Comparative Analysis of Collective Communication Algorithms

Presented By :
Mohit Garg
ISM2013009

OVERVIEW

- Objective
- Collective Operations
- Two Tree Algorithm
- Implementations
- Comparison Results
- Conclusion

Objective

Implement

To implement different collective communication algorithms using MPI (Message Passing Interface).

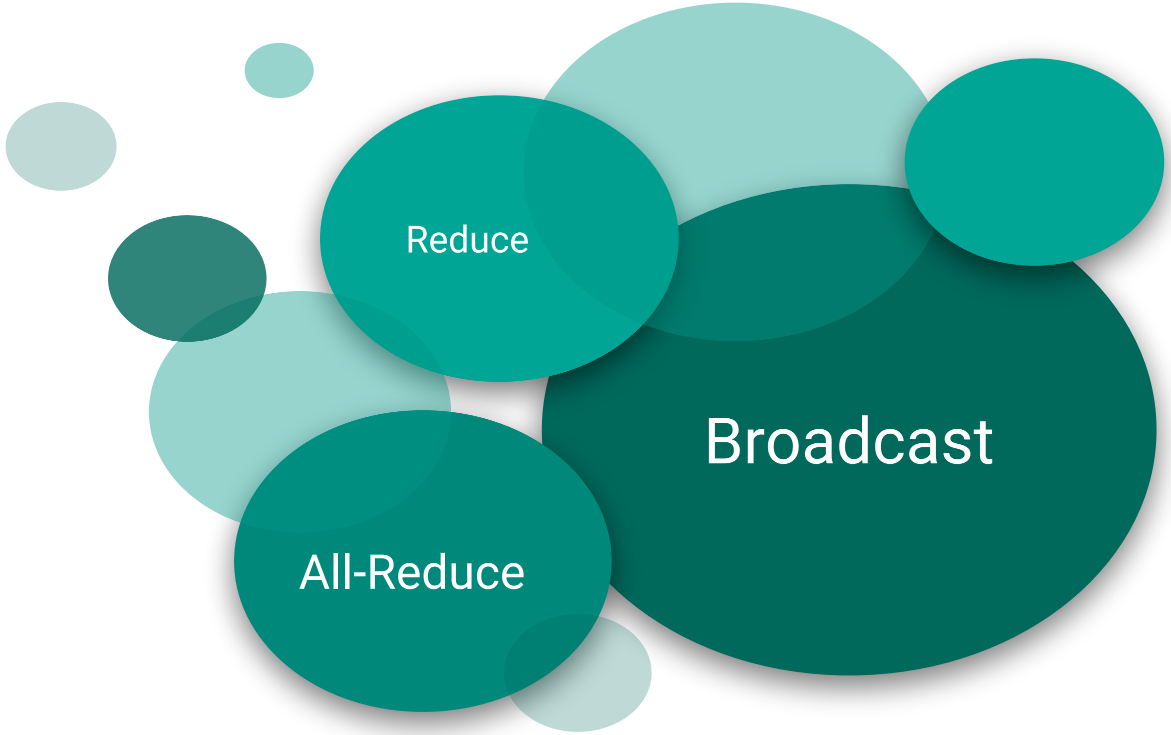
Analyse

To analyse these algorithms on the basis of bandwidth utilization (runtime) under different parameters such as number of process and message size.

Improvise

After analyzing each algorithm we intend to make suitable changes to the implementation so as to increase the performance.

Analyzed Collective Operations



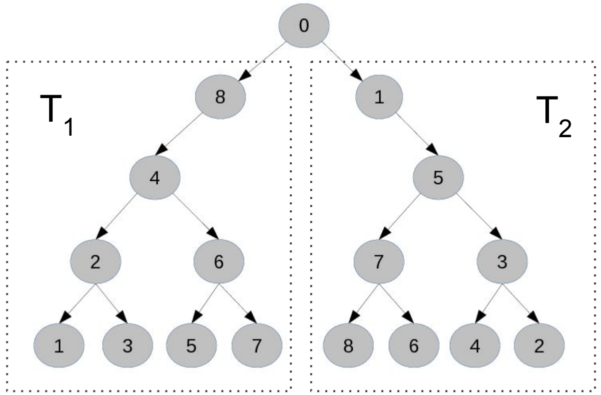
Two Tree Algorithm

While pipelined binary trees enhance the general bandwidth use, the leaves in the tree never transmit messages. To utilize the leaf nodes bandwidth, one can utilize two trees with various structure to such an extent that every node sends in the end.

