An asynchronous computing method for solving PDEs at extreme scale

Konduri Aditya

Assistant Professor
Computational Flow Physics Laboratory
Department of Computational and Data Sciences
Indian Institute of Science, Bengaluru

Case study: ACM Winter School 2019 on High Performance Computing (HPC), IIT Kanpur

11 December 2019

Acknowledgements

- Preeti Malakar (IITK)
- Ashish Kuvelkar (C-DAC)

Collaborators:

- Diego A. Donzis (TAMU)
- Torsten Hoefler (ETH Zurich)
- Hemanth Kolla (Sandia)
- Jacqueline H. Chen (Sandia)

CDS

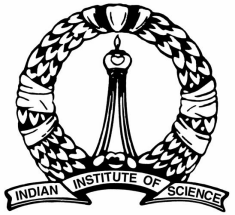
Department of Computational and Data Sciences

http://cds.iisc.ac.in

Computational and Data Sciences (CDS)

- Part of interdisciplinary research division of IISc
- Formed on December 7, 2015 (> 5 years into existence)
- All faculty/research labs/academic programs that were part of SERC were moved to CDS
- Located in SERC building
- Comes under faculty of engineering

Post Graduate Programs at CDS



- ➢Ph.D. – Doctoral program in Engineering
- ➢M.Tech. (Research)–Masters program with thesis
- ➢M. Tech (2 years): Masters program in Computational and Data Science

- **Faculty**
  - ❑Professors 5
  - ❑Associate Professors 5
  - ❑Assistant Professors 6

- **Major Research Funding**
  - ❑MHRD
  - ❑DST, DIT, DBT, CSIR
  - ❑MSR, IBM, Google, Yahoo, Boeing, TCS, AMD, Intel, …

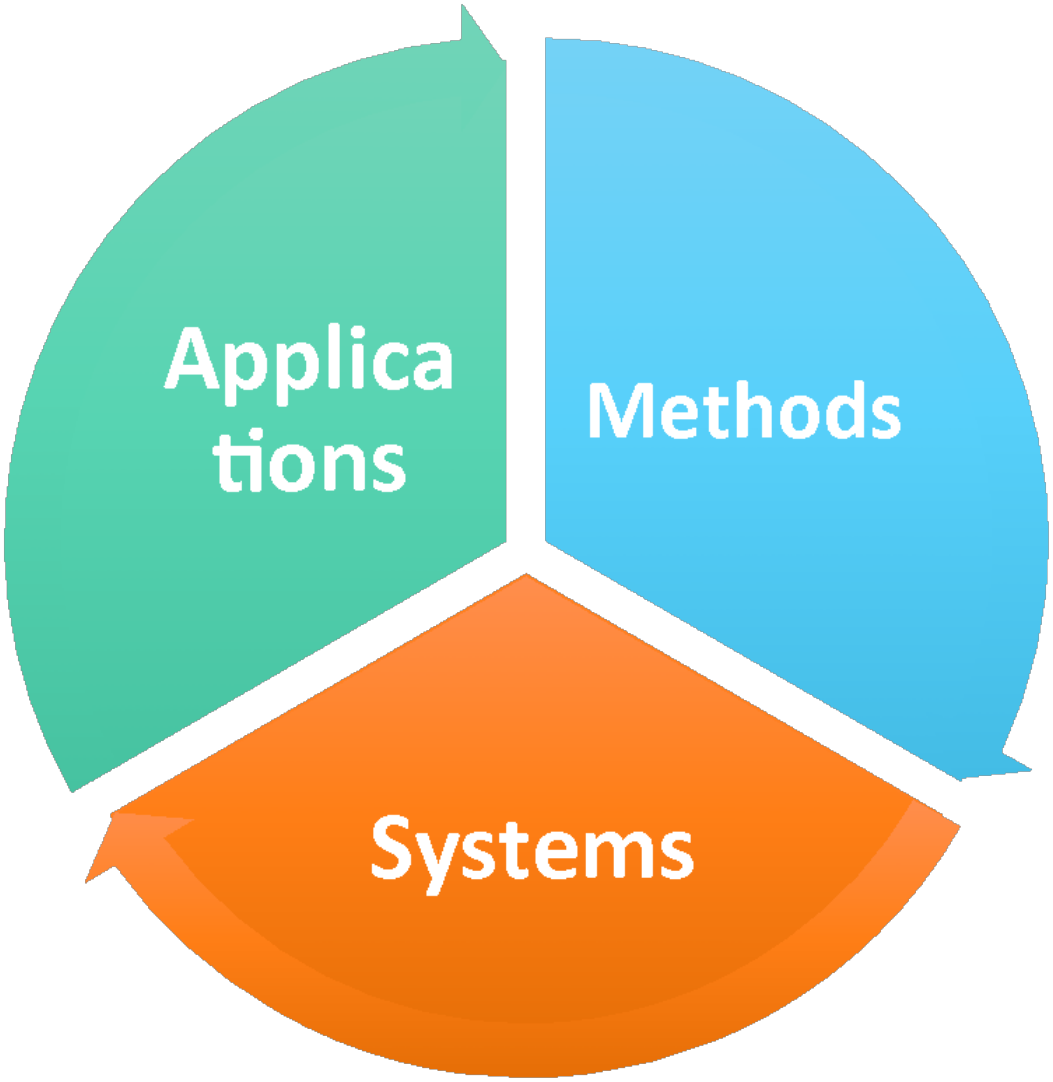
CDS

- M.Tech. (Computational and Data Science) – First program in data science from the premier institutes of India
- Several private scholarships
  - Narayana Nethrayala Research Award
  - Microsoft Data Science Fellowships for M.Tech. (Research)
  - Intuit M.Tech. (Research) Fellowship
  - Cargill M.Tech. Fellowships
  - Target India M.Tech. Fellowship
  - TESCO Technology M.Tech. Fellowship
  - GE Healthcare M.Tech. Fellowship
  - Ericsson M.Tech. Fellowships

http://cds.iisc.ac.in/resources/fellowships/

CDS
Department of Computational and Data Sciences

CDS Research



Research Areas @ CDS

Department of Computational and Data Sciences

Computer&Data Systems

- CAD for VLSI
- Grid Application Research
- Cloud and Distributed Computing
- Machine Learning
- Database Systems
- Video Analytics
- Computer Vision

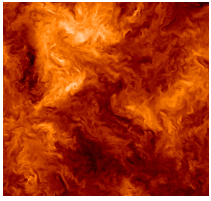
Computational Science

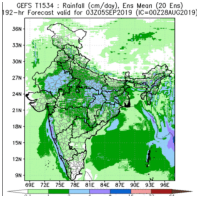
- Computational Electromagnetics
- Computational Photonics
- Medical Imaging
- Scientific Computing and Mathematical Libraries
- Computational Fluid Dynamics
- Computational Biology and Bioinformatics
- Numerical Linear Algebra

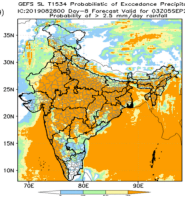
Outline

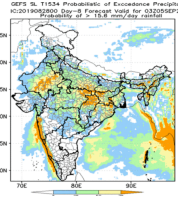
Outline

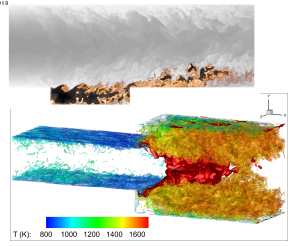
Scientific computing



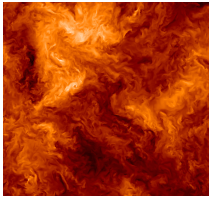


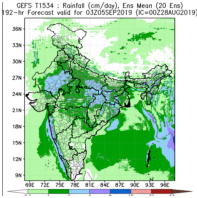


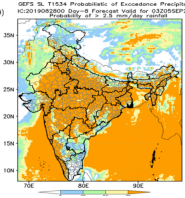


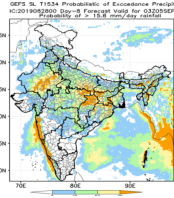


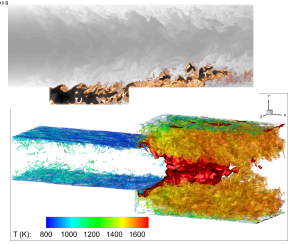
Scientific computing





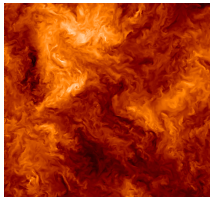


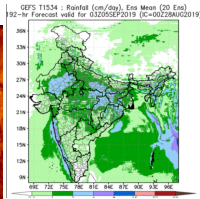


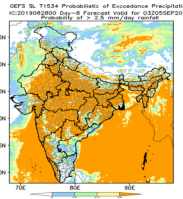


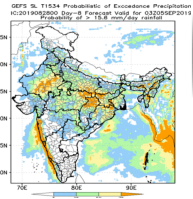
Governing equations → Numerical method → Simulations → Data analysis

