Design Document

Problem Statement

Exercise I

Consider the problem of computing the following operation in parallel:

A_1 OR A_2 OR … OR A_k

where each **A_i** is a task computing a Boolean value. Since only one result is required all tasks can be terminated as soon as one **A_j** results in TRUE for some **j** unless all tasks results in FALSE.

SPMD Template:

Scope:

In case of SPMD all Ai will be doing the same task. The data will be divided amongst all the Ai's equally.

To improve performance we introduce the provision of data skewing where in the user can specify whether or not the complexity of the sequential algorithm depends on input size of element or not i.e. data needs to be divided on the basis of uniform cost model or log model.

If data skewing is to be done, the data has to be sorted based upon the variable on which complexity depends.

Two example problems chosen by us that can be solved using the SPMD model are:-

Problem-1

Search for an element in an array.

Problem-2

Find a prime number in a range of numbers by testing each number for primality.

Four stages of Design Methodology:

Partitioning:
The main aim of partitioning step is to achieve fine granularity and to create as many independent tasks as possible.
In case of SPMD, algortihm performs a single task. Hence we have used data decomposition.

Eg. If there are n elements in the array or in the range then n independent tasks can be created where each task can handle an element.

Communication:

As each task is independent of other task only master-slave communication is required.
Master process will distribute the elements to the slave tasks and slave tasks after doing computation will send the result to master.

Once the master receives true value from any of the worker process, it sends an MPI_Abort() which kills all the worker processes.

Reason for using Master-Slave model: Once the master process has distributed data to all the worker tasks, had it itself performed computation as well on its own share of data, our program would have been inefficient because it would have to wait for its computation to get over and then receive the result from the other processes. So if any process would have completed its computation before the master and would have returned true, we wouldn't have utilised this to our benefit.

Agglomeration & Mapping:

In this phase, if there are P processors and N tasks then P groups can be made where each group will handle N/P elements.

Further if C threads are spawned at each processor then each thread will handle N/(P*C) elements.

Cons - Data skewing can't be handled if we divide the elements uniformly to all tasks for eg. in case of primality testing.

To handle the case of algorithms like primality testing, we propose that the data be divided in partition that decrease by a factor of 2, we solve for the equation for x:
2^(x-1) + 2^(x-2) + ... + 2^(x-P) = N
The solution to the above equation will give us the partitions required for the data.

Advantages of Aggloeration:
Communication is reduced.(important as overhead of communication is significant).
Additional computation per task is not very costly.
Tradeoff between Communication and Computation.

Disadvantages of Agglomeration:
Overhead of thread creation.
Every process needs to store the subarray containing the elements on which computation has to be done.

MPMD Template:

Scope:
The template will be able to handle the following cases:

-
- If there is any Ai whose part of computation is dependent on output of 1 or more Ai's and part is independent then after doing its computation it will have to wait for those Ai's to complete their computation and then receive result of computation from those Ai's.
-
- If any Ai is completely dependent on output of other Ai then those Ai's will be clubbed together and will need to be run sequentially.
-
- Ai is completely independent of other Ais.

Problem-1

Given an Abstract Syntax Tree and a Symbol Table, check whether any semantic rule is violated.

Problem-2

Applying N Firewall Rules on stream of incoming packets in a network and discard it if any rule is violated. Some rules maybe dependent on one another.

Four stages of Design Methodology:

Partitioning:

It is to be done in a way such that it exposes the maximum concurrency possible so in case of MPMD as there are multiple tasks to perform first functional decomposition is done. Then in each task more tasks will be created among which data will be decomposed.

Communication:

Master will communicate with each task and will give it a function pointer and the data that need to be processed.
A task whose computation depends on the output of another task will have to communicate with master that it requires computation of which function and the data on which it needs to be done. Master will then schedule it on a processor and will pass that result to the required process which needed the result.

