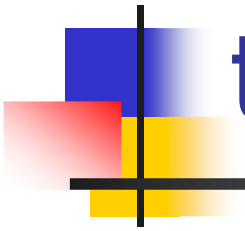
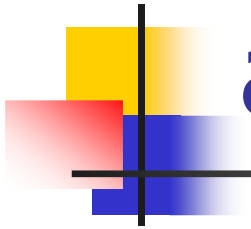
CSE 633 2010 Fall semester

Author: Zhihong wei
Email: zhihongwei@buffalo.edu
Professor: Russ Miller
12/09/2010



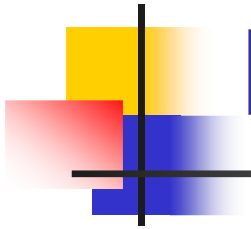
topic

Study of parallel sorting algorithm



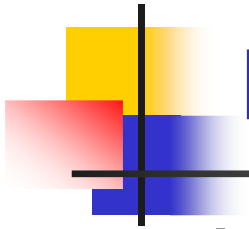
abstract

- Sorting data is a basic, widely used function
- Hard to improve performance by modify sequential sorting algorithm
- Using parallel method to shorten running time is a good choice



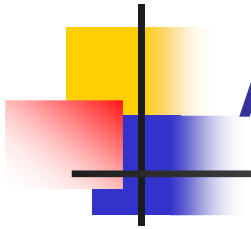
Review project plan

- Step 1: make a basic parallel quickSort programme by using openMP and C++, compare parallel quickSort with sequential quickSort
- Step 2: improve programme to achieve better performance
- Step 3: make a parallel BitonicSort programme, compare with quickSort



My Medium-grained quickSort

- A little different from Hypercube medium-grained quickSort
  1. store all data in a array at first
  2. break this array into 2 parts, low part and high part.
  3. repeat step 2 until we have N arrays and data store in array 0 < array 1<…<array N
     N=node number
  4. each node load the a array and sort it independently



