Parallel Bitonic Sort Implementation

CSE 702 Programming Massively Parallel Systems

Course Instructor – Dr. Russ Miller

Prepared by – Sajid Khan(UB person no: 50248743)

Agenda

- Introduction to Bitonic Sort
- MPI Implementation
- Example Comparison
- Results and Analysis
- Challenges
- Outcome
- References

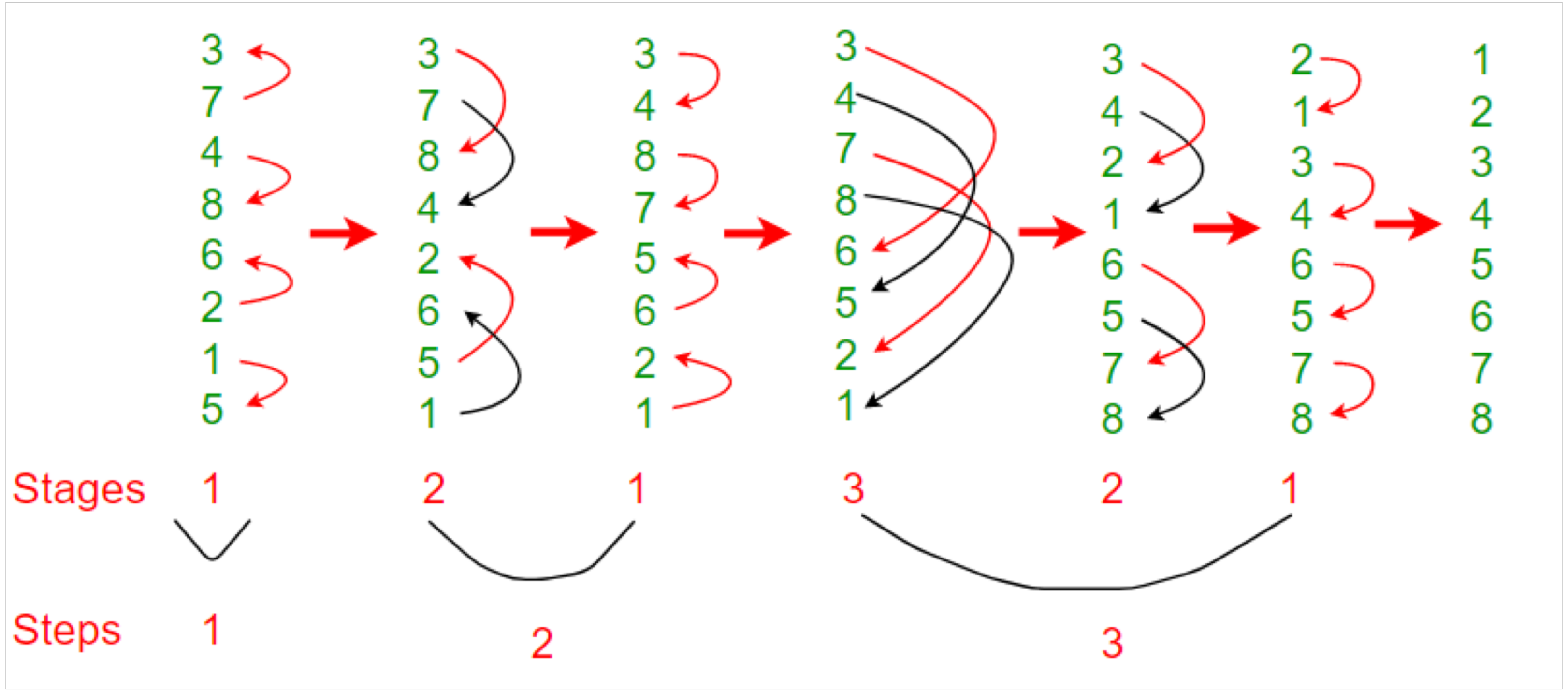
What is Bitonic Sort?

