Analysis and Implementation of Asynchronous Finite Difference Scheme for Advection - Diffusion Equation

Kumar Saurabh [1] Prof. S. Sundar[1] Prof. Dr. Martin Frank [2]

[1] Department of Mathematics
IIT Madras

[2] Math CCES
RWTH Aachen

May 16, 2016

Acknowledgement: DAAD, RWTH IT Center

Outline

Literature

- Diego A. Donzis and Konduri Aditya. Asynchronous Finite Difference Scheme for Partial Difference Equations *Journal of Computational Physics*. 274(0):370-392,2014
- Thomas Camminady. CES Seminar Paper on Asynchronous Finite Difference Scheme for Partial Difference Equation. January 9,2015

Motivation

- Many natural and engineering systems can be described with PDEs
  - Fluid mechanics, Electromagnetism, Quantum Mechanics
- Analytical Solution not known. Need to solve these problems numerically.

[1] Maitham Alhubail, Qiqi Wan. The swept rule for breaking the latency barrier in time advancing PDEs

Motivation

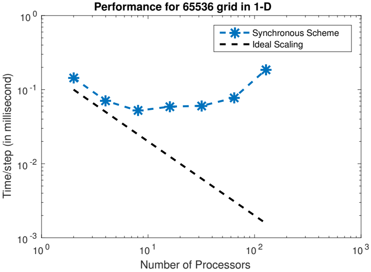
- Many natural and engineering systems can be described with PDEs
  - Fluid mechanics, Electromagnetism, Quantum Mechanics
- Analytical Solution not known. Need to solve these problems numerically.
- Large number of numerical methods: Finite Difference, Finite Volume, Spectral Method, etc.
  - The complexity of systems at realistic conditions typically requires massive computational resources.
  - The problem is decomposed into a large number of Processing Element (PEs).
  - Extreme-scale computer clusters can solve PDEs using over 1,000,000 cores. [1]
  - The communication is required between the PEs to solve the PDEs to compute spatial derivatives.
- Computation rates are much faster than communication
  - Exascale: Communication likely to be bottleneck.

[1] Maitham Alhubail, Qiqi Wan. The swept rule for breaking the latency barrier in time advancing PDEs

- Direct Numerical Simulation (DNS)
  - Resolve all scales in space and time.
  - Computationally very expensive.
  - Communication or synchronization takes upto 50 - 70 % of the total computation time.[2]

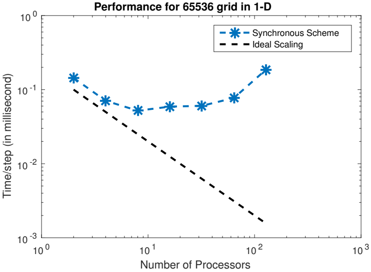
[2] Jagannathan & Donzis (XSEDE 2012), Sankaran et al. (SC 2012), Lee et al. (SC 2013)

- Direct Numerical Simulation (DNS)
  - Resolve all scales in space and time.
  - Computationally very expensive.
  - Communication or synchronization takes upto 50 - 70 % of the total computation time.[2]



[2] Jagannathan & Donzis (XSEDE 2012), Sankaran et al. (SC 2012), Lee et al. (SC 2013)

- Direct Numerical Simulation (DNS)
  - Resolve all scales in space and time.
  - Computationally very expensive.
  - Communication or synchronization takes upto 50 - 70 % of the total computation time.[2]



- Machine Failure
  - Petascale computation spreads over various nodes.
  - What if one of the node fails?

[2] Jagannathan & Donzis (XSEDE 2012), Sankaran et al. (SC 2012), Lee et al. (SC 2013)

Simulations at exascale

Issues with the current Method:

- Communications.
- Synchronization at various time level.

New approach:

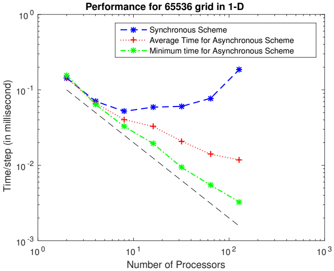
- Can we relax Synchronization?
- Asynchronous numerical Schemes.
- Objective: trade-off accuracy and performance quantitatively and predictably.

Issues with the current Method:

- Communications.
- Synchronization at various time level.

New approach:

- Can we relax Synchronization?
- Asynchronous numerical Schemes.
- Objective: trade-off accuracy and performance quantitatively and predictably.

