Recent Advances in High Performance Algorithms for Computational Science and Engineering

Alvaro Coutinho
alvaro@nacad.ufrj.br

High Performance Computing Center
COPPE/Federal University of Rio de Janeiro
www.nacad.ufrj.br

Contents

- **First things first: Who we are!**
- **Motivation: Multiphysics and Multiscale**
- **Advanced FEM algorithms**
  - Advanced Nonlinear solvers
  - Adaptive Timestep Control
- **Parallel Sub-domain Implementation**
  - Performance measurements, Communication Models, Linear and Nonlinear Solvers, Preconditioners, Adaptivity
- **Parallel Octree Mesh Generation**
- **Scientific Workflows and Many-Task-Computing**

WHO WE ARE

UFRJ Innovation Ecosystem

- COPPE - Federal University of Rio de Janeiro
- CENPES - PETROBRAS Research Center
- Start-ups Incubator
- Rio Technology Park (11 R&D Centers + COPPE Labs)
- Innovation Tower (design phase)

HPC Center's Mission

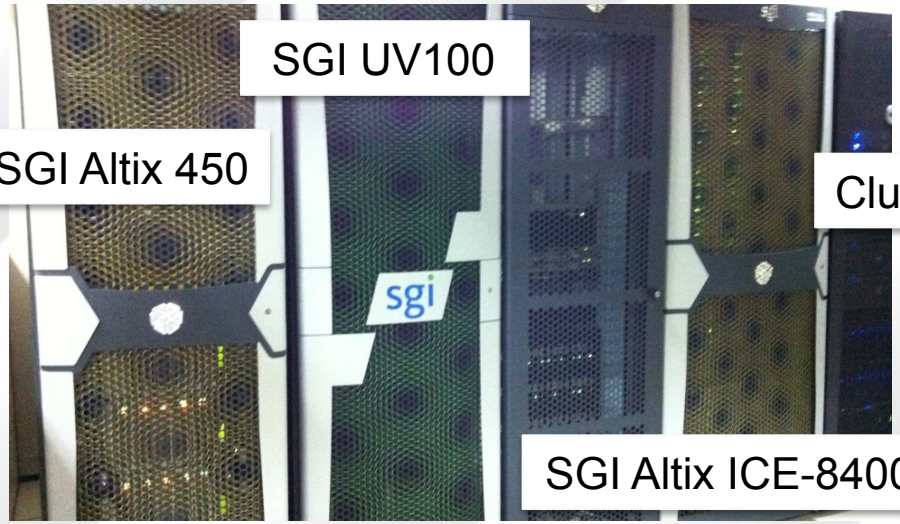
- Foster HPC and CSE at our University and work with our partners
- Operate an advanced HPC and Viz infrastructure
- Develop and support R&D multidisciplinary projects with academia and industry mainly in:
  - Energy: Oil and Gas, Electrical
  - Engineering, Civil, Mechanical, Material Science, Naval
  - Environment, Meteorology, Oceanography
  - Computer science, Databases, Data Mining, e-Science
  - Biological and Life Sciences

HPC Center Systems

SGI UV100

SGI Altix 450

Cluster Dell



SGI Altix ICE-8400

NetApp Storage 100TB

Galileu, Oracle Server #457 TOP500

Dell's Tiled Wall Display

MOTIVATION: MULTIPHYSICS AND MULTISCALE

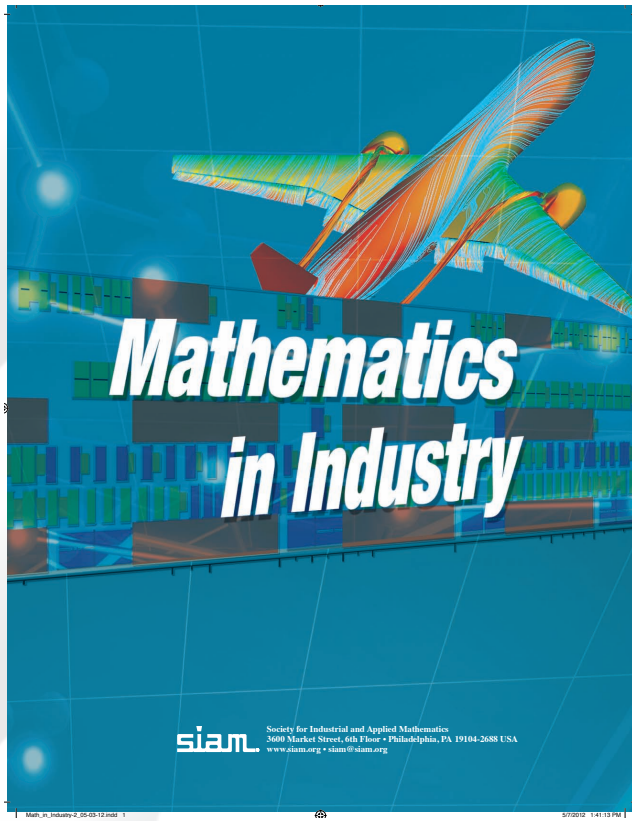
Multiphysics and Multiscale

Argonne NATIONAL LABORATORY

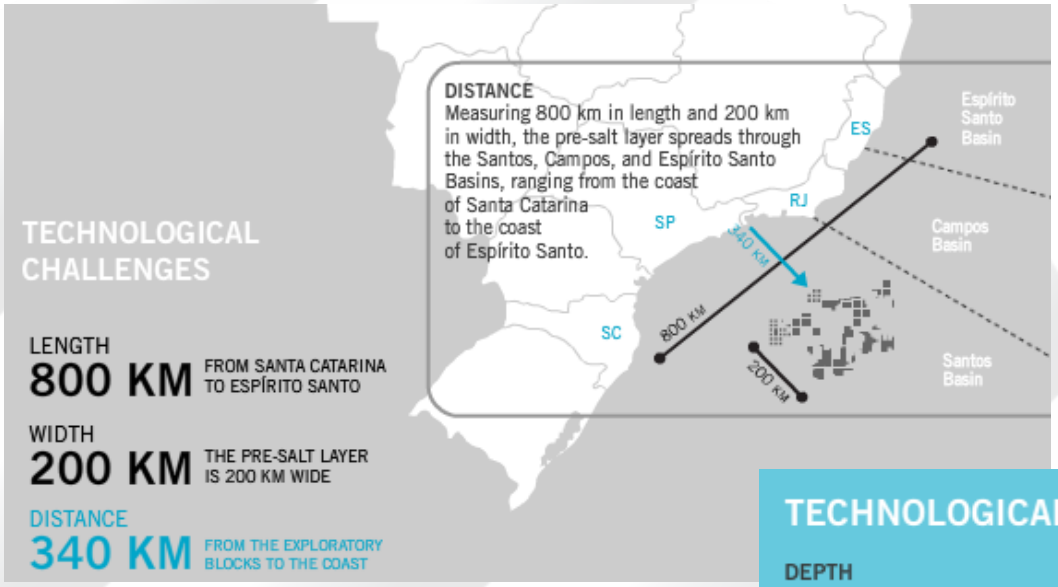
ANL/MCS-TM-321

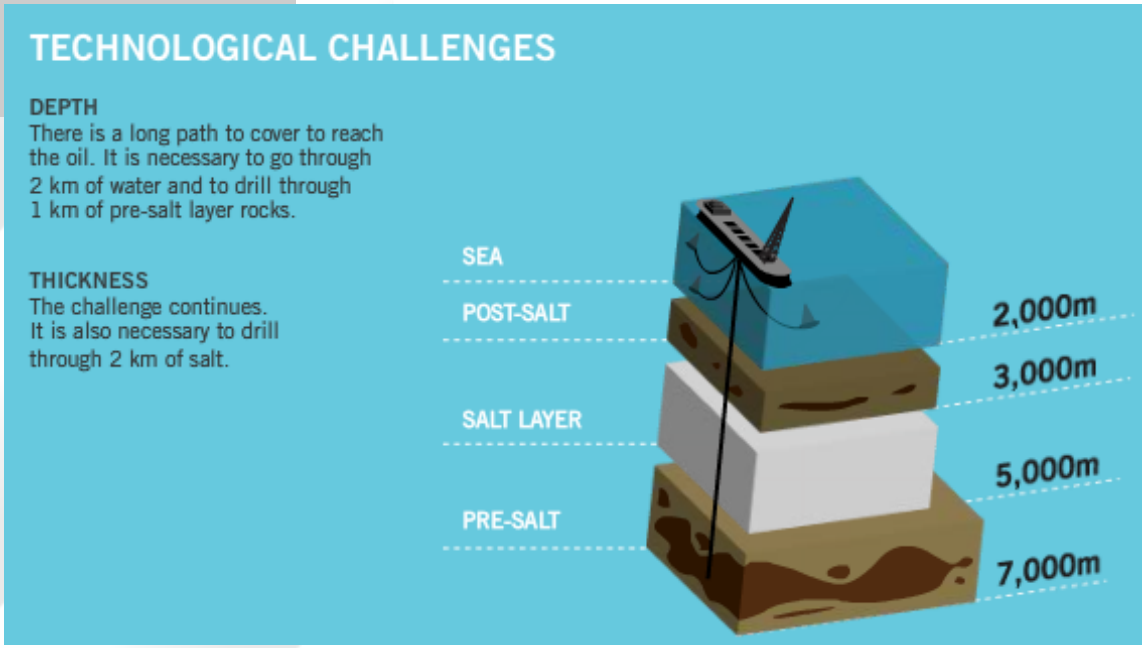
Multiphysics Simulations: Challenges and Opportunities

Mathematics and Computer Science Division



Technological Challenges: Pre-Salt Oil Discoveries





Source: www.petrobras.com

ADVANCED FEM ALGORITHMS

FE Simulation Code: Major Components

```
ALGORITHM 1
Pre-processing
Input data
        Time integration loop
            Nonlinear iteration loop
                Form system of linear equations
                Solve system of linear equations
            End NL loop
            Update time step
            Output results
        End time loop
Post-processing
Visualization
```

Complexity $O(n^{4/3})$, n #unknowns

Optimal solvers require O(n) work per time step, and time accurate integration often implies $O(n^{1/3})$ time steps.

EdgeCFD®
Coupled Fluid Flow/Free-surface/FSI Solver

General:

- Edge based data structure. *"EDE has been proving to be more efficient than other FEM data structures like CSR or EBE"*
- Segregated predictor-multicorrector time marching;
- Adaptive time stepping with PID controller;
- Supports hybrid parallelism (MPI, OpenMP or both at the same time);
- Unstructured grids with linear tetrahedra for velocity, pressure and scalar transport;
- Mesh partitioning performed by Metis or ParMetis;
- Best data reordering defined by EdgePack® in a preprocessing phase;
- Thermal-flow coupling with Boussinesq approximation; FSI
- Input/Output file formats: ANSYS/Ensight/Paraview, neutral files, Xdmf/hdf5

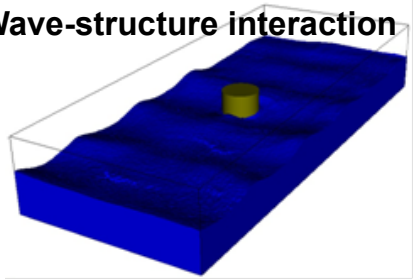
Incompressible Flow:

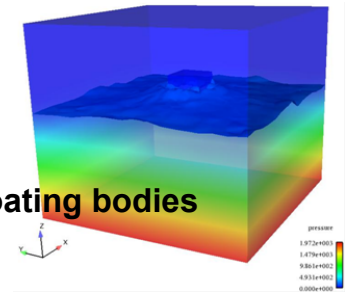
- SUPG/PSPG/LSIC stabilized finite element method in Eulerian or ALE frames
- Fully coupled **u**-*p* system (4-dofs per node/non-symmetric);
- Inexact Newton-GMRES;
- LES (Smagorinsky, Dynamic Smagorinsky), ILES, RB-VMS
- Newtonian or non-Newtonian flows (Power Law, Bingham and Hershel-Buckley)

Transport:

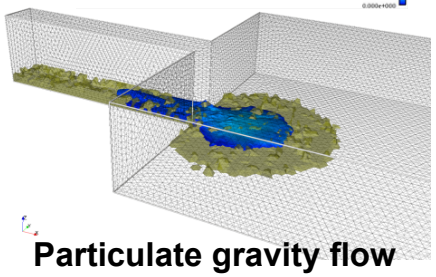
- SUPG/CAU/YZBeta stabilized finite element method in Eulerian and ALE frames
- Supports free-surface flows through Volume-Of-Fluid and Level-Sets.
- (UFMM) Unstructured Fast Marching Method for fast computation of signed distance functions
- PDD: Parallel dynamic deactivation. *"Restrict the computation only in regions with high solution gradients"*

Wave-structure interaction





Floating bodies



Particulate gravity flow

Advanced Nonlinear Solvers

Algorithm 2: Inexact Krylov-Newton Backtracking Method - INKB

1 Set η_0;
2 $k = 0$;
3 $\tau_{NL} = \tau_{res}\|\mathbf{F}(\mathbf{x}^k)\|_2$;
4 **while** $\|\mathbf{F}(\mathbf{x}^k)\|_2 > \tau_{NL}$ **do**
5 Compute $\mathbf{J}(\mathbf{x}^k)$;
6 Solve $\mathbf{J}(\mathbf{x}^k)\mathbf{s} = -\mathbf{F}(\mathbf{x})$ by a Krylov method with tolerance η_k;
7 Set $\lambda_1 = 1$;
8 Compute $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_1\mathbf{s}$;
9 $i = 0$;
10 $\mathbf{x}^t = \mathbf{x}^{k+1}$;
11 **while** $\|\mathbf{F}(\mathbf{x}^t)\|_2 > (1 - \alpha\lambda_i)\|\mathbf{F}(\mathbf{x}^{k-1})\|_2$ **and** $i \leq nbt$ **do**
12 Choose λ_{i+1};
13 Update $\mathbf{x}^t = \mathbf{x}^k + \lambda_{i+1}\mathbf{s}$;
14 $i = i + 1$;
15 **endw**
16 **if** $i < nbt$ **then**
17 Update $\mathbf{x}^{k+1} = \mathbf{x}^t$;
18 **else**
19 backtracking loop rejected ;
20 **endif**
21 $k = k + 1$;
22 Select η_k;
23 **endw**

P-GMRES(m)

Basic idea: "... adapt inner solver tolerance linear according to the convergence history of nonlinear iterations..."

Oversolving control:

$\eta_k = \min\{\eta_0, \max\{\eta_k, \rho \ \ \tau_{NL}/\|\mathbf{F}(\mathbf{x}^k)\|\}\}, \quad \rho = 0.5.$

Adaptive Time Step Control

- **How to speed-up time marching schemes?**
  - Adapting time step according to the solution evolution;
  - Track the solution with interventions (feedback) when needed.
  - Particularly useful in stiff systems
- **Key idea:**
  - Tracking × Feedback → Controllers
  - Theory on ODE solvers
    - *G. Soderlind, Automatic control and adaptive time-stepping. Numerical Algorithms, 2002; 31:281–310*

PID Controller to Adapt Time Steps

- **PID controller**

$$\Delta t = \left(\frac{e_{n-1}}{e_n}\right)^{k_P} \left(\frac{1}{e_n}\right)^{k_I} \left(\frac{e_{n-1}^2}{e_n e_{n-2}}\right)^{k_D} \Delta t_{prev}$$

Controller transfer function

$$e_n = \max(e_{\mathbf{u}}, e_p, e_\phi)$$

$$e_{\mathbf{u}} = \frac{e_{\mathbf{u}}^*}{tol_{\mathbf{u}}}, \qquad e_{\mathbf{u}}^* = \frac{\|\mathbf{u}_n - \mathbf{u}_{n-1}\|}{\|\mathbf{u}_n\|}$$

$$e_p = \frac{e_p^*}{tol_p}, \qquad e_p^* = \frac{\|p_n - p_{n-1}\|}{\|p_n\|}$$

$$e_\phi = \frac{e_\phi^*}{tol_\phi}, \qquad e_\phi^* = \frac{\|\phi_n - \phi_{n-1}\|}{\|\phi_n\|}$$

Measuring what we want to control

"measuring solution change"

We can do better, but this is cheap!

Sometimes better to control CFL

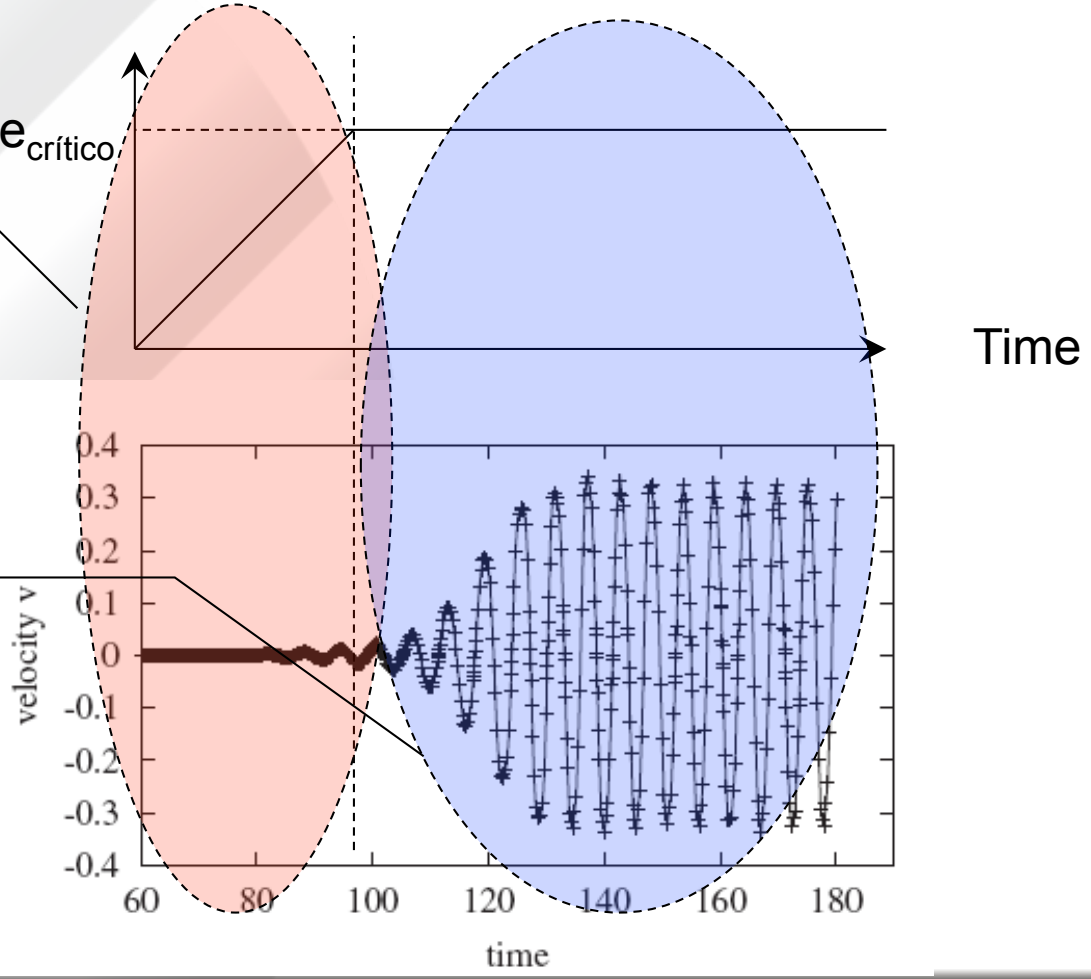
See: A. M. P. Valli, R. N. Elias, G. F. Carey, A. L. G. A. Coutinho, PID adaptive control of incremental and arclength continuation in nonlinear applications, Int. J. Numer. Meth. Fluids 2009; 61:1181–1200

PID Controller in Action:

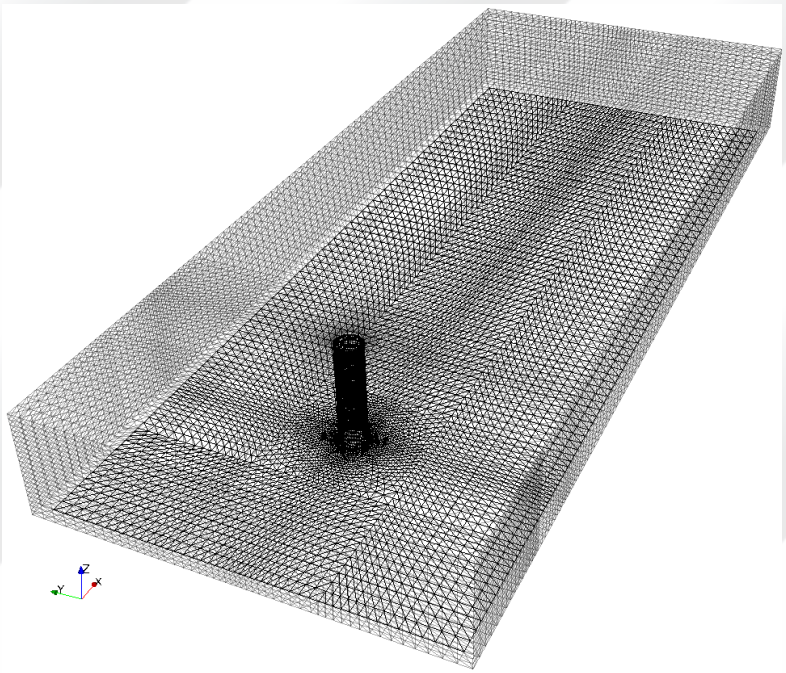
- **Vortex-shedding around a cylinder at Re=100**

Region of slow varying solution where stepsize can be big without compromising solution accuracy.