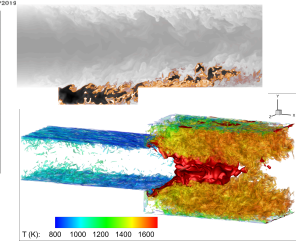
Scientific computing









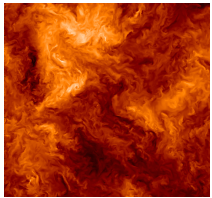


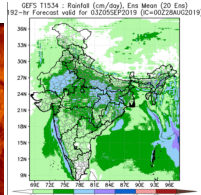
Governing equations → Numerical method → Simulations → Data analysis

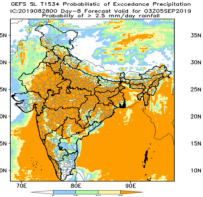
HPC

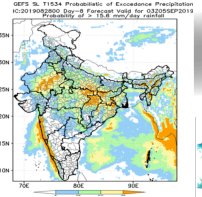
ML / AI

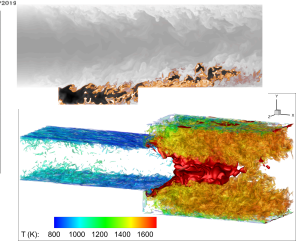
Scientific computing











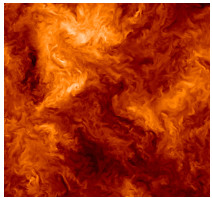
Governing equations → Numerical method → Simulations → Data analysis

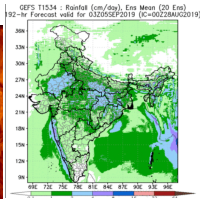
High resolution
~billions of grid points

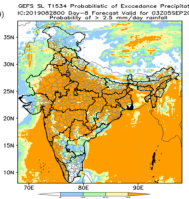
HPC

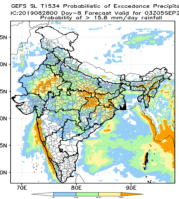
ML / AI

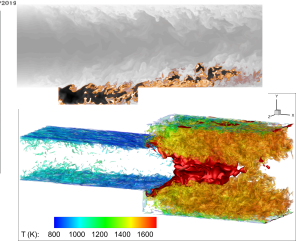
Scientific computing











Governing equations → Numerical method → Simulations → Data analysis

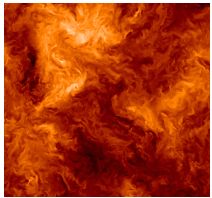
Large volume ~TB/PB

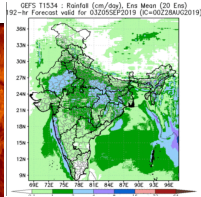
High resolution ~billions of grid points

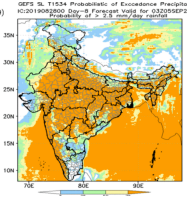
HPC

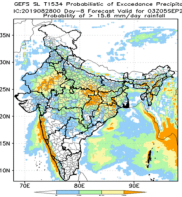
ML / AI

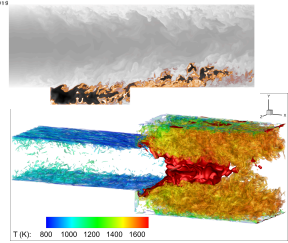
Scientific computing











Governing equations → Numerical method → Simulations → Data analysis

High resolution
~billions of grid points

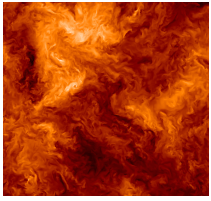
Large volume
~TB/PB

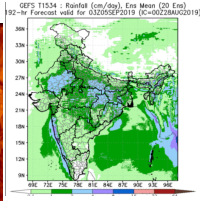
- Detect pattern, anomalies
- Create sub-grid model

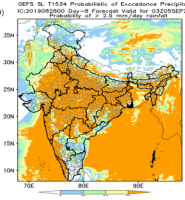
HPC

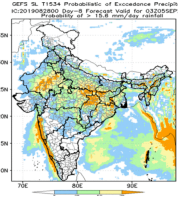
ML / AI

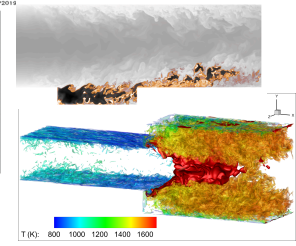
Scientific computing











Governing equations → Numerical method → Simulations → Data analysis

High resolution
~billions of grid points

Large volume
~TB/PB

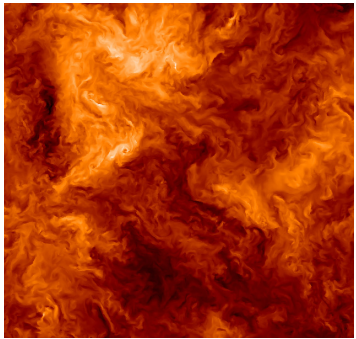
- Detect pattern, anomalies
- Create sub-grid model

HPC

ML / AI

Distributed data

Turbulence



A. F. Maqui & D. A. Donzis (2011)

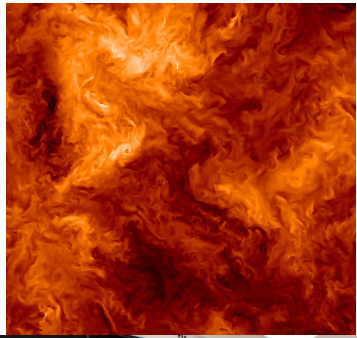
- Direct Numerical Simulations: resolve all scales both in time and space
  - Large scale (L)
  - Kolmogorov small scale (η)
  - Using classical scaling (Kolmogorov 1941)

$$\frac{L}{\eta} \sim Re^{3/4}$$

  where $Re = uL/\nu$ is the Reynolds number
- Typically very high in applications
- Computational effort (C)

$$C \sim Re^3$$

Turbulence



A. F. Maqui & D. A. Donzis (2011)

- Direct Numerical Simulations: resolve all scales both in time and space
  - Large scale (L)
  - Kolmogorov small scale (η)
  - Using classical scaling (Kolmogorov 1941)

$$\frac{L}{\eta} \sim Re^{3/4}$$

  where $Re = uL/\nu$ is the Reynolds number
- Typically very high in applications
- Computational effort (C)

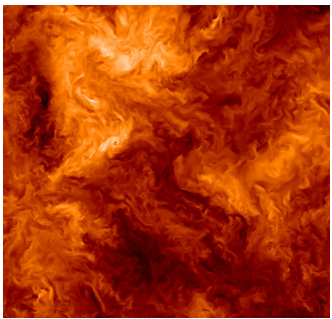
$$C \sim Re^3$$

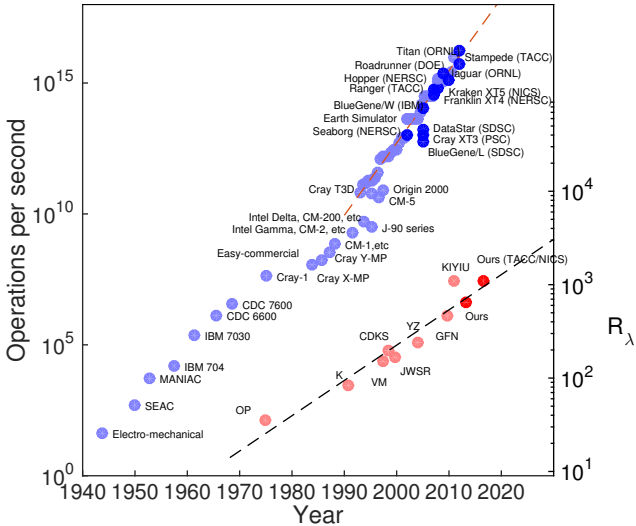
- Parallel computing

Turbulence simulations

Isotropic turbulence

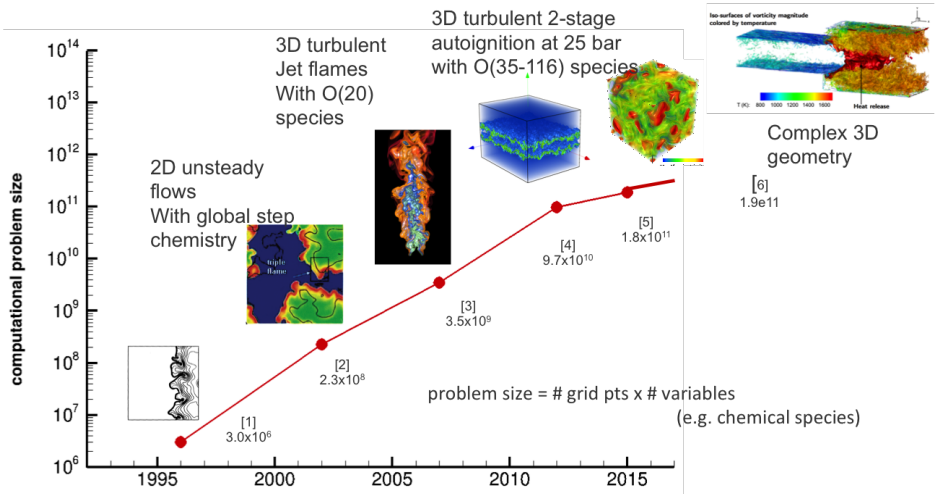
- Direct numerical simulations
  - Pseudo-spectral method
  - Runge-Kutta





Turbulent combustion simulations (S3D code)

