Parallel Bitonic Sort

David Jegan Abishek
Instructor - Dr. Russ Miller
CSE 702 Programming Massively Parallel Systems
November 21

University at Buffalo The State University of New York

Agenda

- Introduction to Bitonic Sort
- Example Comparison
- Results and Analysis
- Future Work

What is Bitonic Sort?

- Bitonic Sequence:

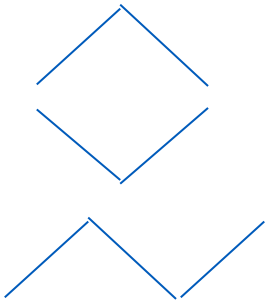
  How to make a given sequence Bitonic?

  A sequence a = (a1, a2, . . ., ap) of p numbers is said to be bitonic if and only if

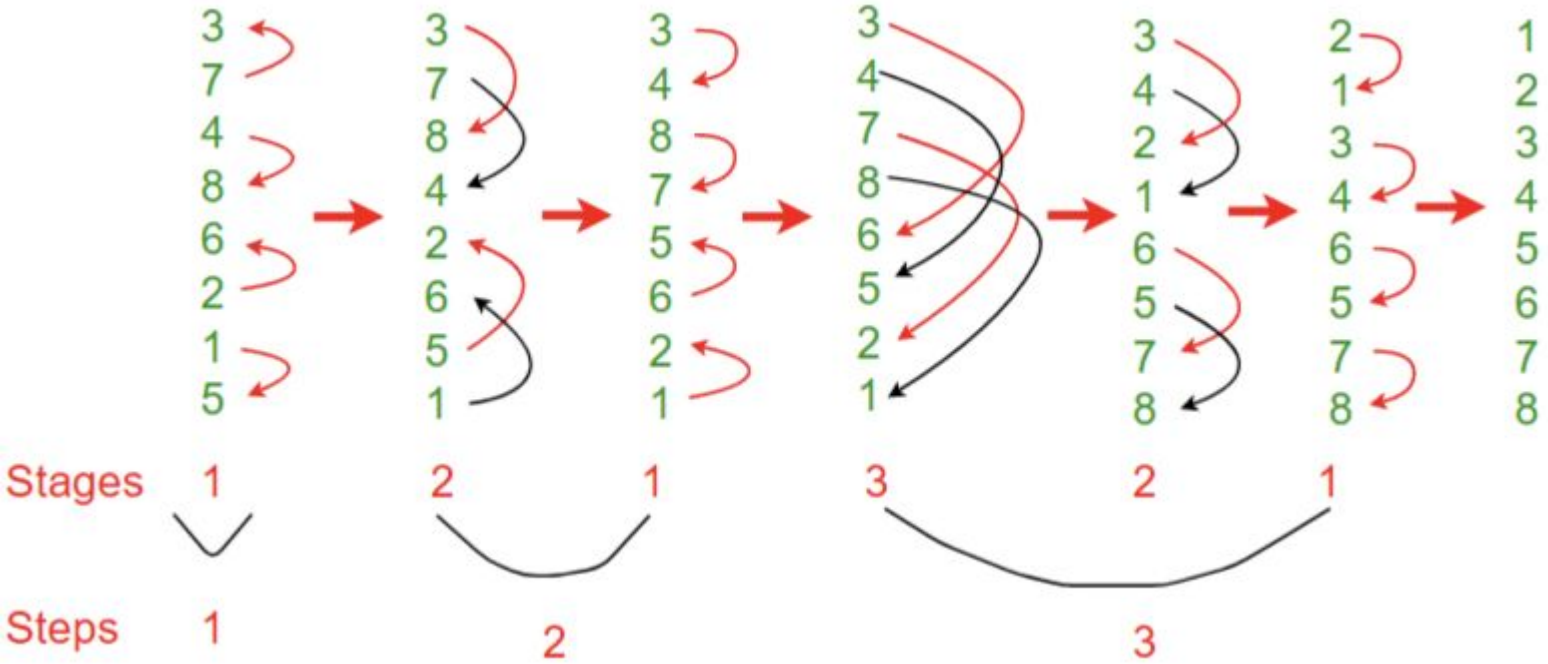
  • a1 ≤ a2 ≤ … ≤ ak ≥ … ≥ ap, for some k,1<k < p, or

  • a1 ≥ a2 ≥ … ≥ ak ≤ … ≤ ap, for some k,1<k < p, or

  • ‘a’ can be split into two parts that can be interchanged to give either of the first two cases.