Analyses of quickSort programme

- Data size and processors number

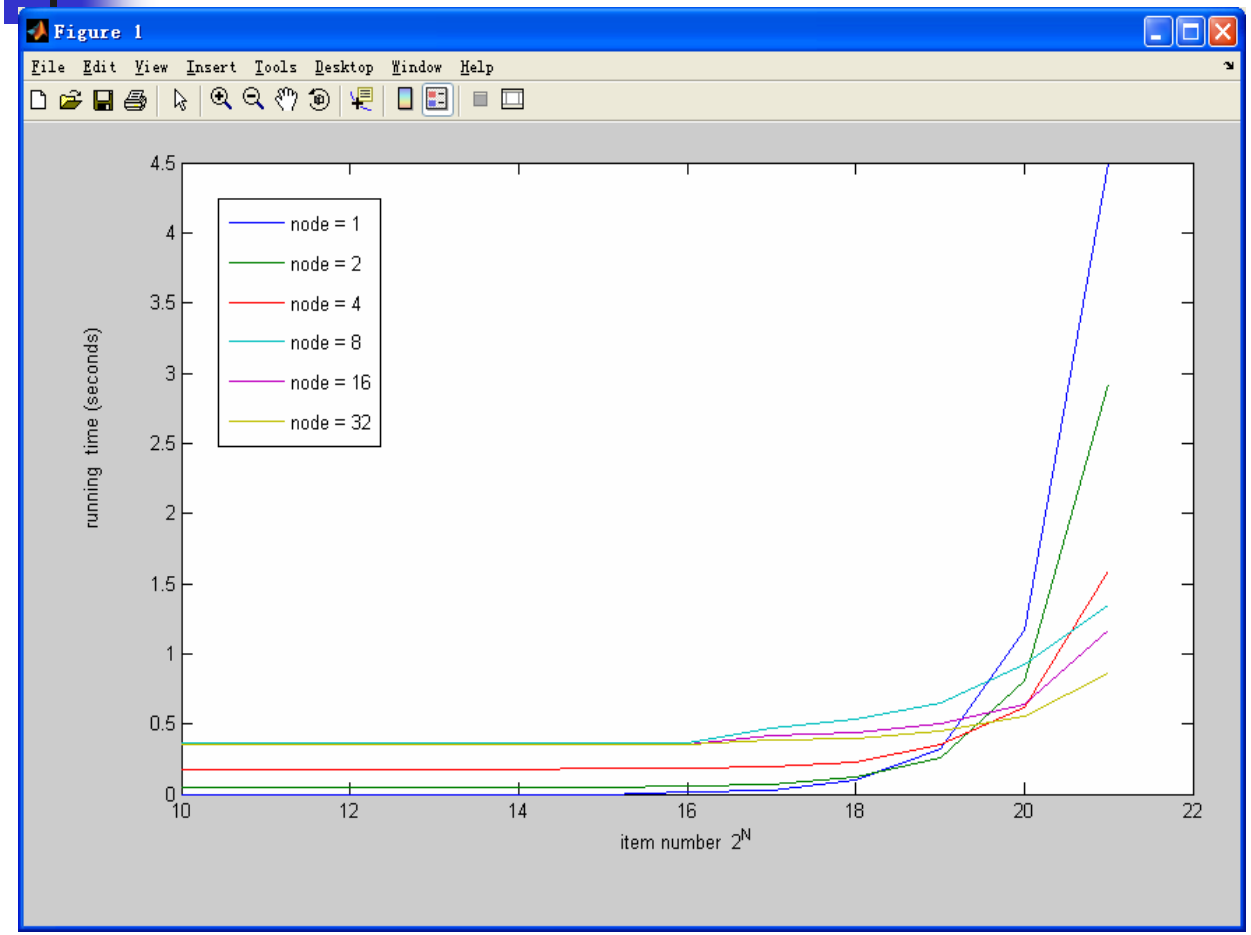
  data size: $2^{10} \sim 2^{21}$

  processor number: 1 ~ 32

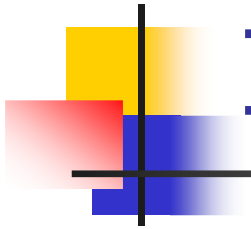
- Running time depend on data

  worst case running time is much larger than expected running time

Analyses of quickSort programme