Direct Numerical Simulations



Outline

Governing equations

$$\frac{\partial \rho}{\partial t} = -\nabla_\beta \cdot (\rho \mathbf{u}_\beta),$$

$$\frac{\partial (\rho \mathbf{u}_\alpha)}{\partial t} = -\nabla_\beta \cdot (\rho \mathbf{u}_\alpha \mathbf{u}_\beta) + \nabla_\beta \cdot \boldsymbol{\tau}_{\beta\alpha} - \nabla_\alpha p + \rho \sum_{i=1}^{N_s} Y_i \mathbf{f}_{\alpha i},$$

$$\frac{\partial (\rho e_0)}{\partial t} = -\nabla_\beta \cdot \left[\mathbf{u}_\beta (\rho e_0 + p)\right] + \nabla_\beta \cdot (\tau_{\beta\alpha} \cdot \mathbf{u}_\alpha) - \nabla_\beta \cdot \mathbf{q}_\beta + \rho \sum_{i=1}^{N_s} Y_i \mathbf{f}_{\alpha i} \cdot (\mathbf{V}_{\alpha i} + \mathbf{u}_\alpha),$$

$$\frac{\partial (\rho Y_i)}{\partial t} = -\nabla_\beta \cdot (\rho Y_i \mathbf{u}_\beta) - \nabla_\beta \cdot (\rho Y_i \mathbf{V}_{\beta i}) + W_i \dot{\omega}_i,$$

Solution methods

- **Finite difference**
- Finite volume
- Finite element
- Spectral element
- Pseudo-spectral

Procedure:

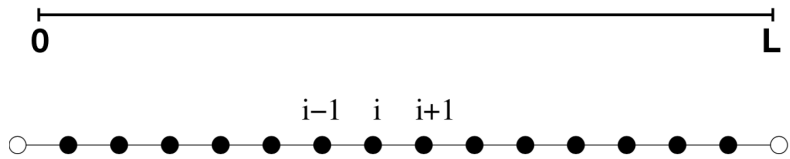
- Approximation of equations
- Discretize domain
- Initial/boundary conditions
- Algorithm

Finite difference method

Consider the simple 1D heat equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

Here $u = u(x, t)$



For discrete domain $u = u(x_i, t_n)$

Finite difference method

Approximations:

- Time derivative: first order accurate forward difference

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t} + \mathcal{O}(\Delta t)$$

- Space derivative: second order accurate central difference

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$

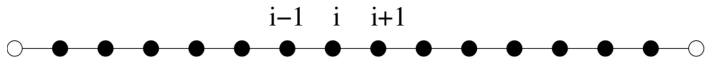
$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$

Finite difference method

Explicit scheme:

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

- Can be written in vector form as $U^{n+1} = f(U^n)$



- Pseudo code

$for(n = 0; n < NT; n++)\{$
$for(i = 0; i < NX; i++)\{$
$u[i][n+1] = a[0] * u[i+1][n] + a[1] * u[i][n] + a[2] * u[i-1][n]$
$\}$
$\}$

Numerical properties

- Stability
- Consistency
- Accuracy

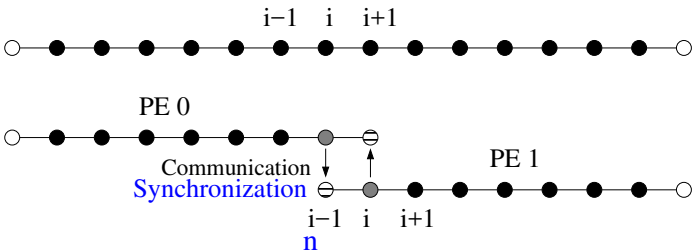
Parameters:

- Grid resolution (Δx, N)
- Nature of PDE (hyperbolic, parabolic, elliptic)
- Non-dimensional parameters (CFL, Reynolds number, etc.)

Parallel implementation

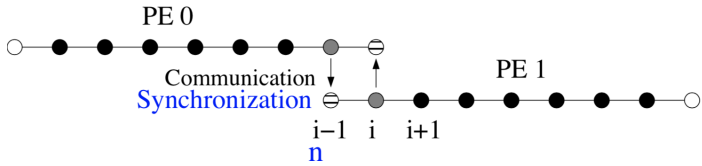
- Data parallelism: domain decomposition
- FD equation - forward in time and central in space

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



- Interior point
- Physical boundary point
- PE boundary point
- Buffer point

Parallel implementation



Point to point communication

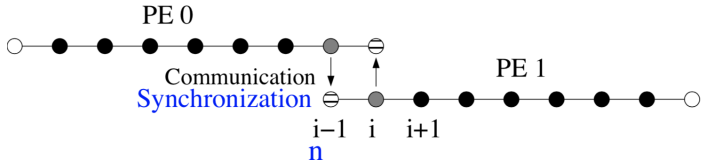
- MPI_Send - MPI_Recv
- MPI_SendRecv

Example:

1. Communicate data: MPI_Isend - MPI_Irecv CPUs remain idle
2. Compute solution at all points

These are blocking communications, cannot achieve computation-communication overlap

Asynchronous (non-blocking) communication



Point to point communication

- MPI_Isend - MPI_Irecv - MPI_Wait

Example:

1. Start communication by posting MPI_Isend - MPI_Irecv
2. Compute solution at interior points
3. Synchronize communication using MPI_Wait
4. Computer solution at PE boundary points

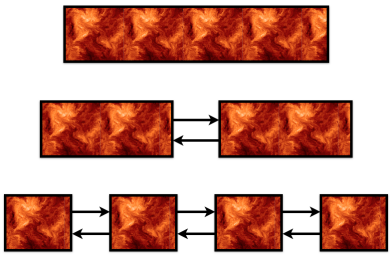
Asynchronous communication

Point to point communication

- MPI_Isend - MPI_Irecv - MPI_Wait
  - two-sided communications
  - MPI functions called by both source and destination
- MPI_Put, MPI_Get
  - one-sided communications
  - MPI function is called by either source or destination

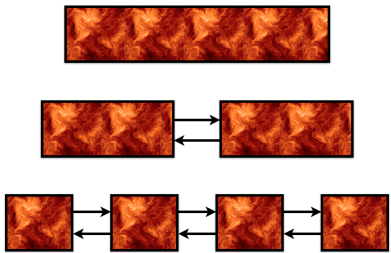
Parallel computing

- Distributed memory - domain decomposition/data parallelism



Parallel computing

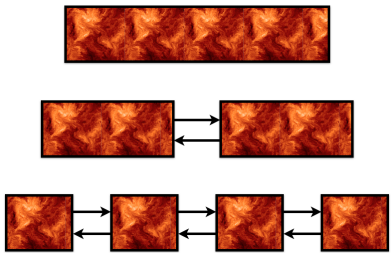
- Distributed memory - domain decomposition/data parallelism

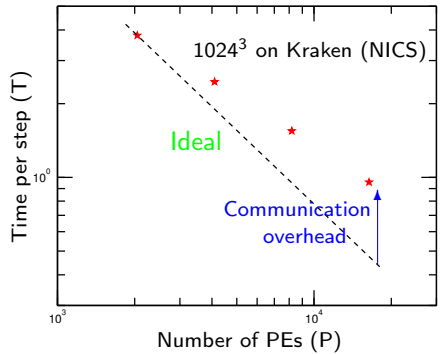


Ideal scaling:
Number of processing elements (P)
Computational effort (C)
Total computing time $T \sim C/P$

Parallel computing

- Distributed memory - domain decomposition/data parallelism

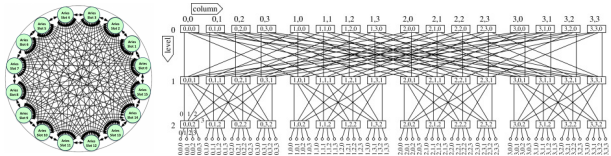




Ideal scaling:
Number of processing elements (P)
Computational effort (C)
Total computing time $T \sim C/P$

Towards Exascale computing: issues

Millions of processing elements (CPU/GPU/FPGA), complex network interconnect



- Large number of communications
  - increase communication time: affects scalability
- Performance variations in processing elements
  - potential issues: noise, inadequate cooling
- Synchronizations enforced at multiple levels (communication, mathematics, etc.)
- System faults
  - failure of a compute node

J. Dongarra et al. (IJHPCA 2011)

Towards Exascale computing: solutions

Parameters:

- Minimize data movement
- Reduce synchronization
- Express data locality and independence

Towards Exascale computing: solutions

Parameters:

- Minimize data movement
- Reduce synchronization
- Express data locality and independence

Research:

- Software level
  - Asynchronous runtime systems
  - Asynchronous communication/programming models

Towards Exascale computing: solutions

Parameters:

- Minimize data movement
- Reduce synchronization
- Express data locality and independence

Research:

- Software level
  - Asynchronous runtime systems
  - Asynchronous communication/programming models
- Mathematical level
  - Asynchronous numerical methods

Towards Exascale computing: solutions

Parameters:

- Minimize data movement
- Reduce synchronization
- Express data locality and independence

Research:

- Software level
  - Asynchronous runtime systems
  - Asynchronous communication/programming models
- Mathematical level
  - Asynchronous numerical methods
- Co-design (integrated solution)
  - Asynchronous computing method