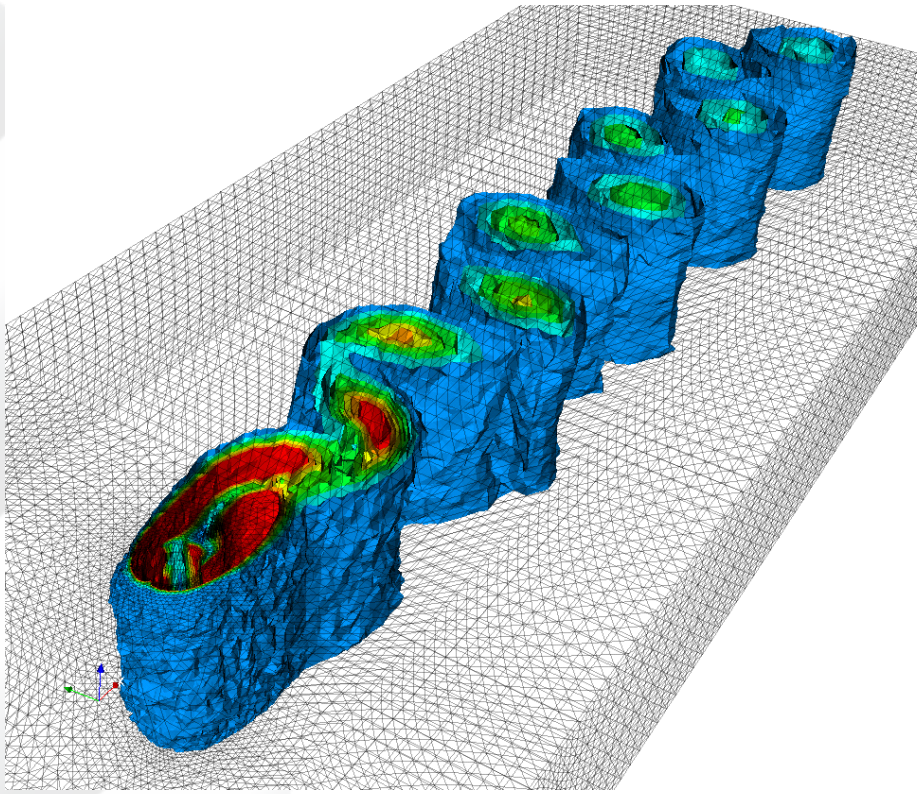
Periodic regime region. Controller will produce a signal to correct such oscillations, generating smaller time steps.



Incompressible Flow past a Circular Cylinder (Re=100)

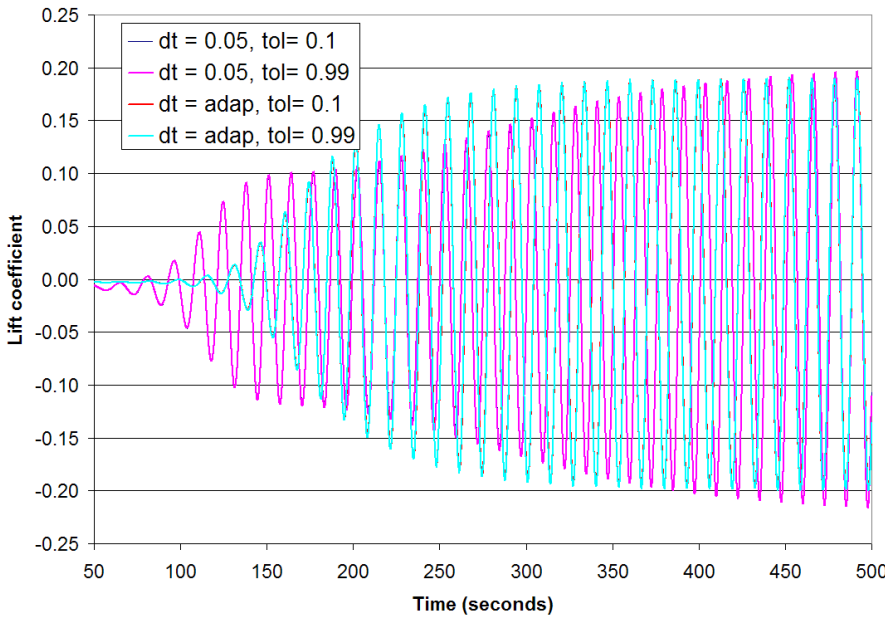


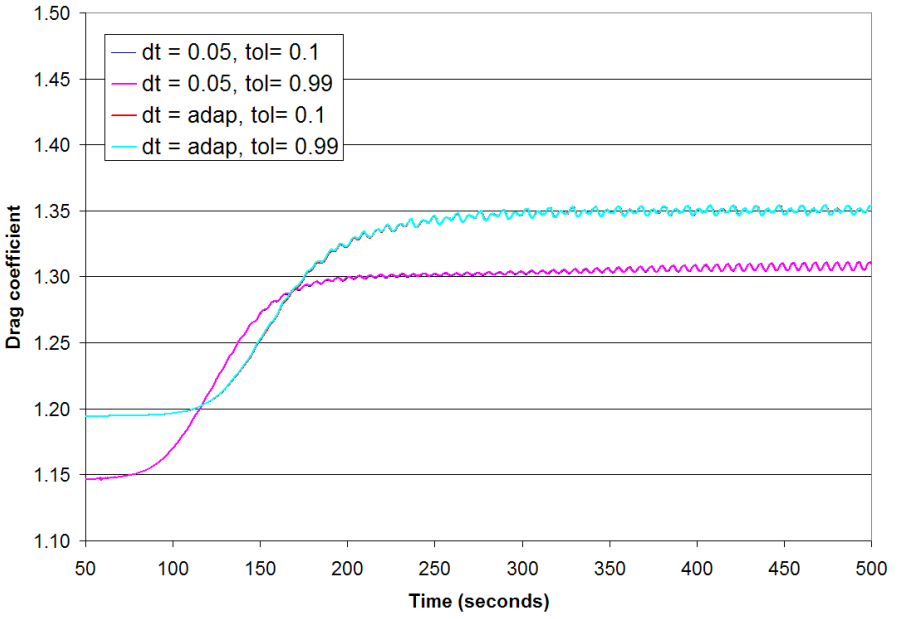
Finite element mesh:
446,662 tetrahedra
1,010,367 edges
81,991 nodes



Snapshot of cylinder surface mesh
and vorticity, showing the development of the
von Karman vortex streets, S_t = 0.16

Flow past a Circular Cylinder: Effects of Time Step Control and IN Methods





Reference solutions:

L. Baranyi, 2003 C_L=0.228, C_D=1.346, S_t=0.163

C. Norberg, 2003

$$S_t \approx 0.2663 - 1.019/\sqrt{\mathrm{Re}}$$

$$C_L \approx \left(\frac{\varepsilon}{30} + \frac{\varepsilon^2}{90}\right)^{1/2}, \quad \varepsilon = (\mathrm{Re} - 47)/47$$

S_t=0.1644, C_L=0.2274

Coupled Fluid Flow and Heat Transfer
Rayleigh-Benard 4:1:1 Container

MESH SIZE:

Elements.....: 93,925

Nodes.....: 21,384

Edges.....: 120,306

Flow equations.....: 70,536

Transport equations...: 19,008

DIMENSIONLESS NUMBERS:

Prandtl.....: 0.72

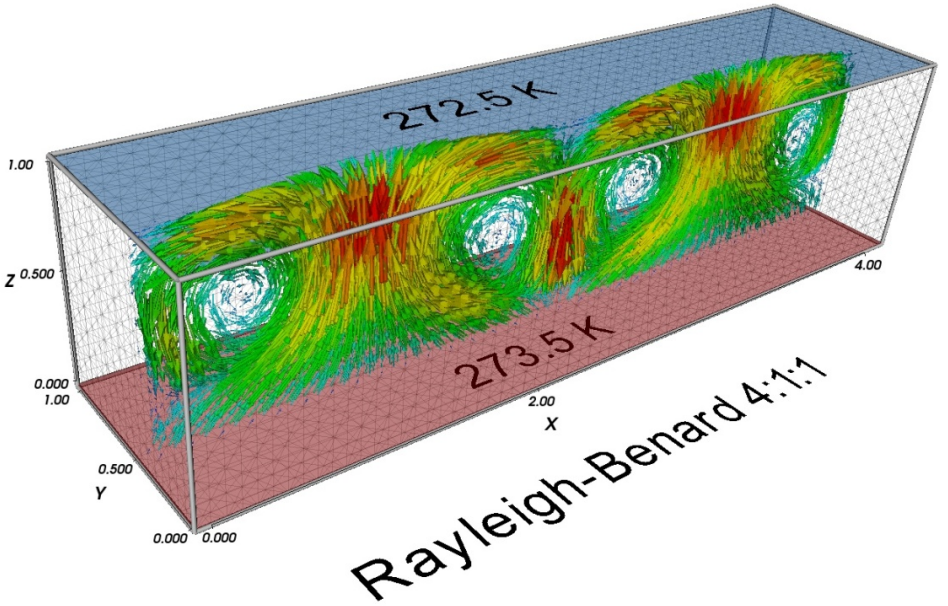
Rayleigh.....: 30,000

NONLINEAR SOLVER PARAMETERS

NLTOL .....: 10^{-3}

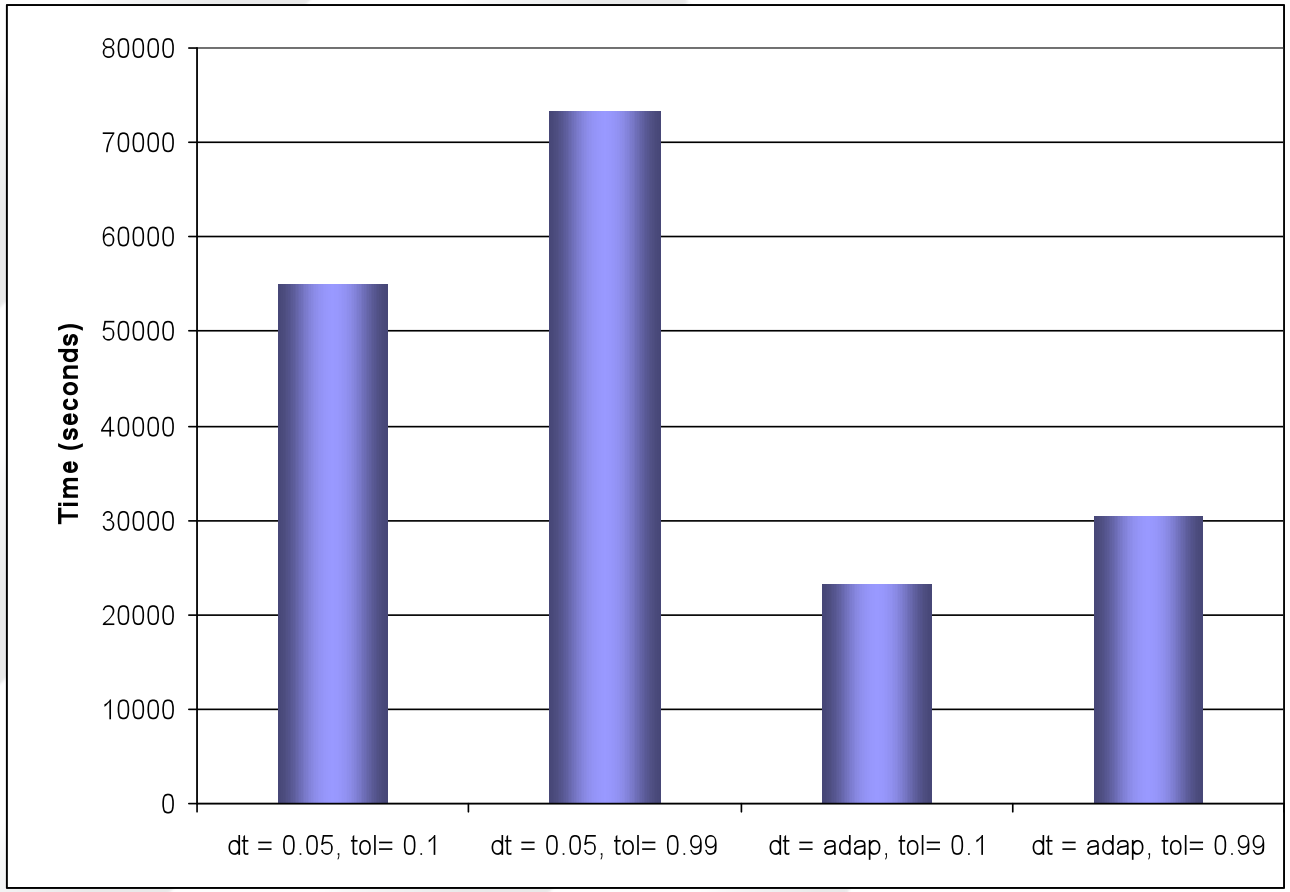
BKTITER .....: 5

2 < CFL < 5, TMAX=2.0, P-GMRES(35)



Solution Data	$\eta = 10^{-6}$	$\eta = 10^{-6}$, BKT	$\eta_{max} = 10^{-1}$, BKT
NL Iter	2,588	2,028	2,574
GMRES Iter	524,623	413,885	38,711
Time Steps	436	436	431
CPU Time	1.0	0.78	0.30

Flow past a Circular Cylinder: Effect of Time Step Control and IN Methods



Adaptive Time Stepping and IN Performance Rayleigh-Benard 4:1:1 Container

Max IN Tol	CFL min	CFL max	SS time	time steps	wall time
0.001	2	10	0.5686	109	1110.49
		5	0.2410	90	855.42
		2	0.1829	171	1184.81
0.1	2	10	0.5635	108	803.89
		5	0.2418	90	637.71
		2	0.1840	172	1017.02
0.99	2	10	0.7436	142	1652.06
		5	0.2417	90	991.96
		2	0.1839	172	1501.97

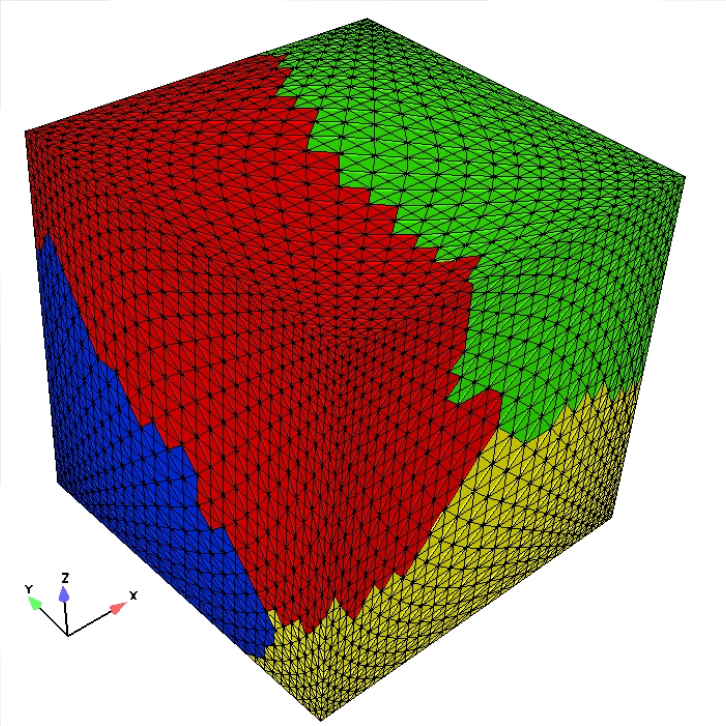
SS Tolerance: 10^{-5} Transport P-GMRES Tol: 10^{-3} # Krylov Space Vectors: 25

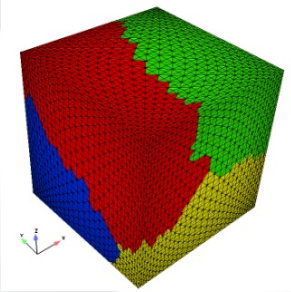
See: A. M. P. Valli, R. N. Elias, G. F. Carey, A. L. G. A. Coutinho, PID adaptive control of incremental and arclength continuation in nonlinear applications, Int. J. Numer. Meth. Fluids 2009; 61:1181–1200

PARALLEL COMPUTATIONS

Hybrid parallelism

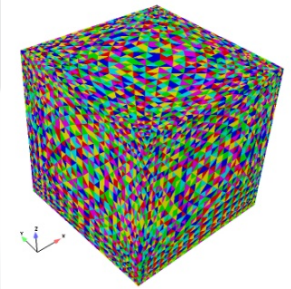
- How do we work with data partitioning and memory dependency?





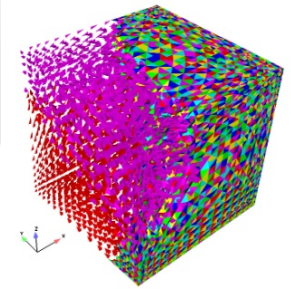
Mesh partitioning (Metis/ Parmetis)

Distributed memory (MPI)



Mesh coloring

Threaded parallelism (OpenMP)

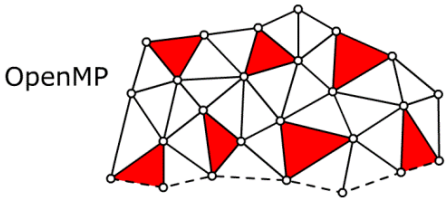


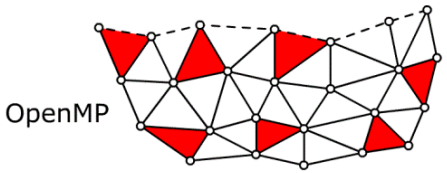
Data partitioning + Mesh coloring

Hybrid matrix-vector product (OpenMP+MPI)

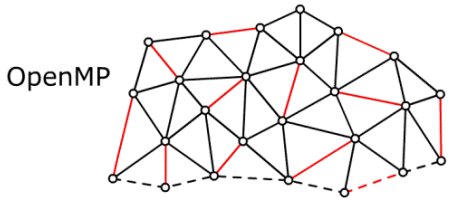
```
iside = 0
DO iblk = 1, nedblk
nvec  = ia_edblk(iblk)
!dir$ ivdep
!$OMP PARALLEL DO
   DO ka = iside+1, iside+nvec, 1
      ...MATVEC computations...
   ENDDO
!$OMP END PARALLEL DO
ENDDO
...over interface nodes...
#ifdef MPICODE
call MPI_AllReduce
#endif
```

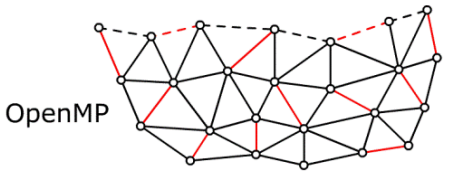
Element-by-Element





Edge-by-Edge

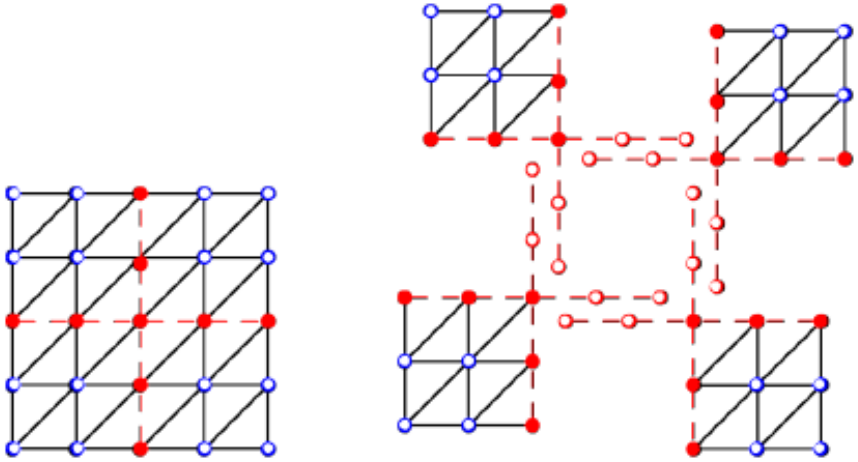




MPI Collective Communications

MPI collective

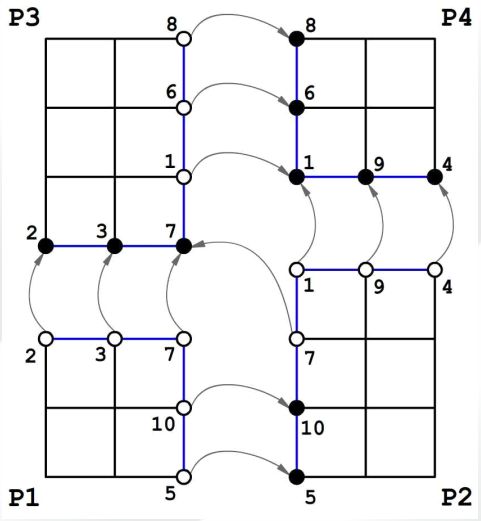
1. All shared equations are synchronized in just one collective operation;
2. Easy to implement;
3. Poor performance for massive parallelism;
4. Some (small) improvements can be done (... *but there's no miracle...*).



