Comparison of MPI and GA on Glenn1 and Glenn2

Final Project Report
CSE – 721
Introduction to Parallel Computing

Karthik Raj Saanthalingam

Abstract

- Compare the Message passing model - MPI and the Partitioned Global Address Space model
- Analyze work sharing, communication tradeoffs
- Analyze communication pattern, scalability, interoperability with other libraries etc.

Message Passing Interface

- Processes coordinate through messages
- Processes should coordinate
- Data is explicitly associated with each processor
- Suitable for clusters with higher cost for accessing non-local memory
- Communication overhead is transparent

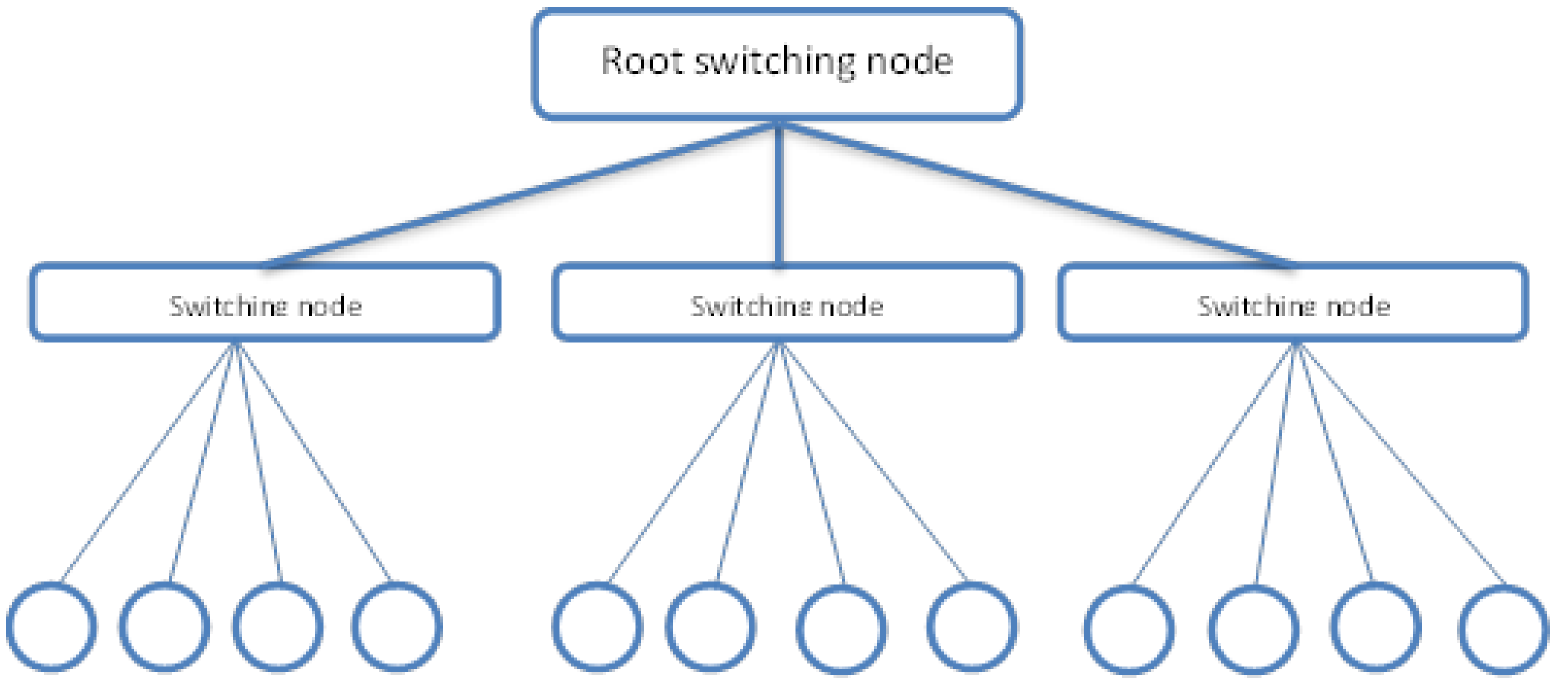
Partitioned Global Address Space - Global Arrays

- Abstraction of a shared memory model
- One sided communication
- No need of coordination between the processes unless explicitly required by application
- Data consistency – managed explicitly
- Applicable only for arrays!

