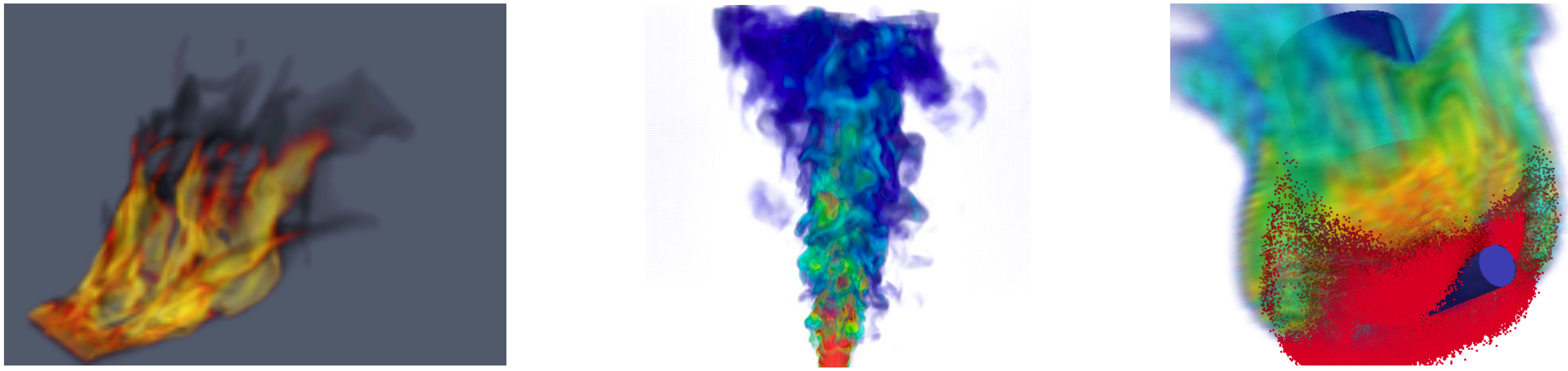
SAND2012-5312C

Exceptional service in the national interest

Sandia
National
Laboratories



A Performance-based Code Assessment for Low Mach Large Eddy Simulations

Stanford University

July 11th, 2012

Stefan P. Domino

Thermal/Fluids Computational Engineering

Sandia National Laboratories, NM

Abstract

A Performance-based Code Assessment for Low Mach Large Eddy Simulations

Stefan P. Domino, 1541 Sandia National Laboratories; July 11th, 2012, Stanford University

A Large Eddy Simulation (LES) treatment of fluid turbulence is required for qualification efforts at the laboratory centered on aerodynamics, fire environments, and captive-carry loading. Due to the inherent unsteady nature of the typical flows within the Abnormal/Thermal, Normal and Delivery environments, LES is required for accurate environment prediction as other less expensive techniques, such as Reynolds-Averaged Navier-Stokes (RANS) simulations, have proven to be inadequate. In general, LES calculations require significantly more computing resources than the RANS calculations needed for aerodynamic design. For example, resolution of vortex/fin interactions will likely require O(200) million element meshes while the characterization of fire environments, requiring resolution of Rayleigh/Taylor instabilities to accurately capture the large-scale plume core collapse (pool diameters of 5-10 meters), typically requires sub-centimeter resolution.

Software design often requires trade-offs between generality and performance since highly specialized code can be optimized for its problem specific application space. The goal of this project is to improve the performance of an acoustically incompressible LES capability while providing adequate generality to address key needs of the Lab's customer base. Efforts have ensured that the fluid dynamics module of Sierra (Sierra/FD) is well positioned for follow on activities for scaling to hundreds of thousands of cores, and efficient operation on next generation multiprocessors.

This seminar will provide a performance-based assessment of the current ASC Sierra Thermal/Fluids code base. Detailed code performance, cast within weak and string scaling studies, will be presented. Low Mach mixture fraction-based LES scaling simulations, which have been run on meshes ranging from 2 Million elements to well over 1 Billion elements on core counts of up to 65,536, will be presented. The cost of code generality compared to specific code implementations has been explored and lessons learned will be provided. Finally, a set of new low Mach fluid discretizations and couplings will be presented with detailed verification to serve to guide future paths of algorithmic code development. A proposed path forward to resolve scaling bottlenecks and matrix assembly will be provided.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Presentation Overview

- Project Guiding Principle
- Milestone Description and Completion Criteria
- Physics, Discretization, Algorithmic Behavior and Coupling Approaches
- The Cost of Code Generality
- Performance Results
- Summary of Accomplishments
- Lessons Learned
- Proposed Path Forward

SNL Cross Cutting L2 Team

Much appreciation to the talents of….