Running time to sorting random data from 2^{10} to 2^{21} by 1 to 32 processors



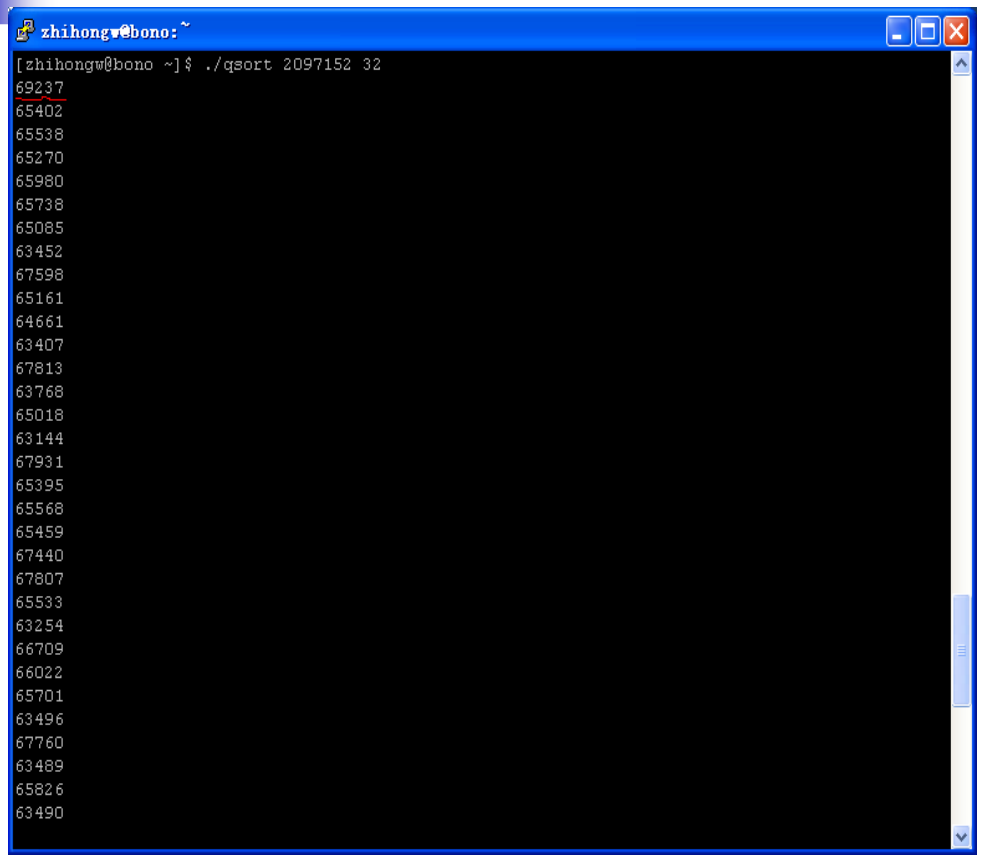
Improvement

- To avoid w-c running time, we can use a simple method to assign data randomly to every nodes

assign the (i*N)th data to node 0
assign the (i*N +1)th data to node 1
.
.

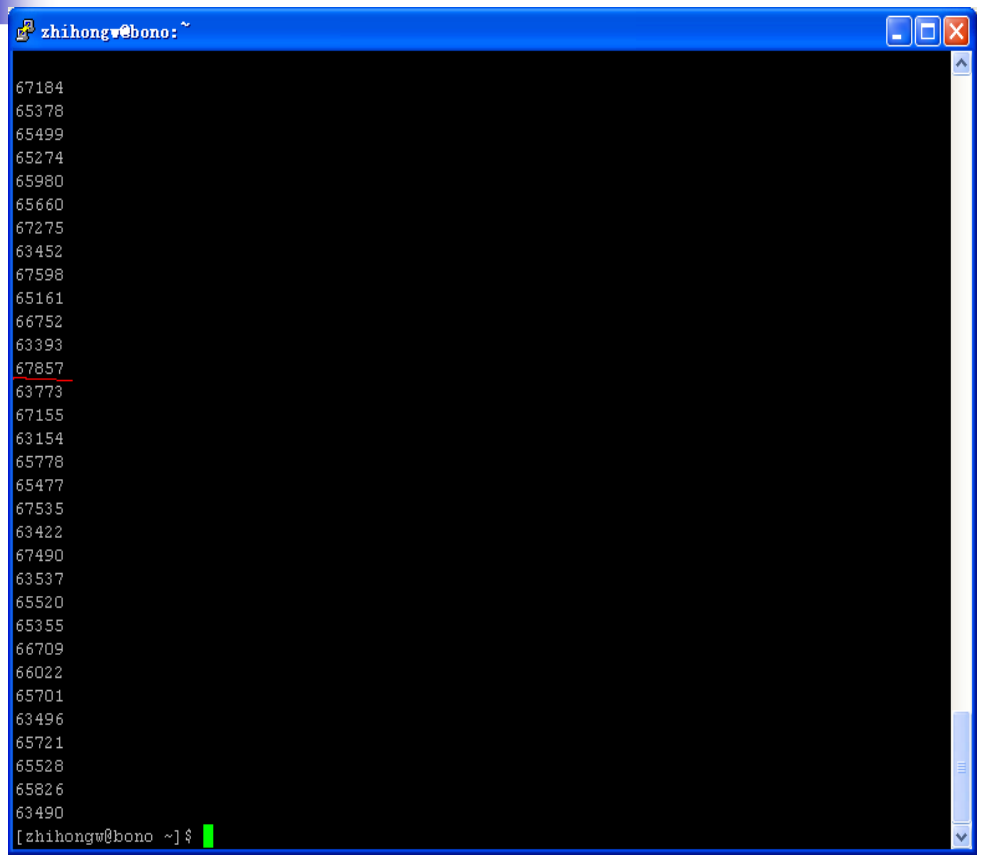
assign the (i*N +N-1)th data to node N-1