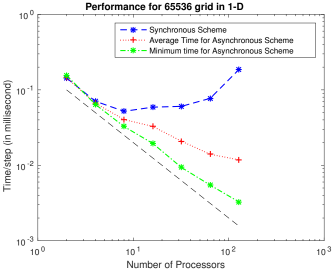


Issues with the current Method:

- Communications.
- Synchronization at various time level.

New approach:

- Can we relax Synchronization?
- Asynchronous numerical Schemes.
- Objective: trade-off accuracy and performance quantitatively and predictably.



Performance Improvement

- Computation time
  - Hardware: Faster hardware, Larger Memory Size.
  - Numerical schemes: Fewer operations.
- Communication time
  - Hardware: Network topology, switches, etc.
  - Numerical Scheme:
    - Fewer communications, Larger messages
    - Asynchronous Scheme: Stable and Consistent.

FD Schemes

FD method can be used to discretize and solve PDEs.
Consider 1D advection - diffusion equation:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \alpha\frac{\partial^2 u}{\partial x^2} \tag{1}$$

which can be discretized according to FTCS:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^n - u_{i-1}^n) = \frac{\alpha}{\triangle x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \mathscr{O}(\triangle x^2, \triangle t) \tag{2}$$

FD Schemes

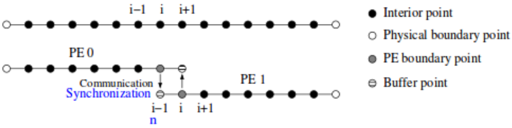
FD method can be used to discretize and solve PDEs.
Consider 1D advection - diffusion equation:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{1}$$

which can be discretized according to FTCS:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^n - u_{i-1}^n) = \frac{\alpha}{\triangle x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \mathcal{O}(\triangle x^2, \triangle t) \tag{2}$$

Stencil needs information not on the PE for boundary nodes.



Asynchronous Case

Problem: To compute u_i^{n+1} we need values of u that are possibly not on the PE.
Solution:

- For the left boundary

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^n - u_{i-1}^{\tilde{n}}) = \frac{\alpha}{\triangle x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^{\tilde{n}}) \tag{3}$$

- For the right boundary nodes:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^{\tilde{n}} - u_{i-1}^n) = \frac{\alpha}{\triangle x^2}(u_{i+1}^{\tilde{n}} - 2u_i^n + u_{i-1}^n) \tag{4}$$

where $\tilde{n}$ is the last available value for a particular node.

Problem: To compute u_i^{n+1} we need values of u that are possibly not on the PE.
Solution:

- For the left boundary

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^n - u_{i-1}^{\tilde{n}}) = \frac{\alpha}{\triangle x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^{\tilde{n}}) \quad (3)$$

- For the right boundary nodes:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^{\tilde{n}} - u_{i-1}^n) = \frac{\alpha}{\triangle x^2}(u_{i+1}^{\tilde{n}} - 2u_i^n + u_{i-1}^n) \quad (4)$$

where $\tilde{n}$ is the last available value for a particular node.

- Regarding $\tilde{n}$
  - Synchronous when $\tilde{n} = n$
  - $\tilde{n}$ can be $n, n-1, n-2, ...$
  - Concrete value of $\tilde{n}$ depends on hardware, network, traffic, (possible) unpredictable factors,...
  - $\tilde{n}$ is in fact a principle random variable.

Asynchronous Case

- Asynchronous Scheme - Stable and Accurate?

[3] At constant Courant Number: r_α

- Asynchronous Scheme - Stable and Accurate?
- Stability
  - Stable if the Synchronous Scheme is stable, irrespective of delay statistics.

[3] At constant Courant Number: r_α

- Asynchronous Scheme - Stable and Accurate?
- Stability
  - Stable if the Synchronous Scheme is stable, irrespective of delay statistics.
- Truncation Error
  - Not homogeneous in space and random.
  - Need for statistical description of the truncation error.
  - $\langle E \rangle$: Spatial average taken over the entire domain.
  - $\overline{E}$: Ensemble average taking into account the stochastic nature of delay.
- It can be shown that [3]:

$$\langle E \rangle \approx \underbrace{\langle K_S \rangle \Delta x^2}_{\text{Synchronous part}} + \underbrace{\frac{N_B}{N}\left(-r_\alpha \langle \dot{u} \rangle_B + r_\alpha \langle \dot{u}' \rangle_B \Delta x - \frac{r_\alpha \langle \dot{u}'' \rangle_B}{2} \Delta x^2 \right) \overline{\tilde{k}}}_{\text{Asynchronous Part}} \quad (5)$$

$$\langle \overline{E} \rangle \approx -\overline{\tilde{k}} \frac{P}{N} \propto \overline{\tilde{k}} P \Delta x \quad (6)$$

[3] At constant Courant Number: r_α

- Dependence on Scaling:
  - Strong Scaling: $\langle\overline{E}\rangle \sim O(\Delta x)$
  - Weak Scaling: $\langle\overline{E}\rangle \sim O(1)$
  - Verified by numerical experiments.