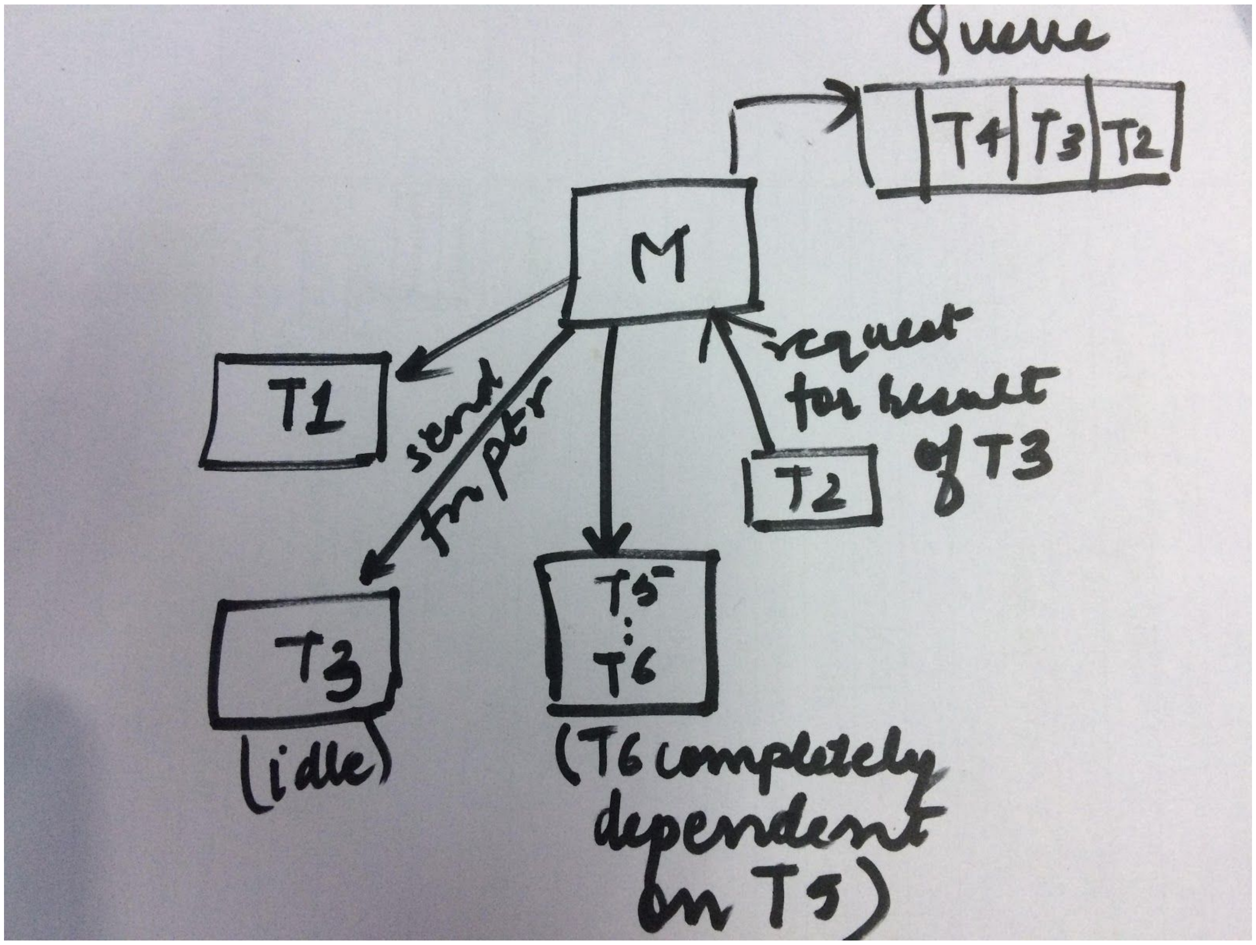
Agglomeration Mapping and Scheduling:

If a task is completely dependent on output of other task then group such tasks together in a single task and assign them to a processor. If number of such tasks exceed the number of processors then use dynamic scheduling in which master will act as a scheduler and when it will recieve a message that a processor is idle it will schedule a task from the queue.

If the number of processors is greater than the number of tasks, then divide the data of each task in ceil(P/t) parts where each part along with a function pointer is assigned to a processor and if any partition is left then put it in the queue.

If a task needs output of computation done by another task then it can tell the master that it wants a particular task to be done on a set of data. The master stores the result returned by a process so that in case other process needs that value it can send it directly. Otherwise the master will schedule it on a processor if their is a idle processor else it will put the task in queue.

If these computations need to be performed on a large set of data where on each data unit we want the same processing then in each function further threads can be spawned which will perform the same operation on different data.



Each box represents a processor.