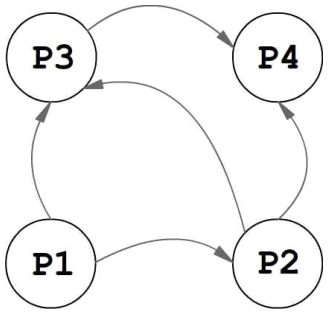
(a) Original mesh (b) Redundant communication

P2P Subdomain Communication

Master-Slave subdomain relationship



(a) Mesh Partition



(b) Communication Graph

Exchange information between neighboring processors implemented in two stages:
(i) slaves processes send their information to be operated by masters
(ii) solution values are copied from masters to slaves.

EdgeCFD uses non-blocking send and receive MPI primitives

Performance on Current Processors

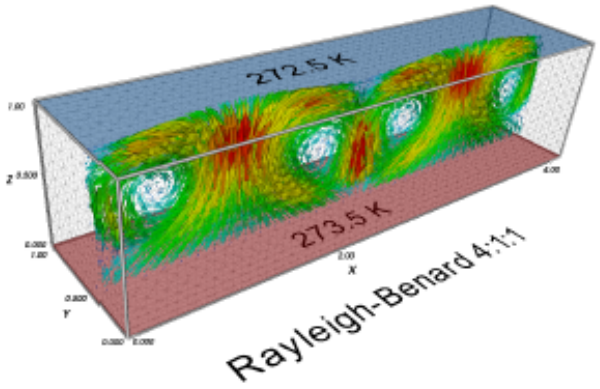
- **Rayleigh-Benard (Ra=30,000, Pr=0.71)**

Assumptions:

1. All mesh entities were reordered to explore memory locality;
2. Same mesh ordering to all systems;
3. Same compiler (Intel) and compilation flags (-fast)

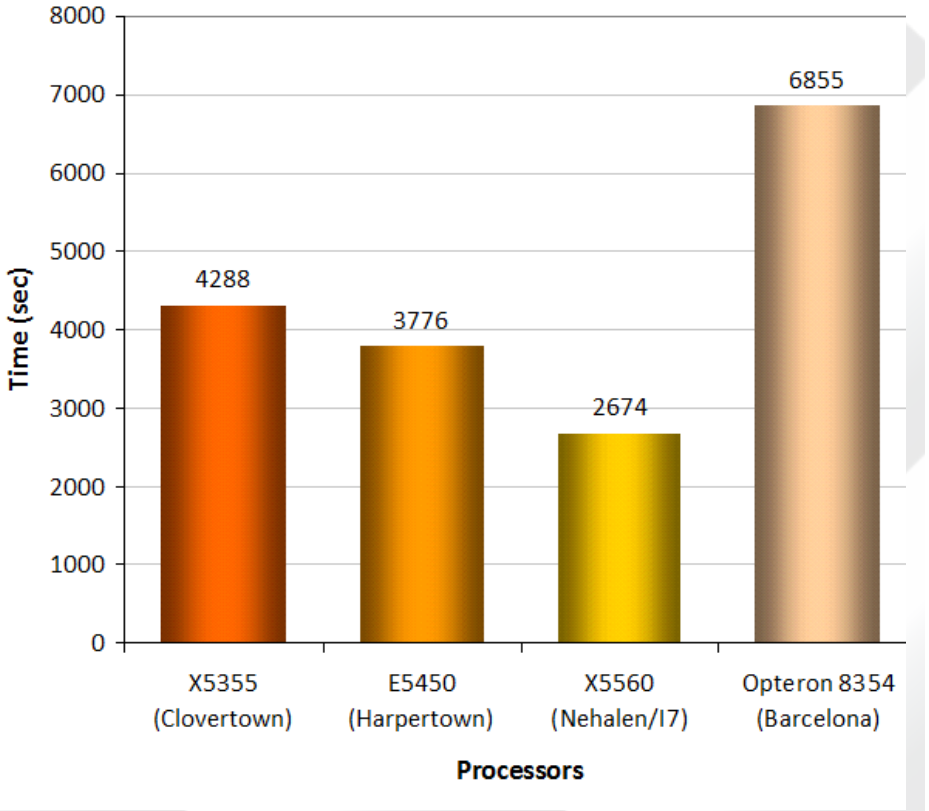
Mesh sizes

	MSH1	MSH2
Tetrahedra	178,605	39,140,625
Nodes	39,688	7,969,752
Edges	225,978	48,721,528
Flow equations	94,512	31,024,728
Transport equations	36,080	7,843,248

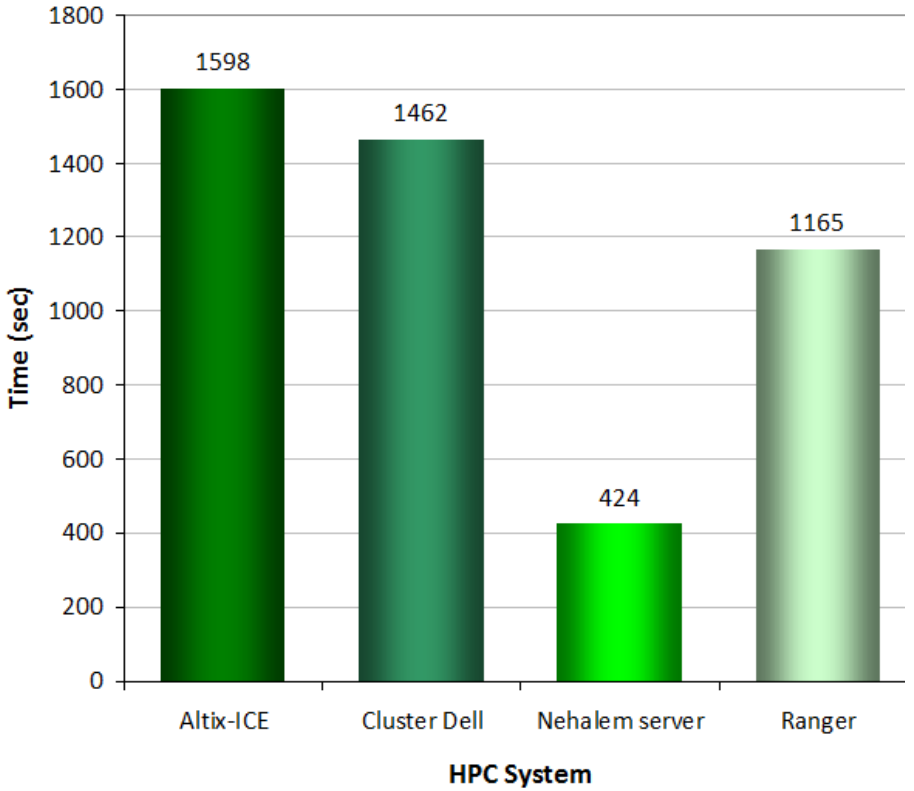


Natural convection problem

Processor performance



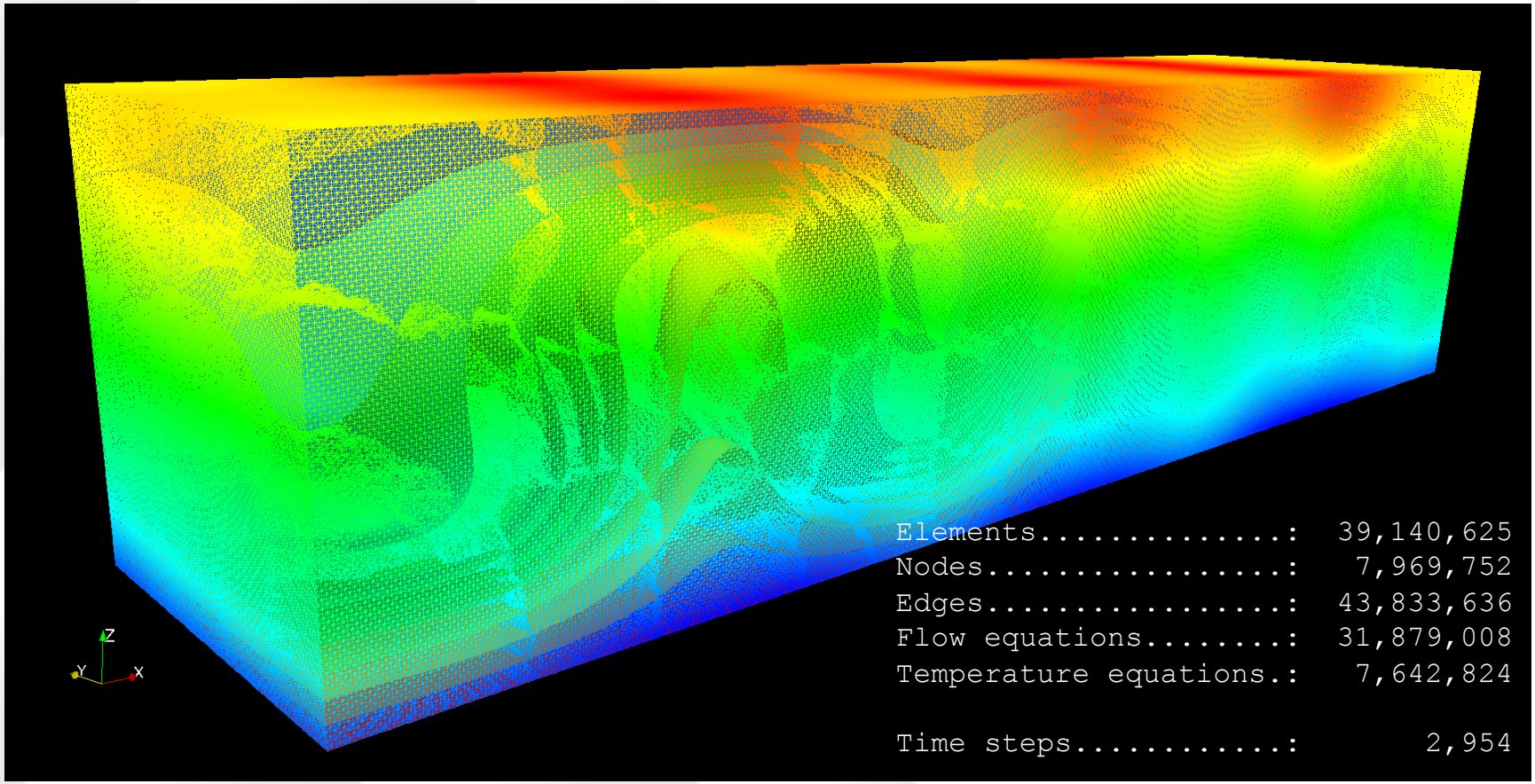
(a) CPU (serial run)



(b) System (8 cores, 1 node)

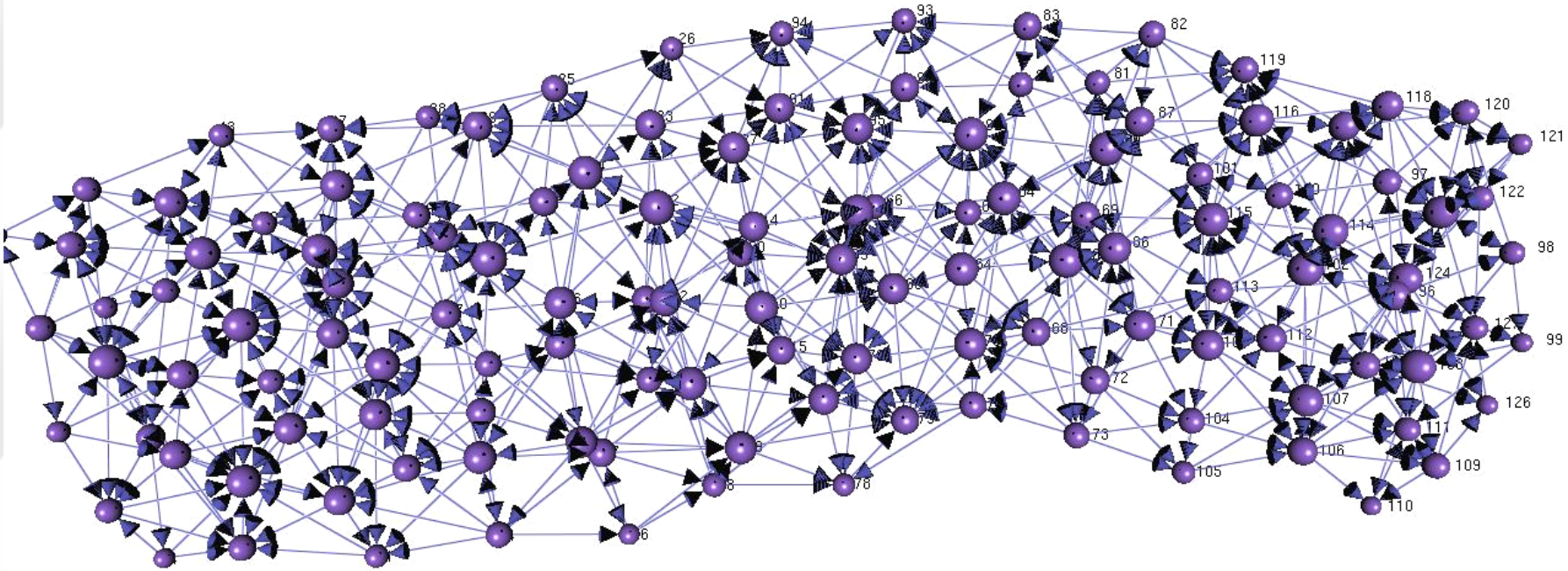
OMP Nehalem Server: slight better than MPI

Rayleigh-Benard 4:1:1 - 501×125×125 mesh



Mesh generation: SGI Altix 450 128GB RAM
Solver: SGI Altix ICE 8400, 128 cores, Cluster Dell 64 cores, MPI-P2P

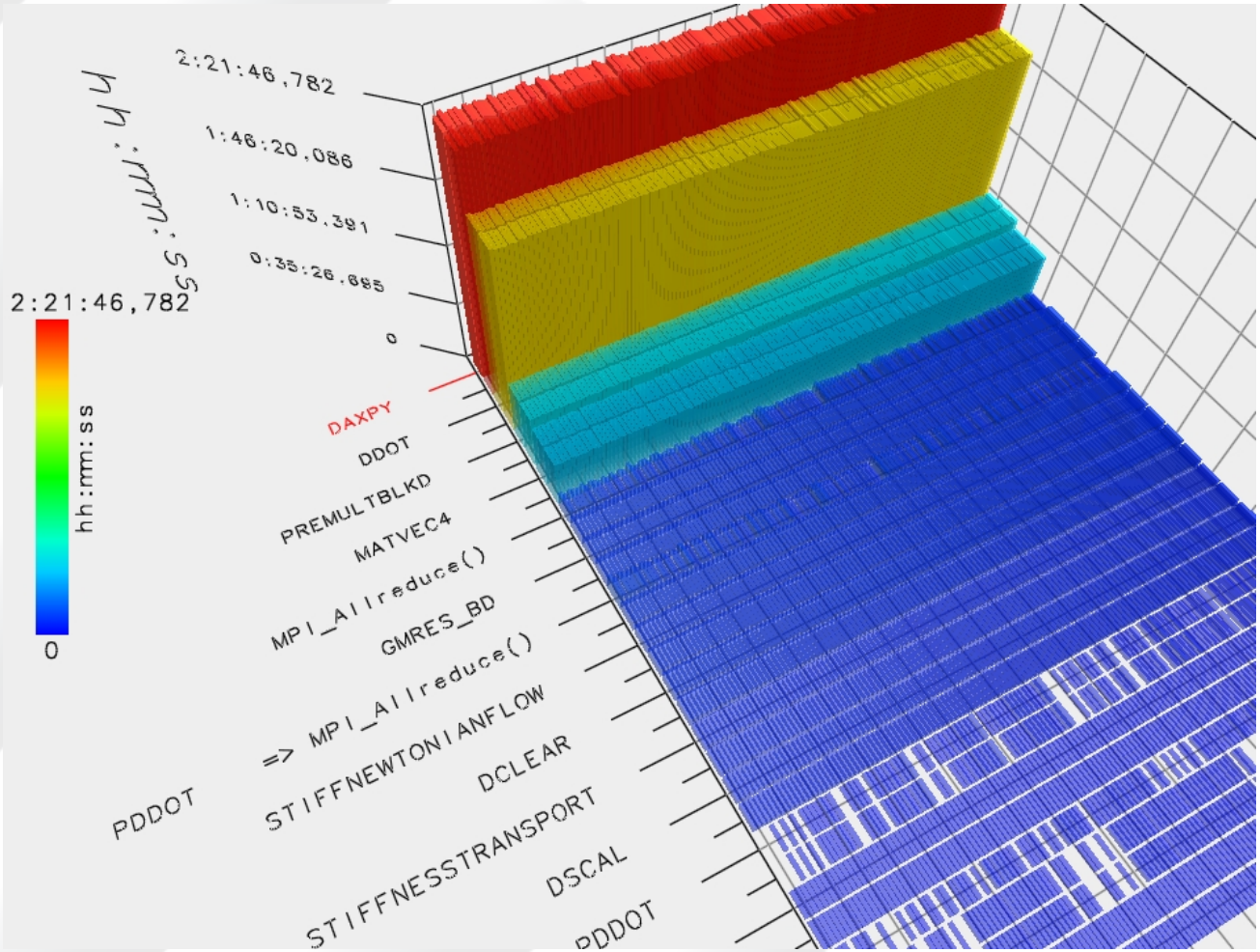
P2P Communication Graph
SGI Altix ICE 8200, 128 cores



Arrows indicate communication direction in sending operations.
Communication is reversed in receiving