Outline

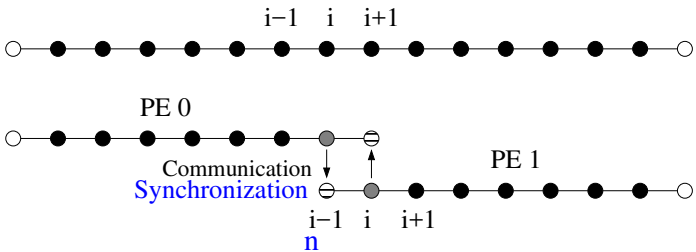
Finite difference method

Consider the simple 1D heat equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

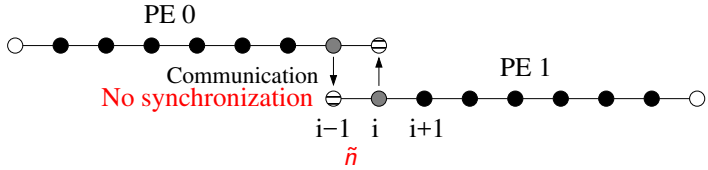
FD equation - forward in time and central in space

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

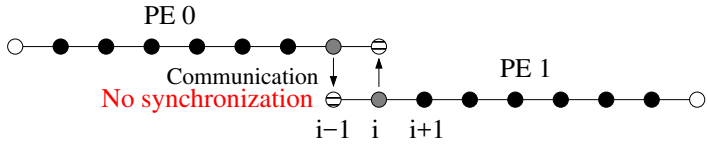


- Interior point
- Physical boundary point
- PE boundary point
- Buffer point

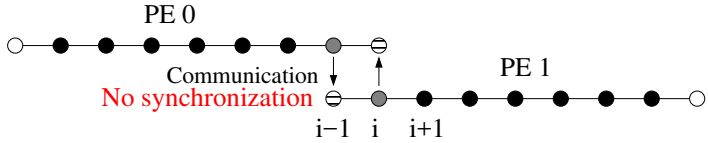
Asynchronous computing



Asynchronous computing



Asynchronous computing



- $u_{i-1}^{\tilde{n}}$ is the available data at $(i-1)$th grid point
- $\tilde{n}$ could be n, $(n-1)$, $(n-2)$, etc. time levels

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^{\tilde{n}}}{\Delta x^2} + \mathcal{O}(\Delta t^?, \Delta x^?)$$

- Interior points used in computations are at nth level
- Buffer points are at $\tilde{n} = n - \tilde{k}$ th level
- What are the properties of such a scheme?

Parameters

Numerical properties depend on

- Grid resolution (N)

Parameters

Numerical properties depend on

- Grid resolution (N)
- Stencil size (S)
- Number of PE (P)
- Number of delay levels (L)
- Probability of level k (p_k)
  - Communication performance: Network latency, bandwidth; message characteristics; MPI implementation; ETC.

$\tilde{n} = n - \tilde{k}$ is a random variable at PE boundaries

Parameters

Numerical properties depend on

- Grid resolution (N)
- Stencil size (S)
- Number of PE (P)
- Number of delay levels (L)
- Probability of level k (p_k)
  - Communication performance: Network latency, bandwidth; message characteristics; MPI implementation; ETC.

$\tilde{n} = n - \tilde{k}$ is a random variable at PE boundaries

Example: $L = 2$, $\tilde{k} = \{0, 1\}$
If $p_0 = 0.6$, then
$p_1 = 1 - p_0 = 0.4$

Numerical experiments

Advection-diffusion equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^{\tilde{n}} - u_{i-1}^{\tilde{n}}}{2\Delta x} = \alpha\frac{u_{i+1}^{\tilde{n}} - 2u_i^n + u_{i-1}^{\tilde{n}}}{\Delta x^2}$$

Initial condition

$$u(x,t) = \sum_\kappa A(\kappa)\sin(\kappa x + \phi_\kappa)$$

- Multi-scale with given spectrum
- Random phases: averages; avoid very special circumstances

Analytical solution

$$u_a(x,t) = \sum_\kappa e^{-\alpha\kappa^2 t} A(\kappa)\sin(\kappa x + \phi_\kappa - ct)$$

Boundary condition: periodic

Numerical experiments

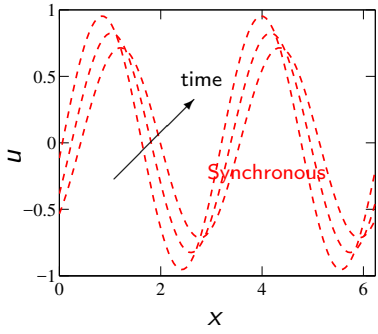
Finite difference equation (at left boundary of each PE)

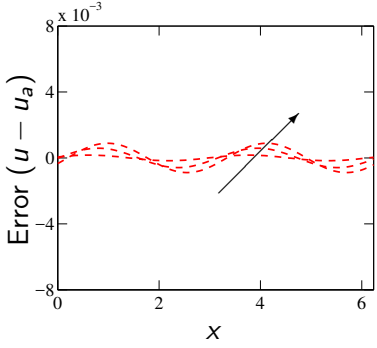
$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_{i-1}^{\tilde{n}}}{2\Delta x} = \alpha\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^{\tilde{n}}}{\Delta x^2}$$

- $\tilde{n} = n - \tilde{k}$
- $\tilde{k}$: time level delay
  - location and time dependent
- Delays simulated using random number generator
  - probability of k is p_k
- Time step $\Delta t = r_\alpha \Delta x^2/\nu$
  - stability considerations (synchronous)

Numerical experiments

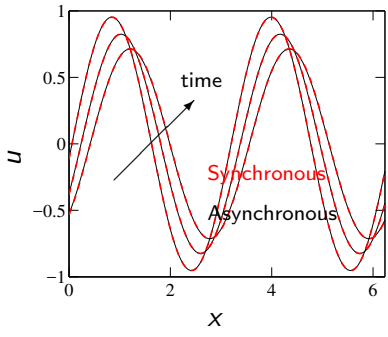
- Time evolution of solution and error

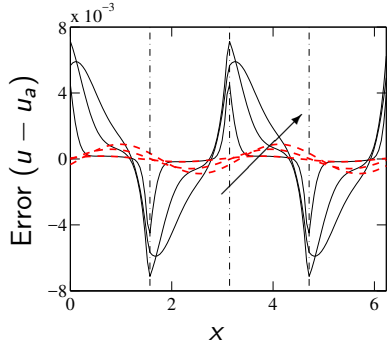




Numerical experiments

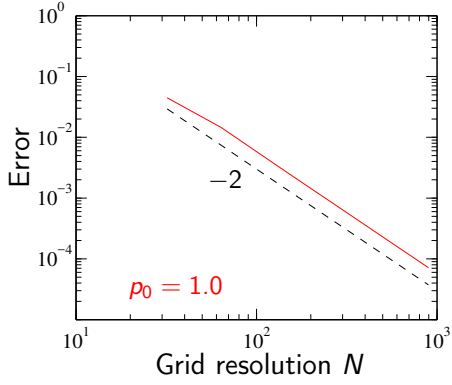
- Time evolution of solution and error





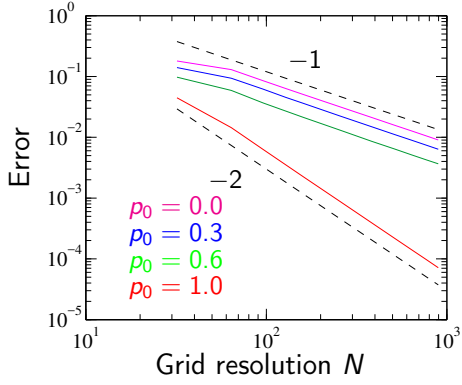
Parameters: $P = 4,\ L = 2,\ p_k = \{0.0, 1.0\}$

Accuracy



[1]K. Aditya and D. A. Donzis (SC 2012), [2]D. A. Donzis and K. Aditya (JCP 2014)

Accuracy



- Solution is stable
- Accuracy drops to first order with asynchrony

[1]K. Aditya and D. A. Donzis (SC 2012), [2]D. A. Donzis and K. Aditya (JCP 2014)

Numerical properties

Feasibility of asynchronous schemes: stability, consistency, accuracy

- Need new framework to analyze asynchronous schemes
- Stability
  - stable if synchronous scheme is also stable[2]
- Consistency
  - can be readily shown[2]
- Accuracy
  - strongly affected[1,2]
  - not much known about:
    - higher order schemes
    - spectral resolution
    - error propagation

[1]K. Aditya and D. A. Donzis (SC 2012), [2]D. A. Donzis and K. Aditya (JCP 2014)

Accuracy

- Using Taylor series

$\tilde{\mathbf{n}} = \mathbf{n}$

$$E_i^n = -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 + \mathcal{O}(\Delta t^2, \Delta x^4).$$

Accuracy

- Using Taylor series

$\tilde{\mathbf{n}} = \mathbf{n}$

$$E_i^n = -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 + \mathcal{O}(\Delta t^2, \Delta x^4).$$