- Development Team:
  - Matt Bettencourt (1426)
  - Jon Clausen (1514)
  - David Glaze (1541)
  - Steve Kennon (Numericus Group LLC)
  - Paul Lin (1426)
  - David Noble (1514)
  - Pat Notz (1541)
  - Greg Wagner (8365)
- Performance Consulting
  - Alan Williams (1543)
  - Kendall Pierson (1542)
- Solver Support
  - Mike Heroux (1426)
  - Jonathan Hu (1426)
- Management Support
  - Ryan Bond (1541)
  - Rob Hoekstra (1426)
- T/F Team including
  - Sheldon Tieszen (retired)
  - Kim Mish (1542)

Project Guiding Principle

- The SIERRA Mechanics Integrated Code (IC) tool suite is being developed under the Department of Energy's (DOE) Advanced Scientific Computing (ASC) program to support *Science-based Stockpile Stewardship* (SBSS)
- Other aspects of SBSS include:
  - Physics and engineering model development, creation of high quality validation data sets, algorithm development and Uncertainty Quantification (UQ)
- The guiding principle for this combined project is to provide a *predictive* capability for high consequence accident scenarios
- The ASC project deliverables are managed by Milestone efforts across the fully supported ASC application space

Milestone Description

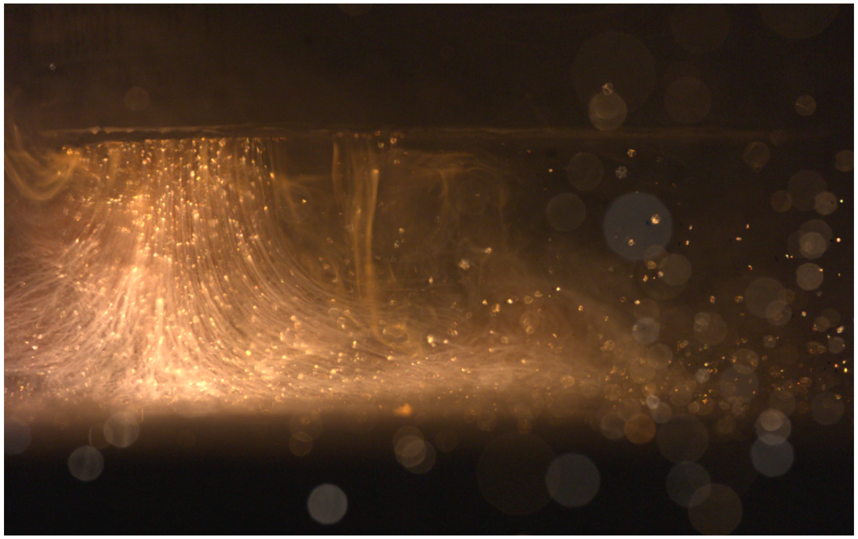
- Milestone Details:
  - Completion of an acoustically incompressible LES capability within Sierra/FD with adequate CPU performance
  - Performance and scalability enhancements to Sierra/FD will be achieved through maturation of the code on representative test problems. This will be done in such a way as to maintain Sierra/FD's readiness to address a broad set of applications
  - A single species LES simulation that demonstrates the required CPU performance and scalability.
- Four specific criteria for completion:
  - Scaling demonstrated on ***tens of thousands*** of cores
  - Performance study for a mixture fraction-based LES
  - Adequate description scaling and performance
  - Lessons learned and proposed path forward

Synergistic Projects

- Funding streams that have been leveraged
  - ASC Integrated Codes
    - Collective T/F team; General T/F consolidation support
  - ASC Algorithms
    - Trilinos solvers
    - Shadid/Cyr/Lin; Physics-based preconditioners
    - Dohrmann; general solver techniques
    - Domino; New coupling/discretization algs (PC, GMOS, etc.)
  - ASC CSAR
    - Wagner; Higher order CVFEM on Unstructured Meshes
  - SNL Early Start LDRD
    - Clausen; Developing Highly Scalable Fluid Solvers
  - ASCR Wind Energy UQ
    - UQ and high fidelity CFD for wind energy applications (SNL, Stanford, Purdue)