- To understand Bitonic Sort, we must first understand what is Bitonic Sequence and how to make a given sequence Bitonic.
- A sequence is called Bitonic if it is first increasing, then decreasing. In other words, an array arr[0..n-i] is Bitonic if there exists an index i where 0<=i<=n-1 such that

  x0 <= x1 …..<= xi and xi >= xi+1….. >= xn-1

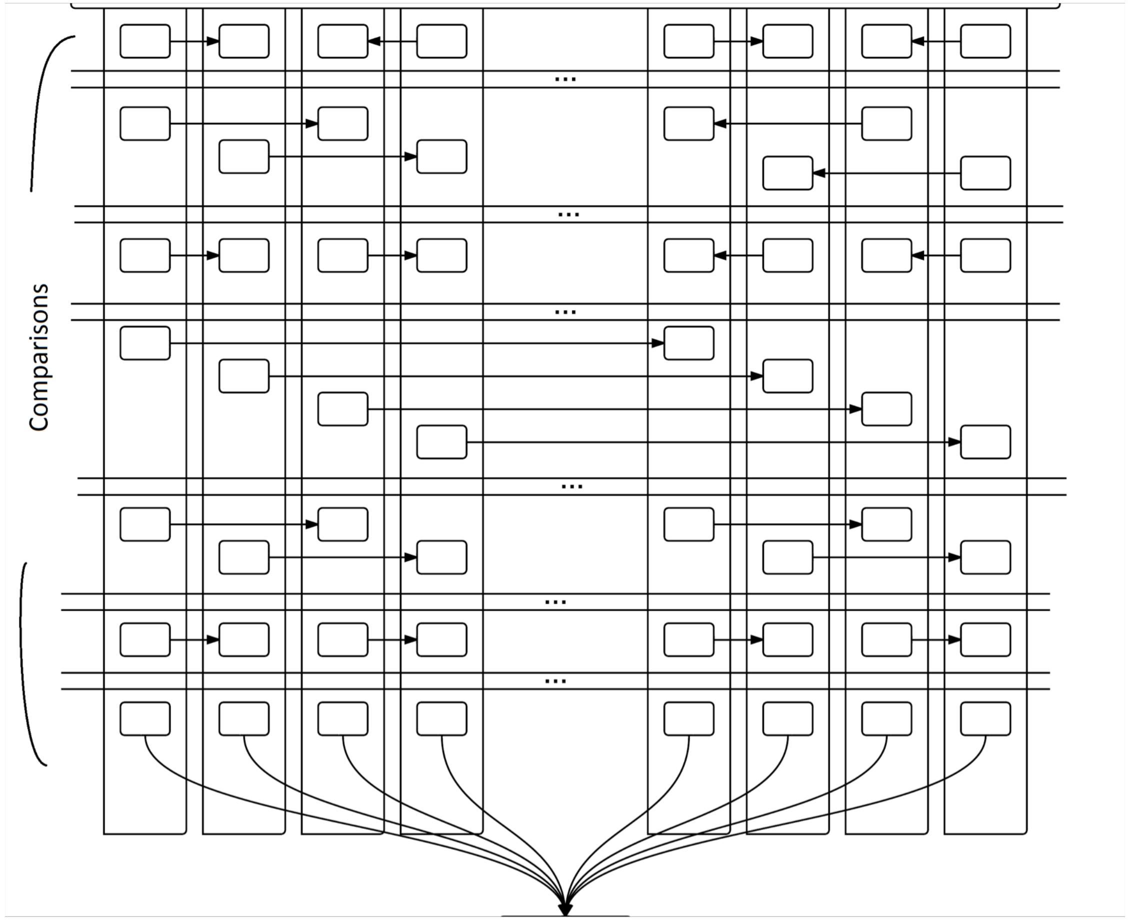
- A sequence, sorted in increasing order is considered Bitonic with the decreasing part as empty.

How to make a sequence Bitonic?