- Dependence on Scaling:
  - Strong Scaling: $\langle \overline{E} \rangle \sim O(\Delta x)$
  - Weak Scaling: $\langle \overline{E} \rangle \sim O(1)$
  - Verified by numerical experiments.

Delay	Strong Scaling	Weak Scaling
0(sync)	-2.0195	-2.0034
1	-1.0764	-0.0845
2	-1.0371	-0.0749
4	-1.0117	-0.0490
6	-1.0033	-0.0214
8	-0.9995	-0.0685

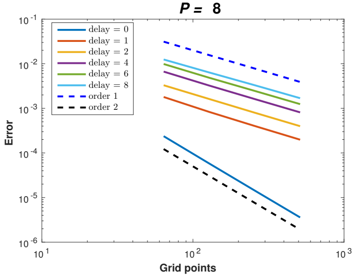


Figure : Strong Scaling

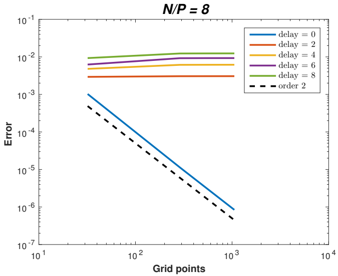


Figure : Weak Scaling

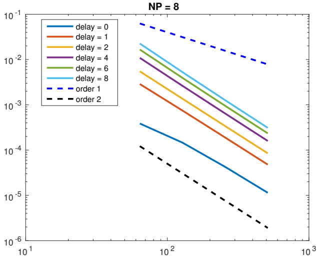
Order Recovery

- Order of Convergence falls to first order in case of Asynchronous Scheme.
- Need to recover the order.
- Can be achieved by analysis of Truncation Error and eliminating terms that effect accuracy.
  - Changing the time step size Δt.
  - New Asynchrony Tolerant Scheme.

Order Recovery

- Approximate: $\Delta t \sim \Delta x^3$

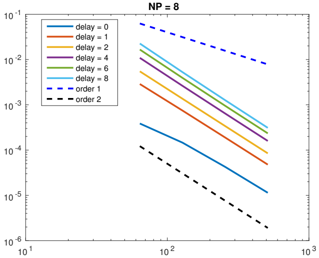
- Approximate: $\Delta t \sim \Delta x^3$



Delay	Async	Changing Δt
0(sync)	-2.0195	-1.9008
1	-1.0764	-1.9712
2	-1.0371	-2.0048
4	-1.0117	-2.0313
6	-1.0033	-2.0488
8	-0.9995	-2.0609

Order Recovery

- Approximate: $\Delta t \sim \Delta x^3$



Delay	Async	Changing Δt
0(sync)	-2.0195	-1.9008
1	-1.0764	-1.9712
2	-1.0371	-2.0048
4	-1.0117	-2.0313
6	-1.0033	-2.0488
8	-0.9995	-2.0609

- Drawback: (Space resolution) X 2 $\sim$ (Time resolution) X 8
  - More Computation Time.

The resulting Stable Scheme [4] obtained after truncation error analysis is:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^{\tilde{n}} - u_{i-1}^n) = \frac{\alpha}{2\triangle x^2}(u_{i+2}^{\tilde{n}} - u_{i+1}^{\tilde{n}} - u_i^n + u_{i-1}^n) + \mathcal{O}(k\Delta t, \Delta x, k\Delta t/\Delta x) \tag{7}$$

[4] Stable for $r_\alpha \leq 0.5$ and $r_c \leq \sqrt{2r_\alpha} - r_\alpha (r_c = \frac{c\Delta t}{\Delta x})$

The resulting Stable Scheme [4] obtained after truncation error analysis is:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^{\tilde{n}} - u_{i-1}^n) = \frac{\alpha}{2\triangle x^2}(u_{i+2}^{\tilde{n}} - u_{i+1}^{\tilde{n}} - u_i^n + u_{i-1}^n) + \mathscr{O}(k\Delta t, \Delta x, k\Delta t/\Delta x) \quad (7)$$

- Assumption: Only one of the neighbouring processor faces delay.
  - Basis: No buffering of message.

