

CSE 457: Parallel Sorting Worksheet -II

Assume that $P < N$ and P is a power of 2, i.e., $P = 2^k$ for some integer k . Assume that each processor starts out with a `mylist` of size $m = N/P$; the numbers in the original list of size N are divided arbitrarily into these P lists of size $m = N/P$. Let `pqsort?(startprocessor, endprocessor)` refer to a sorting function to sort in parallel the global list comprising all local lists (`mylist`) at processors with ranks `startprocessor` to `endprocessor`. Let `quicksort(mylist)` be the standard serial Quicksort function which returns `mylist` in sorted order; on a list of size m , this function costs $m \log_2 m$.

In the last class we discussed `pqsort1` which has an unacceptably high computation cost. Now consider converting it to a more efficient parallel sorting scheme.

1. Let `split(mylist, pivot, leftlist, rightlist)` be a function that partitions elements in `mylist` so that `leftlist` contains all numbers smaller than or equal to `pivot` and `rightlist` contains the remaining numbers.

Let `merge(mylist, list1, list2)` be a function that merges the elements in `list1` and `list2` such that `mylist` contains all numbers in sorted order.

Can you use these two functions to change `pqsort1` so that its computation costs are decreased significantly (without increasing communication costs)? Provide your algorithm by editing `pqsort1` appropriately (we will call it `pqsort2`).

```
quicksort(mylist)
pqsort2(0, P-1)
.....
pqsort2(startprocessor, endprocessor)

processorgroupsize = endprocessor - startprocessor + 1
partnerdistance = processorgroupsize/2
midprocessor = startprocessor+partnerdistance

if ( partnerdistance >= 1)
  if (myprocessorid = startprocessor)
    pick pivot as the median of my sorted list mylist
  end if
  broadcast pivot from startprocessor to processors in the range
  startprocessor to endprocessor
  split(mylist, pivot, leftlist, rightlist)
  if (myprocessorid >= midprocessor)
    mypartner = myprocessorid - partnerdistance
    sendreceive exchange: send my leftlist and receive partner's rightlist
    to replace my leftlist
    merge(mylist, leftlist, rightlist)
    pqsort2(midprocessor, endprocessor)
  else
    mypartner = myprocessorid + partnerdistance
    sendreceive exchange: send my rightlist and receive partner's leftlist
    to replace my rightlist
    merge(mylist, leftlist, rightlist)
    pqsort2(startprocessor, midprocessor-1)
```

```
        end if
    end if
end pqsort1
```

Typically, the median of a randomly sampled subset of numbers in a list is very close to the median of the list. For purposes of easy analysis assume that the pivot selection using the median of the local list is such that it perfectly splits each local list into equal sized left and right halves.

- a) Using P processors, how many times is the function `pqsort2` invoked at a processor? Provide expressions.
- b) What is the size of `mylist` at each invocation at a processor?
- c) What is the total computation cost at a processor? Provide expressions.
- d) Consider communication costs using t_s for the startup and t_w for the per word cost. Provide an expression for the communication cost at a processor during a single call to `pqsort` using p processors.
- e) Provide an expression for the total communication costs at a processor over all calls to `pqsort2` starting with P processors. Show your work. Simplify to show the higher order term in the cost; you may assume that $m/2 \geq \log_2(P)$.
- f) Provide an expression for T_P the cost (time) on P processors for the parallel algorithm. Which costs, computation or communication, dominate for `pqsort1` and `pqsort2`?