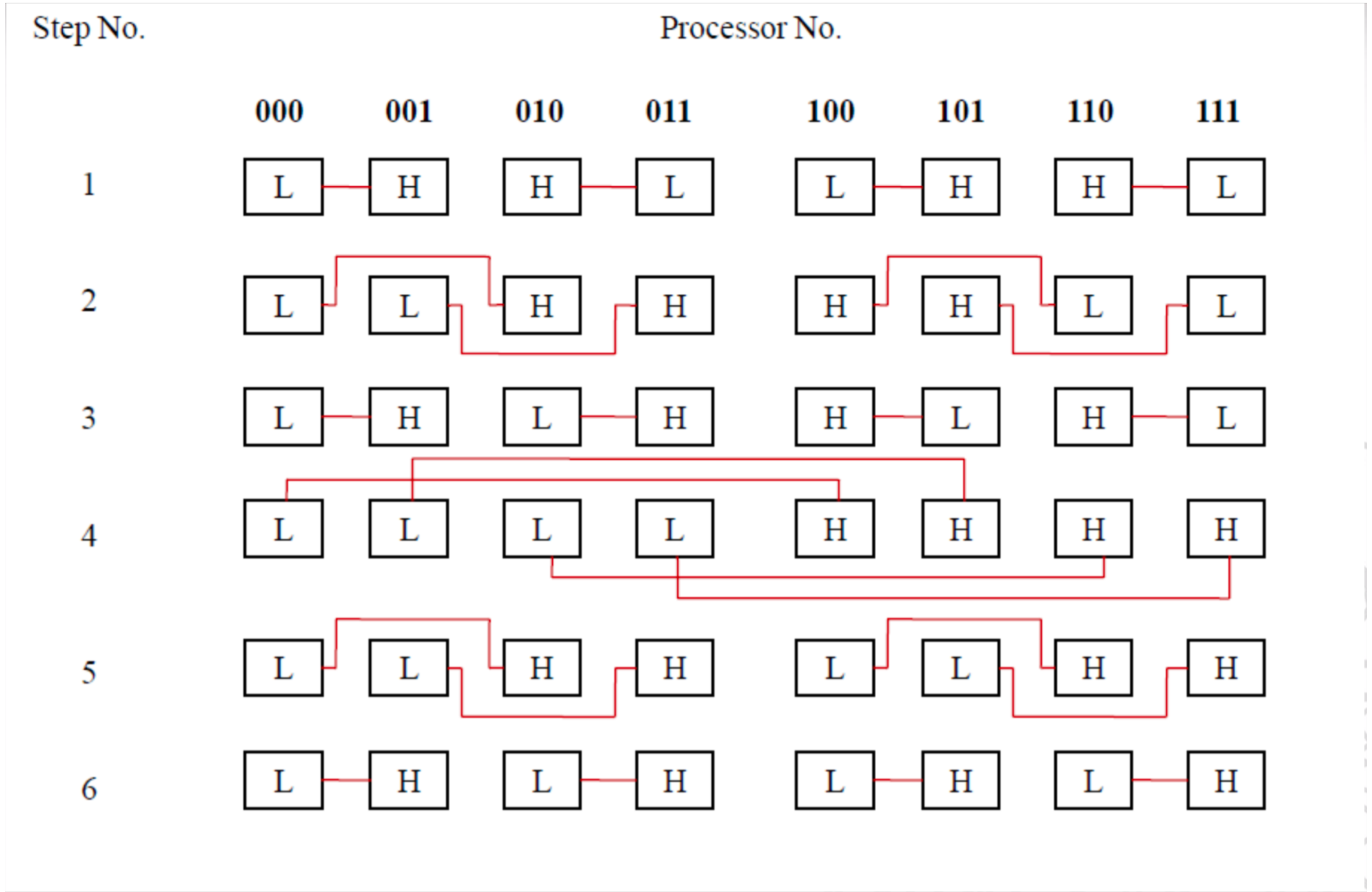


Time complexity = (Logn)^2 comparisons

MPI Implementation



Whom to compare with?