Topology (As proposed by Sanders and Traff)

Let $h = \lceil \log(p + 2) \rceil$
We construct two binary trees T_1 and T_2 of height $(h - 1)$ with the following properties :

- PEs are assigned to the nodes of these trees such that they form an in-order numbering of both T_1 and T_2.
- T_2 is dual to T_1 in the sense that its non leaf nodes are the leaves of T_1 and vice versa.



Two Tree Sanders

Only defined for even no of processes (Fig: 1)

For odd number the last node is added either as root for both T_1 & T_2 or at the bottom in both trees.

Synchronization Overhead

As defined in paper, their algorithm is synchronized using coloring method which adds extra waiting time.

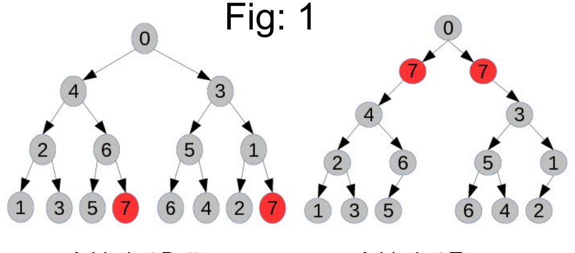
Imbalancing (Fig: 2)

Their topology does not provide balancing to the tree, which should results in reduced bandwidth utilization.

Complex Tree Construction

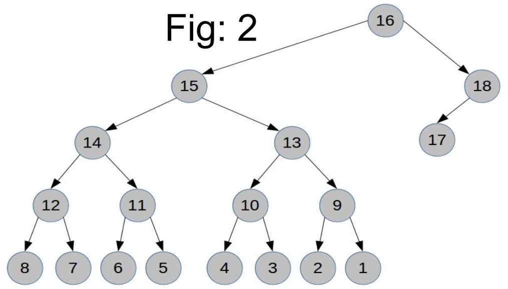
Tree construction using this method is complex and also takes a significant amount of time which impacts on the overall performance.

Fig: 1



Added at Bottom Added at Top

Fig: 2



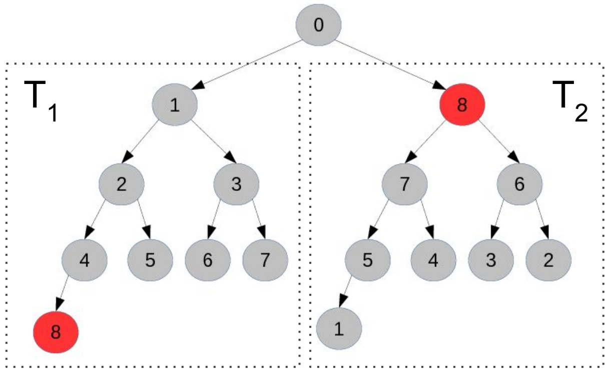
Two Tree Complete

As we analyzed Two Tree Sanders topology, we came across several problems as discussed in previous slide. So to overcome these issues we proposed a simpler and effective solution to create a Two Tree Topology. As our method always make complete binary Tree we have named this version as TwoTreeComplete.

Topology (Two Tree Complete)

Let h = log(p + 1)
We construct two complete binary trees T_1 & T_2 of height h with following property :

- PEs are assigned to the nodes of these trees such that they form a level order(increasing sequence) numbering in T_1 and level order(decreasing sequence) in T_2.
- This itself ensures the property that inner node in T_1 are leaf nodes in T_2.
- Also it automatically solves the problem for odd number of nodes.



