Chapter 3

Parallel Merge Sort

3.1 Objectives:

At the end of this lecture the learner will be able to:

- Apply Parallel Merge Sort algorithm to sort a given list.

3.2 Parallel Merge Sort Algorithm

Procedure parallelMergeSort
begin
Create processors P_i where i = 1 to n
if i $>$ 0 then recieve size and parent from the root
recieve the list, size and parent from the root
endif
midvalue= listsize/2
if both children is present in the tree then
send midvalue, first child
send listsize-mid,second child
send list, midvalue, first child
send list from midvalue, listsize-midvalue, second child
call mergelist(list,0,midvalue,list, midvalue+1,listsize,temp,0,listsize)
store temp in another array list2
else
call parallelMergeSort(list,0,listsize)
endif
if i$>$0 then
send list, listsize,parent
endif
end

3.3 Example

In the figure given below, the given problem is divided into sub problems of equal size approximately. If each of the sub problems can be solved independently, then the speed of execution can be increased significantly using parallel computing. Here each node of the tree represents a processor or process. The time taken by the sequential merge sort algorithm is O(nlogn). The number of steps taken by the corresponding parallel version is 2 logn steps as shown below. Each step of the execution need to perform more than one operation based on the cardinality of the list.

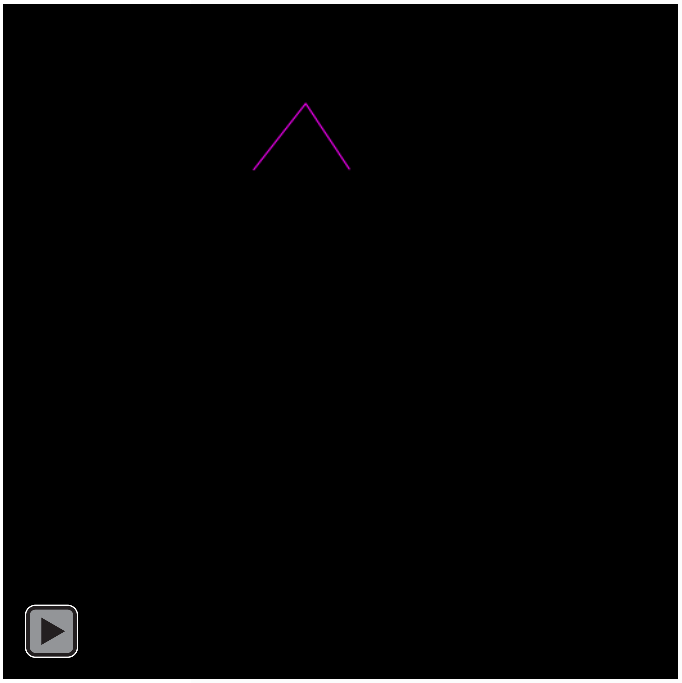

Figure 3.1: Parallel Merge Sort Example

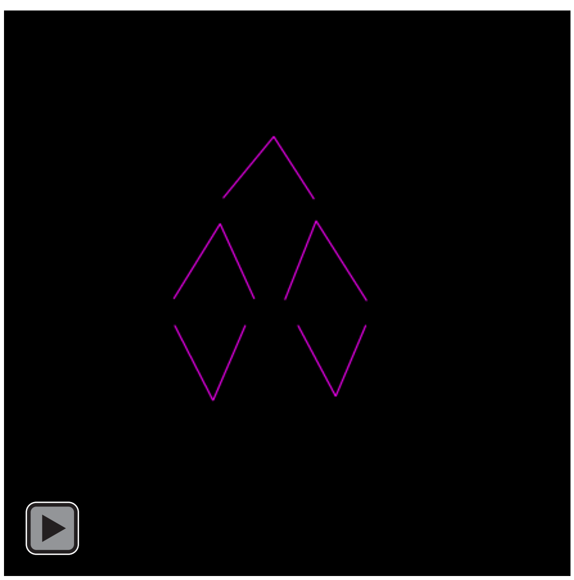

Figure 3.1(a): Processor Allocation during Partition and Merging

3.4 Analysis

Let us assume that the sublists are communicated to the respective processors and the merging takes place automatically.

3.4.1 Communication Complexity

During the Partition phase, in the first step n/2 data is sent to processor P_2. In the next step n/4 data is sent from P_0 to P_1 and from P_2 to P_3. The total number of steps is log p, where p is the number of processors.
During Merging phase, n/4 and n/2 data are sent during the two steps. Again the total number of steps is log p, where p is the number of processors.
Therefore the time taken for communication is $\text{Time}_{Communication}$ = 2($\text{Time}_{Startup}$ +(n/2)Time_{data} + $\text{Time}_{Startup}$ + (n/4)Time_{data} +...)
$\text{Time}_{Communication}$=2(log p)$\text{T}_{Startup}$ + 2n Time_{data}

3.4.2 Computation Complexity

Computation takes place only in the merging phase. During the first step only one computation step. That is P_0 and P_2. During the next step there are three computation steps. That is P_0.
Therefore $\text{Time}_{Computation}$ = $\sum_{i=0}^{logp}(2^i - 1)$