Abnormal/Thermal Environment

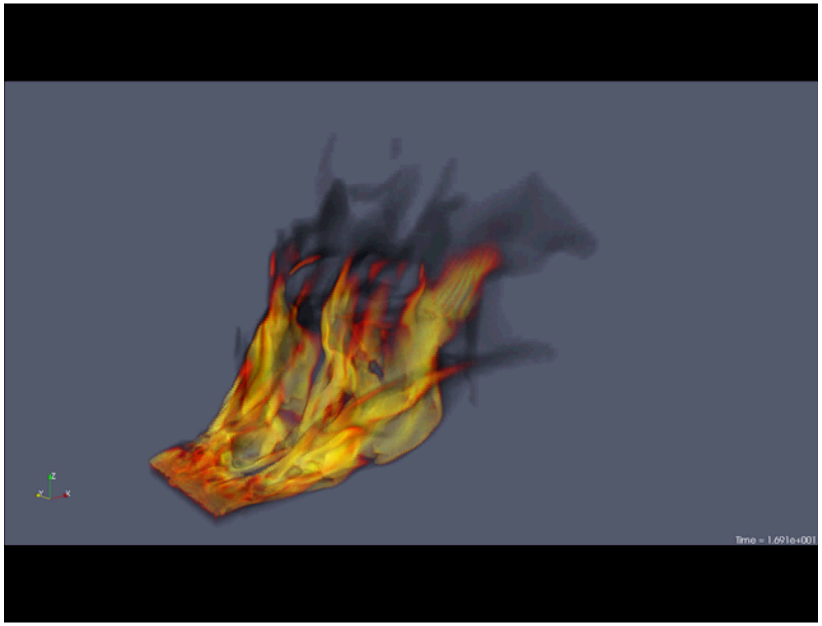
- Hydrocarbon JP-8 10 m fire
- Aluminum propellant fire



- ☐ Lead experimentalists: Jim Nakos
- ☐ Lead experimentalists: Walt Gill

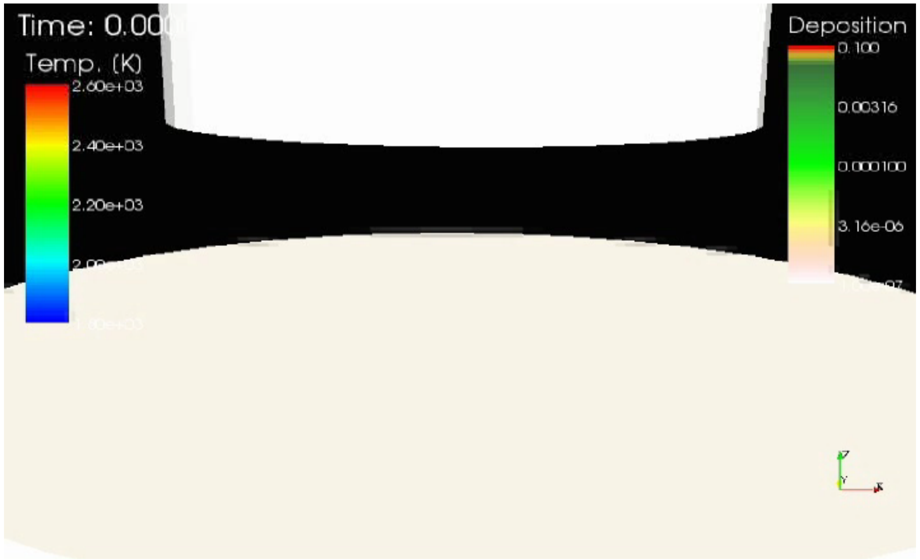
Simulation Capability

- Hydrocarbon JP-8 pool fire



- ☐ Multi-physics pool fire simulation that supported previous qualification effort

- Aluminum propellant fire



- ☐ Multi-physics propellant fire simulation that is supporting a variety of applications

Physics of Interest

- The variable density, low Mach set of equations are solved in which the acoustics have been filtered, thereby, allowing density to be a function of the spatially constant, possible variable in time thermodynamic pressure