[4] Stable for $r_\alpha \leq 0.5$ and $r_c \leq \sqrt{2r_\alpha} - r_\alpha (r_c = \frac{c\Delta t}{\Delta x})$

Asynchrony Tolerant Scheme

The resulting Stable Scheme [4] obtained after truncation error analysis is:

$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^{\tilde{n}} - u_{i-1}^n) = \frac{\alpha}{2\triangle x^2}(u_{i+2}^{\tilde{n}} - u_{i+1}^{\tilde{n}} - u_i^n + u_{i-1}^n) + \mathscr{O}(k\Delta t, \Delta x, k\Delta t/\Delta x) \tag{7}$$

- Assumption: Only one of the neighbouring processor faces delay.
  - Basis: No buffering of message.

Possible Situation	Processor 1		Processor 2		Processor 3		Processor 4	
	LB	RB	LB	RB	LB	RB	LB	RB
Y	×	×	×	×	×	×	×	×
Y	✓	×	✓	×	✓	×	✓	×
N	×	✓	✓	×	×	×	×	×

[4] Stable for $r_\alpha \leq 0.5$ and $r_c \leq \sqrt{2r_\alpha} - r_\alpha$ ($r_c = \frac{c\Delta t}{\Delta x}$)

The resulting Stable Scheme [4] obtained after truncation error analysis is:

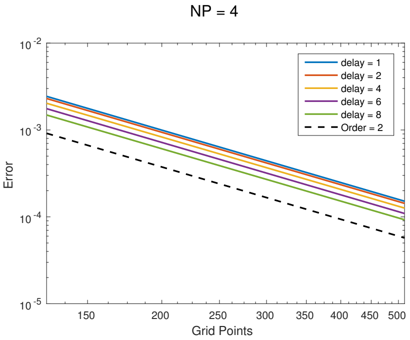
$$\frac{1}{\triangle t}(u_i^{n+1} - u_i^n) + \frac{c}{2\triangle x}(u_{i+1}^{\tilde{n}} - u_{i-1}^n) = \frac{\alpha}{2\triangle x^2}(u_{i+2}^{\tilde{n}} - u_{i+1}^{\tilde{n}} - u_i^n + u_{i-1}^n) + \mathscr{O}(k\Delta t, \Delta x, k\Delta t/\Delta x) \quad (7)$$

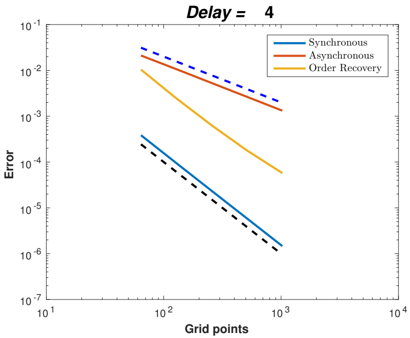
- Assumption: Only one of the neighbouring processor faces delay.
  - Basis: No buffering of message.

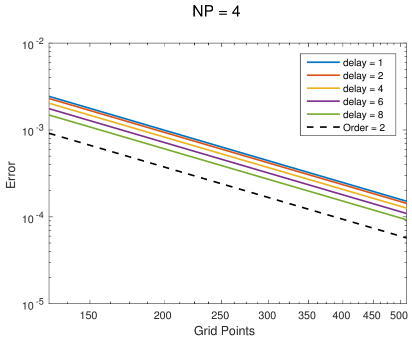
Possible Situation	Processor 1		Processor 2		Processor 3		Processor 4	
	LB	RB	LB	RB	LB	RB	LB	RB
Y	×	×	×	×	×	×	×	×
Y	✓	×	✓	×	✓	×	✓	×
N	×	✓	✓	×	×	×	×	×

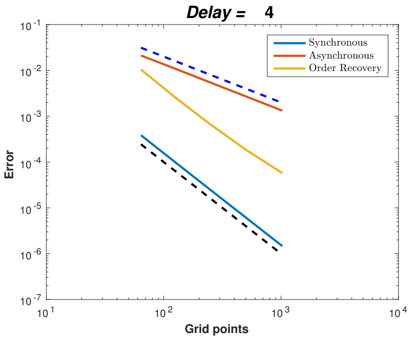
- Algorithm:
  - Communicate the time step along with the data.
  - If the processor faces delay compute with AT schemes for boundary nodes.
  - If the processor do not faces delay communicate with FTCS scheme.
  - For interior nodes, compute with FTCS scheme.