$\tilde{\mathbf{n}} = \mathbf{n} - \mathbf{1}$

$$\begin{aligned}\tilde{E}_i^n|_1 &= -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 - \alpha\dot{u}\frac{\Delta t}{\Delta x^2} + \alpha\dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha\dot{u}''}{2}\Delta t \\ &\quad + \mathcal{O}(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q)\end{aligned}$$

Accuracy

- Using Taylor series

$\tilde{\mathbf{n}} = \mathbf{n}$

$$E_i^n = -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 + \mathcal{O}(\Delta t^2, \Delta x^4).$$

$\tilde{\mathbf{n}} = \mathbf{n} - \mathbf{1}$

$$\begin{aligned}\tilde{E}_i^n|_1 &= -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 - \alpha\dot{u}\frac{\Delta t}{\Delta x^2} + \alpha\dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha\dot{u}''}{2}\Delta t \\ &\quad + \mathcal{O}(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q)\end{aligned}$$

$\tilde{\mathbf{n}} = \mathbf{n} - \tilde{\mathbf{k}}$

$$\begin{aligned}\tilde{E}_i^n|_k &= -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 - \alpha k\dot{u}\frac{\Delta t}{\Delta x^2} + \alpha k\dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha k\dot{u}''}{2}\Delta t \\ &\quad + \mathcal{O}(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q),\end{aligned}$$

Delay k appears as prefactor, does not affect order with asynchrony

Accuracy

- Overall error in the domain

$$\langle \overline{E} \rangle = \frac{1}{N} \sum_{i=1,N} \overline{E_i^n}.$$

$$\langle \overline{E} \rangle = \frac{1}{N} \left[\sum_{i \in I_I} E_i^n + \sum_{i \in I_B} \overline{\tilde{E}_i^n |_{\tilde{k}}} \right]$$

Interior points (N_I) PE boundary points (N_B)

Accuracy

- Overall error in the domain

$$\langle \overline{E} \rangle = \frac{1}{N} \sum_{i=1,N} \overline{E_i^n}.$$

$$\langle \overline{E} \rangle = \frac{1}{N} \left[\sum_{i \in I_I} E_i^n + \sum_{i \in I_B} \overline{\tilde{E}_i^n |_{\tilde{k}}} \right]$$

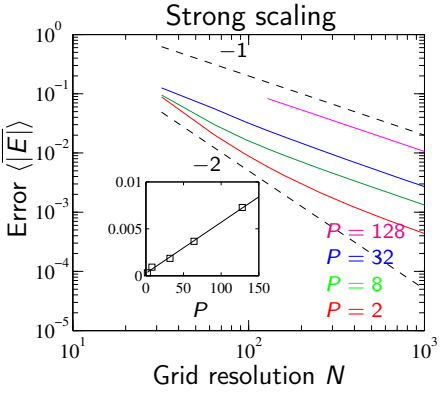
Interior points (N_I) PE boundary points (N_B)

- When error due to asynchrony dominates, it scales as:

$$\langle \overline{E} \rangle \sim \overline{\tilde{k}} \frac{P}{N} \sim \overline{\tilde{k}} P \Delta x$$

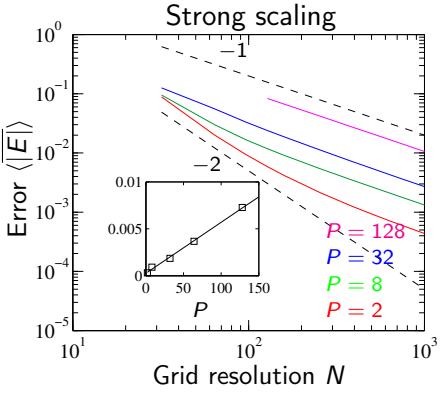
- Strong scaling: $\langle \overline{E} \rangle \sim \mathcal{O}(\Delta x)$
- Weak scaling $\langle \overline{E} \rangle \sim \mathcal{O}(1)$
- Independent of accuracy of original scheme: drastic effect of asynchrony
- Validation of this theoretical framework: **D. A. Donzis and K. Aditya (JCP 2014)**

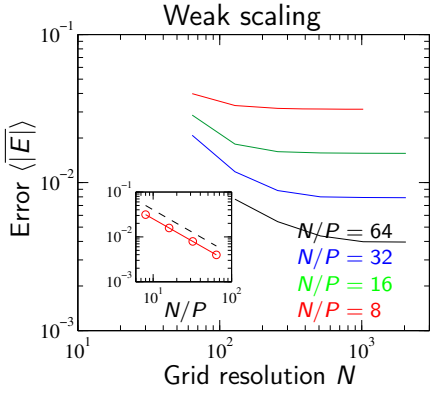
Accuracy - effect of P



$$\langle|\overline{E}|\rangle \sim \frac{P}{N}$$

Accuracy - effect of P





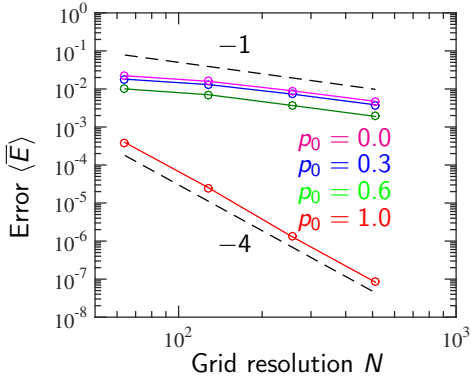
$$\langle|\overline{E}|\rangle \sim \frac{P}{N}$$

Agreement between theory and numerical experiments

- Zeroth-order accuracy on weak scaling

Accuracy

- Results from fourth order accurate simulations

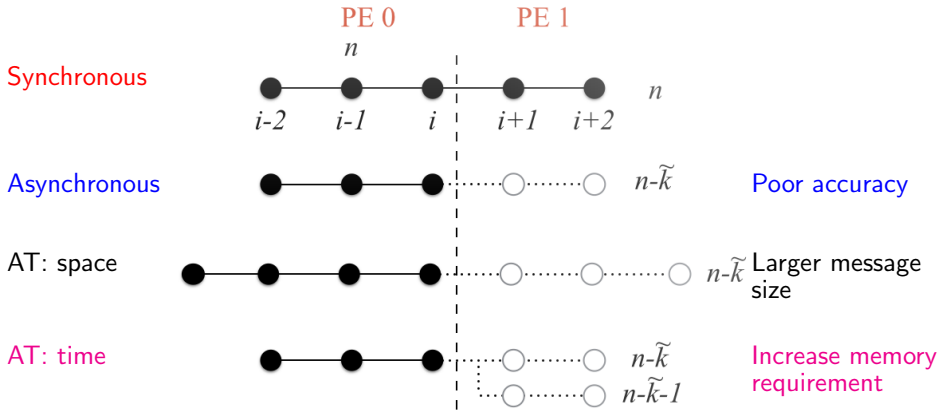


$$\langle \overline{\overline{E}} \rangle \sim \overline{\overline{\tilde{k}}} \frac{P}{N} \sim \overline{\overline{\tilde{k}}} P \Delta x$$

Asynchrony-tolerant schemes

Asynchrony-tolerant schemes

Stencil structure:

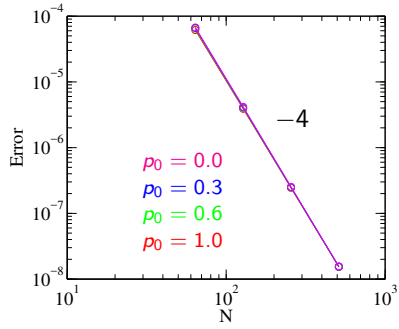


Asynchrony-tolerant schemes

Time: Runge-Kutta second order, space: fourth order AT scheme

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{24\Delta x^2}\Big[(k+1)(k+2)(-u_{i-2}^{n-k}+16u_{i-1}^{n-k}-30u_i^n+16u_{i+1}^n-u_{i+2}^n) - k(k+2)(-u_{i-2}^{n-k-1}+16u_{i-1}^{n-k-1}-30u_i^n+16u_{i+1}^n-u_{i+2}^n) + k(k+1)(-u_{i-2}^{n-k-2}+16u_{i-1}^{n-k-2}-30u_i^n+16u_{i+1}^n-u_{i+2}^n)\Big]$$

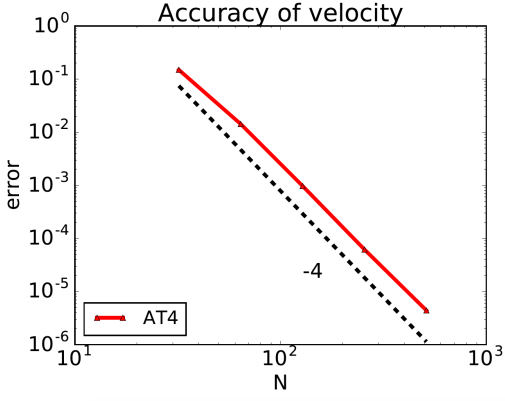
$$\langle \overline{E} \rangle \sim P\Delta x^{a+1} \sum_{m=1}^{\mathcal{N}} \gamma_m \overline{\tilde{k}^m}$$