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_j}{\partial x_j} = 0$$

$$DOFs = \tilde{u}_x, \tilde{u}_y, \tilde{u}_z, p, \tilde{z}$$

$$\frac{\partial \bar{\rho} \tilde{z}}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_j \tilde{z}}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\bar{\rho} D \frac{\partial \tilde{z}}{\partial x_j} - \tau_{zu_j}\right)$$

$$\frac{\partial \bar{\rho} \tilde{u}_i}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_j \tilde{u}_i}{\partial x_j} = -\frac{\partial}{\partial x_i} p + \frac{\partial}{\partial x_j}\left(\overline{\tau}_{ij} - \tau_{u_i u_j}\right) + (\bar{\rho} - \rho^r) g_i$$

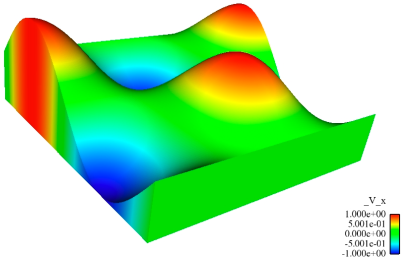
Turbulence closure models required for turbulent diffusive flux vector and subgrid stress tensor

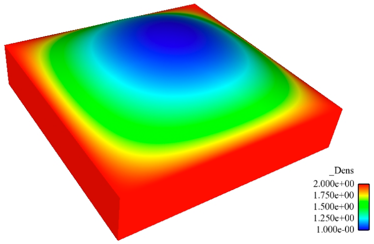
$$\bar{\rho} = \frac{1}{\dfrac{\tilde{z}}{\rho(\tilde{z}=0)} + \dfrac{(1-\tilde{z})}{\rho(\tilde{z}=1)}}$$

☐ Regardless of coupling techniques (monolithic or pressure-projection) an elliptic pressure system is created

Discretization and Coupling

- A variety of code discretizations have been implemented and verified using the Method of Manufactured Solutions
- Discretizations include:
  - Vertex centered Control Volume methods
  - Cell centered Control Volume methods
  - Finite Element
- Couplings range from
  - explicit pressure projection
  - operator split pressure projection
  - monolithic (fully coupled)
- More details to come….





☐ Variable density MMS; Ux (T) density (B)

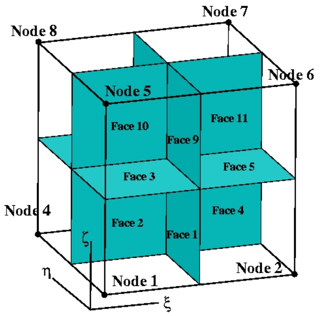
CVFEM Discretization

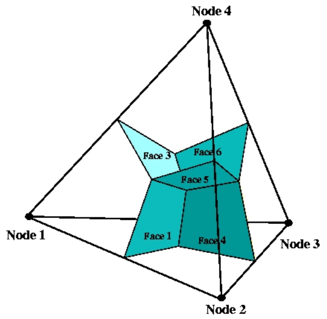
- The core discretization used in the low Mach code base has been the Control Volume Finite Element Method, CVFEM
- An elemental basis is defined from which interpolation and gradients within the element are determined
- The test function is defined to be piece-wise constant
- This method can best be described as a Petrov-Galerkin method
- The canonical 27-point stencil is recovered

$$\int w \frac{\partial \bar{\rho}\tilde{u}_j\tilde{\phi}}{\partial x_j} d\Omega = -\int \bar{\rho}\tilde{u}_j\tilde{\phi}\frac{\partial w}{\partial x_j} d\Omega + \int w\bar{\rho}\tilde{u}_j\phi n_j \, d\Gamma$$

$$w = w_I ; \frac{\partial w_I}{\partial x_j} = -\delta(x - x_{scs})$$

$$\int w \frac{\partial \bar{\rho}\tilde{u}_j\tilde{\phi}}{\partial x_j} d\Omega = \sum_{ip} \left(\bar{\rho}\tilde{u}_j\right)_{ip} \tilde{\phi}_{ip} n_j dS = \sum_{ip} \dot{m}_{ip}\tilde{\phi}_{ip}$$

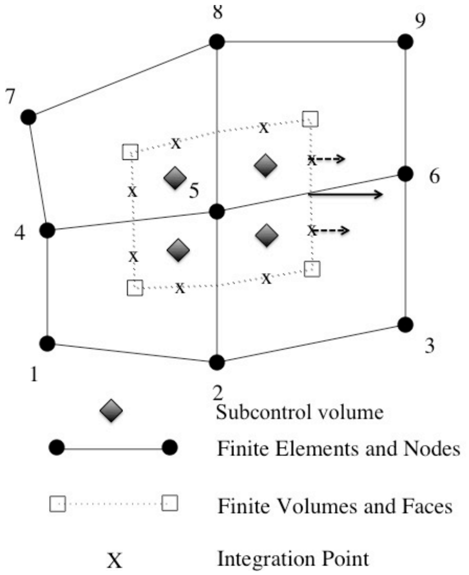




$$\phi(\vec{x}) = \sum_J N(\vec{x})\phi_J$$

Edge-Based Discretization

- In this method, the dual mesh is defined to establish geometric values at the edge midpoint (area vector) and node (volume)