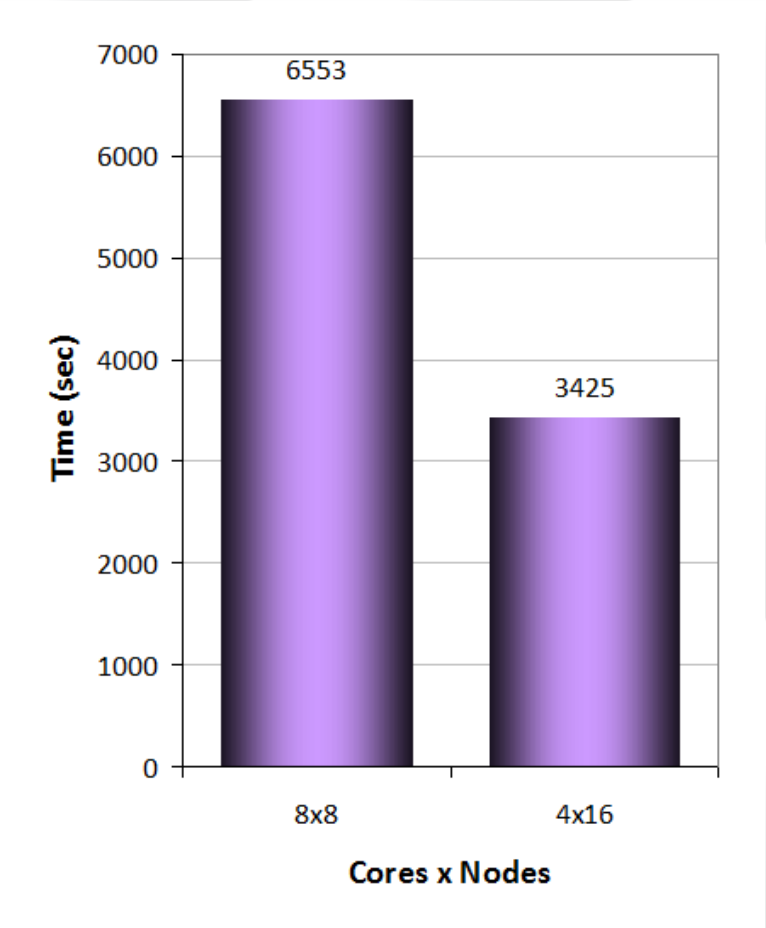
Graph visualization tool: Nodes3D, http://brainmaps.org/index.php?p=desktop-apps-nodes3d

TAU Parallel Profiling

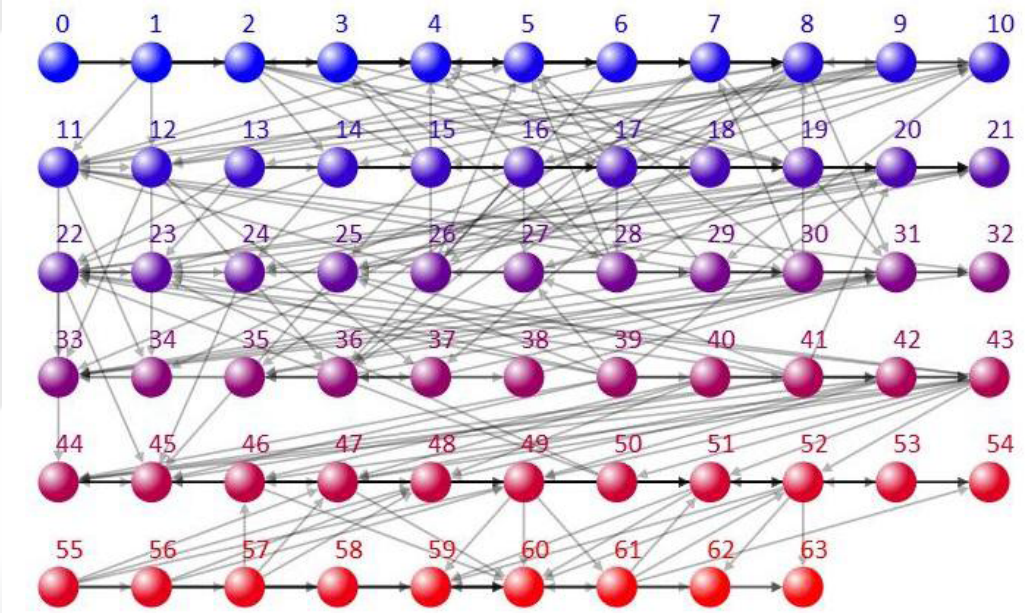


SGI Altix ICE 8400, 128 cores

Cluster Dell, 64 cores, skew core allocation



Time spent in 10 time steps



Communication graph

The LibMesh Library

- The LibMesh library (http://libmesh.sourceforge.net) is an C++ and object-oriented tool for numerical simulation of partial differential equations, on serial and parallel platforms, using the finite element method.
- LibMesh allows discretization of one, two, and three dimensional transient problems using several types of elements.
- Krylov solvers through PETSc library: GMRES with incomplete factorization preconditioners and others
- The LibMesh library provides support for Adaptive Mesh Refinement and Coarsening (AMR/C)
- Development initiated at ICES, UT Austin, USA

Parallel Preconditioners Performance

- **Engineering and scientific flow simulations often require the solution of large and sparse linear systems of equations**

- **Improving that operation involves:**
  - *Methods based on Krylov subspace combined with some efficient preconditioning.*
  - *Parallel computing*

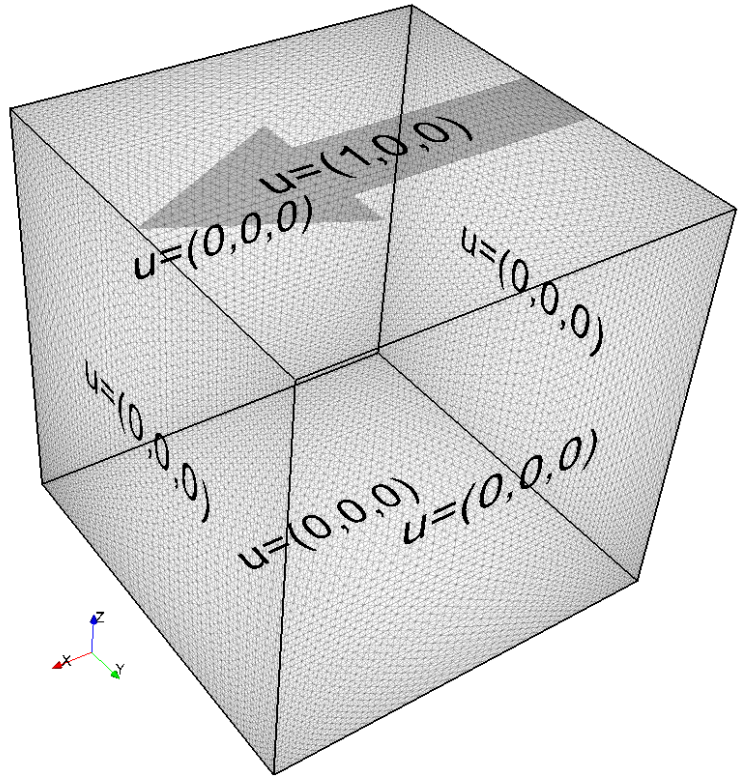
- **Parallel Preconditioners:**
  - *Block-Jacobi*
  - *Block-Jacobi+Lower-upper symmetric Gauss-Seidel (LU-SGS)*
  - *Block-Jacobi+ILU(0)*
  - *All implemented in libMesh and PETSc*

Numerical Results

- **Numerical problem:**
  - *Lid-Driven cavity Flow at Re= 100 and 1000*
- **Computational Resources:**
  - *Uranus – SGI Altix ICE 8200 at NACAD/COPPE/UFRJ:*
    - *cluster with 64 CPUS quad core Intel Xeon with 256 cores connected with InfiniBand network*
  - *Ranger – SUN Constellation at TACC/UT Austin:*
    - *configured with 3,936 16-way SMP compute-nodes (blades) connected with InfiniBand technology in a full-CLOS topology*

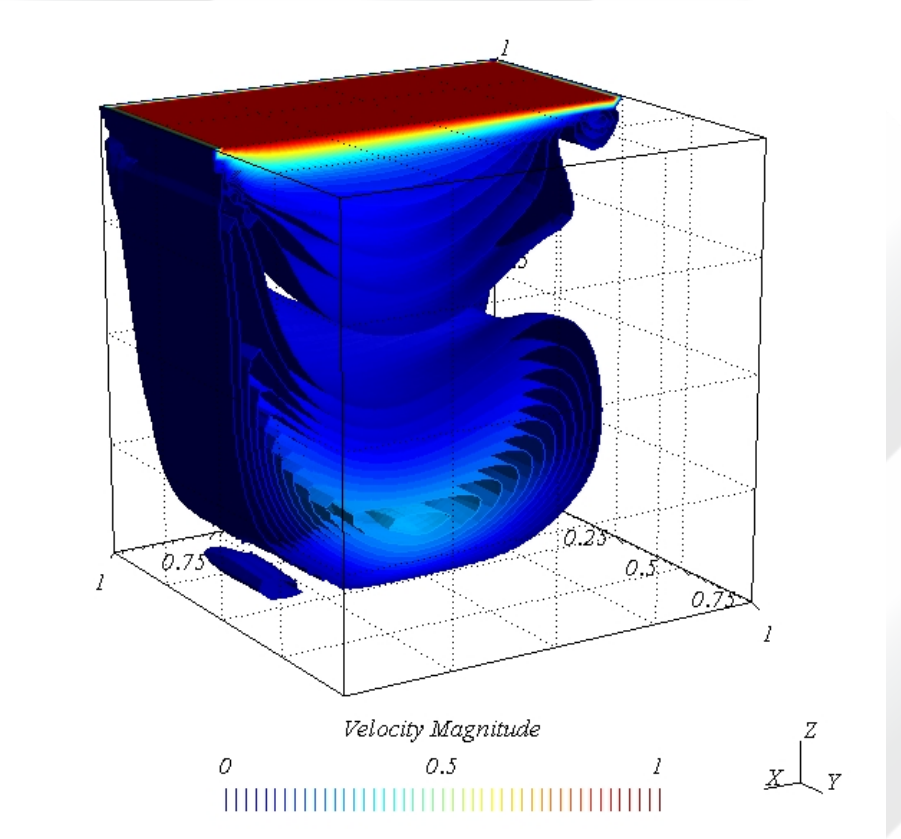
Lid Driven Cavity Flow

- Reynolds 100 and 1000
- Structured Mesh: 40 x 40 x 40
- Quadratic hexahedron (HEX27) for velocity and linear hexahedron(HEX8) for pressure
- GMRES(30)
  - Block-Jacobi
  - Block-Jacobi+ILU(0)
  - Block-Jacobi+LU-SGS
- Natural ordering
- Linear solver tolerance: 10^{-6}
- Nonlinear tolerance: 10^{-3}

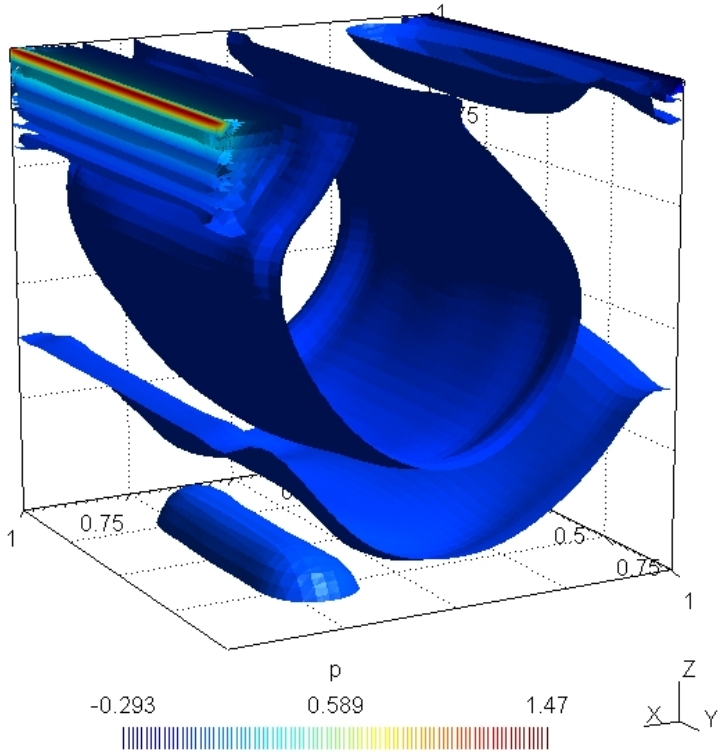


Problem Specification

Lid Driven Cavity Flow



Velocity Magnitude



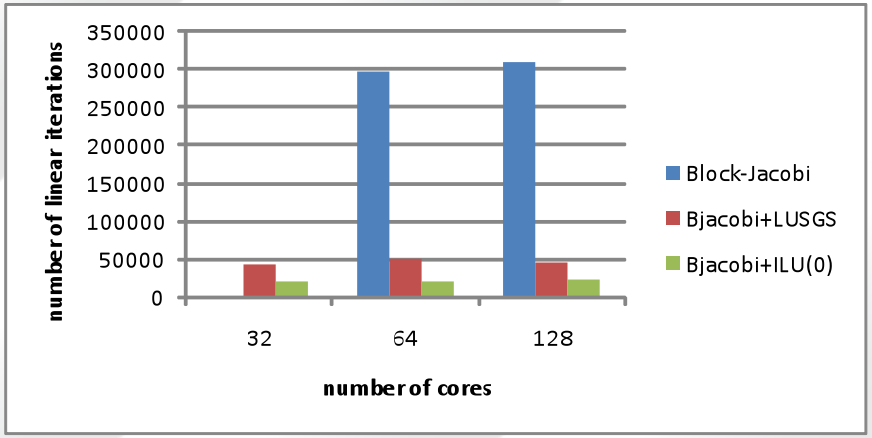
Pressure Field

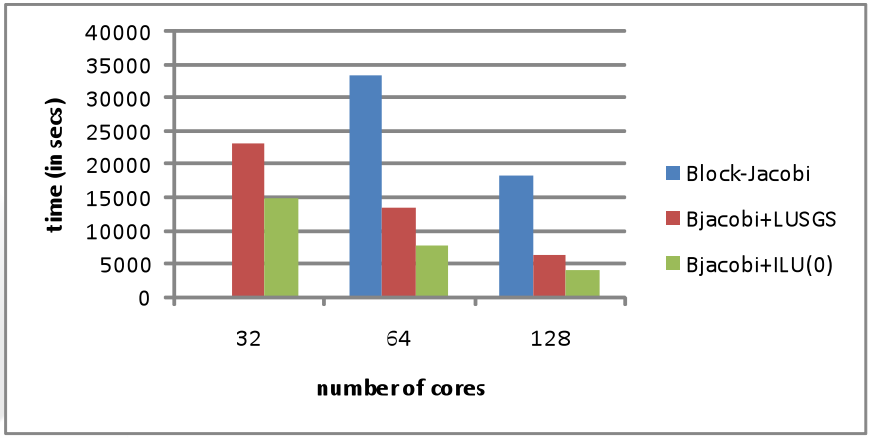
Perfomance (Re=1000)

Number of Linear Iterations

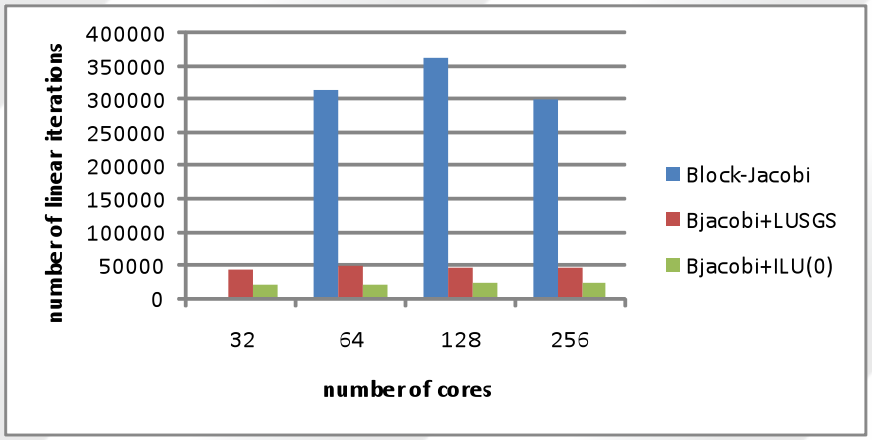
Elapsed time taken by solver

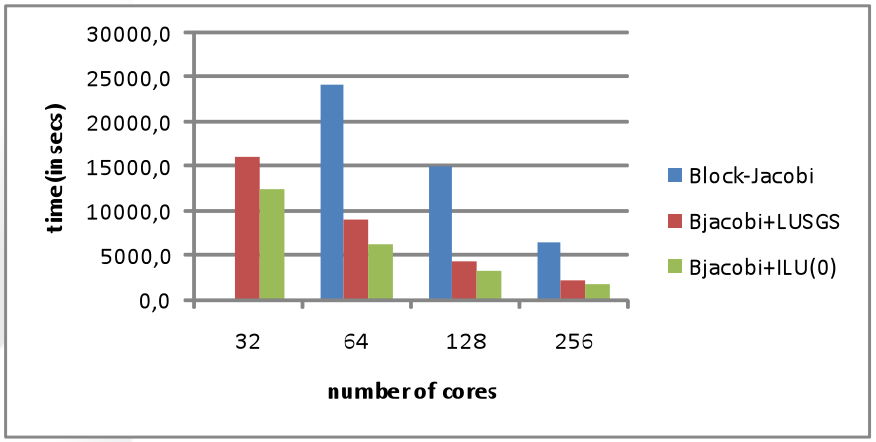
Uranus





Ranger

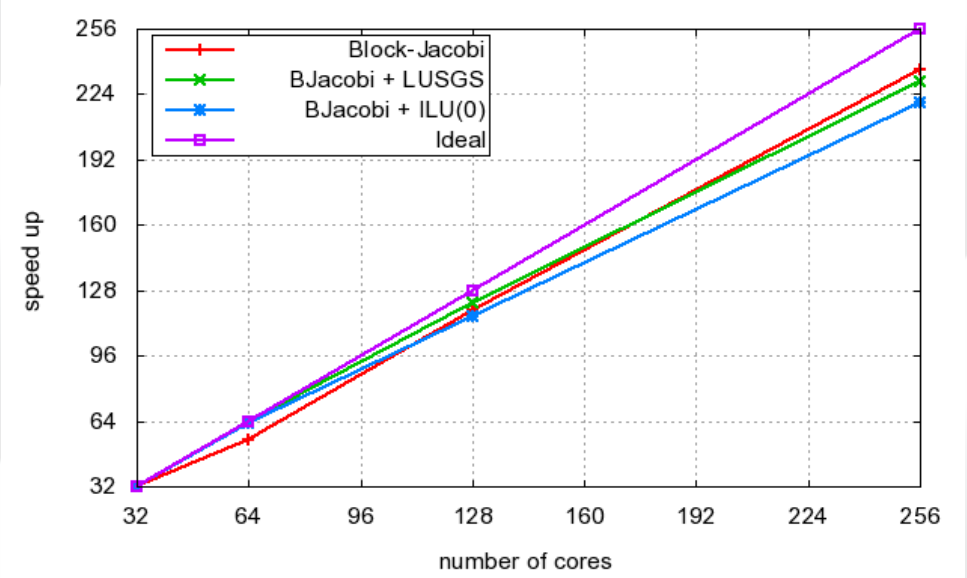




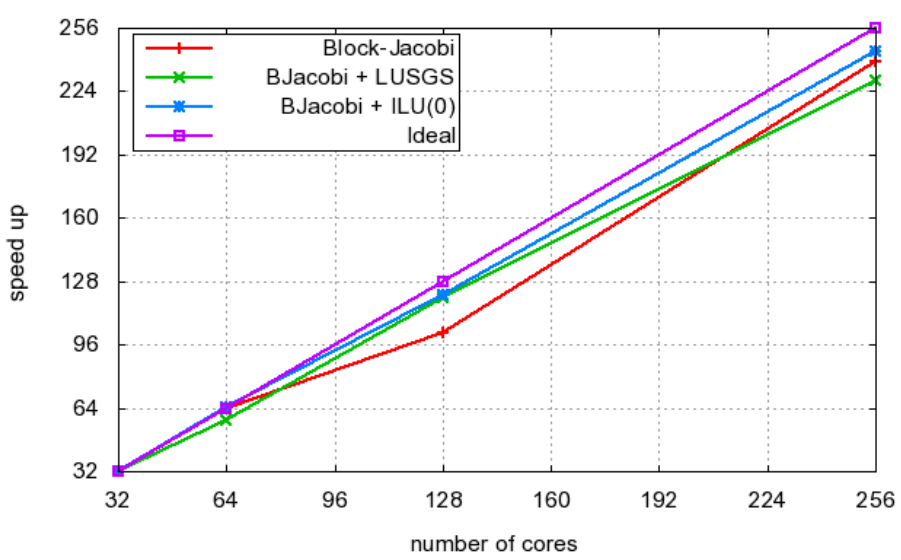
BJacobi+ILU(0) has the smaller computational effort
Block-Jacobi did not converge with 32 processors
Block-Jacobi requires 6 times more iterations than BJacobi+LU-SGS