[4] Stable for $r_\alpha \leq 0.5$ and $r_c \leq \sqrt{2r_\alpha} - r_\alpha (r_c = \frac{c\Delta t}{\Delta x})$


NP = 4
Error
Grid Points
delay = 1
delay = 2
delay = 4
delay = 6
delay = 8
Order = 2


Delay = 4
Error
Grid points
Synchronous
Asynchronous
Order Recovery





Delay	Async	Changing Δt	AT scheme
0(sync)	-2.0195	-1.9008	-2.0195
1	-1.0764	-1.9712	-2.0050
2	-1.0371	-2.0048	-2.0049
4	-1.0117	-2.0313	-2.0044
6	-1.0033	-2.0488	-2.0039
8	-0.9995	-2.0609	-2.0029

Asynchronous Algorithm

- Deterministic Asynchronous Scheme
  - Error: Deterministic - Independent of runs
  - Exchange the information after a certain amount of steps. (SYNC_STEP)
  - Naive way of Implementation

Asynchronous Algorithm

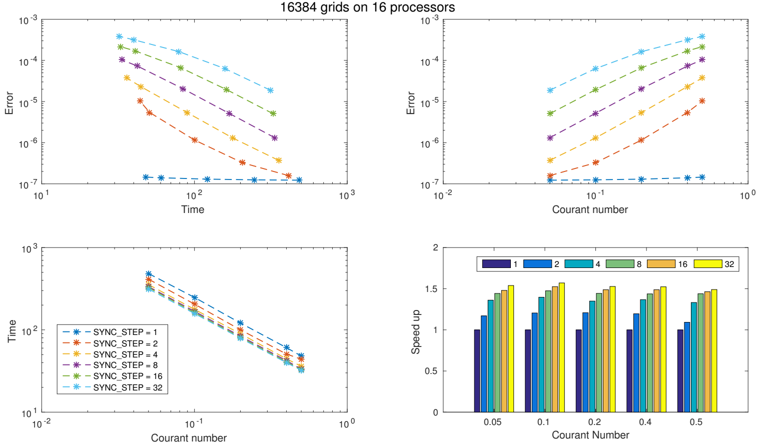
- Deterministic Asynchronous Scheme
  - Error: Deterministic - Independent of runs
  - Exchange the information after a certain amount of steps. (SYNC_STEP)
  - Naive way of Implementation
- Stochastic Asynchronous Scheme
  - Error: Different for different runs.
  - Do not wait for the communication to complete.
  - Use the latest time values.
  - Dependent on delay statistics.

Asynchronous Algorithm

- Deterministic Asynchronous Scheme
  - Error: Deterministic - Independent of runs
  - Exchange the information after a certain amount of steps. (SYNC_STEP)
  - Naive way of Implementation
- Stochastic Asynchronous Scheme
  - Error: Different for different runs.
  - Do not wait for the communication to complete.
  - Use the latest time values.
  - Dependent on delay statistics.
- Can in practice, be accomplished using MPI.
- Performance Matrix: Speedup

$$Speedup = \frac{\text{Time taken by Synchronous Scheme}}{\text{Time taken by scheme under relaxed synchronization}} \tag{8}$$

- Effect of Courant Number r_α:

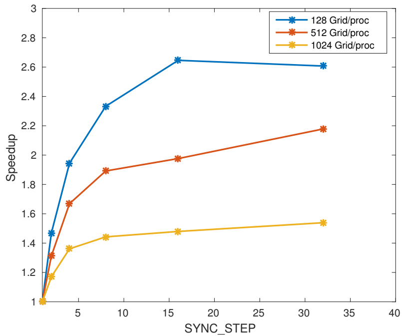


Speedup for Deterministic Case

- Computation Cost: X
- Synchronization cost/ Synchronization : Y
- Number of Time Steps: N_T
- Total Number of Synchronization: $\frac{N_T}{SYNC\_STEP}$
- Speedup = $\frac{X+N_T Y}{X+\frac{N_T}{SYNC\_STEP} Y}$