Implementations

		BroadCast	Reduce	All-Reduce
1	State of the Art	• Binomial Tree (Short msg) • Scatter + Allgather (Large msg) (Binomial+Recursive Doubling)	• Binomial Tree (Short msg) • ReduceScatter + Gather (Large msg) (Recursive Halving+Binomial)	• Recursive Doubling (Short msg) • ReduceScatter + AllGather (Large msg) (Recursive Halving + Recursive Doubling)
2	Other Known Algorithms	• Pipelined Binary Tree • Linear Pipeline	• Pipelined Binary Tree • Linear Pipeline	• Pipelined Binary Tree • Ring Pipeline
3	Proposed & Improvised Algorithms	• Two Tree SandersTopSync • Two Tree SandersBottomSync • Two Tree Complete • TwoTreeSandersBottom	• Two Tree Sanders • Two Tree Complete	• Two Tree Sanders • Two Tree Complete • Two Tree CompleteOptimal • Two Tree Sanders+Complete

Note :

SandersTop	: Using Sanders Topology and when odd no. of nodes, last node is added at top.
SandersBottom	: Using Sanders Topology and when odd no. of nodes, last node is added at bottom.
Sync	: Algorithm is synchronized using coloring method (Unsync if not written)
Complete	: Using our version of Two Tree Topology.

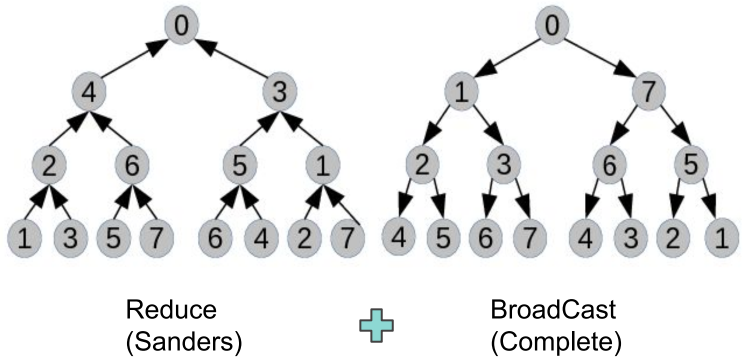
Implementations

All the implementations in the table are either known or can be understood by their name with the exception of the last two in allreduce , which we have improvised looking at the results. We will now explain both of them.

1. Two Tree Sanders+Complete

We have divided the allreduce operation in two parts: Reduce and Broadcast.
In this version we uses different topologies for these two parts :

