has to be further controlled. There exist numerous basic models:

- PRIORITY CRCW: the processors are allocated fixed separate priorities and the processor with the highest priority is allowed to complete WRITE.
- ARBITRARY CRCW: one arbitrarilyselected processor is permitted to complete WRITE. The algorithm may make no expectations about which processor was selected.
- COMMON CRCW: all processors are permissible to comprehensive WRITE if all the values to be written are equal. Any algorithm for this model has to getguaranteed that this situation is fulfilled. If not, the algorithm is illegitimate and the device state will be indeterminate.

For example:-

Solving binary search using this model will result the same as EREW because multiple processors can read *'k'* simultaneously in *O(1)* time. But the final reduction takes *O(n/p)* time using p processors.

Hence the total time taken by the algorithm using this model will be "n/p".

Simulation of huge PRAMs on minor PRAMs

Minor PRAMs can pretend higher PRAMs. Which is relatively simple, there are two simulations which are very useful and disreputably used. The first result says that if we decrease the number of processors, the **cost** of a PRAM algorithm does not change, up to a multiplicative constant.

Lemma

Let *M*<*N*. Any problem which can be resolved on a *M*-processor PRAM in *t* steps can be solved on a *N*-processor PRAM in *TN=O(tM/N)* steps assuming the same size of shared memory.

Proof:

1. Partition *M* simulated processors into *N* groups of size *M/N* each.
2. Associate each of the *N* simulating processors with one of these groups.
3. Each of the simulating processors simulates one step of its group of processors by:
    1. executing all their READ and local computation substeps first,
    2. Executing their WRITE sub steps then.

Lemma

Assume *m'*<*m*. Any problem that can be solved on a *p*-processor and *m*-cell PRAM in *t* steps can be solved on a *max(p,m')*-processor *m'*-cell PRAM in *O(tm/m')* steps.

Proof:

1. Partition *m* simulated shared memory cells into *m'* continuous segments S_i of size *m/m'*.
2. Each simulating processor P'_i, *1<= i<= p*, will simulate processor P_i of the original PRAM.
3. Each simulating processor P'_i, *1<= i<= m'*, stores the initial contents of S_i into its local memory and will use *M'[i]* as an auxiliary memory cell for simulation of accesses to cells of S_i.
4. Simulation of one original READ operation:
   each P'_i, *i=1,...,max(p,m')* repeats for *k=1,...,m/m'*:
    1. write the value of the *k*-th cell of S_i into *M'[i]*, *i=1,...,m'*,
    2. read the value which the simulated processor P_i, *i=1,...,p*, would read in this simulated sub step, if it appeared in the shared memory.

5. The local computation substep of P_i, $i=1,...,p$, is simulated in one step by P'_i.

6. Simulation of one original WRITE operation is analogous to that of READ.