How to find the right pair?

```
for (i = 0; i <dimensions; i++) { // dimensions = log(n), iterates on stages
    for (j = i; j>=0; j--) { // combinations in each stage
        if (((process_rank >> (i + 1)) % 2 == 0 && (process_rank >> j) % 2 == 0)
|| ((process_rank >> (i + 1)) % 2 != 0 && (process_rank >> j) % 2 != 0)) {
                CompareLow(j);
  } else {
                CompareHigh(j);
        }
    }
 }
```

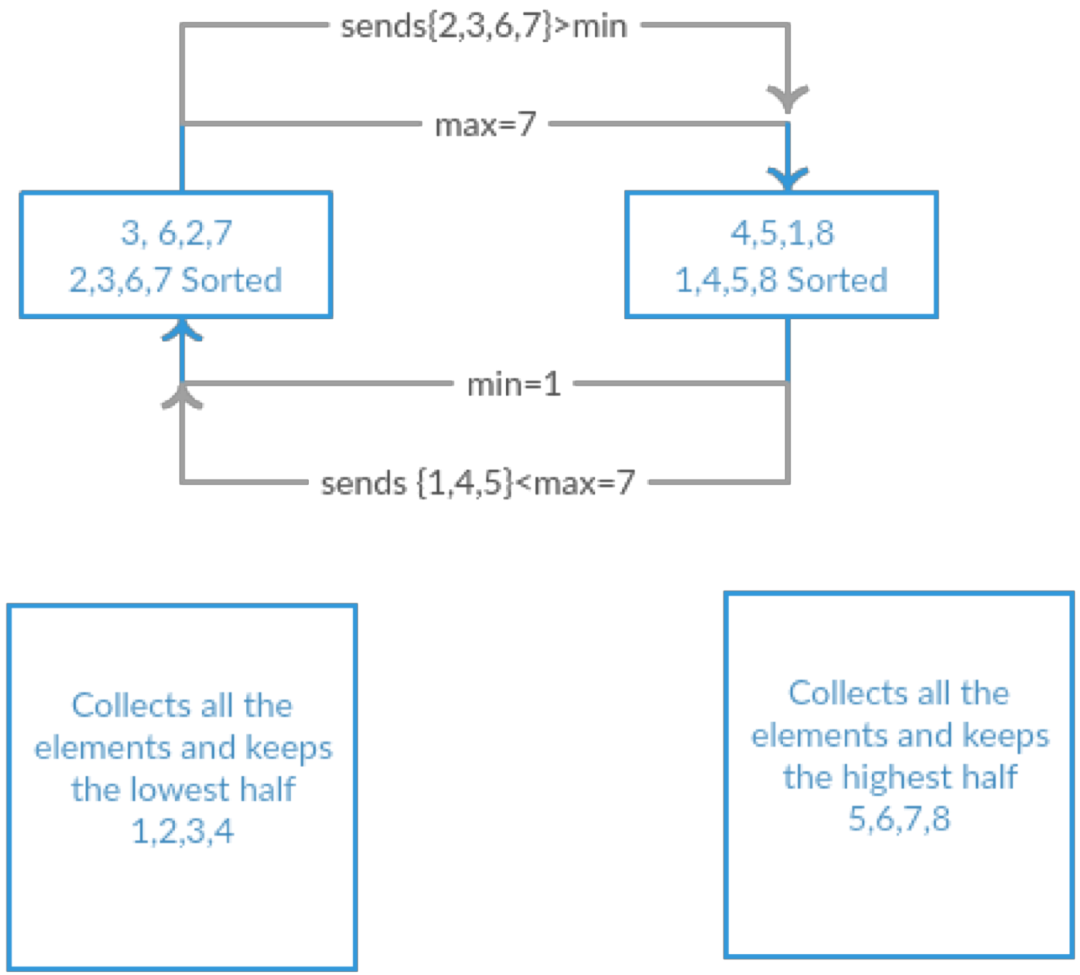
Example comparison

- Consider the first comparison, where process rank in binary is 000
- It finds the partner using bitwise EXOR operation.