- Sanders Two Tree version for Reduce
- Complete Two Tree Version for Broadcast

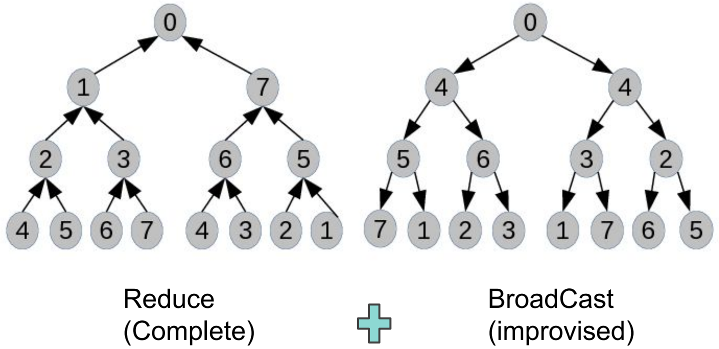


Implementations

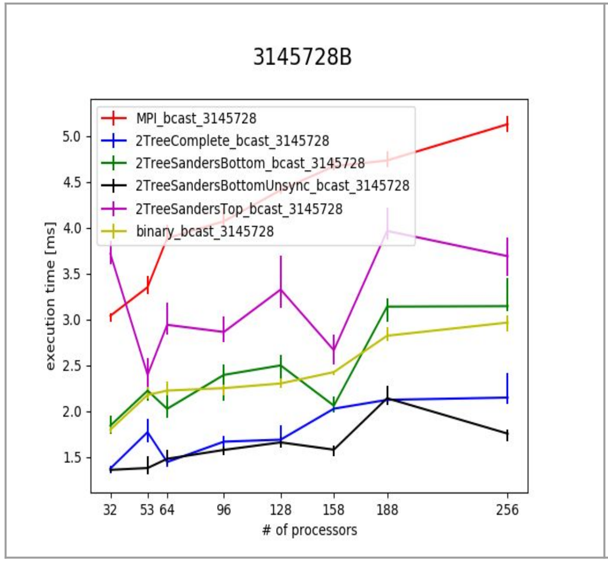
2. Two Tree CompleteOptimal

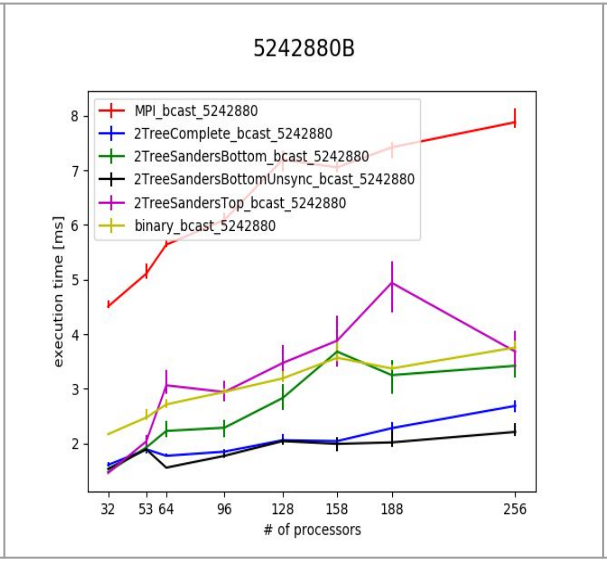
We have divided the allreduce operation in two parts: Reduce and Broadcast.
In this version we uses different topologies for these two parts :

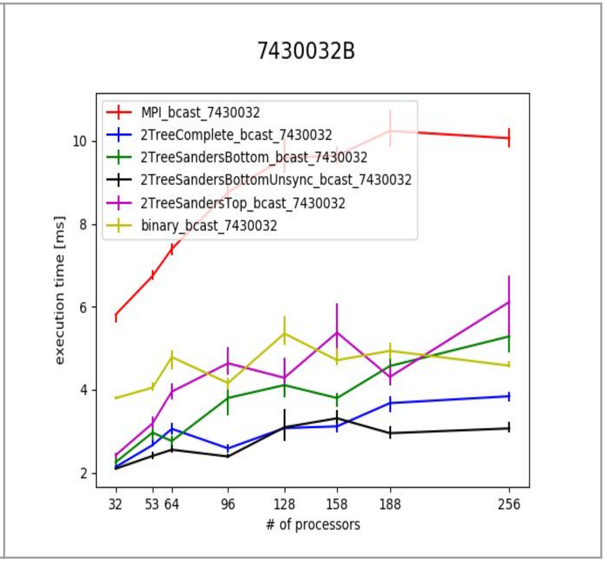
- Sanders Two Tree version for Reduce
- For BroadCast we created a new topology (improvised) in a way such that nodes expected to complete its task early in Reduce part appears high up in Broadcast topology.