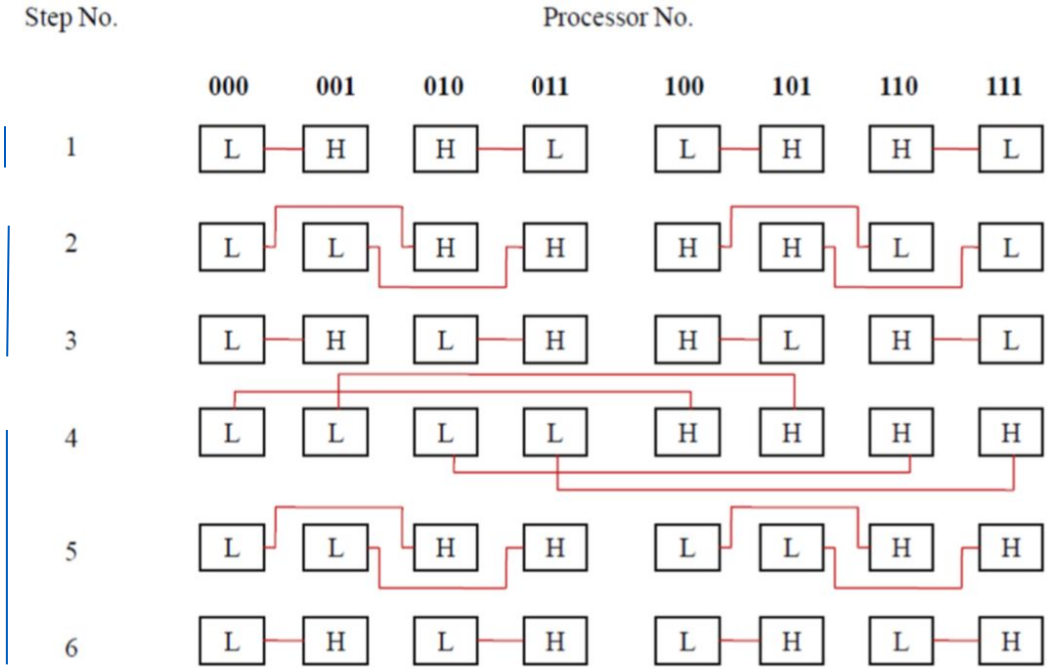
How to make a sequence Bitonic?