    partner’s rank = process_rank ^ (1<<j)

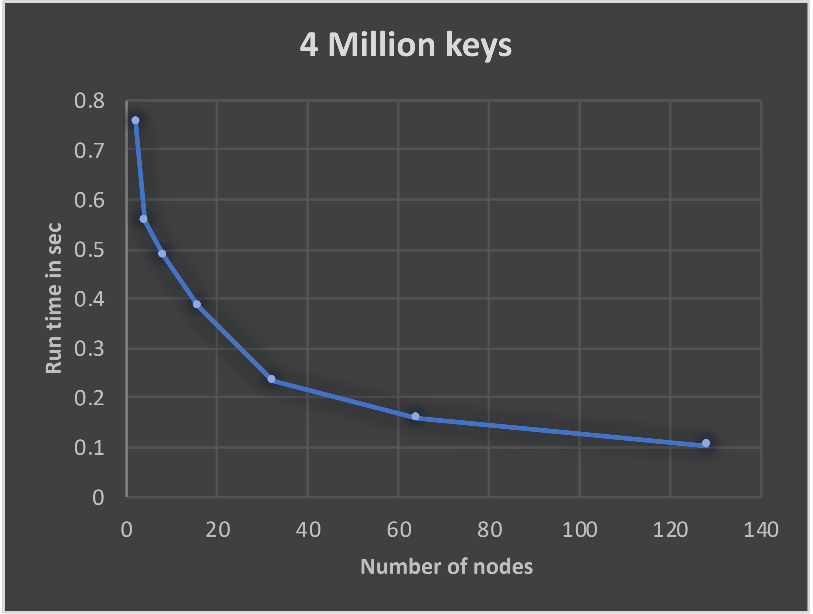
- Where j is comparison bit varies from [0,logn) and n is the number of processors.

What happens in a Comparison?



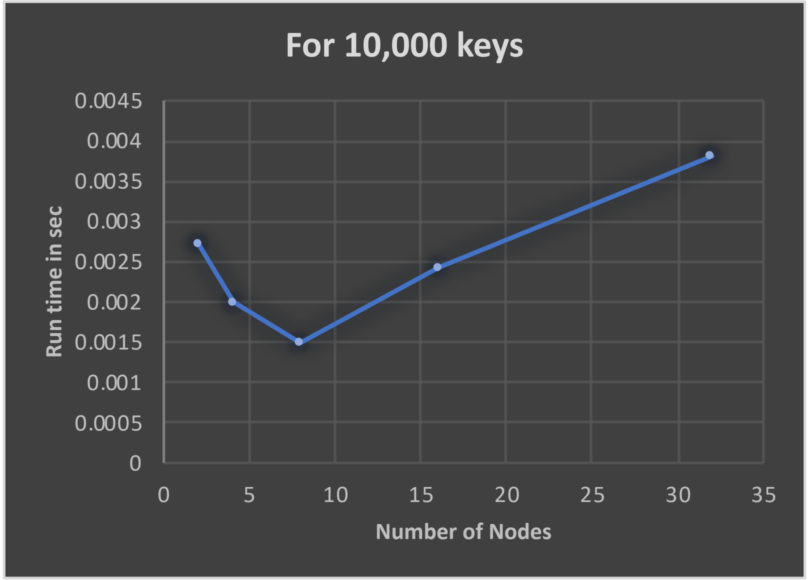
Few Results – 4 million keys

Nodes	Time in sec
2	0.753486
4	0.558978
8	0.484986
16	0.383336
32	0.232034
64	0.156750
128	0.102977