Improvement



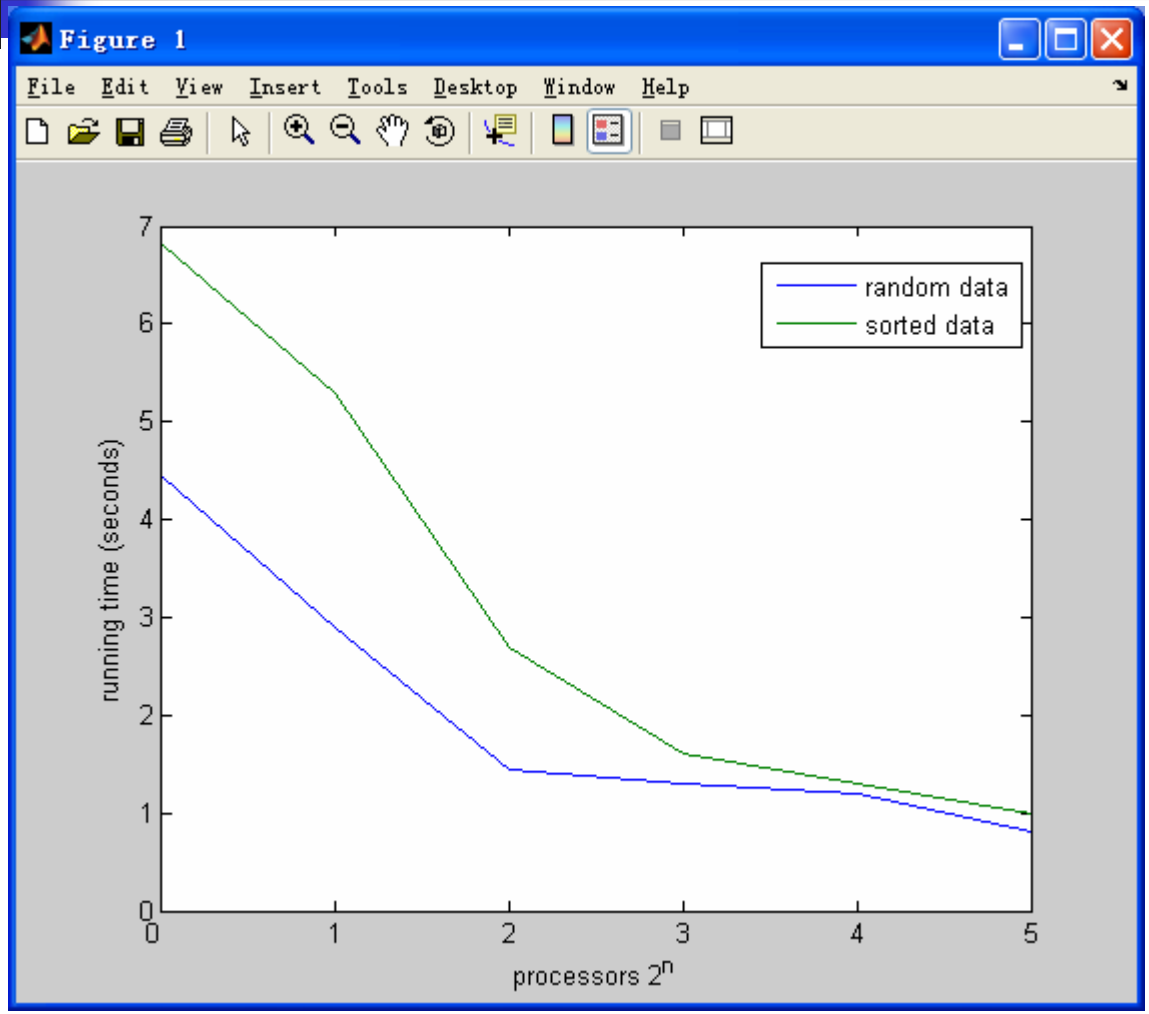
When sorting a random data, after medium-grained quickSort step, every node was assigned about 67000 items, one node has 69237 items, which is larger than any other nodes

Improvement

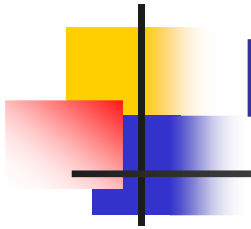


When sorting a sorted data, after medium-grained quickSort step, every node was assigned about 65000 items, one node has 67857 items, which is larger than any other nodes. The worse case did not appear and data were divided more evenly

