# Implementation of Asynchronous Scheme on Parallel Frameworks using MPI

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

[1] Department of Mathematics
IIT Madras

[2] Math CCES
RWTH Aachen

August 11, 2016

Acknowledgement: DAAD, RWTH IT Center

# Outline

# 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.

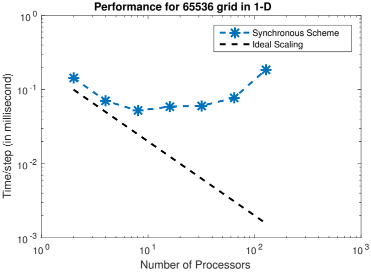
[1] Maitham Alhubail, Qiqi Wan. The swept rule for breaking the latency barrier in time advancing PDEs

- 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)

# Issues in Current Method

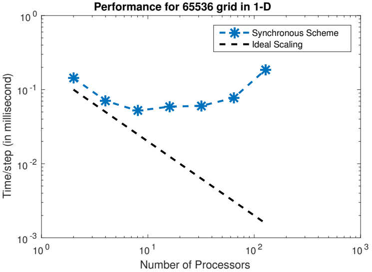
- Direct Numerical Simulation (DNS)
  - Resolve all scales in space and time.
  - Computationally very expensive.
  - Communication or synchronization takes upto 50 - 70 % of the total computation time.[2]




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

# Issues in Current Method

- 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)

**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 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:

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

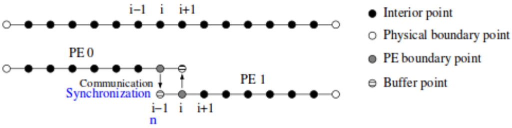
FD method can be used to discretize and solve PDEs.
Consider 1D advection - diffusion equation:

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

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.

# 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.

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

- Asynchronous Scheme - Stable and Accurate?

[3] Diego, A. Donzis and Konduri, Aditya, 2004 “Asynchronous Finite Difference Scheme for Partial Difference Equations”, Journal of Computational Physics. 274(0), pp: 370-392.

[4] At constant Courant Number: $r_\alpha$

- Asynchronous Scheme - Stable and Accurate?
- Stability
  - Stable if the Synchronous Scheme is stable, irrespective of delay statistics. [3]

[3] Diego, A. Donzis and Konduri, Aditya, 2004 "Asynchronous Finite Difference Scheme for Partial Difference Equations", Journal of Computational Physics. 274(0), pp: 370-392.

[4] At constant Courant Number: $r_\alpha$

- Asynchronous Scheme - Stable and Accurate?
- Stability
  - Stable if the Synchronous Scheme is stable, irrespective of delay statistics. [3]
- 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 [4]:

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

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

[3] Diego, A. Donzis and Konduri, Aditya, 2004 “Asynchronous Finite Difference Scheme for Partial Difference Equations”, Journal of Computational Physics. 274(0), pp: 370-392.

[4] At constant Courant Number: $r_\alpha$

- 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 |

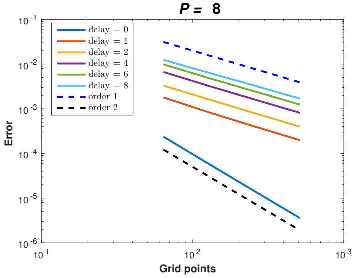



Figure : Strong Scaling

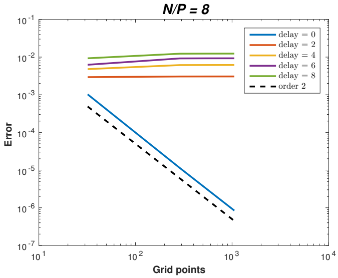



Figure : Weak Scaling

- Need for higher order schemes that are capable to maintain accuracy.
- Truncation Error Analysis.
- Previous work by Mudigree et al. [5], Donzis and Aditya[6] proposed such schemes.

[5] Mudigree, D., Sherleker,S., Ansumali, S., 2014 “Delayed Difference scheme for large scale scientific simulations”, Physical Review Letters, 113(21), 218701.

[6] Diego, A. Donzis and Konduri, Aditya, 2004 “Asynchronous Finite Difference Scheme for Partial Difference Equations”, Journal of Computational Physics. 274(0), pp: 370-392.

- Need for higher order schemes that are capable to maintain accuracy.
- Truncation Error Analysis.
- Previous work by Mudigree et al. [5], Donzis and Aditya[6] proposed such schemes.
- $\langle \overline{E} \rangle \propto \overline{\tilde{k}}$: Higher the delay, higher will be the error.

[5] Mudigree, D., Sherleker,S., Ansumali, S., 2014 “Delayed Difference scheme for large scale scientific simulations”, Physical Review Letters, 113(21), 218701.

[6] Diego, A. Donzis and Konduri, Aditya, 2004 “Asynchronous Finite Difference Scheme for Partial Difference Equations”, Journal of Computational Physics. 274(0), pp: 370-392.