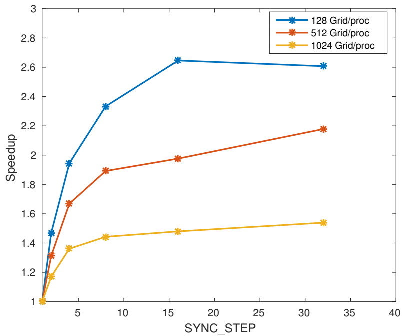
Speedup for Deterministic Case

- Computation Cost: X
- Synchronization cost/ Synchronization : Y
- Number of Time Steps: N_T
- Total Number of Synchronization: $\frac{N_T}{SYNC\_STEP}$
- Speedup = $\frac{X+N_T Y}{X+\frac{N_T}{SYNC\_STEP} Y}$



Speedup for Deterministic Case

- Computation Cost: X
- Synchronization cost/ Synchronization : Y
- Number of Time Steps: N_T
- Total Number of Synchronization: $\frac{N_T}{SYNC\_STEP}$
- Speedup = $\frac{X+N_T Y}{X+\frac{N_T}{SYNC\_STEP} Y}$



- Higher Value of SYNC_STEP ⇒ More Speedup
- More load on the processor ⇒ Less Speedup

Communication calls

- Two Sided Non - blocking Communication
  - Achieved using MPI_Isend / MPI_Irecv / MPI_Test
  - Data to be transferred is stored in the buffer.
  - Buffering of data takes place.
  - Handshaking occurs - Matching tags and rank.

Communication calls

- Two Sided Non - blocking Communication
  - Achieved using MPI_Isend / MPI_Irecv / MPI_Test
  - Data to be transferred is stored in the buffer.
  - Buffering of data takes place.
  - Handshaking occurs - Matching tags and rank.
- One Sided RMA
  - Achieved using MPI_Put / MPI_Lock / MPI_Unlock
  - Target processor exposes its location to memory.
  - No buffering - Source processor writes into the target location before returning.
  - Redundancy of data is prevented by MPI_Lock.
  - No handshaking - Consent of target processor is not required.

Communication calls

- Two Sided Non - blocking Communication
  - Achieved using MPI_Isend / MPI_Irecv / MPI_Test
  - Data to be transferred is stored in the buffer.
  - Buffering of data takes place.
  - Handshaking occurs - Matching tags and rank.
- One Sided RMA
  - Achieved using MPI_Put / MPI_Lock / MPI_Unlock
  - Target processor exposes its location to memory.
  - No buffering - Source processor writes into the target location before returning.
  - Redundancy of data is prevented by MPI_Lock.
  - No handshaking - Consent of target processor is not required.

Delay Statistics

The statistics of delay is measured in terms of k^* which is defined as:

$$k^* = \sum_{i=0}^{i=\infty} i * k_i \tag{9}$$

where k_i is defined as the ratio of the number of Time Steps that faced Delay = i and total number of Time Steps.

Two Sided Non - blocking: PE = 8

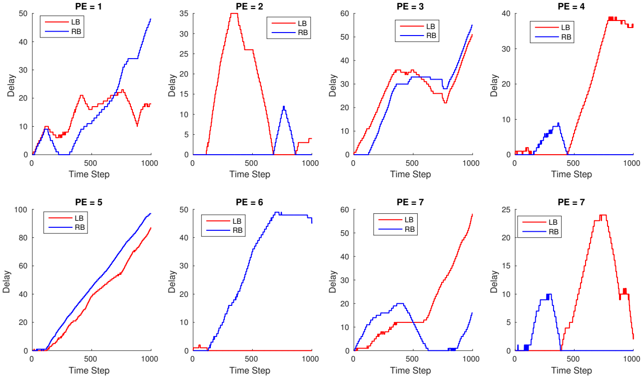


Figure : Evolution of Delay for 1024^3 grid for 1000 time steps. Each process has two boundary point. LB denotes the delay faced by left boundary point and RB the right boundary point