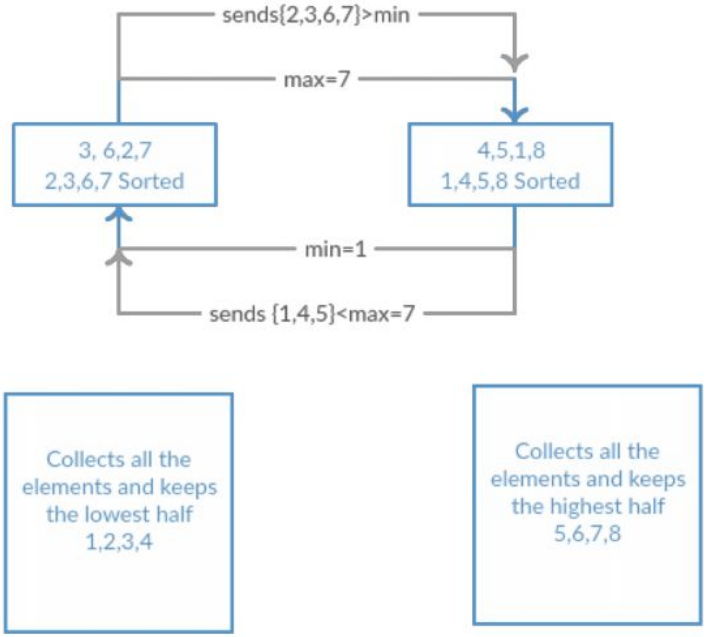
Bitonic Sorting

- To sort an unordered sequence, sequences are merged into larger bitonic sequences, starting with pairs of adjacent numbers.
- By a *compare-and-exchange* operation, pairs of adjacent numbers formed into increasing sequences and decreasing sequences.
- Pairs form a bitonic sequence of twice the size of each original sequences.
- By repeating this process, bitonic sequences of larger and larger lengths obtained.
- In the final step, a single bitonic sequence sorted into a single increasing sequence.

Bitonic Sorting



What happens in a comparison?



Algorithm

- Input: Number of processors, Data length
- Find the ranks of each processor
- Generate data in each processor using randomize function
- Sort the lists generated in the processor
- Compare and exchange data with a neighbor
- The above steps use comparison functions to compare and exchange

Runtime

When (P=n)

$$T_{par}^{bitonic} = \sum_{i=1}^{i=\log n} i = \frac{\log n(\log n + 1)}{2} = O(\log^2 n)$$

When (P << n)

$$T_{par}^{bitonic} = \frac{N}{P}(\log N + \log^2 P)$$

Running it!

```
#!/bin/sh
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=1
#SBATCH --constraint=IB
#SBATCH --partition=general-compute --qos=general-compute
#SBATCH --time=12:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=davidjeg@buffalo.edu
#SBATCH --output=bitonic_sort.out
#SBATCH --job-name=bitonic_sort

module load intel/14.0
module load intel-mpi/4.1.3
module list
mpicc -lm -o bitonic_sort bitonic_sort.c
ulimit -s unlimited

export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

srun bitonic_sort 3355

#
echo "All done!"
```

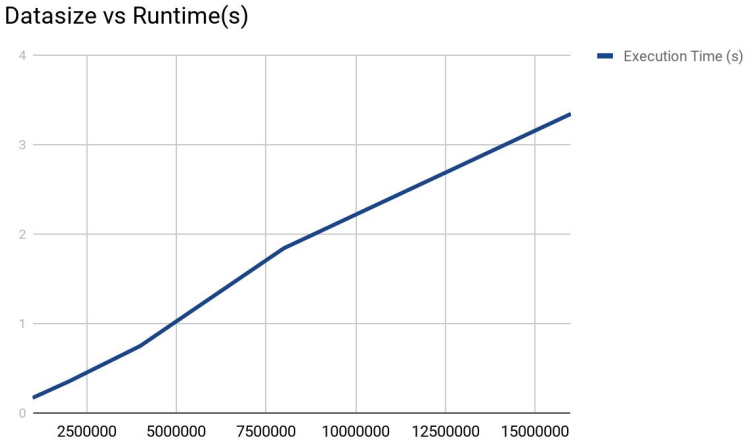
```
Currently Loaded Modules:
  1) intel/14.0   2) intel-mpi/4.1.3


Number of Processes spawned: 2
Time Elapsed (Sec): 0.169214
All done!
```

Results

Keeping the number of processors constant and increasing the datasize.