# 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.

- 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

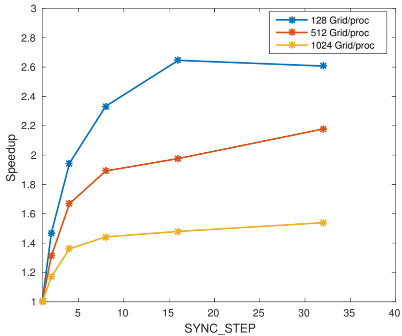
$$Speedup = \frac{\text{Time taken by Synchronous Scheme}}{\text{Time taken by scheme under relaxed synchronization}} \tag{7}$$

# 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 = $\frac{X+N_T Y}{X+\frac{N_T}{SYNC\_STEP} Y}$

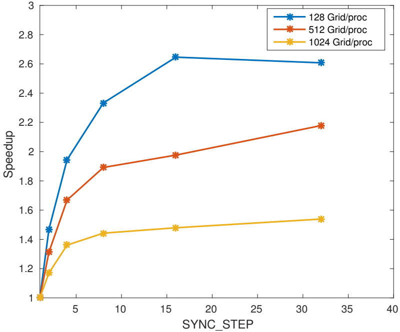
# Speedup for Deterministic Case

- 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:

$$k^* = \sum_{i=0}^{i=\infty} i * k_i \tag{8}$$

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

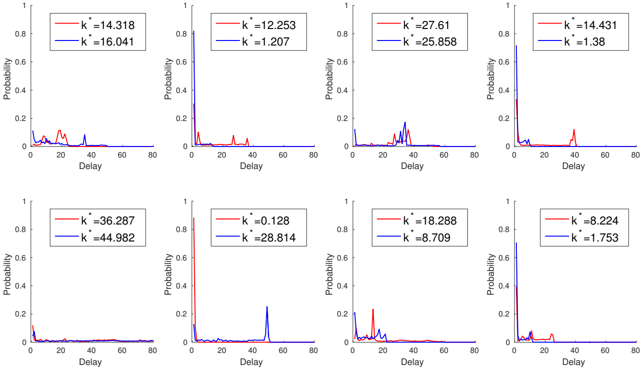



Figure : Delay Distribution for $1024^3$ grid for 1000 time steps

# One - Sided RMA: PE = 8

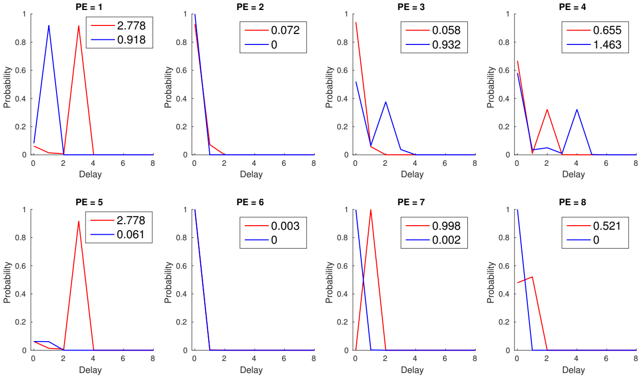



Figure : Delay Distribution for $1024^3$ grid for 1000 time steps

# 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 [7]

[7] 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.
  - $\langle \overline{E} \rangle \propto \tilde{\overline{k}}$
  - Enforce partial/total synchronization when k = L [7]

## Test Case

- Number of Grid Points = 65536.
- Number of Processors = 32
- Courant Number = 0.1
- Final t = 0.08*len/c

[7] L represents the maximum allowable delay.

# 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

## 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 | Time (s) | Maximum Delay | Average Delay |
|---|---|---|---|
| 8 | 1.747674 | 26729 | 1152.118219 |
| 4 | 1.935257 | 63719 | 595.315498 |
| 2 | 2.357168 | 5074 | 23.086279 |
| 1 | 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 | Time (s) | Maximum Delay | Average Delay |
|---|---|---|---|
| 8 | 204.288266 | 38 | 0.152375 |
| 4 | 162.511951 | 42 | 0.013750 |
| 2 | 122.102558 | 44 | 0.075125 |
| 1 | 93.667287 | 20 | 0.132000 |

- Two-sided Non blocking Send and Receive

| Processor per node | Time (s) | Maximum Delay | Average Delay |
|---|---|---|---|
| 8 | 151.922679 | 65 | 10.7976 |
| 4 | 112.257639 | 123 | 20.1246 |
| 2 | 111.381462 | 137 | 17.8507 |
| 1 | 84.915095 | 110 | 13.0063 |

# Observations

- Memory latency Vs Communication time.
- Distribution of Processor on different nodes $\Rightarrow$ Dependent on Memory Requirement.[8]
- Delay Statistics: Random
- Increasing load $\Rightarrow$ Lower value of Average Delays.
- RMA operation $\Rightarrow$ more favourable for Asynchronous Operation.

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

# Conclusion

- Original Scheme:
  - Second Order Convergence without delays.
  - First Order Convergence in presence of delays.
- Asynchrony Tolerant Scheme: Higher delay means higher error.
- 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.

*Thank You for your attention!!!*