Strong Scalability

- **Speedup roughly proportional to the number of used processors**



Re=100



Re=1000

In all schemes, speedups close to ideal

Memory Requirements

- **memory consumption (in MB) for the preconditioners and the number of equations per processor (NDOFS)**

Procs.	*NDOFS*	*BJacobi*	*Bjacobi+LU-SGS*	*Bjacobi+ILU(0)*
32	58844	148.9	148.9	282.5
64	28937	71.6	71.6	135.9
128	15699	40.0	40.0	73.2
256	7944	19.5	19.5	36.0

ILU(0) demands almost two times the memory since it needs to store the incomplete matrix factorization

AMR/C and Solver Performance

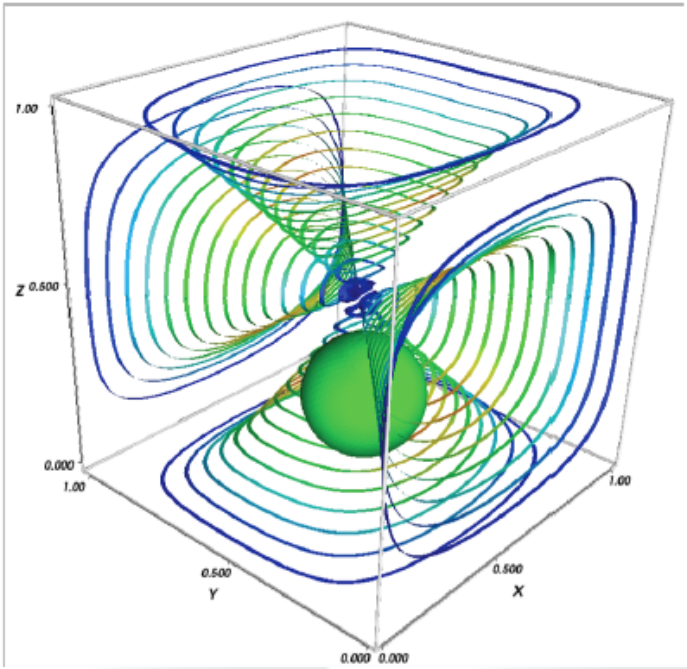
- **Implicit time integration schemes require a solution of large and sparse linear system**
- **Krylov subspace methods + preconditioning strategies**
  - Incomplete LU factorizations methods
  - Powerful method in terms to improve convergence
  - Complex parallel implementation
- **Making ILU more suitable for parallel architectures**
  - Block preconditioners with local ILU factorizations
  - Domain decomposition preconditioners: Additive-Schwartz and
  - Block Jacobi
- **How the choice of block ILU affects the simulation performance using AMR/C using different unknowns orderings**

Domain Decomposition Preconditioners

- **Provide high level of concurrency and are very simple to implement**
- **Permits combination with incomplete LU factorizations**
- **Additive Schwarz**
  - Each processors solves a local subsystem including bordering variables
  - Each block overlaps to neighboring block the amount δ levels
- **Block-Jacobi**
  - Zero-overlap form of Additive-Schwarz.
  - Non-overlapping ILU factorizations is performed over the main diagonal block of local part of the A.

The three-dimensional deformation problem

- The sphere is deformed by vortices and stretched out very thinly.
- The velocity field is given by

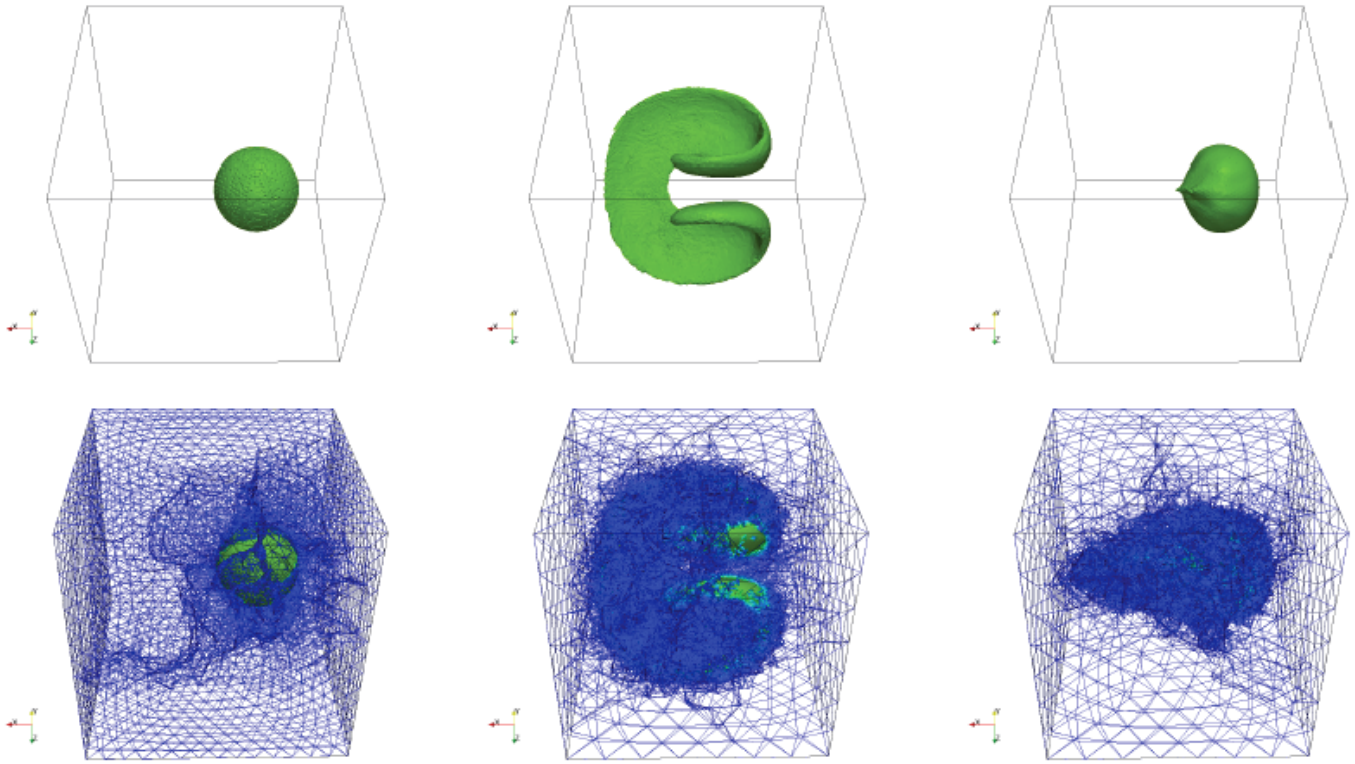


- AMR Parameters:
  - Initial Mesh: 1,130,949 linear tetrahedra
  - Refine fraction: 0.9
  - Coarse fraction: 0.1
  - $h-level$: 4
  - AMR every 5 time steps
- Solver Parameters:
  - GMRES(30) - relative tolerance 10^{-6}.
  - Preconditioners: ASM $(1-level)$ and Block-Jacobi
  - Local ILU(0) and ILU(1)

SUPG FEM with linear tets

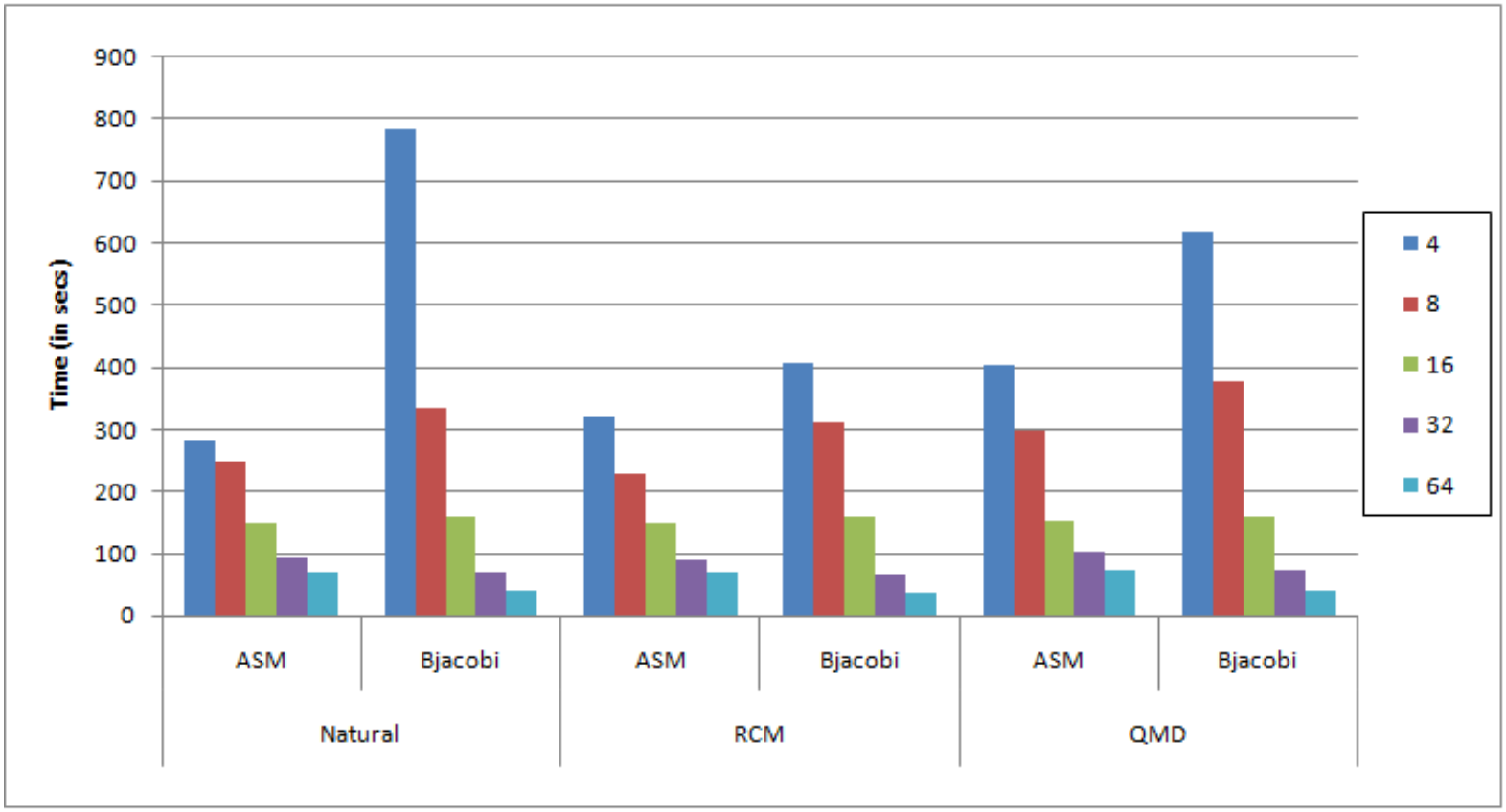
Results 3D deformation problem

- Solution and mesh configuration of the 3D deformation problem at three times, $T = 0$, $T = 1.5$, $T = 3.0$ time units.



Results 3D deformation problem

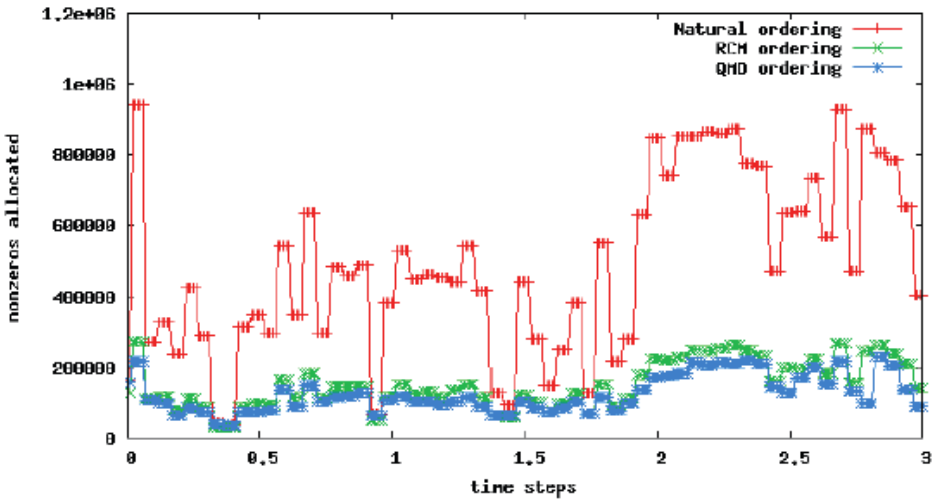
GMRES(30) + local ILU(0) performance using Block-Jacobi and ASM on Altix ICES 8200 [1]



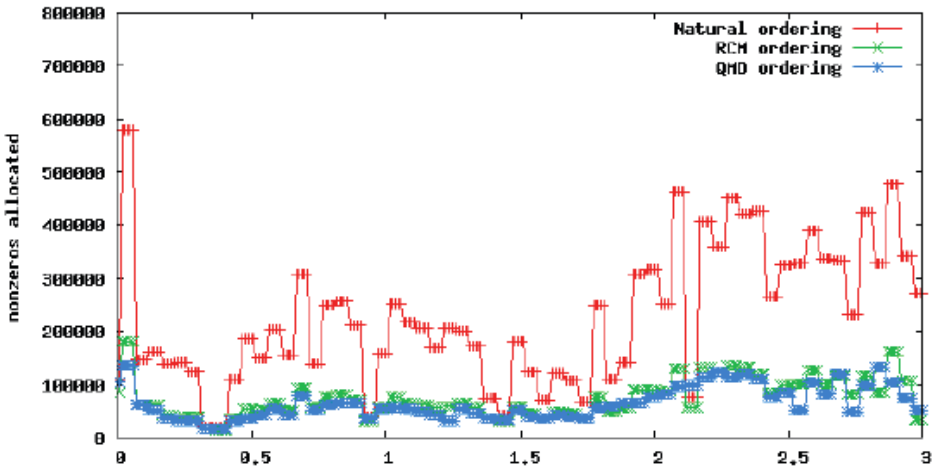
[1]Each compute node has two Intel Quad-Core 2.66GHz, 8MB L2 cache on-die and 16GB of memory

Nonzeros allocated using ILU(1) preconditioner and 64 processors

ASM

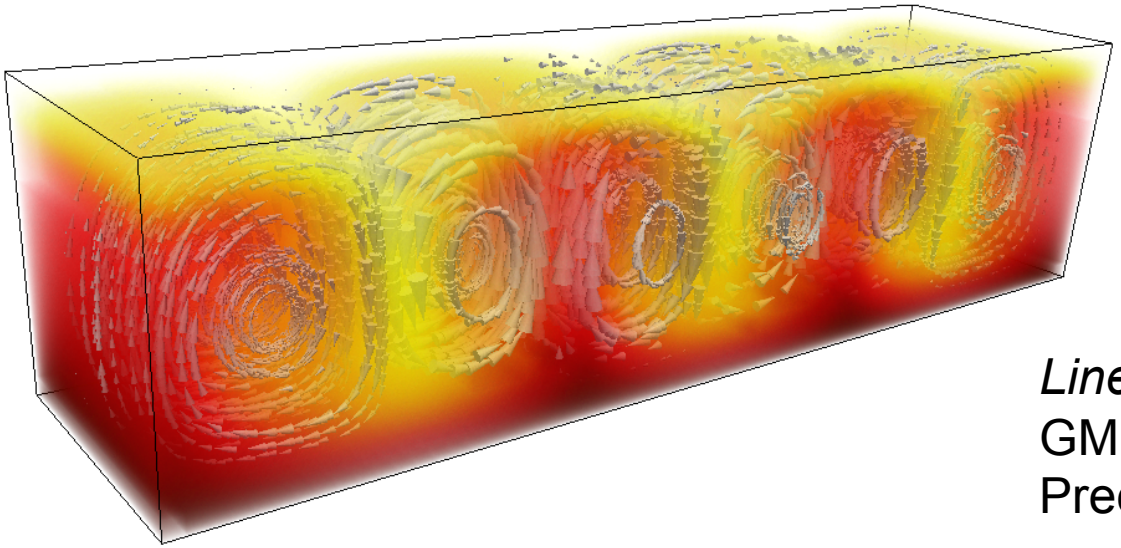


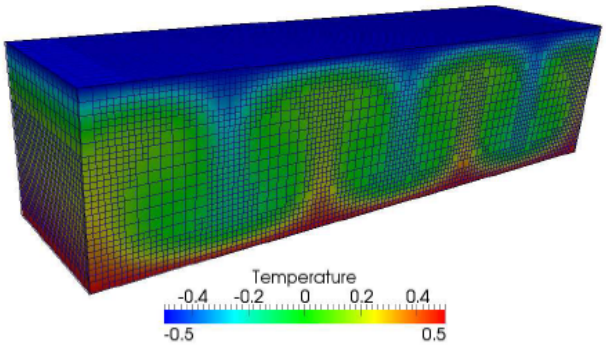
Block-Jacobi



Restricted 3D Rayleigh-Benard flow

- Rayleigh-Benard flow in a container.





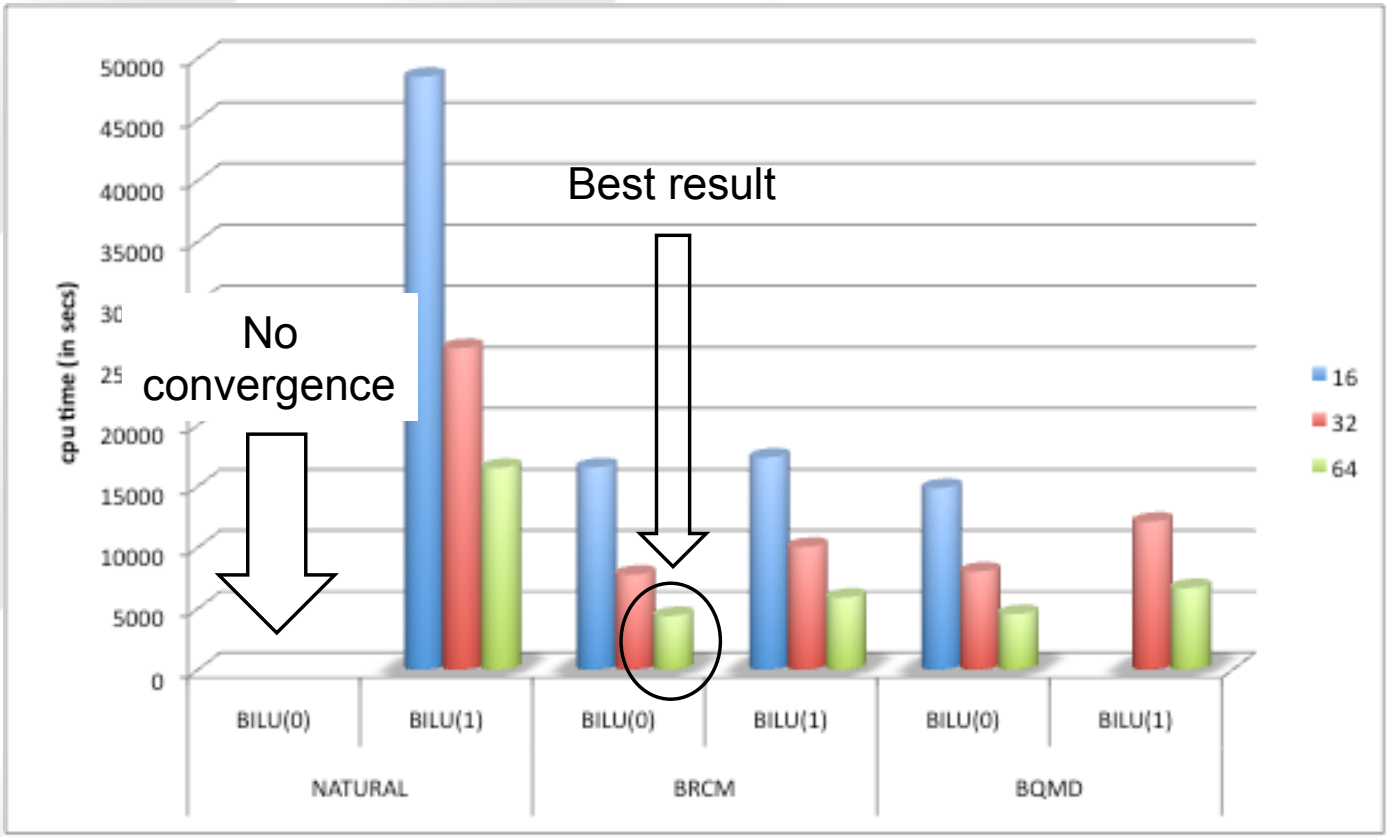
Linear Solver parameters:
GMRES(30) – Tol: 10^{-6}
Preconditioners: BILU(0) e BILU(1)