Two Sided Non - blocking: PE = 8

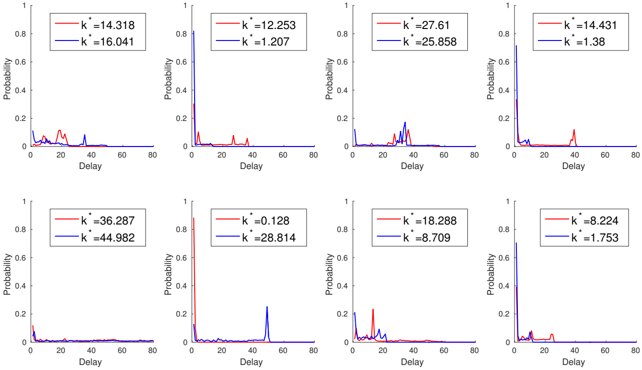


Figure : Delay Distribution for 1024^3 grid for 1000 time steps

One - Sided RMA: PE = 8

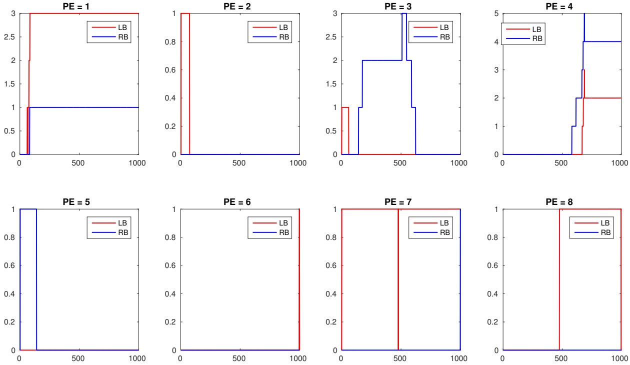


Figure : Evolution of Delay for 1024^3 grid for 1000 time steps. Each process has two boundary point. LB denotes the delay faced by left boundary point and RB the right boundary point

One - Sided RMA: PE = 8

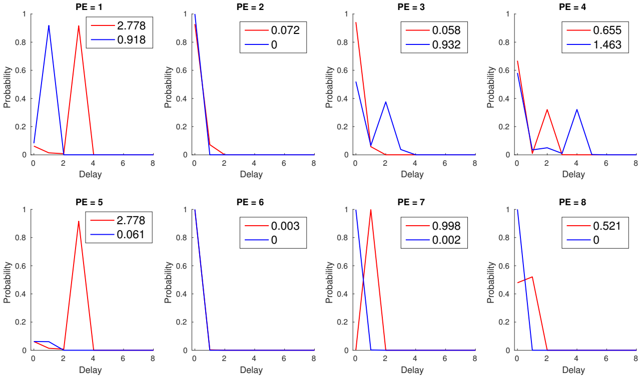


Figure : Delay Distribution for 1024^3 grid for 1000 time steps

Stochastic Asynchronous Scheme

- Requirements:
  - "Ghost" cells: value should be either old or new value
  - Buffer to store old Time Stamp Values.
  - Need timestamp information to be communicated along with data.
  - Need an error control knob.
  - $\langle \overline{E} \rangle \propto \tilde{\overline{k}}$
  - Enforce partial/total synchronization when k = L [5]

[5] L represents the maximum allowable delay.

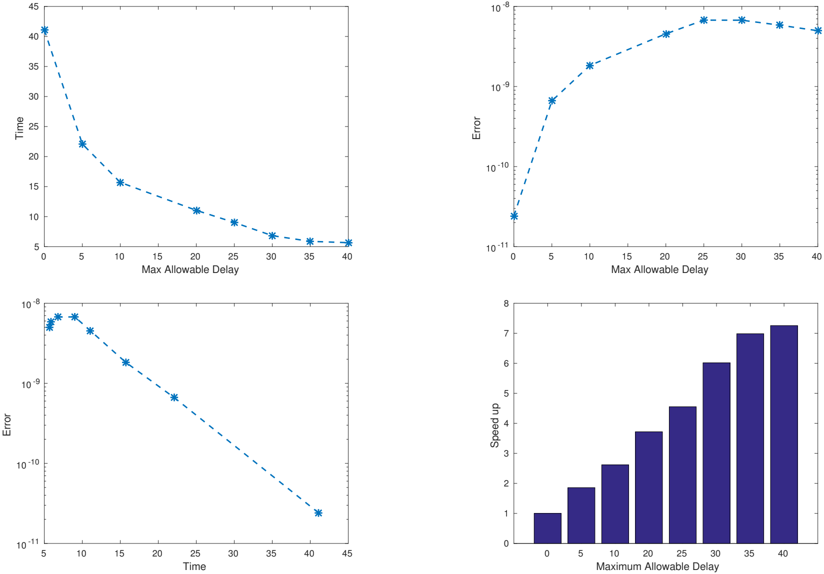
- Requirements:
  - "Ghost" cells: value should be either old or new value
  - Buffer to store old Time Stamp Values.
  - Need timestamp information to be communicated along with data.
  - Need an error control knob.
  - $\langle \overline{E} \rangle \propto \overline{\tilde{k}}$
  - Enforce partial/total synchronization when k = L [5]

Test Case

- Number of Grid Points = 65536.
- Number of Processors = 32
- Courant Number = 0.1
- Final t = 0.08*len/c
- Study effect of Maximum Allowable Delay on Error and Time

[5] L represents the maximum allowable delay.