Results BroadCast

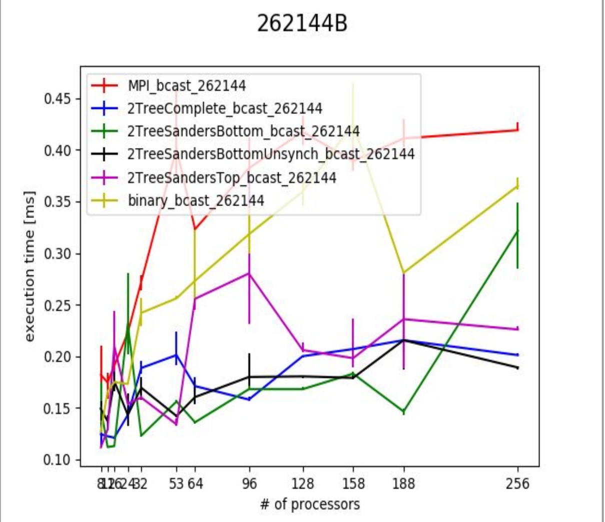


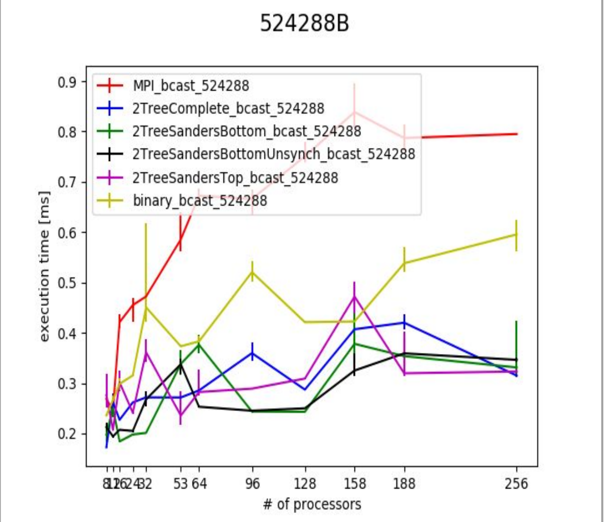


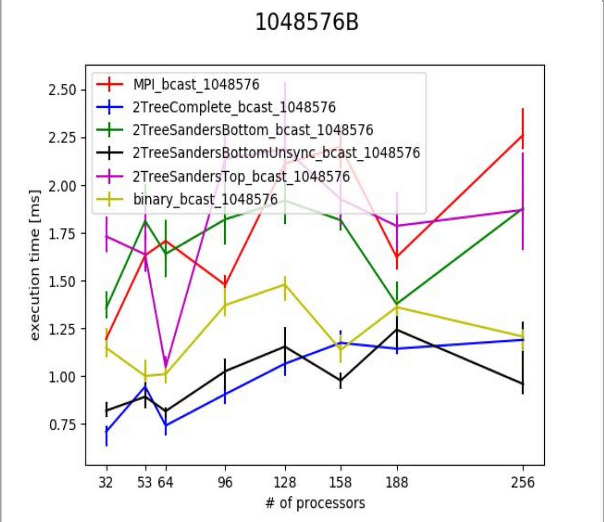


Remark : Clearly 2-Tree algorithms (Sanders and Complete) without synchronization is a better choice for broadcast when dealing with this range of message sizes.