Improvement

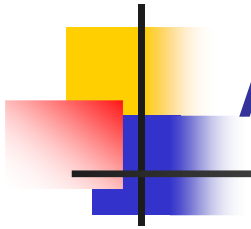


Running time of sorting random data and sorted data



Parallel BitonicSort

- Distribute data items evenly to all nodes
- Every nodes sort data by bitonicSort
- Merge data

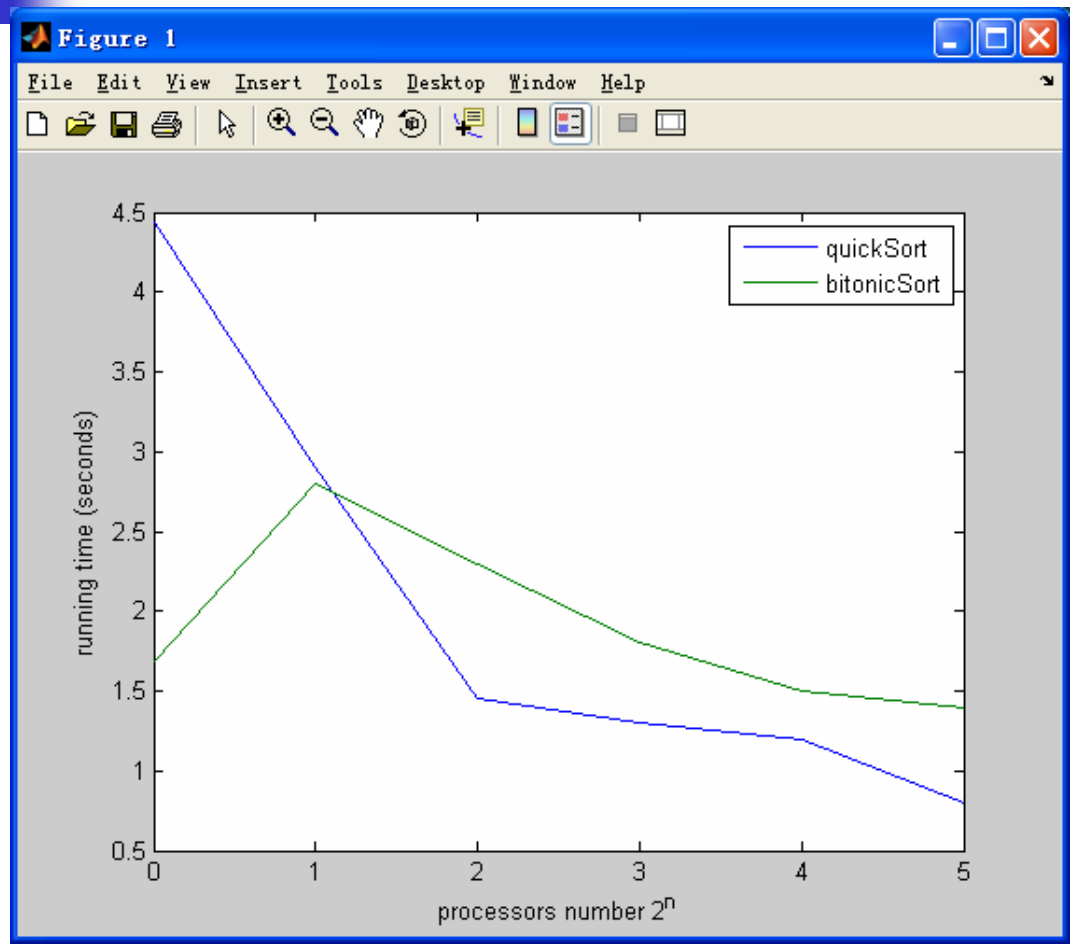


Analyses of Parallel BitonicSort

- Data size and processors number

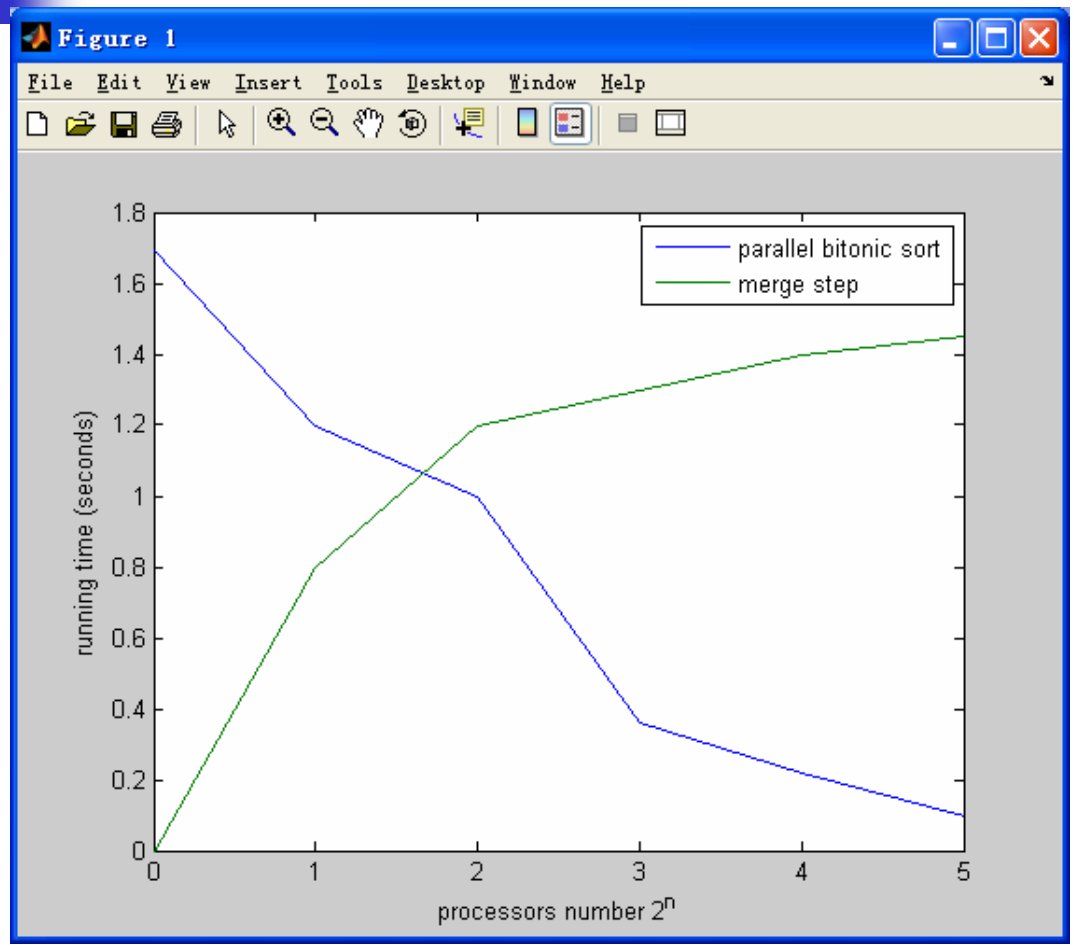
  As same as quickSort

Analyses of Parallel BitonicSort



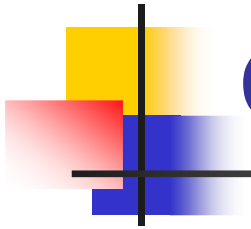
Comparing quickSort and bitonicSort when sorting 2^21 items, with the increasing of processors number, quickSort become more efficient.

Analyses of Parallel BitonicSort



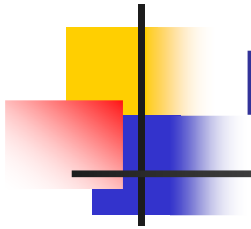
Comparing running time of parallel bitonicSort step and merge step,

with the increasing of processors number, merge step running time approach to a constant, which dominate the whole running time



disadvantage

- Data set size still too small
- Inefficient of BitonicSort merge step



reference

- R.Miller, L.Boxer “Algorithms sequential and parallel ” second edition

Thanks !