Glenn1 Vs Glenn2

- Glenn1
  - Semi-fat tree
  - 4 processors per node
  - 8 GB shared memory
- Glenn2
  - Semi-fat tree
  - 8 processors per node
  - 24 GB shared memory

Glenn1 Vs Glenn2



Glenn1 cluster

Benchmark 1

- **Matrix Multiplication**
  - MPI implementation using asynchronous communication
    - 1D row partition
    - One-all communication of Matrix B
    - Local computation
    - Root process receives the results

Benchmark 1

- **Matrix Multiplication**
  - GA implementation using asynchronous communication
    - 2D partition
    - One-all initializes the matrices
    - Local computation
    - No more coordination between processes

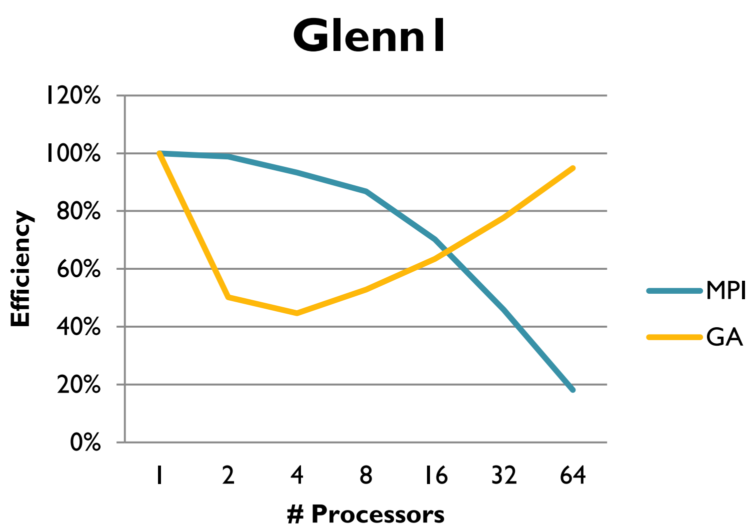
Communication overhead

- MPI
  - The amount of communication required by the program is $3*N^2$.
    - N^2 - for broadcasting array B
    - N^2 - for sending out each interval of rows $(P*(N^2/P) = N^2)$
    - N^2 - for receiving the results back from the worker processors

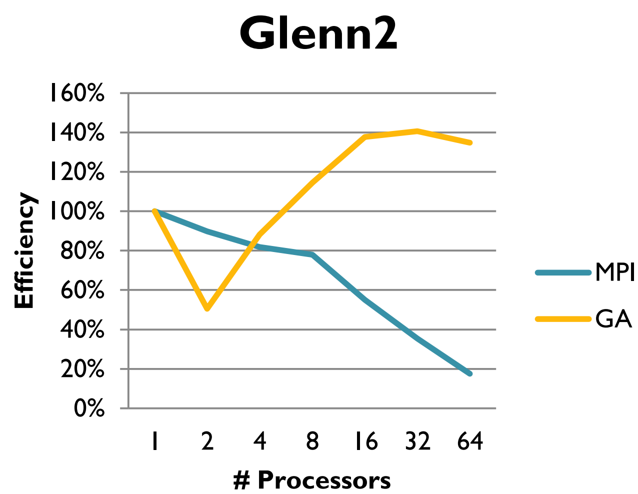
Communication overhead

- GA
  - The amount of communication required by the program is N^2
    - $2N^2$ - for initializing matrices

Efficiency

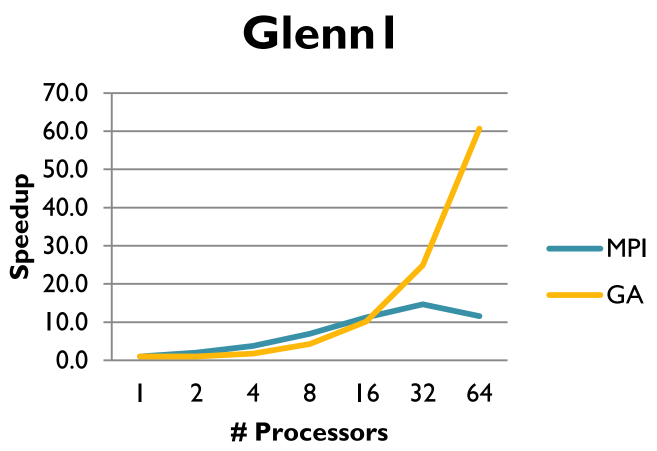


Glenn1
120%
100%
80%
60%
40%
20%
0%
Efficiency
1
2
4
8
16
32
64
Processors
MPI
GA