Results BroadCast



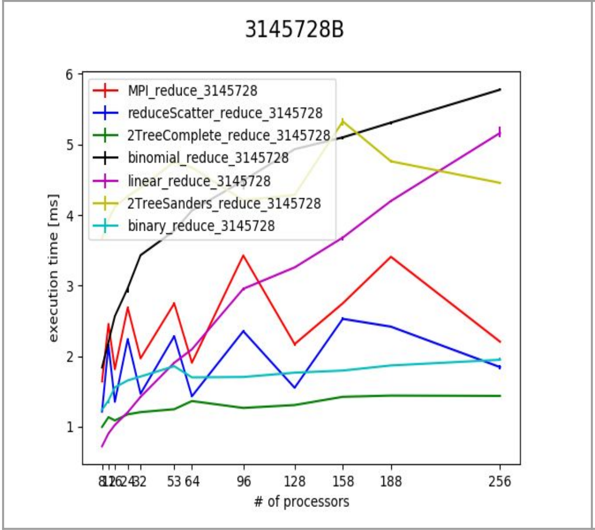


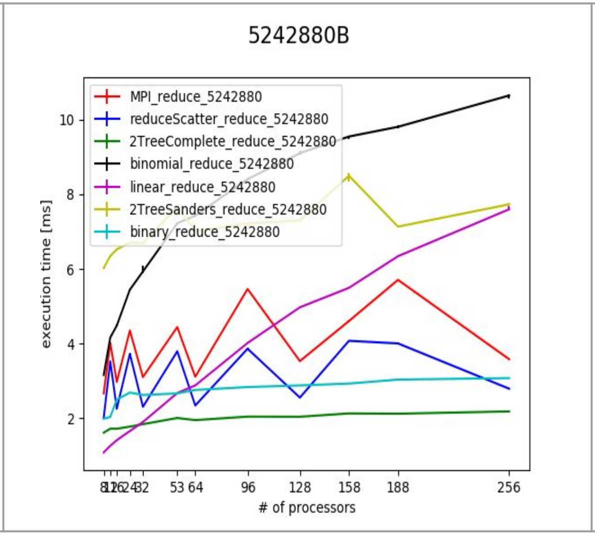


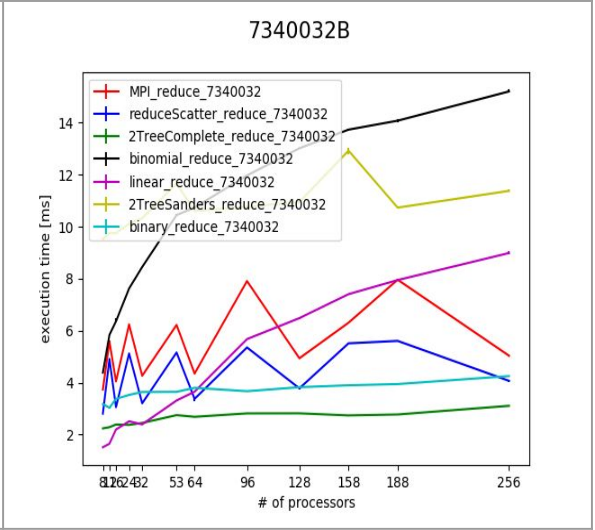
Remark : We can see here that for smaller message sizes, Two Tree Sanders with synchronization is almost equal (as in 524288B) or is better (as in 262144B) than others.

Results

Reduce

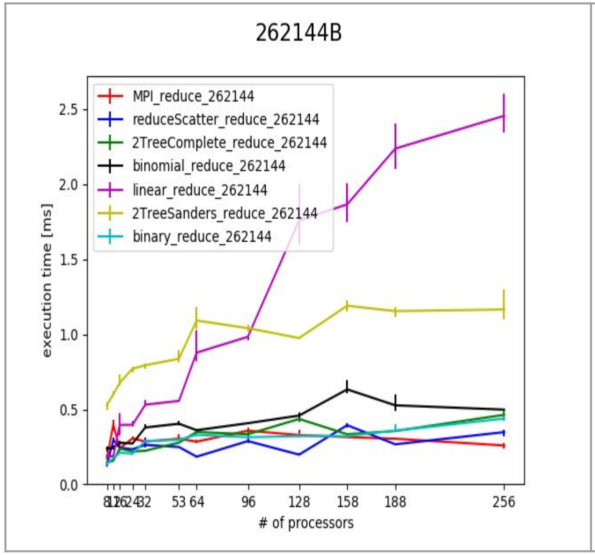


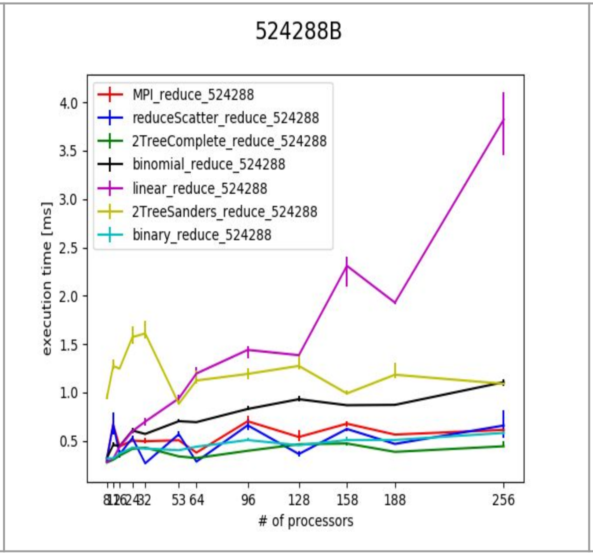


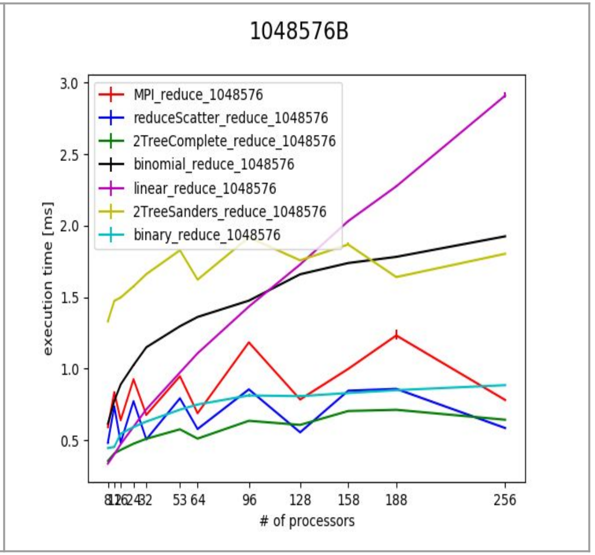


Remark : Two Tree Complete is clearly a good choice for reduce operation when dealing with large message sizes. Also a point to notice is that sanders topology suffers in reduce operation even without synchronization.