Result of Stochastic Implementation



Comparison of Deterministic and Stochastic Asynchronous Algorithm

Deterministic Implementation with 2048 Grid Points per processor		Stochastic Implementation with 2048 Grid Points per processor	
SYNC_STEP	Speedup	Maximum Allowable Delay	Speedup
5	1.3615	5	1.8589
10	1.4474	10	2.6174
20	1.4889	20	3.7192
30	1.5300	30	6.0154

Table : Comparison of Deterministic and Stochastic Asynchronous Implementation

- Nodes
  - Called "host", "computer", "machine"
  - A certain amount of memory (RAM) is physically allocated on each node
  - Each node contains multiple sockets.
- Sockets
  - Collection of cores with a direct pipe to memory
  - Contains multiple cores.
- Cores
  - Single processing unit capable of performing computations.
  - Smallest unit of resource allocation.

Test Case

- Number of Grid Points = 1024.
- Number of Processors = 8
- Courant Number = 0.1
- Number of Time Steps = 150000
- Communication: RMA
- Study effect of Process Distribution on Various Nodes on Time and Delay Statistics

Test Case

- Number of Grid Points = 1024.
- Number of Processors = 8
- Courant Number = 0.1
- Number of Time Steps = 150000
- Communication: RMA
- Study effect of Process Distribution on Various Nodes on Time and Delay Statistics

Processor per node	Minimum Time	Maximum Time	Average Time	Maximum Delay	Average Delay
8	1.632993	1.898601	1.747674	26729	1152.118219
4	1.306848	2.515238	1.935257	63719	595.315498
2	2.310587	2.394653	2.357168	5074	23.086279
1	2.575294	2.636949	2.604869	1912	49.434085

Test Case

- Number of Grid Points = 268435456.
- Courant Number = 0.1
- Number of Time Steps = 500
- Study effect of Process Distribution on Various Nodes on Time and Delay Statistics

Test Case

- Number of Grid Points = 268435456.
- Courant Number = 0.1
- Number of Time Steps = 500
- Study effect of Process Distribution on Various Nodes on Time and Delay Statistics

- RMA operation

Processor per node	Minimum Time	Maximum Time	Average Time	Maximum Delay	Average Delay
8	196.022283	210.897438	204.288266	38	0.152375
4	157.093253	168.151056	162.511951	42	0.013750
2	112.703603	126.833151	122.102558	44	0.075125
1	91.371902	96.440170	93.667287	20	0.132000

- Two-sided Non blocking Send and Receive

Processor per node	Minimum Time	Maximum Time	Average Time	Maximum Delay	Average Delay
8	143.154700	158.406454	151.922679	65	10.7976
4	95.791654	137.700051	112.257639	123	20.1246
2	85.375067	141.165011	111.381462	137	17.8507
1	68.627403	105.315875	84.915095	110	13.0063

Observations

- Memory latency Vs Communication time.
- Distribution of Processor on different nodes $\Rightarrow$ Dependent on Memory Requirement.[6]
- Delay Statistics: Random
- Increasing load $\Rightarrow$ Lower value of Average Delays.
- RMA operation $\Rightarrow$ more favourable for Asynchronous Operation.

[6] MPI for Big Data: Dominique LaSalle, Parallel Computing, 2014

Conclusion

- Original Scheme:
  - Second Order Convergence without delays.
  - First Order Convergence in presence of delays.
- Second Order Convergence in presence of delays.
  - Reducing Δt
  - Newer Scheme - Asynchrony Tolerant Scheme
- Fewer Communication calls $\Rightarrow$ More speedup.
- Higher Load on processor $\Rightarrow$ Communication cost less significant.
- Stochastic Implementation - More Speedup.
- Study of Delay Statistics.
  - MPI_RMA: better for asynchronous case in terms of average delay statistics.
  - Dependence of load on the processor. More frequent communication $\Rightarrow$ Larger Delays.
  - Impact of Computer Architecture.