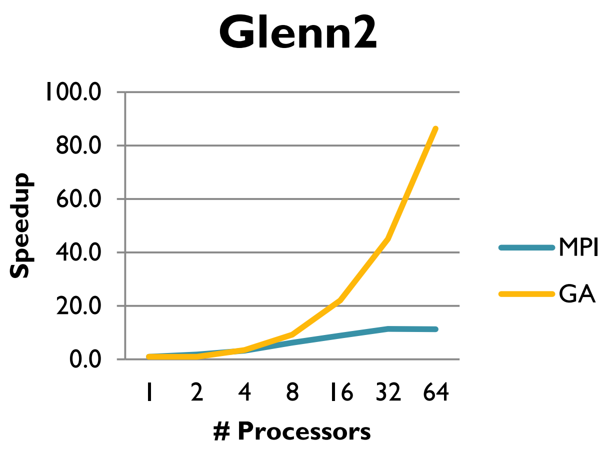


Glenn2
160%
140%
120%
100%
80%
60%
40%
20%
0%
Efficiency
1
2
4
8
16
32
64
Processors
MPI
GA

Speedup



Glenn1
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
Speedup
1
2
4
8
16
32
64
Processors
MPI
GA



Glenn2
100.0
80.0
60.0
40.0
20.0
0.0
Speedup
1
2
4
8
16
32
64
Processors
MPI
GA

Benchmark 2

- Merge sort
  - MPI implementation using synchronous communication
    - 1D row partition
    - Processes initialize the data for the range they own
    - Local computation
    - Communicate sorted list to neighbor – Send & Receive
    - Work sharing - Half processes are idle!

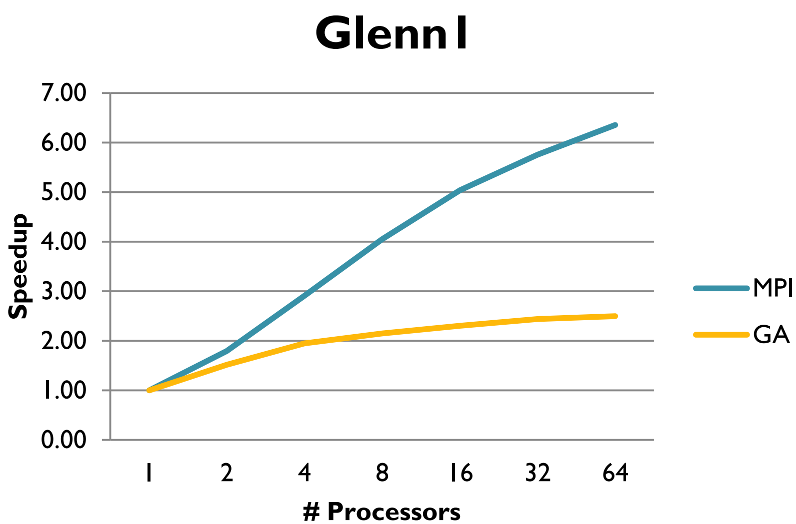
Benchmark 2

- Merge sort
  - GA implementation using
    - 1D row partition
    - Processes initialize the data for the range they own
    - Local computation
    - Communicate sorted list to neighbor – one sided communication
    - Work sharing - Half processes are idle!