Number of processors = 2

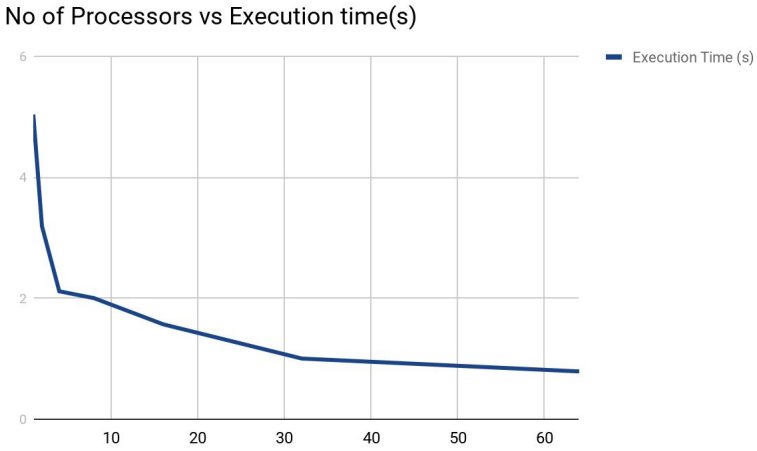


Data size	*Execution Time (s)*
1,000,000	0.169214
2,000,000	0.350628
4,000,000	0.749664
8,000,000	1.841554
16,000,000	3.344101

Results

Keeping the data size constant and increasing the number of processors.

Data size = 16 Million (16,000,000)

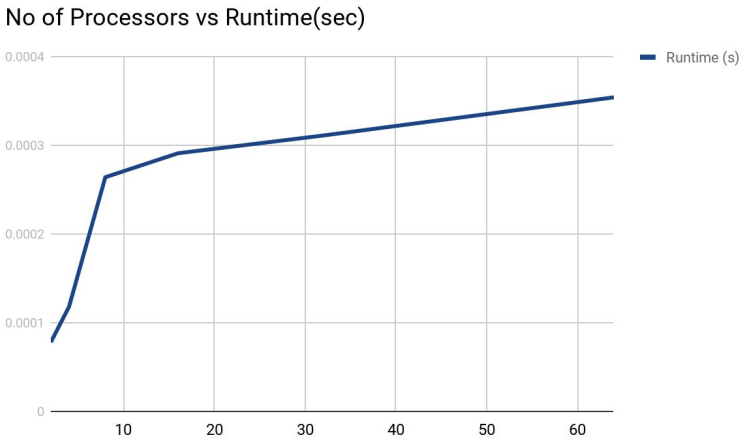


No of Processors	*Execution Time (s)*
1	5.038071
2	3.191084
4	2.115728
8	2.002834
16	1.569341
32	1.003489
64	0.790844

Results

Keeping the number of processors equal to the data and analyzing the execution time.

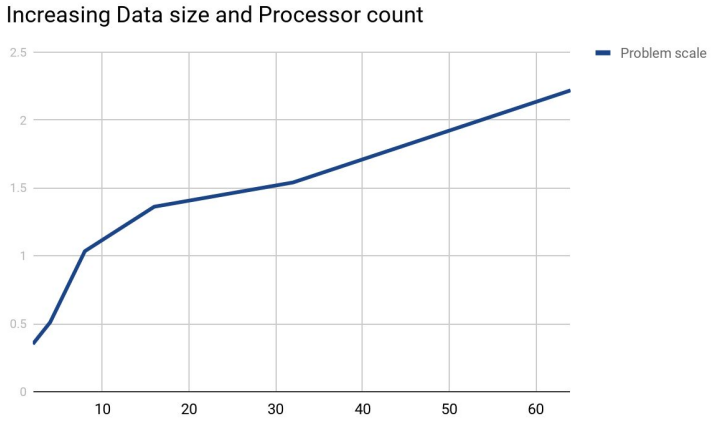
Data size = Number of processors



No of Processors	*Execution Time (s)*
2	0.000078
4	0.000118
8	0.000274
16	0.000291
32	0.000311
64	0.000354

Results

Keeping the data per processor constant as number of processors increase and analyzing the execution time.



No of Processors	*Data size*	*Execution Time (s)*
2	2000000	0.350628
4	4000000	0.510892
8	8000000	1.034621
16	16000000	1.362129
32	32000000	1.540692
64	64000000	2.218754

Future Work

- Compare MPI (distributed memory models) with OpenMP (shared memory models)
- Compare with other sort algorithms

References

- Algorithms Sequential and Parallel: A Unified Approach by Russ Miller and Laurence Boxer
- http://en.wikipedia.org/wiki/Bitonic_sorter
- CCR: Resources and Tutorial Materials
- http://www.cs.rutgers.edu/~venugopa/parallel_summer2012/bitonic_overview.html

Thank you!