Results Reduce



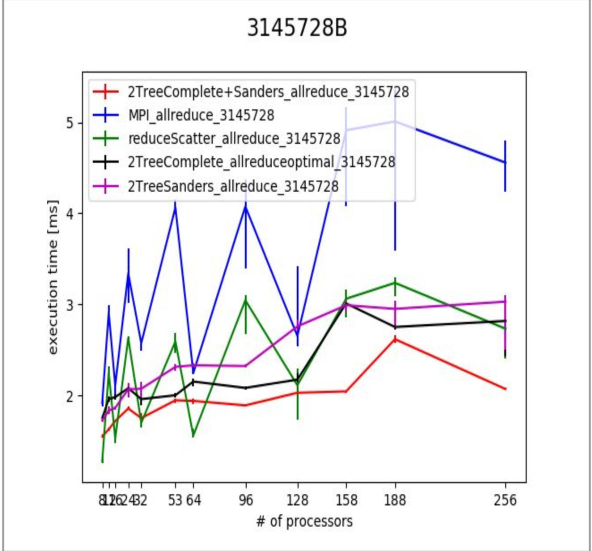


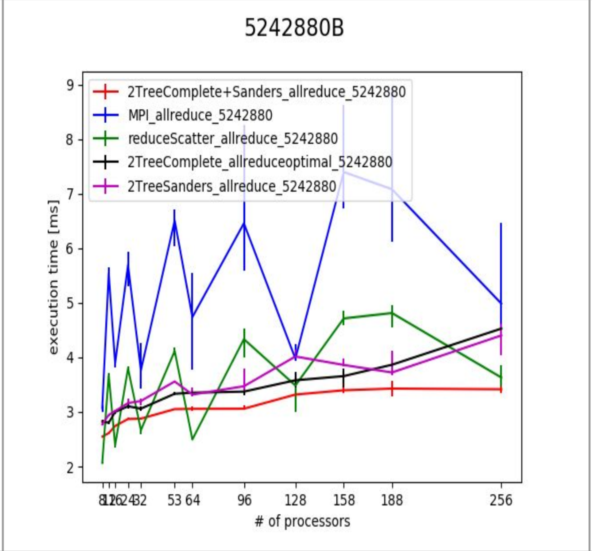


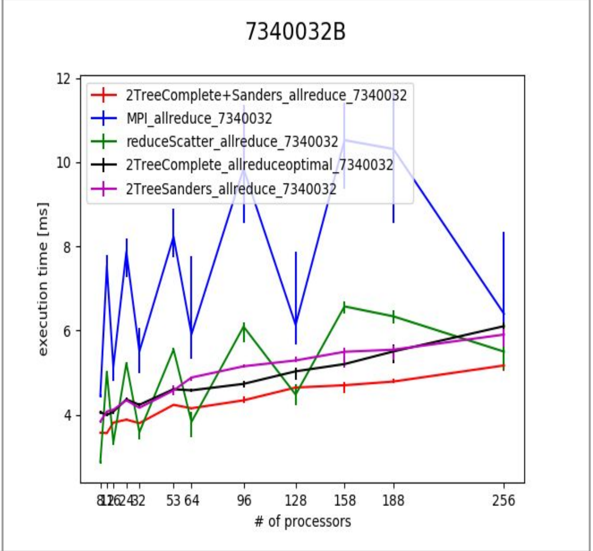
Remark : For smaller message sizes reduceScatter+Gather algorithm also seems to work fine especially for power-of-two number of processes. Otherwise Two Tree Complete is an excellent choice.