Problem parameters:
Reynolds number: 2,025.8
Rayleigh number: 30,000
Prandtl number: 0.72

AMR parameters:
Initial mesh: 64 x 16 x 16 linear hexahedra
Refine fraction: 0.5
Coarse fraction: 0.01
h-level: 1

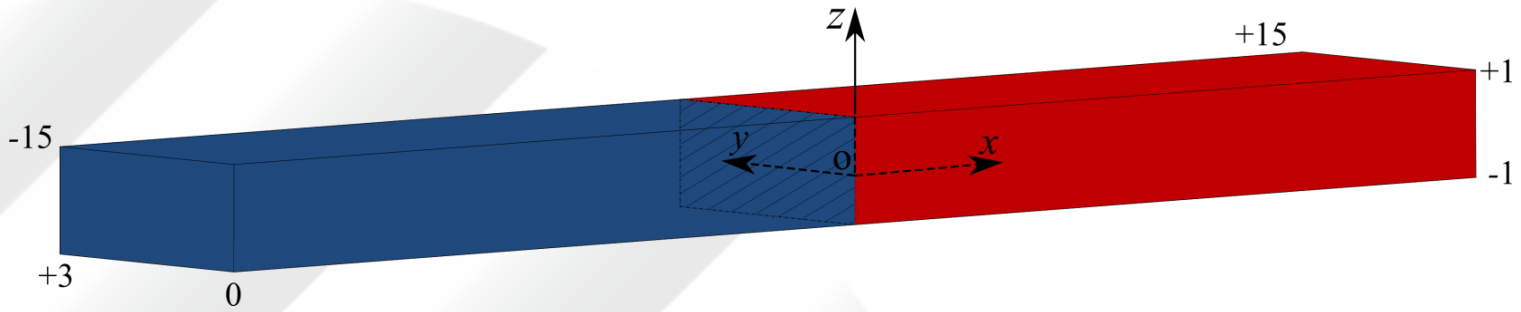
Restricted 3D Rayleigh-Benard flow

- **Ordering evaluation: CPU time for the preconditioned linear solve iterations in the Navier-Stokes solver**



Computed at TACC's Ranger

Planar 3D Lock-Exchange with AMR/C



Geometry, boundary and initial conditions:

- ✓ Simulation Domain: $\Omega = [-15,15]\text{x}[0,3]\text{x}[-1,1]$
- ✓ No-slip on top and bottom walls; slip on all others
- ✓ Half right filled with “heavy” fluid $(\rho_h = 1)$ and left half filled with “light” fluid $(\rho_l = 0)$

Dimensionless parameters:

$\text{Gr} = 1.5 \times 10^6 \qquad Sc = 0.71$

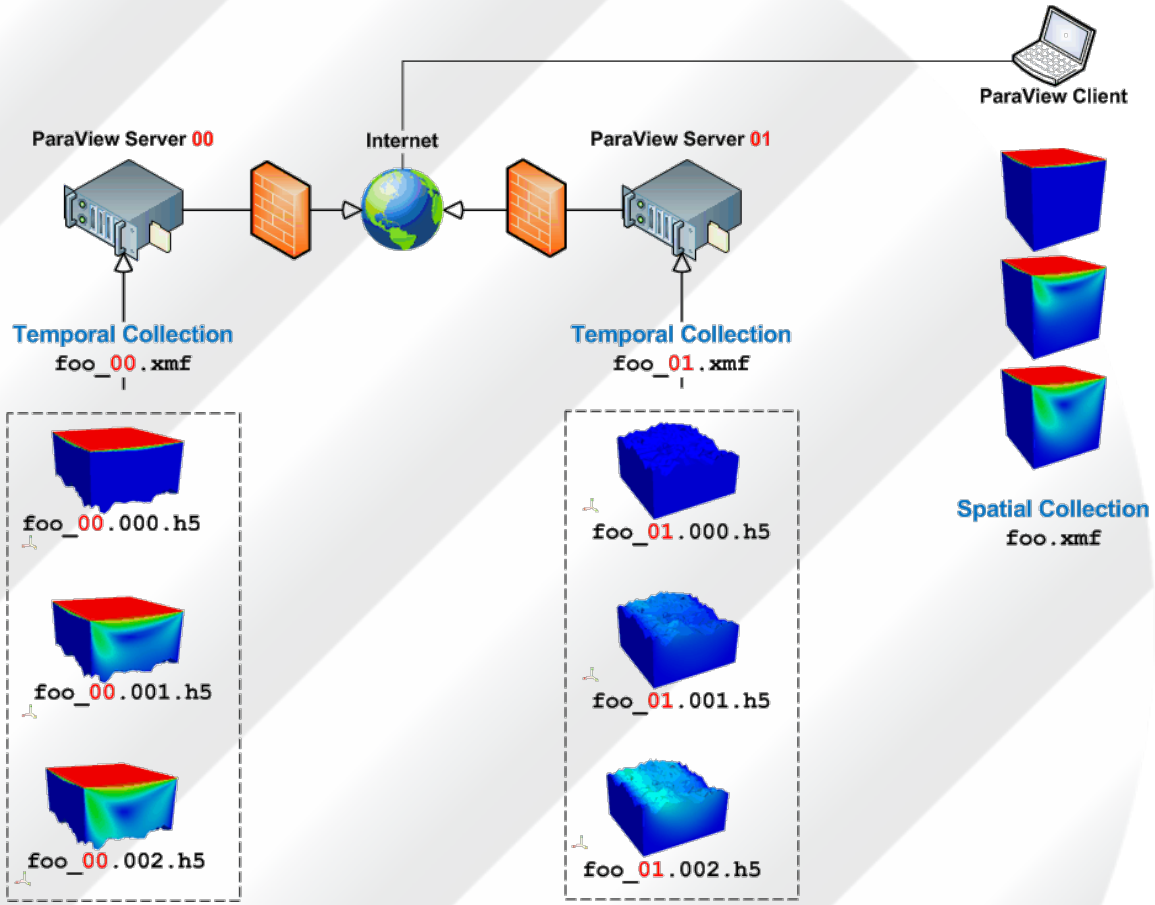
$\Delta t = 0.025$

Run on 240 cores, Viz with ParaView, parallel rendering

GMRES(30)
BILU(1)
RCM reordering

See also Camata et al, IJNMF, 2012

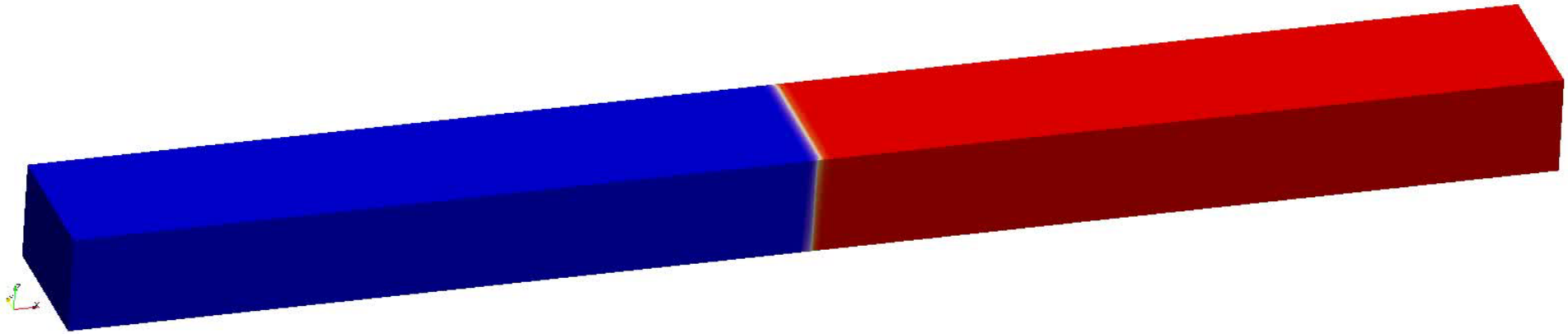
Parallel Visualization

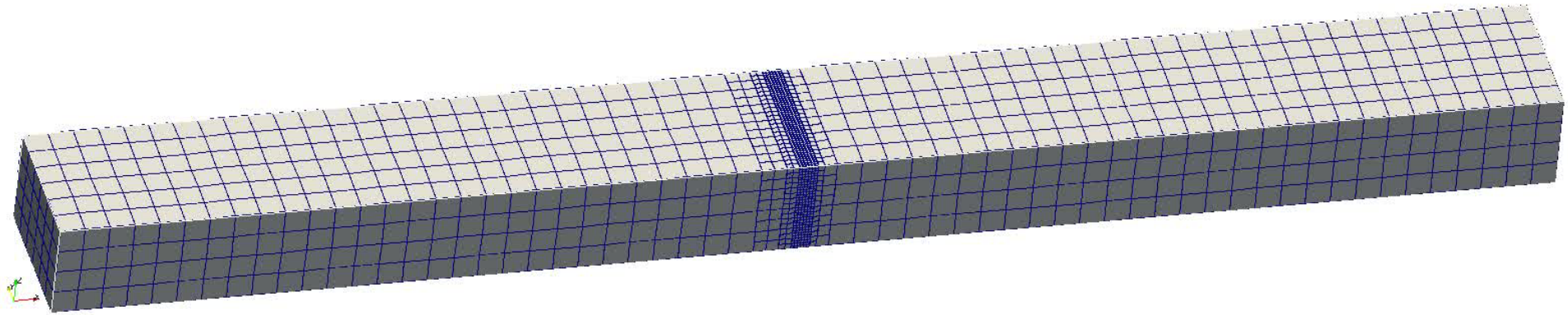


Data for ParaView Parallel Remote Visualization in XDMF/HDF5 formats
Spatial collection of temporal collections

More details: Elias et al, ParCFD09

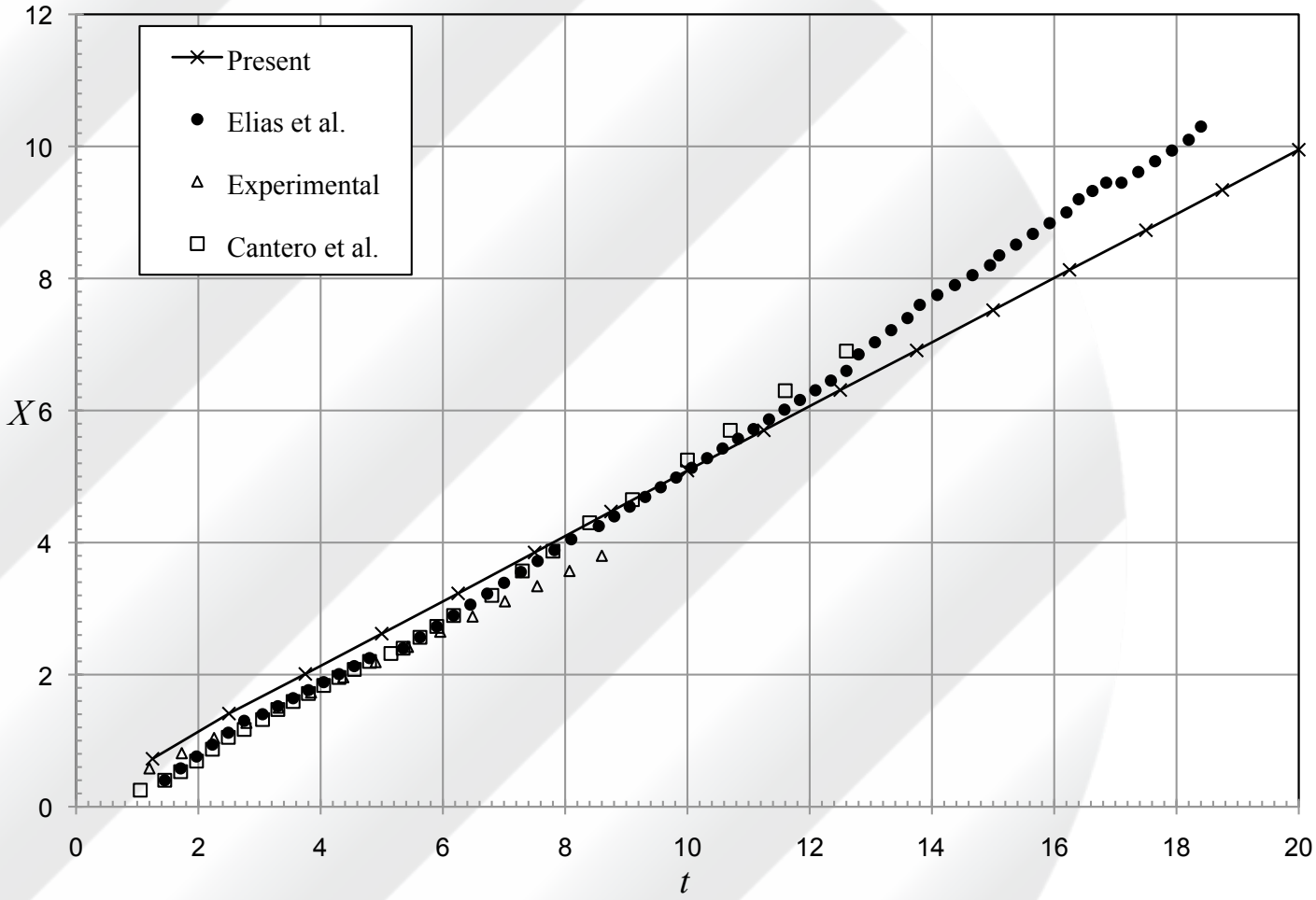
Planar 3D Lock-Exchange with AMR/C





Planar 3D Lock-Exchange with AMR/C

$X = |x|$ Front evolution



Froude Number

$$Fr = \frac{u_f}{u_b}$$

Work	Fr
Present	0.505
Elias *et al.*	0.579
Härtel *et al.*	0.576
Cantero *et al.*	0.570
Experimental	~0.48

PARALLEL OCTREE MESH GENERATION

Parallel Octree Mesh Generation

- **Growing availability of parallel machines**
- **Improvements on scalability of numerical solutions (FEM, FVM)**
- **Scalable Octree meshing and solver strategies**
  - Dendro lib: scales oup to 4,000 cores (Sundar et al, 2007)
  - Tu et al, runs in 67,000 cores (2005)
  - p4est: scales up to 220,320 cores, Burstedde et al (2011)
  - Park & Shin (2012): octree meshes using GPGPU
- **Our objectives**
  - Present a parallel octree generator able to representing arbitrary surfaces
  - Extract conforming tetrahedral meshes from the resulting octree
  - Perform a parallel scalability analysis

Linear Octree

- An octree is a tree data structure in which internal nodes has exactly eight children [3] (Figure 1)
- Often used to partition a three dimensional domain space by recursively subdividing it into eight octants
- Linear Octree
  - ✓ complete list of leaf nodes
  - ✓ octants are encoded by a scalar key: Morton Code
- Advantages:
  - ✓ do not require to store internal nodes
  - ✓ reduce overhead associated with pointers use

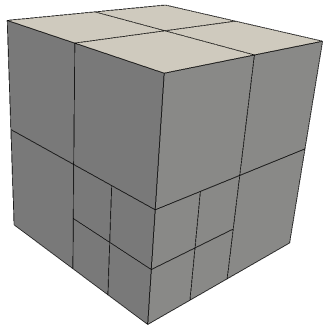


Figure 1: Octree – sudivision of a cube into octants

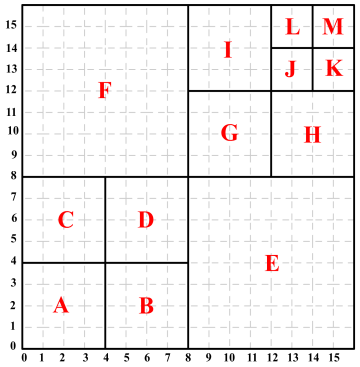


Figure 2: Quadtree - 2D structure

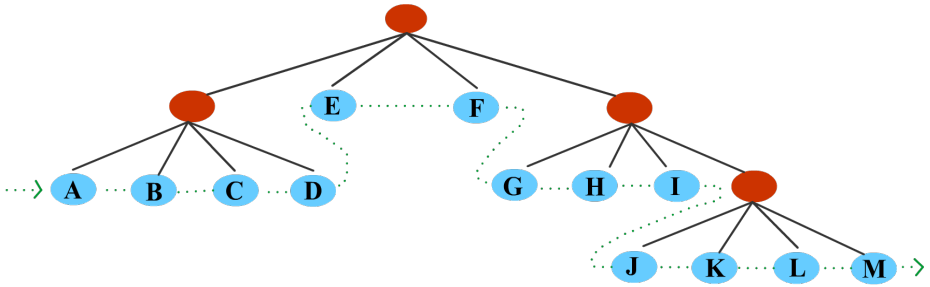
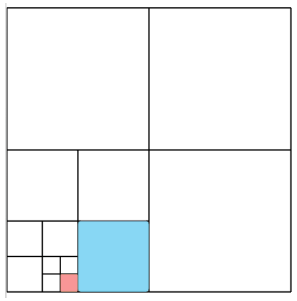


Figure 3: tree structure

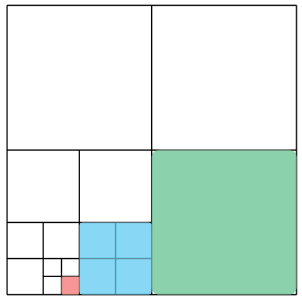
Morton code
- ✓ D's left lower corner: (4,4)
- ✓ Binary representation: (100,100)
- ✓ Interleave the bits: 110000
- ✓ Append D`s level: 110000 ← **10**

Octree Algorithms

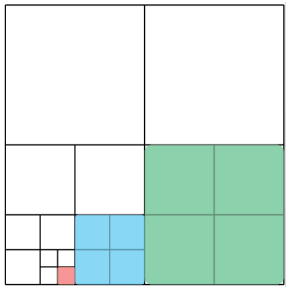
- Octree Construction
  - ✓ generate a list of octants equally distributed between the cores.
- Octree partition
  - ✓ redistribution of the octants among processes with the objective to reach load balance
  - ✓ each process stores a contiguous chunk of leaf octants
- Octree Refinement
  - ✓ non-recursive refinement algorithm transverses all leaf octants for each local octree replacing an octant with its eight children
- 2:1 Balancing
  - ✓ no leaf octants sharing at level l shares an edge or face with another leaf at level greater than l + 1



Unbalanced octree



Riplet octree



balanced octree

- Meshing
  - ✓ Find anchors and hanging nodes
  - ✓ Find sharing nodes
  - ✓ Get element connectivity

Manipulating STL surfaces

- Necessary to represent complex solid boundaries present in computational solid and fluid mechanics applications
  - ✓ Surfaces are given by a triangulation, typically obtained from CAD package as STL files
- Finding interceptions is based on bounding box threes (BBT)
  - ✓ Advantage: Allows fast overlap rejection test
  - ✓ Computational cost: O(*nlogn*)
- Finding octants located inside the solid boundaries is based on ray tracing