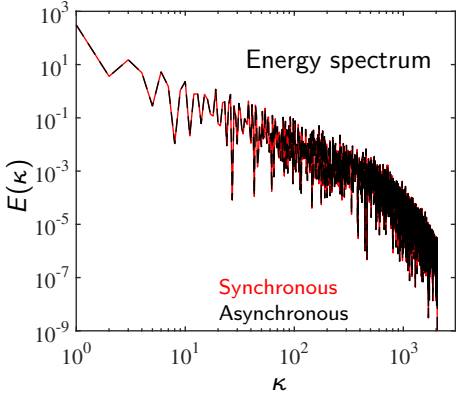


K. Aditya and D. A. Donzis (JCP 2017)

Reacting flows

- Simulation of 1D premixed auto-ignition
- Fourth order central difference and asynchrony-tolerant schemes





Outline

Asynchronous algorithm

Consider for example:

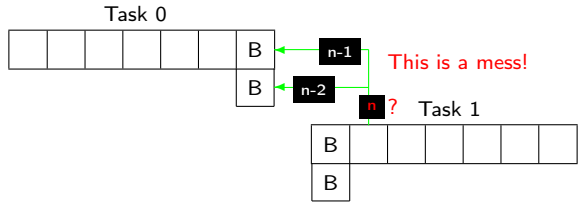
$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{2\Delta x^2}\Big[&u_{i+3}^n - 2u_{i+2}^n + u_{i+1}^n \\ &+ \left(\tilde{k}+1\right)\left(u_{i-1}^{n-\tilde{k}} - 2u_{i-2}^{n-\tilde{k}} + u_{i-3}^{n-\tilde{k}}\right) \\ &- \tilde{k}\left(u_{i-1}^{n-\tilde{k}-1} - 2u_{i-2}^{n-\tilde{k}-1} + u_{i-3}^{n-\tilde{k}-1}\right)\Big]\end{aligned}$$

What is it required to implement such a scheme efficiently?

- Communication: non-blocking, synchronization only when $\tilde{k} > L$
- Store data from multiple time levels
- Knowledge of time level ($\tilde{k}$) of each message
- Atomicity while accessing values

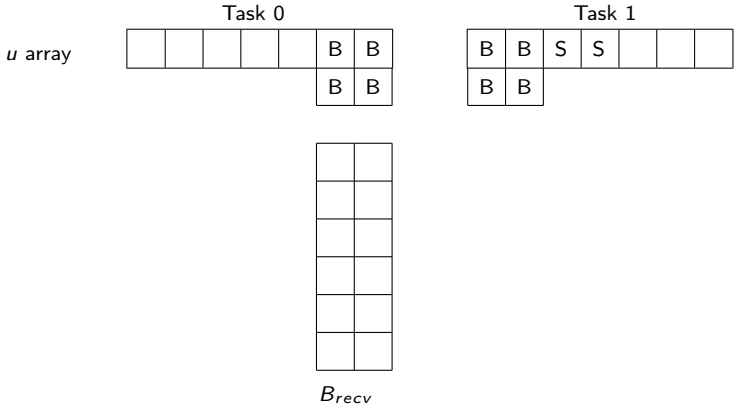
Asynchronous communications

- Traditional setting: MPI_Isend/Irecv
- Use MPI_Tag for time stamp $(n - \tilde{k})$

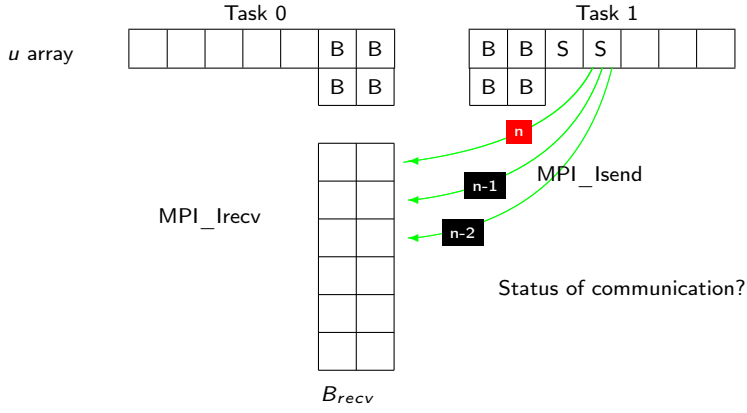


- Messages can arrive in any order
- Very difficult to keep track of time level delay $(\tilde{k})$

Asynchronous communications

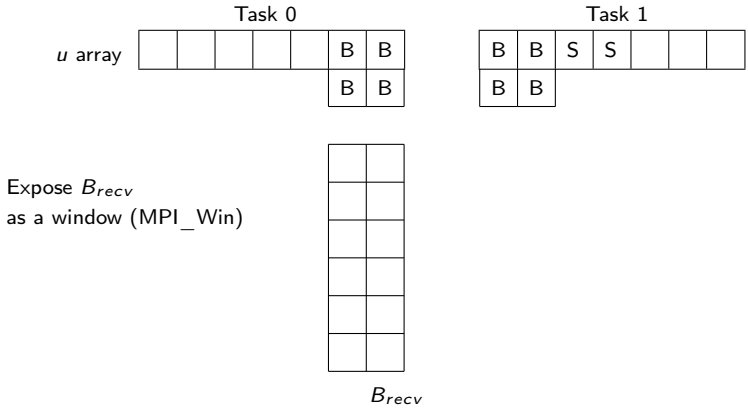


Asynchronous communications



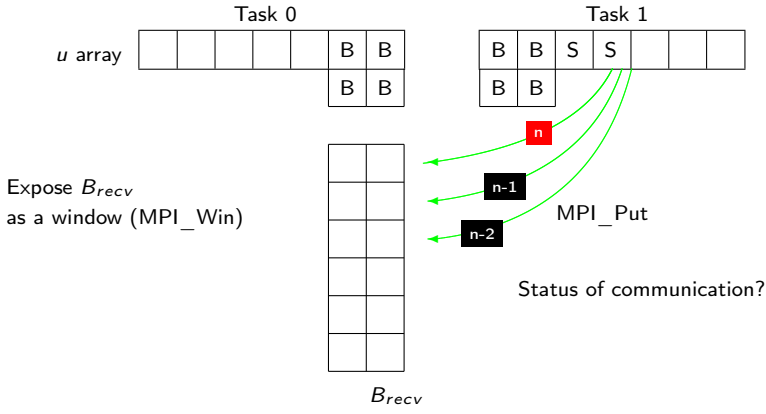
- Use MPI_Test to check the status of communication
- Use MPI_Wait to synchronize communication
- Obtain time level (k) using MPI_Tag

Asynchronous communications



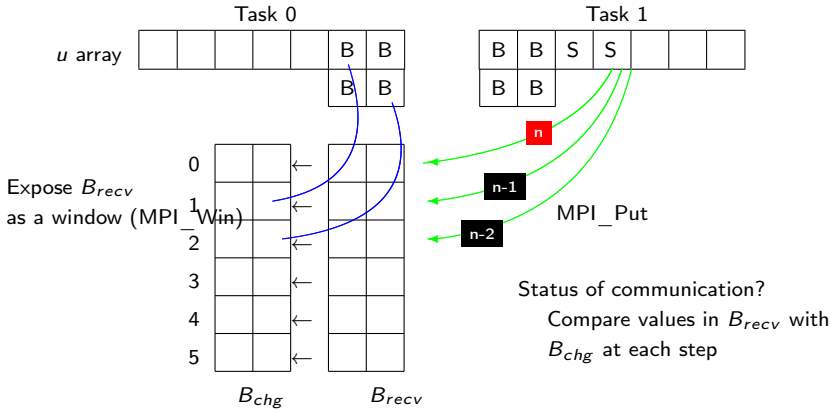
- Only source PE is involved
- Atomicity of data is critical

Asynchronous communications



- Only source PE is involved
- Atomicity of data is critical

Asynchronous communications



- Only source PE is involved
- Atomicity of data is critical
- Obtaining time level (k)
  - Initialize $B_{recv} = B_{chg}$
  - New message: if $B_{recv} \neq B_{chg}$

Asynchronous algorithm

1. *Initialize*
2. *Synchronous time loop to populate* B_{recv}
   a. *Compute* $\boldsymbol{U}_{\boldsymbol{i}}^{n+1} = f\left(\boldsymbol{U}^{n}\right), \boldsymbol{i} \in \boldsymbol{I}$
   b. *Communication updated values*

Asynchronous algorithm

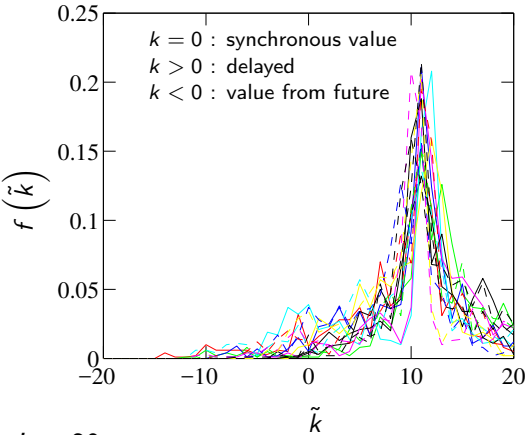
1. *Initialize*
2. *Synchronous time loop to populate* B_{recv}
   a. *Compute* $\boldsymbol{U}_{\boldsymbol{i}}^{n+1} = f\left(\boldsymbol{U}^{n}\right), \boldsymbol{i} \in I$
   b. *Communication updated values*
3. *Index values in* B_{recv} *and create* B_{chg}
4. *Asynchronous time loop*
   a. *Compute* $\boldsymbol{U}_{\boldsymbol{i}}^{n+1} = f\left(\boldsymbol{U}^{n}\right), \boldsymbol{i} \in I_I$
   b. *Check communication status*
   c. *Synchronize if delays* $\tilde{k} > L$
   d. *Compute* $\boldsymbol{U}_{\boldsymbol{i}}^{n+1} = f\left(\boldsymbol{U}^{n}, \boldsymbol{U}^{n-\tilde{k}}, \boldsymbol{U}^{n-\tilde{k}-1}\right), \boldsymbol{i} \in I_B$
   e. *Initiate communication of updated values*