Results

AllReduce(Selected)



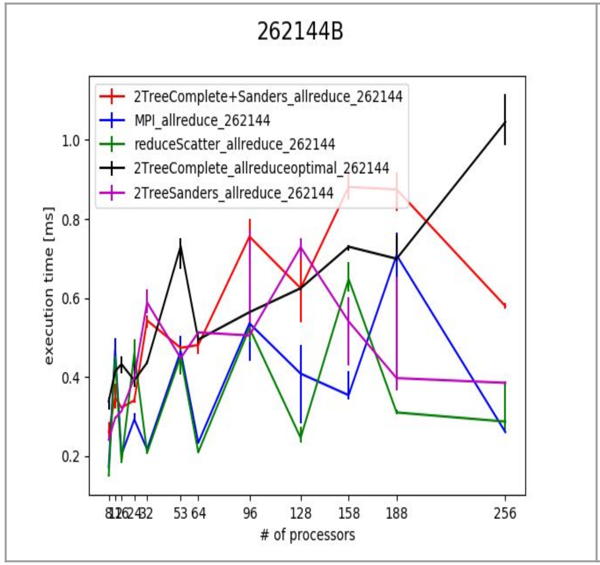


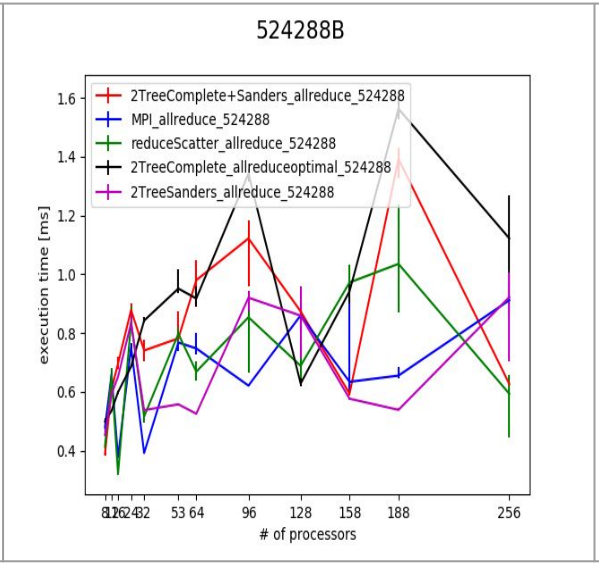


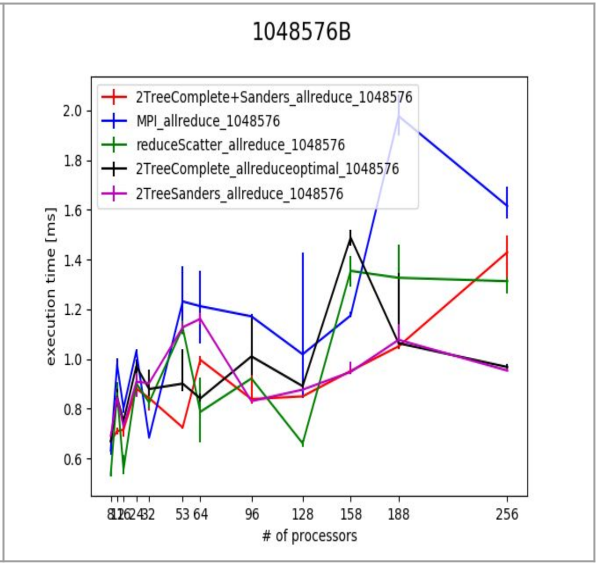
Remark : For All-Reduce operation we can see that for power-of-two number of processes ReduceScatter+Allgather is an optimal choice, for all the other cases 2Tree algorithm(Sanders+Complete)version is best and 2Tree algorithm(Optimized version) also is good.

Results

AllReduce(Selected)







Remark : For smaller message sizes ReduceScatter+AllGather seems to be a better choice even for non-power-of-two processes(for 262144B). Other than that it is hard to choose an algorithm that is good in all cases.

CONCLUSION

- Two Tree algorithm seems to be better choice for most of the cases and the results are even more encouraging given the fact that chunk values choosen may not be optimal, so pipeline algorithms can perform even better.
- All the analysis is done using the sum operators for the reduction steps, study of other operators can also lead to important results.
- It is very clear from the results that MPI library functions are also using reduceScatter variations for reduce and allreduce operations which the Two tree algorithm is outperforming in most cases.
- Two Tree Algorithm is more stable than any other algorithm we have analyzed.

THANK YOU