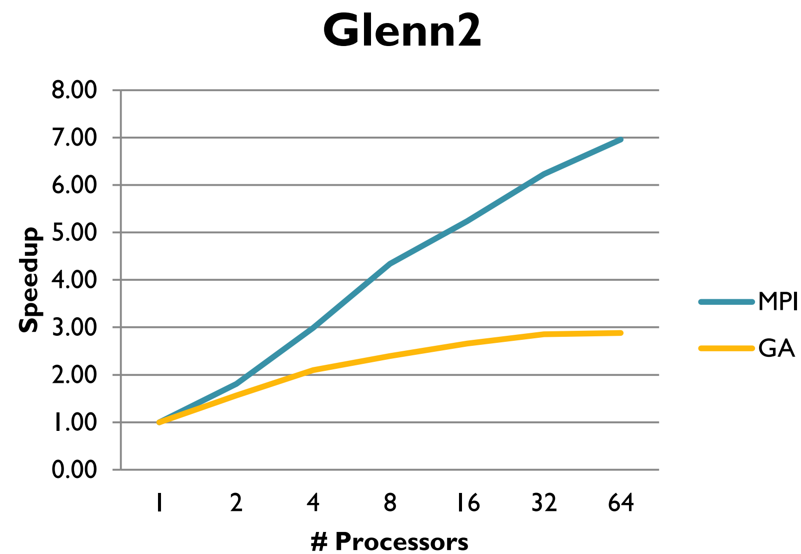
Communication Overhead

- MPI & GA
  - N/P at max
- Root node holds the entire array
- Memory requirement
  - N – Root node
  - N/2 – Node that is at a distance of P/2 from root node
  - etc

Speedup

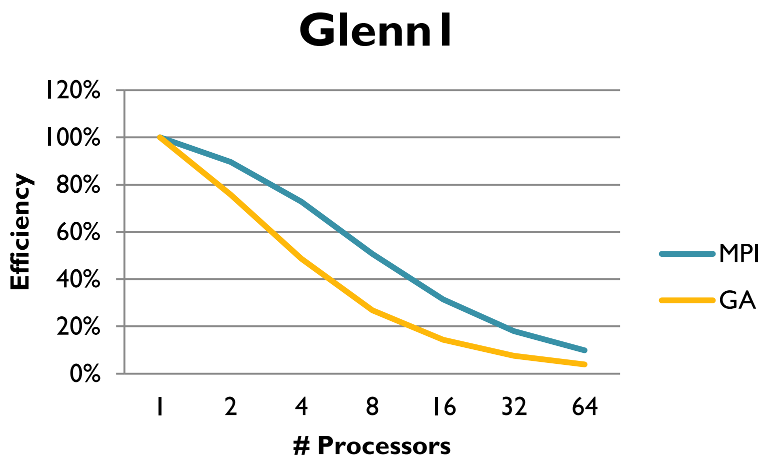


Glenn1
Speedup
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
4
8
16
32
64
Processors
MPI
GA

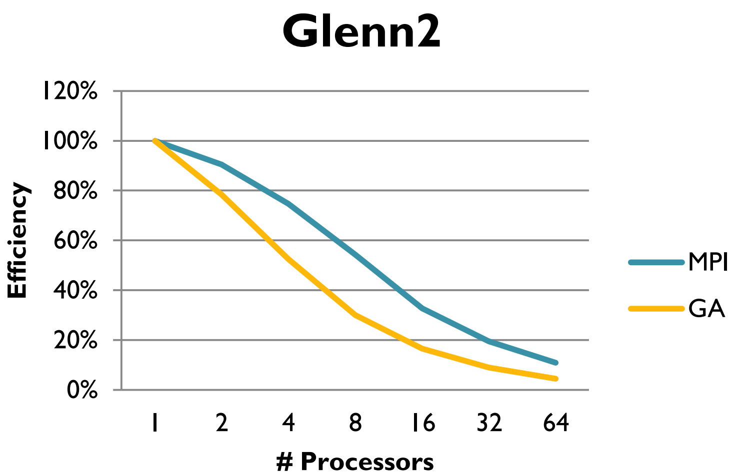


Glenn2
Speedup
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
4
8
16
32
64
Processors
MPI
GA

Efficiency



Glenn1
120%
100%
80%
60%
40%
20%
0%
Efficiency
1
2
4
8
16
32
64
Processors
MPI
GA



Glenn2
120%
100%
80%
60%
40%
20%
0%
Efficiency
1
2
4
8
16
32
64
Processors
MPI
GA

Inferences

- One-sided VS two sided communication
  - Inherent nature of application
- 1D VS 2D partitioning
  - 2D partitioning - scalable
- Inter-node VS intra-node communication
  - Intra-node communication – faster, share memory
- Shared memory
  - Larger memory, larger cache – better performance


Thank You!