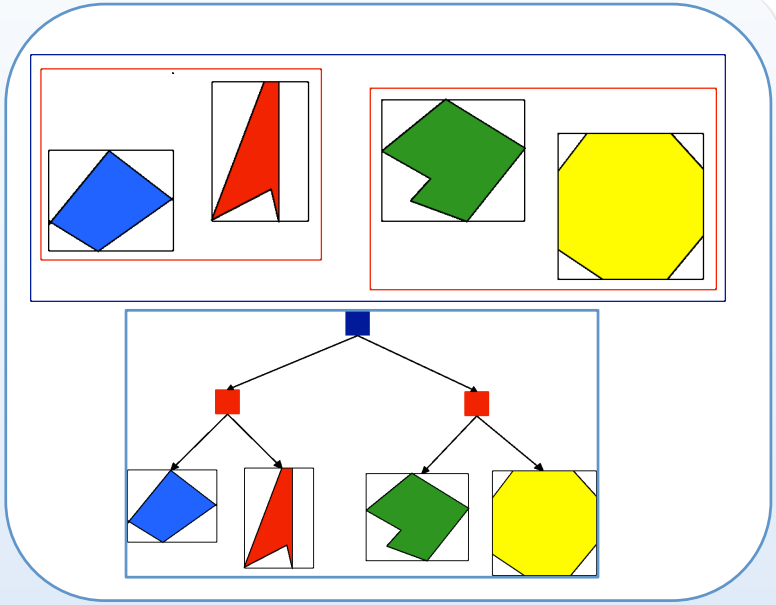


Figure 4: Bounding Box Tree scheme

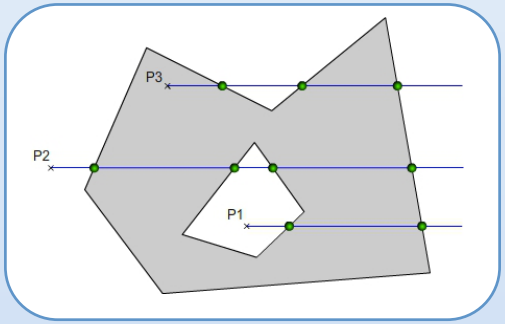
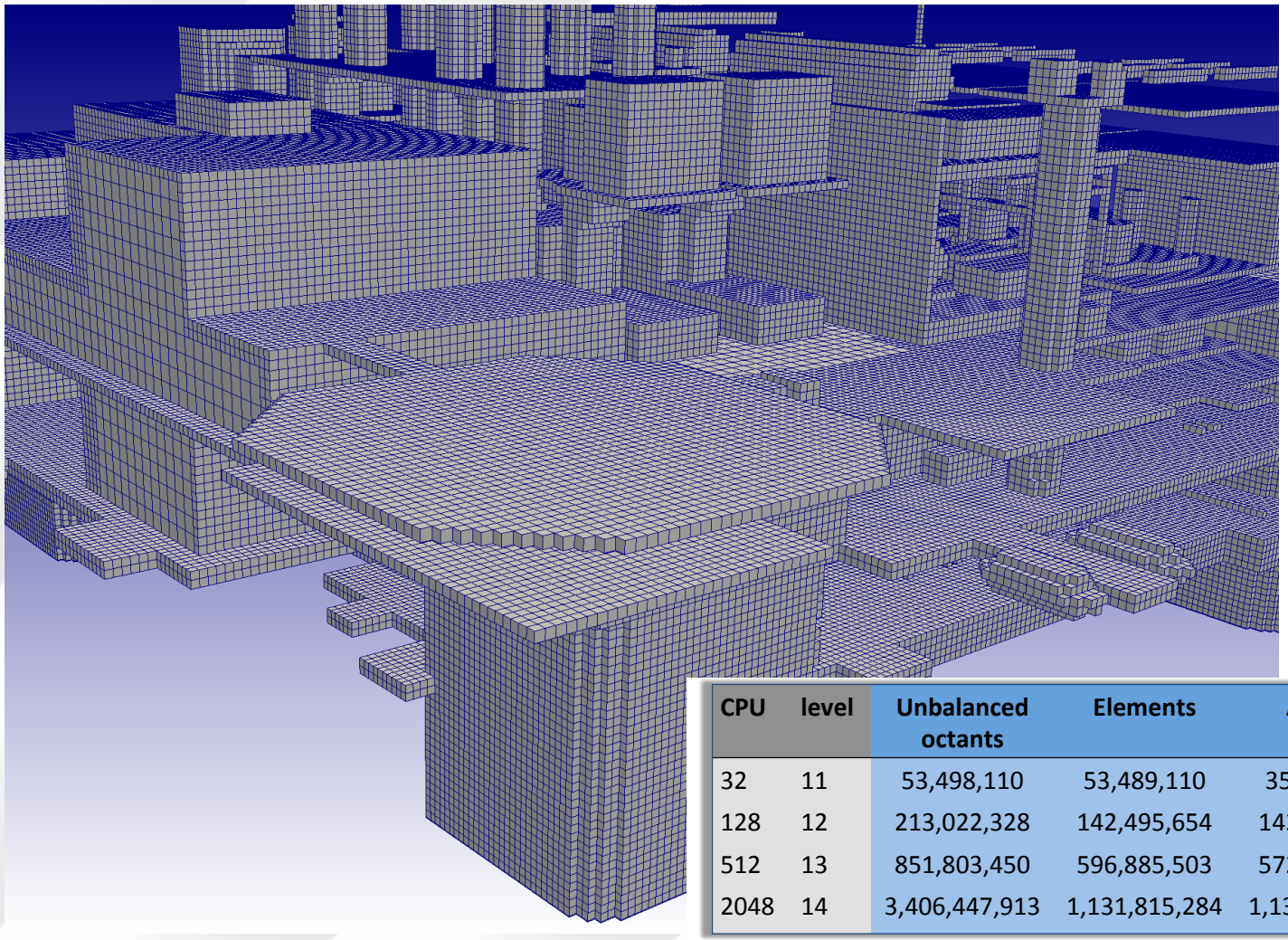


Figure 5: Ray trancing

Parallel Octree Mesh Generation



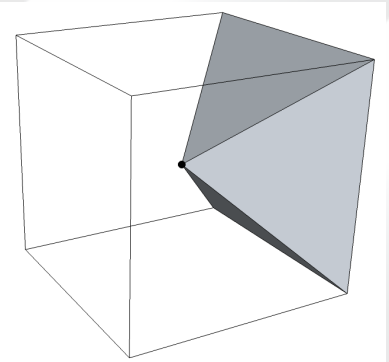
Our scheme was able to generate 3.4 billion octants in less than 10 seconds per 1.6 million octants per core

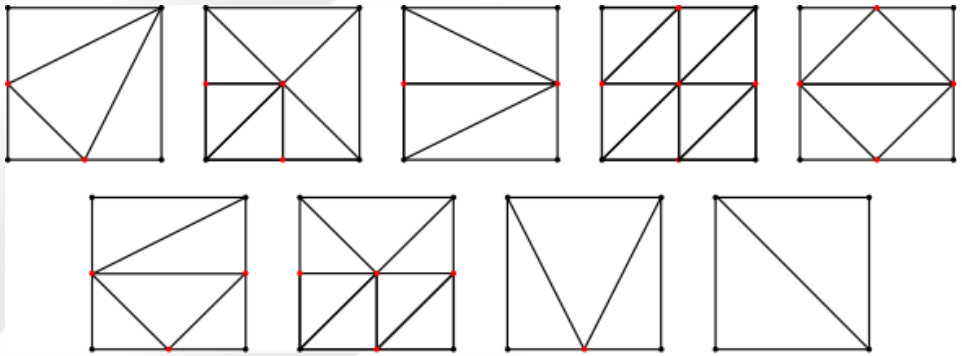
CPU	level	Unbalanced octants	Elements	Anchor nodes	Hanging nodes	Total Time
32	11	53,498,110	53,489,110	35,931,290	15,482,795	106.405
128	12	213,022,328	142,495,654	143,116,154	61,562,108	135.249
512	13	851,803,450	596,885,503	572,099,070	246,023,557	169.129
2048	14	3,406,447,913	1,131,815,284	1,136,125,418	488,655,212	194.051

See Camata and Coutinho, CCPE, 2012

Conforming Techniques

- **FREY e GEORGE (2000) / BERG et. al (1998)**
  - Decompose octants in 6 pyramidal elements by inserting a central node
  - Define 9 templates for face triangulation
  - Connect all face nodes with the central node

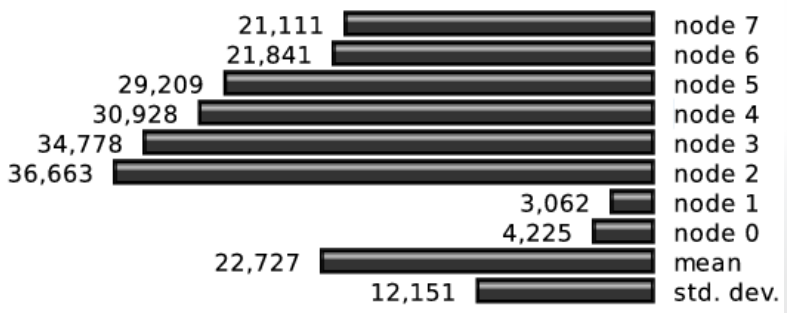




  - Does not require modifications in the octree construction
  - Embarrassing parallel (does not need neighboring info)
  - Templates for all possible hanging nodes configuration

Improving Refinement Load Balancing

- **STL intersection evaluation time**



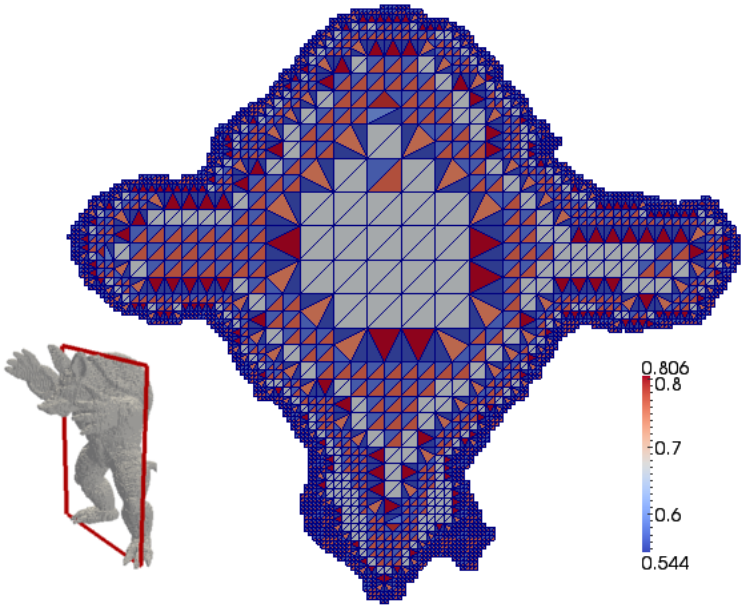
- **Partitioning computational cost**

Num. calls	Refinement	Partitioning
8 cores		
2	45.53	218.11
6	28.85	211.25
9	28.74	213.63
128 cores		
2	12.28	27.66
6	6.18	28.11
9	4.73	28.01

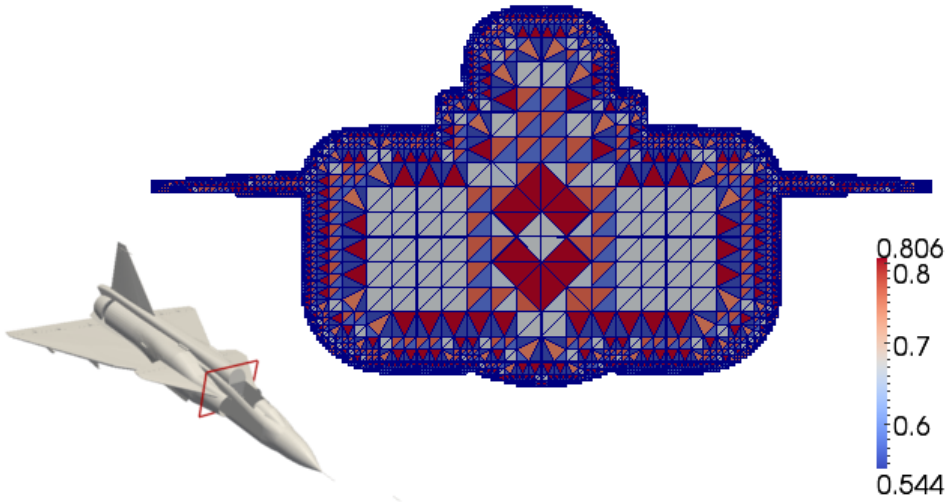
SGI Altix ICE 8400

Mesh Quality

- **BRANETS and CAREY (2005):** $Q_0 = \frac{72\sqrt{3}V}{(\sum_{i=1}^{6} l_i^2)^{3/2}}$



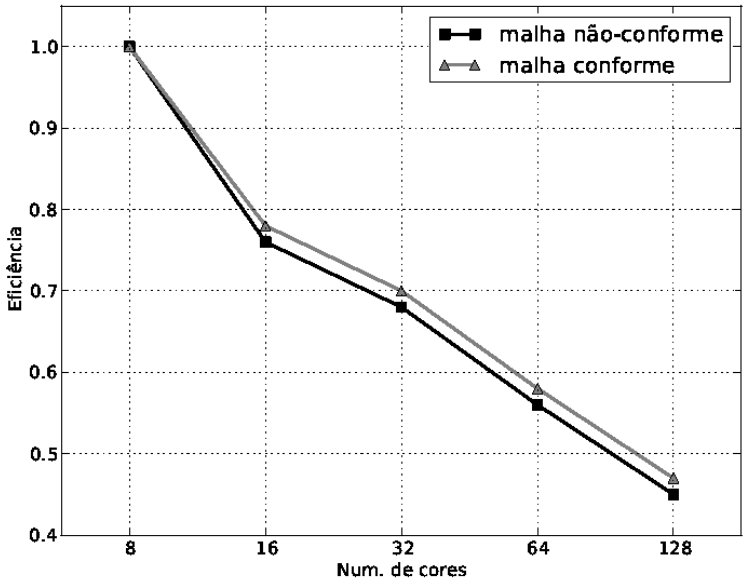
Armadillo



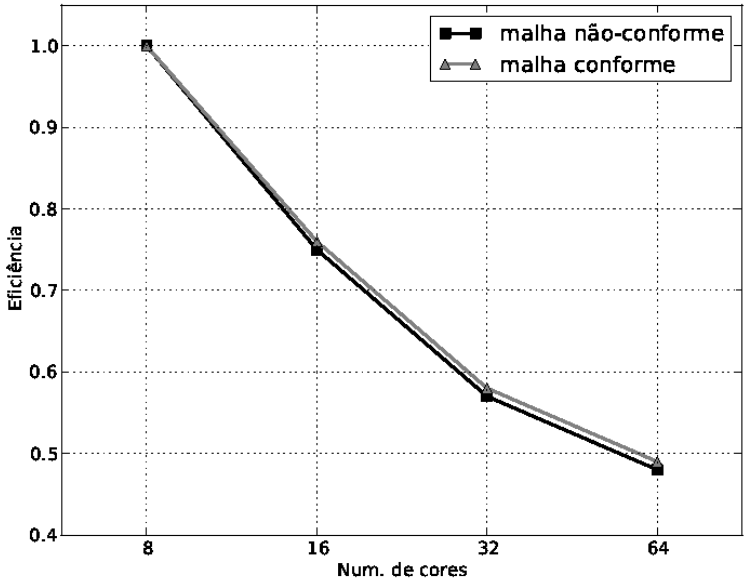
Viggen

- **Mesh quality is predictable due to templates**

Strong Scalability



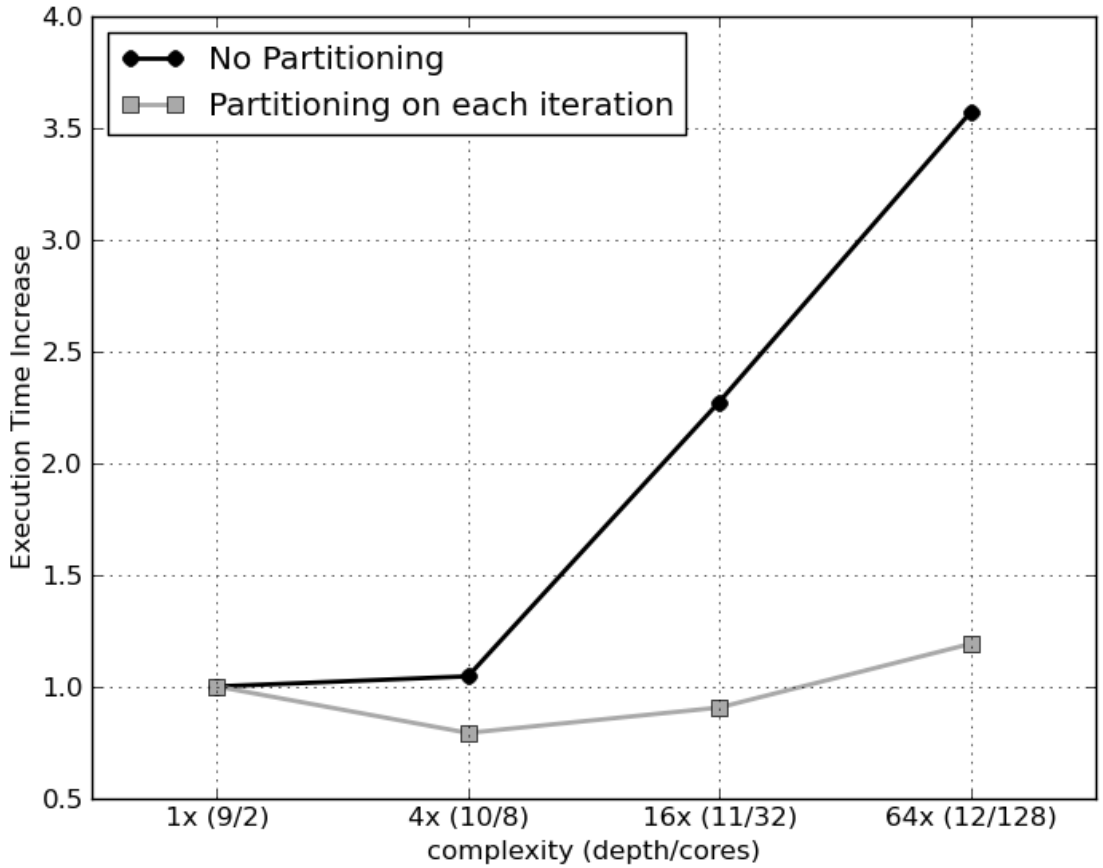
Armadillo 11 ref. levels



Viggen 12 ref. levels

SGI Altix ICE 8400

Weak Scalability Armadillo 11 levels

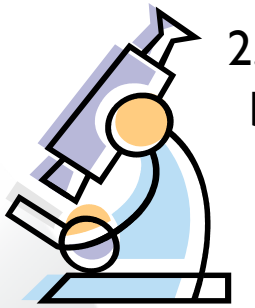


SGI Altix ICE 8400

SCIENTIFIC WORKFLOWS AND MANY-TASK-COMPUTING

Typical scenario: scientific experiment

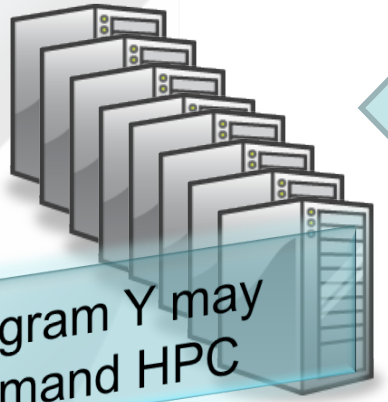
1. Data collection



2. Data analyzed by program X

3. Large Volume of Data Produced ...

4. ...which need to be processed by program Y in a cluster



Program Y may demand HPC

5. Results are analyzed by program Z

This scenario demands the execution of many programs as a chain of activities that may be assisted by scientific workflows

Parallelization difficulties

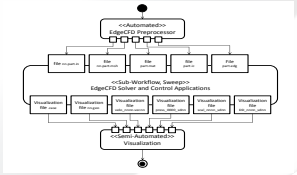
- Controlling parallel execution in distributed environments
- Steering activities in distributed environments
- Provenance gathering in distributed/ heterogeneous environments

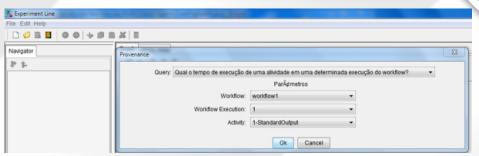
Provenance can support analyzing scientific experiments

- **Before execution:**
  - What programs may be used? Is there any alternative to explore?
  - Is there any dependency between activities? Which activities are mandatory?
- **After execution:**
  - What were the parameters that lead the best result?
  - What was the scientific workflow that lead to the desired result?
  - Where are the output files generated by the distributed activity A using the parameters P?
  - How many times the activity A in version V was used in the experiment E?

all these queries are related to the ability of reproducing and validating a scientific experiment