Outline

Computational performance

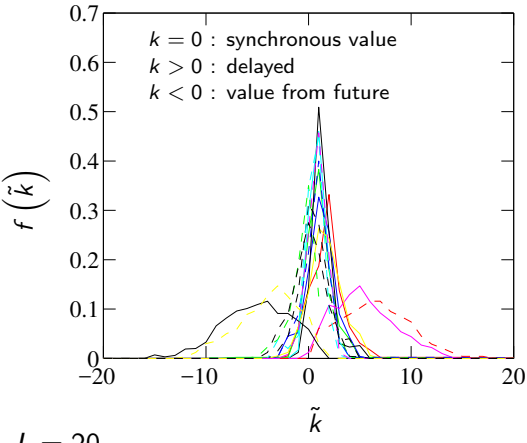
- How much can we gain by solving PDEs asynchronously?
- Need data from realistic problems on state-of-the-art machines with advanced interconnect
  - Distribution of $\tilde{k}$: $f(\tilde{k})$
  - Strong scaling
- Simulation of 1D/3D flow problems
- Parameters close to practical simulations

$\tilde{k}$ distribution: Hopper-MPI



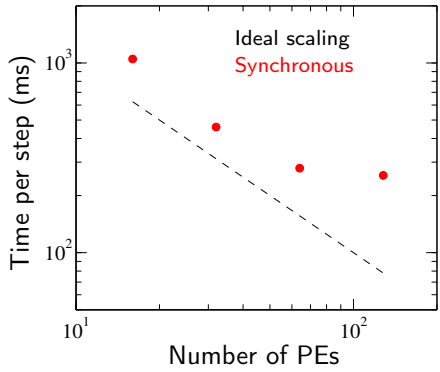
- $N = 128$, $P = 8$, $L = 20$
- Mean: 9.6
- Standard deviation: 5.4

$\tilde{k}$ distribution: Hopper-foMPI



- $N = 128$, $P = 8$, $L = 20$
- Mean: 0.9
- Standard deviation: 1.9

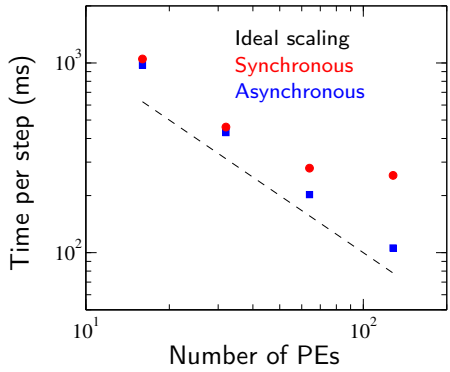
Strong scaling



- Burgers' equations, $N = 32K$, $L = 20$

K. Aditya, D. A. Donzis and T. Hoefler (SC 2013), Outstanding Dissertation Award

Strong scaling

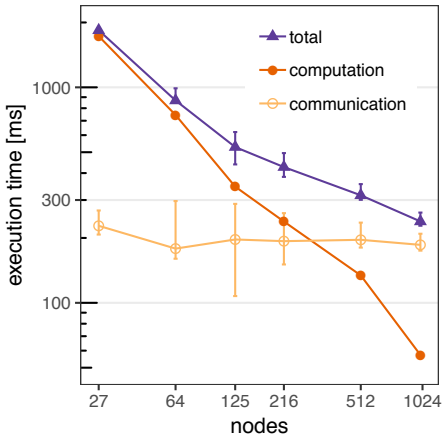


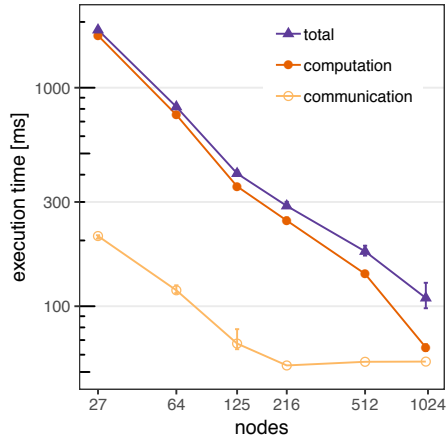
- Burgers' equations, $N = 32K$, $L = 20$
- Nearly ideal scaling with asynchronous computing

K. Aditya, D. A. Donzis and T. Hoefler (SC 2013), Outstanding Dissertation Award

Strong scaling

- 3D turbulence simulation: hybrid OpenMP-MPI code (up to 12 thousand cores)
- Grid: 120^3

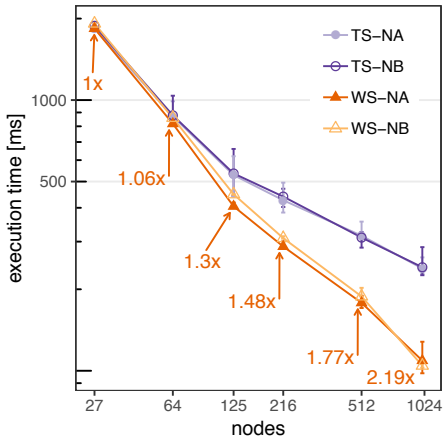




K. Aditya et al. (under review, SISC)

Strong scaling

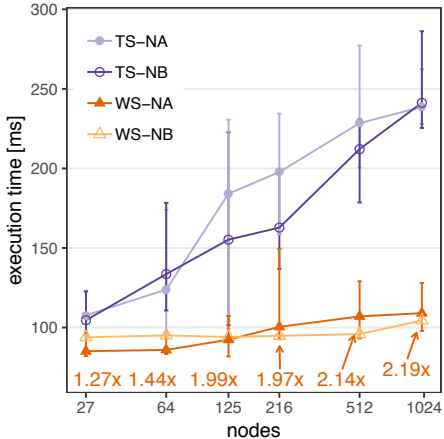
- Traditional synchronous (TS), Weakly synchronous (WS) - asynchronous method
- One-sided (NA), two-sided (NB)



K. Aditya et al. (under review, SISC)

Weak scaling

- Traditional synchronous (TS), Weakly synchronous (WS) - asynchronous method
- One-sided (NA), two-sided (NB)



K. Aditya et al. (under review, SISC)

Outline

Energy

Source: Y. Ju

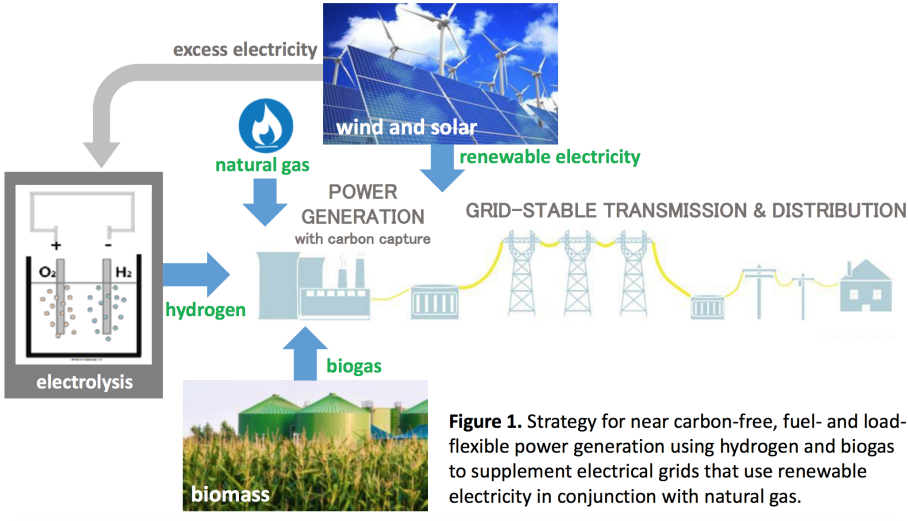
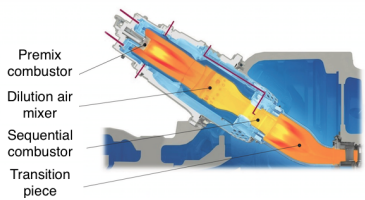


Figure 1. Strategy for near carbon-free, fuel- and load-flexible power generation using hydrogen and biogas to supplement electrical grids that use renewable electricity in conjunction with natural gas.