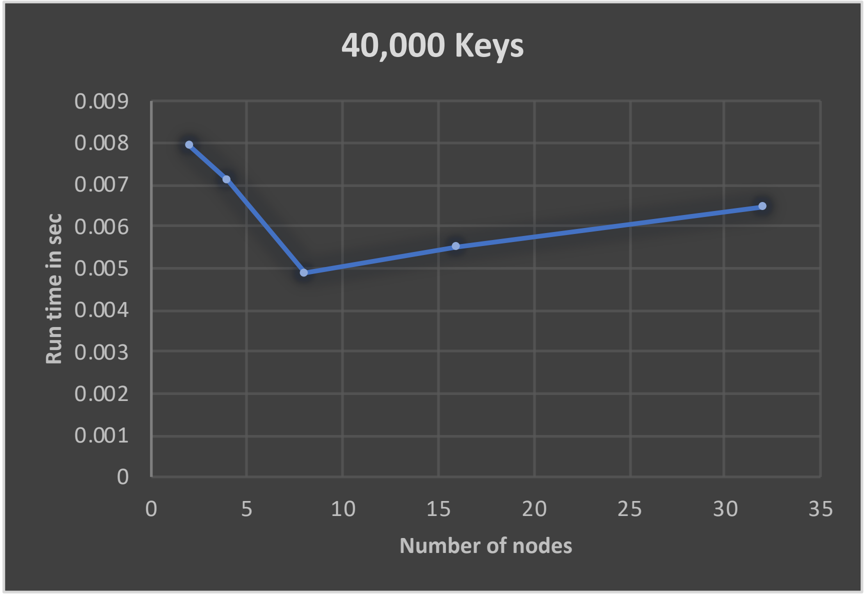
10,000 keys

Nodes	Time in sec
2	0.00272
4	0.00199
8	0.00149
16	0.00242
32	0.00381

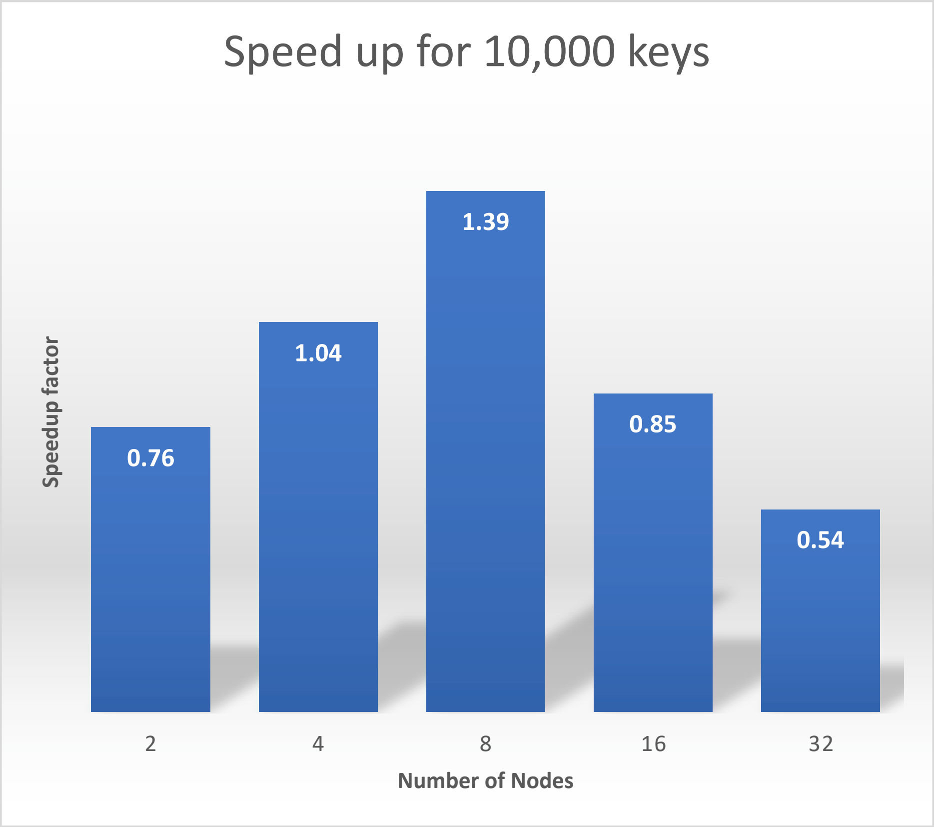


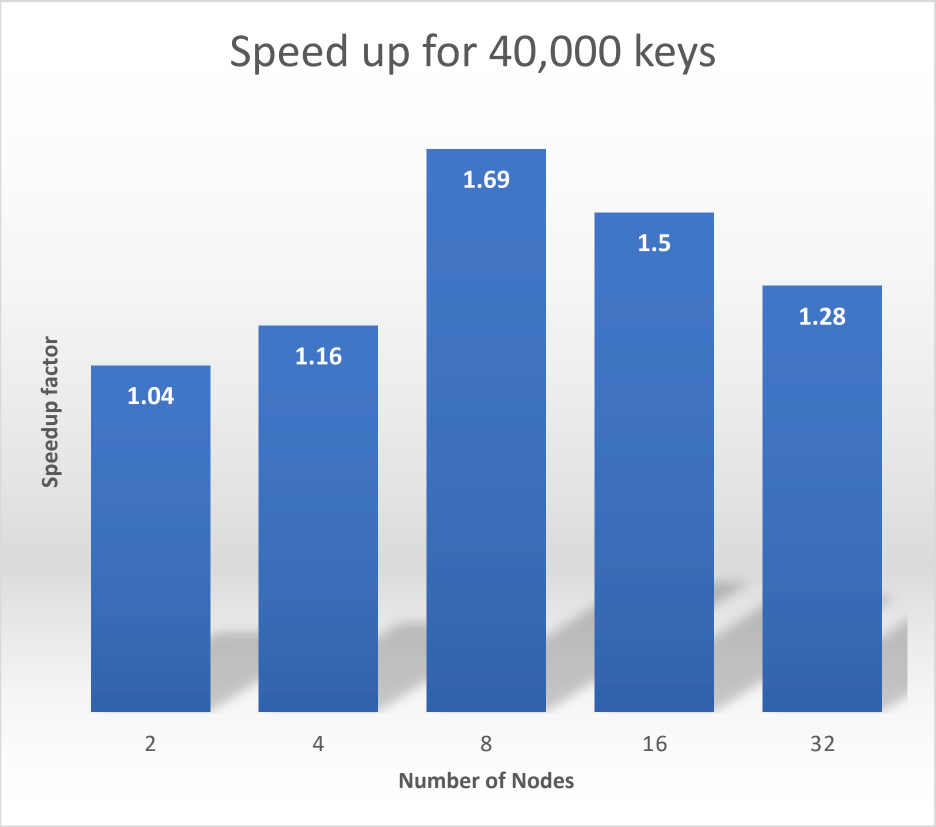
40,000 keys

Nodes	Time in sec
2	0.007921
4	0.007094
8	0.004884
16	0.005504
32	0.006473



Speed up factor





Challenges

- Allocation of higher order nodes 128, 256.
- Difficulty in debugging the Algorithmic flaws.

Outcome

- Found out how parallel implementation can reduce runtime by significant amount compared to sequential runs.
- How runtime behaves as number of cores is increased.
- Observe speedup in latency with Amdahl’s law.
- Knowledge of MPI, Open MPI.

References

- https://ubccr.freshdesk.com/support/solutions/articles/13000026245-tutorials-and-training-documents
- https://ubccr.freshdesk.com/support/solutions/articles/5000688140-submitting-a-slurm-job-script
- https://cse.buffalo.edu/faculty/miller/teaching.shtml
- https://www.geeksforgeeks.org/bitonic-sort/

- Find my code on /github.com/sajid912/MPI-Bitonic-Sort

Thank you!!