Our vision of the experiment life cycle

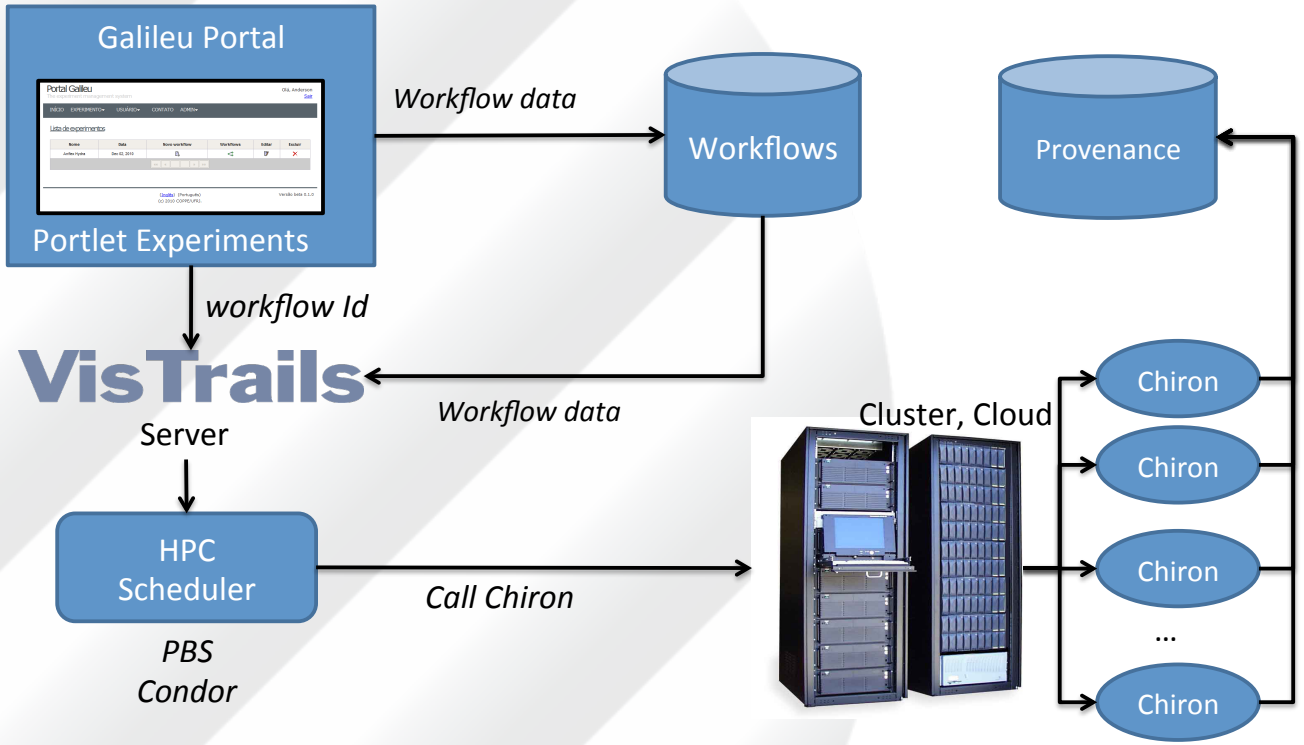




see Mattoso et al, IJ Business Process Integration and Management, 2010

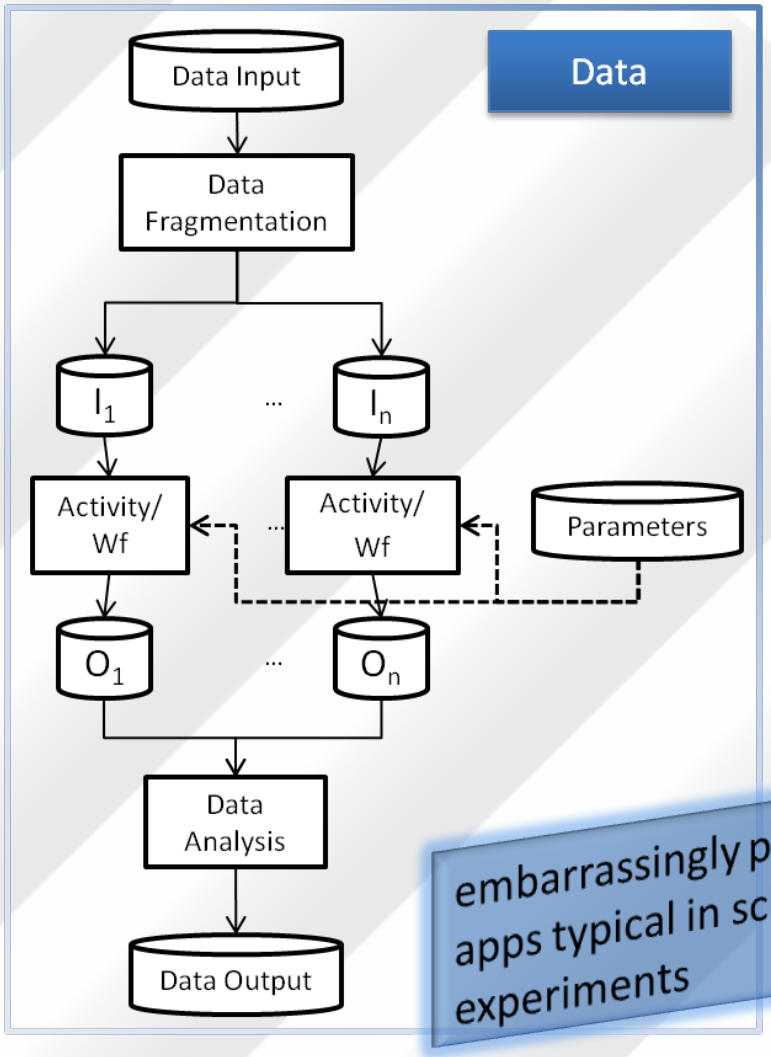
Workflow Execution

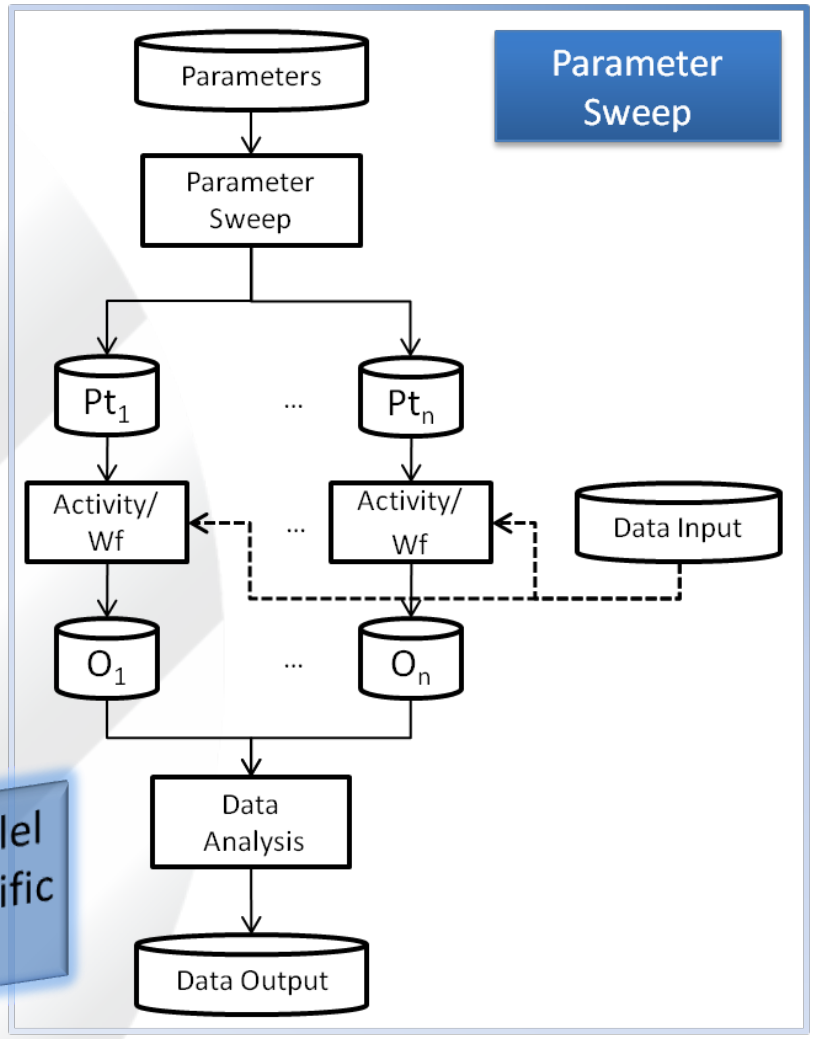
- Chiron middleware solution that bridges the SWfMS to the HPC supporting MTC parallelization strategies



- Goal: reduce the complexity involved in designing and managing activity/ workflow parallel executions while gathering distributed provenance data

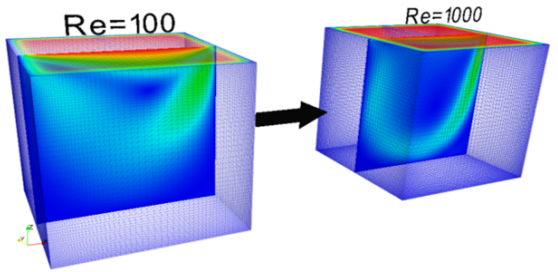
Parallel paradigms supported in Chiron





embarrassingly parallel apps typical in scientific experiments

Parameter sweep VT Workflow using Chiron

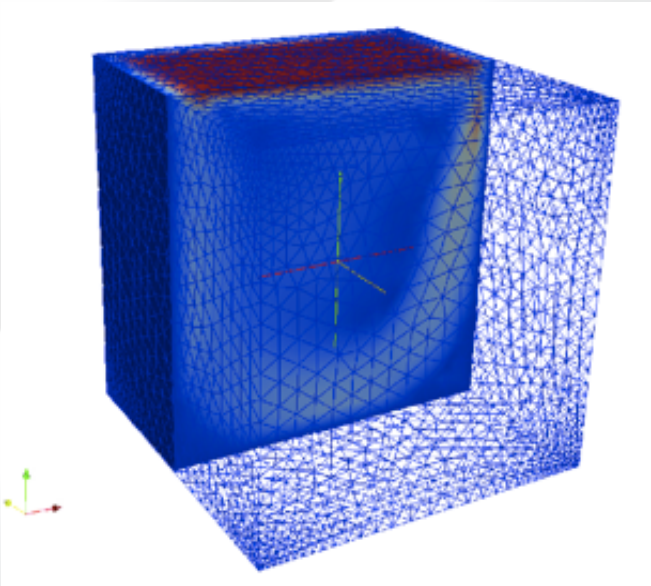


Lid-driven Cavity

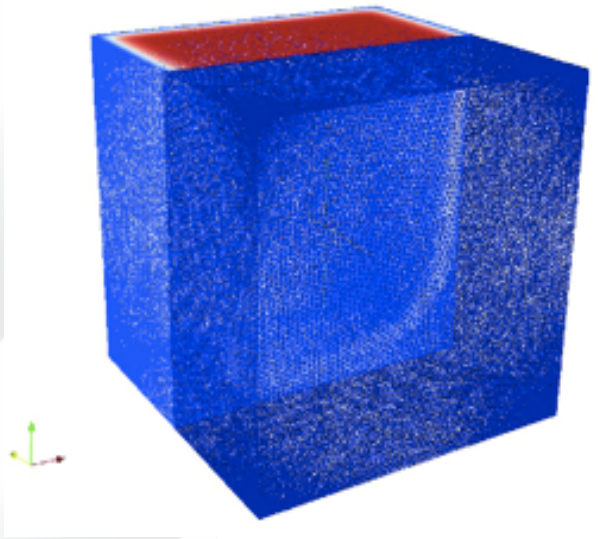
See Ogasawara et al, MTAGS, 2009, Ogasawara et al, VLDB 2011

Large Eddy Simulation Benchmark

- Lid-driven cavity flow at Re=12,000, T=360s



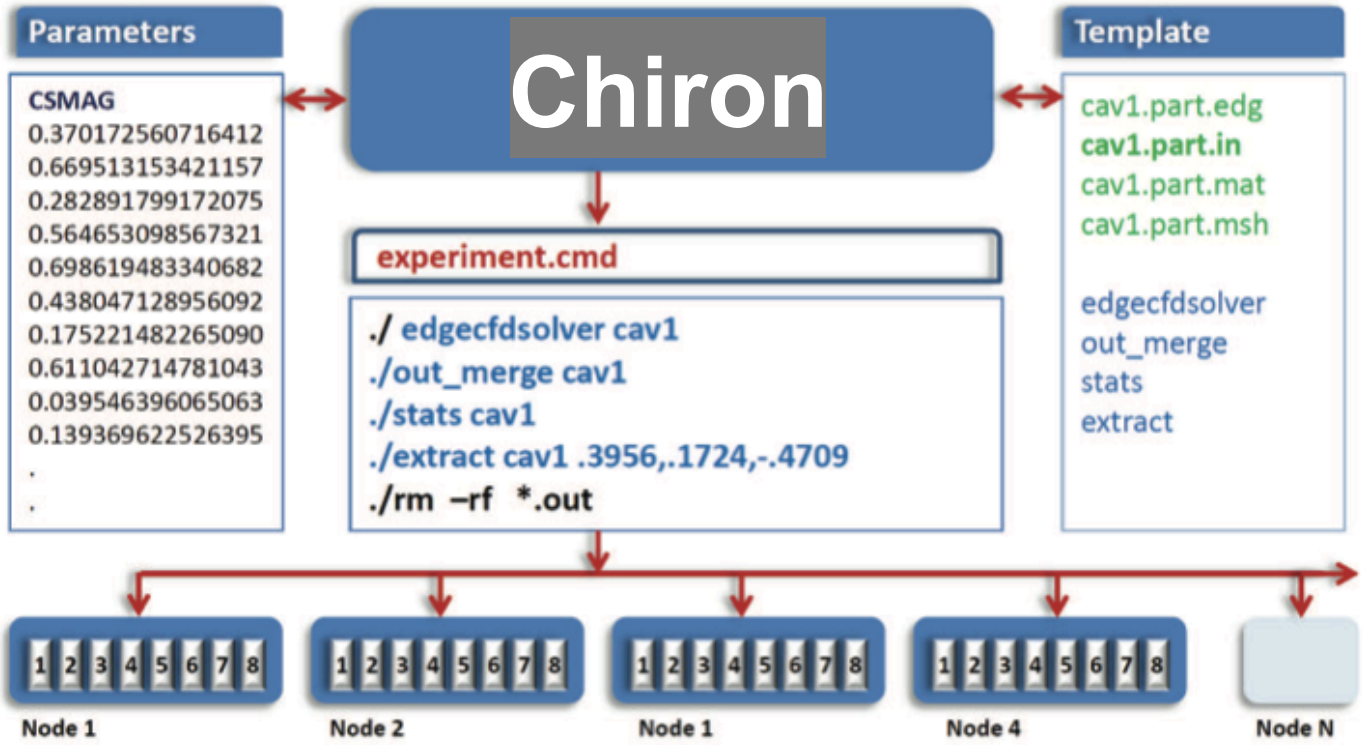
Coarse mesh 32^3 TET4 elements



Fine mesh 128^3 TET4 elements
Reference solution

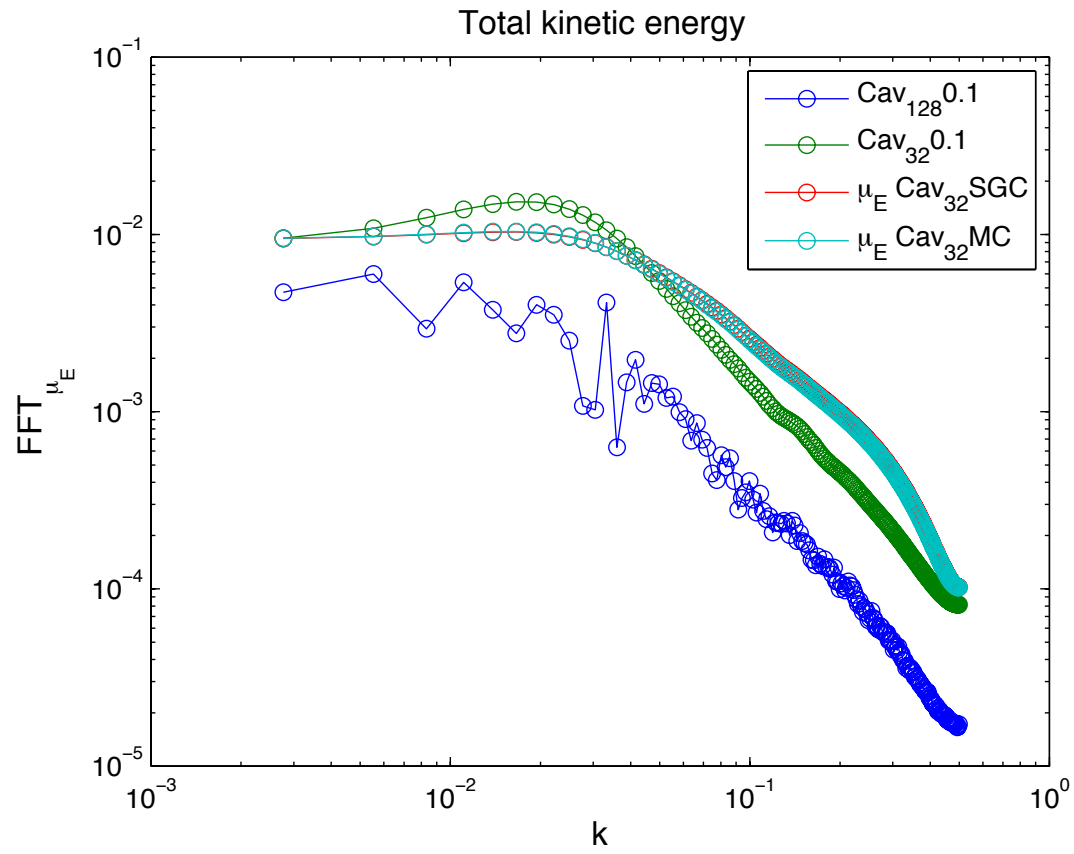
- MC reference solution with 1,000 samples, 126hs in 32 cores
- SGC with 8 approximation levels, 257 support nodes, 39hs (32 cores)
- Difference between the mean of the solutions: $O(1.0x10^{-2})$

Large Eddy Simulation Workflow Parallel Execution in Chiron



- Provenance gathers data to be used to evaluate statistical moments
- We can track the error between MC and SGC and determine an interpolation level to satisfy a minimum error
- Provenance capabilities in Chiron can automatically increase the interpolation level and resubmit the experiment to obtain statistical moments with a prefixed error

LES UQ Analysis



Kinetic energy FFT for MC and SGC solutions

Dissipation Mechanism : where numerics should match the physics

See: Guerra et al, IJUQ, 2012

CONCLUDING REMARKS AND DISCUSSION

- **Advanced Algorithms and Solvers**
  - Advances in nonlinear solvers and time step controllers
  - Scalable solvers for complex engineering problems implemented
  - Communication issues of paramount importance
  - Adaptivity introduces an extra complexity layer
- **Parallel Octree Mesh Generation**
  - Scalable, handles arbitrary immersed surfaces, generates hex and tet meshes
- **Scientific Workflows**
  - Experiments life cycle must be managed as a whole:
    - *Composition, Execution and Analysis*
    - *Prospective and retrospective provenance should be gathered*
  - Hydra can be a bridge between the SWfMS and the HPC environment
    - *Supports workflow data and parameter sweep parallelization*
    - *Evaluated in a real case CFD solver with little overhead*
    - *Supports distributed provenance gathering*

Acknowledgements:

- Faculty:
  - HPC: A. Coutinho (PI), R. Elias
  - Civil Engineering: J. Alves
  - Mechanical Engineering: F. Rochinha
  - Computer Science: M. Mattoso, V. Braganholo
  - External Collaborators: G. Carey, ICES, UT Austin, B. Barth, TACC, UT Austin, A. Valli, L. Catabriga, UFES
- Pos-Docs, Research Staff and Students:
  - C. Alvarez, A. Aveleda, C. Barbosa, D. Barcarolo, O. Caldas, J. Camata, A. Cortes, M. Costa, A. Cruz, G. Guerra, M. Goncalves, J. Gonzalez, N. Guevara Jr, A. Mendonça, E. Lins, R. March, E. Ogasawara, D. Oliveira, F. Seabra, P. Sesini, C. Silva, S. Zio, Igor Ghisi
- Funding: Petrobras, ANP, MCT/CNPq, MCT/FINEP
- Computer Resources: NACAD/COPPE/UFRJ, TACC/UT Austin