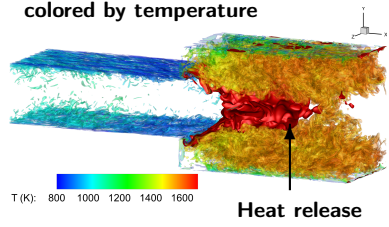
Reheat burner simulations

Ansaldo GT36



Source: Ansaldo Energia

Iso-surfaces of vorticity magnitude colored by temperature

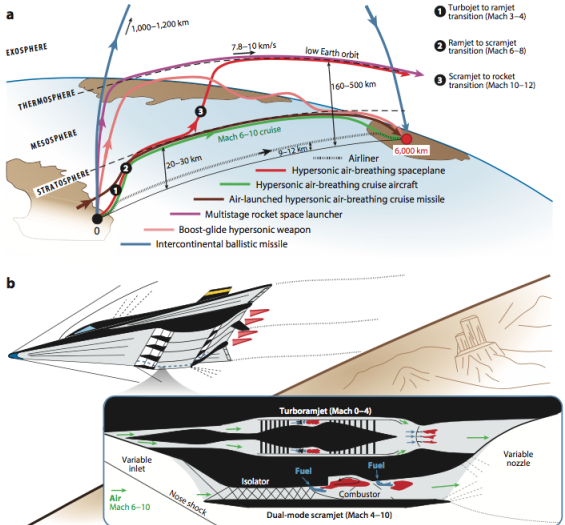


- Aim: achieve high efficiency, low emissions, and fuel flexibility
- Understand flame stabilization mechanism at reheat conditions
- **1.25 billion grid points** run on **50 thousand processors**
- Evaluated the role of mixed mode combustion

K. Aditya et al., Proceedings of the Combustion Institute (2018, accepted)

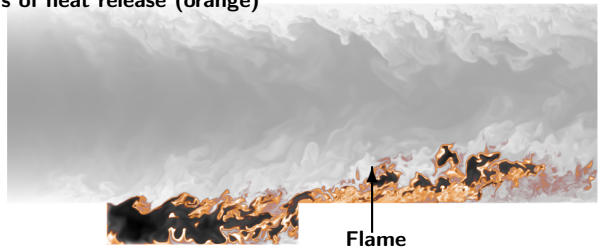
Aerospace

Source: J. Urzay (ARFM 2018)



Scramjet combustion simulations

Contours of temperature (white-black)
Contours of heat release (orange)

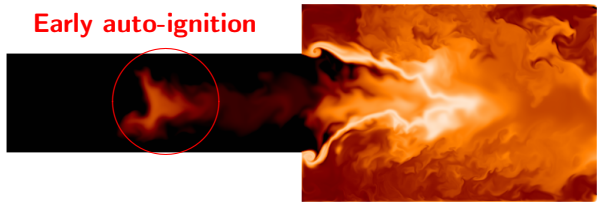


- Aim: evaluate flame stabilization mechanism at lean SCRAM JET conditions
- **2.0 billion grid points** run on **200 thousand processors**
- Study the effect of cavity aspect ratio
- Vorticity dynamics in the shear layer

K. Aditya, H. Kolla, and J. H. Chen, manuscript under preparation

Anomalous or extreme events

Reheat combustor

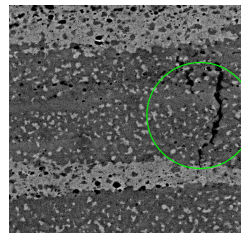


source: K. Aditya et al. (2018)

Cyclones in weather

source: NOAA

Crack propagation in materials

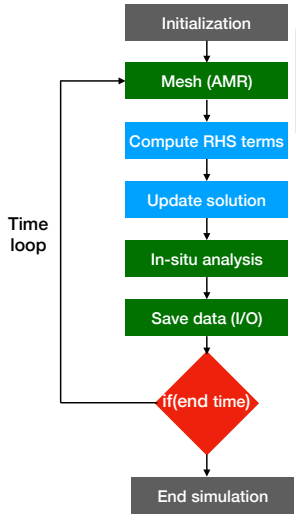


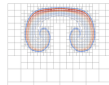
source: K. Satya Prasad

Scientific data

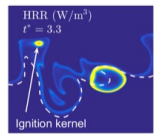
- Multi-variate
- Multi-scale
- Smoothly varying
- Time streaming
- Massive data from both experiments and computations
- Need intelligent workflows to compute, analyze and store data
- Anomalous events:
  - characterized by extreme values in joint distributions
  - appear in highly localized regions in space/time

Simulation workflow





M. A. Kopera et al. (2014)



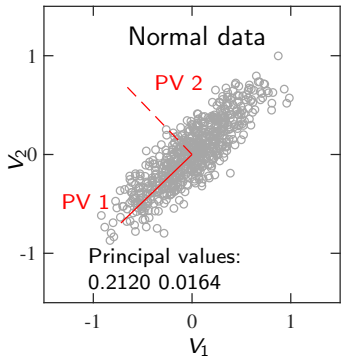
A. Krisman (2016)

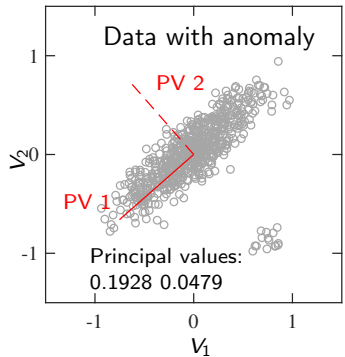
Principal component analysis

- Scale the data: with mean and maximum
- Compute the co-variance matrix (second order joint moment)
- Perform Eigen decomposition to obtain the principal values and vectors

Principal component analysis

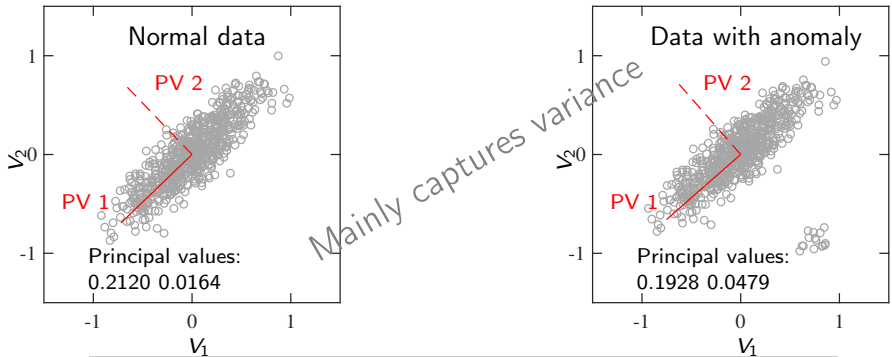
- Scale the data: with mean and maximum
- Compute the co-variance matrix (second order joint moment)
- Perform Eigen decomposition to obtain the principal values and vectors





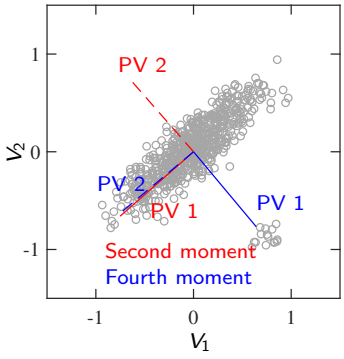
Principal component analysis

- Scale the data: with mean and maximum
- Compute the co-variance matrix (second order joint moment)
- Perform Eigen decomposition to obtain the principal values and vectors

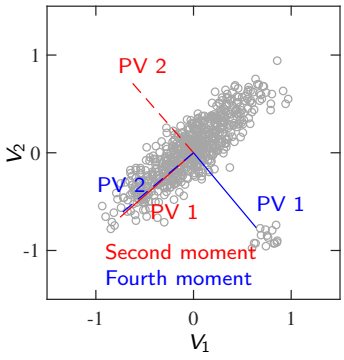


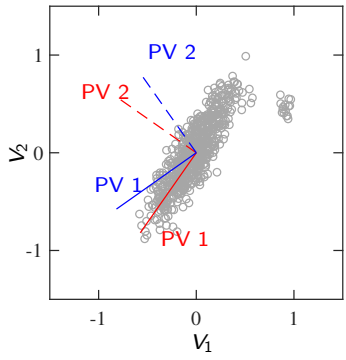
Need higher order moments to capture extreme events

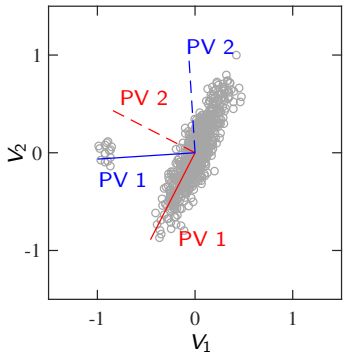
Anomaly detection



Anomaly detection







- First principal kurtosis vector aligns in the direction of anomalies
- Can be used to characterize extreme events

Outline

Conclusions

- Provided a basic overview and parallel implementation PDE solvers
- Asynchronous computing: viable approach to solve PDEs accurately at extreme scales
- Developed theoretical framework: $\langle \overline{E} \rangle \sim \frac{P}{N} \Delta x^a \sum_{m=1}^{\mathcal{T}} \gamma_m \overline{\tilde{k}^m}$
  - Combine numerical as well as machine-specific architectural aspects
  - Understand effects of asynchrony
  - Devise new schemes which are tolerant to asynchrony
- Developed algorithm that has the potential to eliminate virtually all communication overheads at Exascale by eliminating forced synchronizations and overlapping communication and computation

Future work:

- Perform 3D reacting flow direct numerical simulations
- Couple the method with asynchronous runtime models