❑ Quadrature points for edge-based scheme

- Ramifications for the edge-based finite volume (EBFV)structure are as follows:
  - Reduced stencil (27-point to 7-point for structured hex)
  - Simple L/R data structure allows for simple interpolation and orthogonal gradient contributions
  - Lack of elemental basis requires a diffusion operator in terms of orthogonal to the edge and non-orthogonal correction that requires projected nodal gradients

Finite Element Discretization

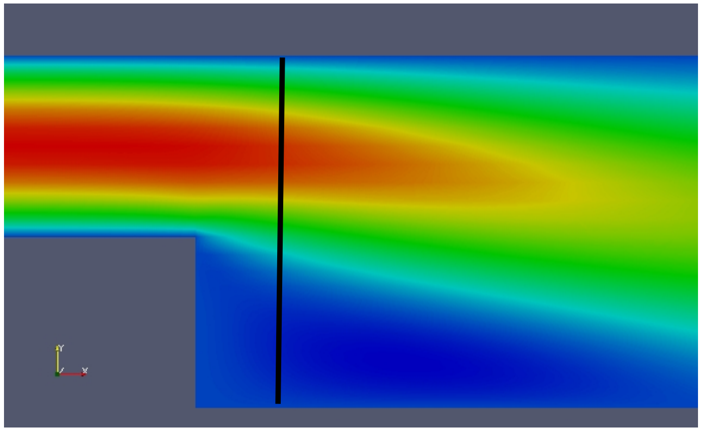
- Classic Equal Order Interpolation with explicit pressure stabilization
- Monolithic or approximate pressure projection couplings exist
- Pressure stabilization can be similar to segregated approach (2nd or 4th order) or PSPG
- Advection stabilization obtained via SUPG

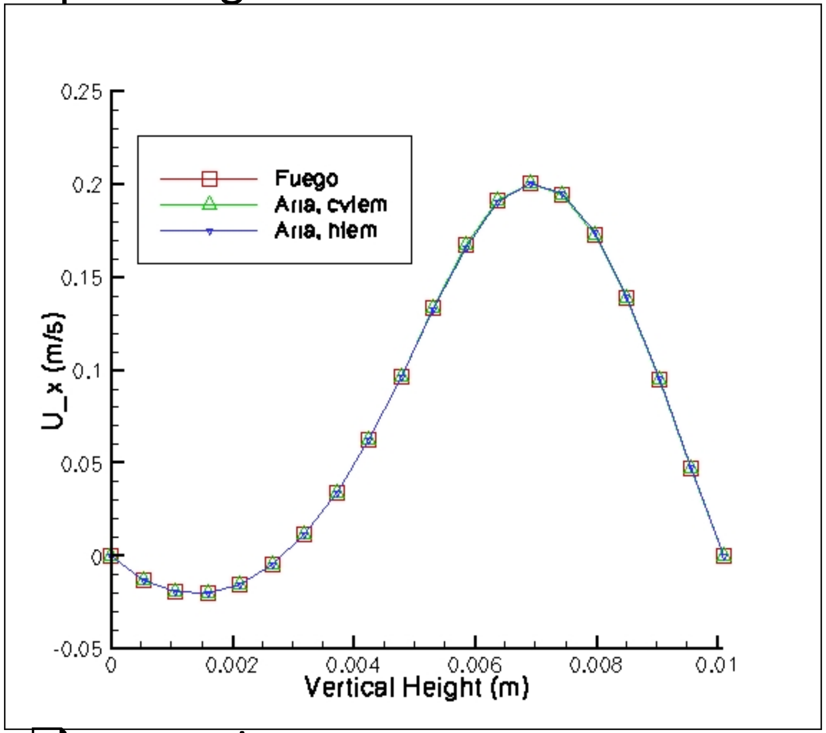
$$\tilde{w} = w + \tau u_j \frac{\partial}{\partial x_j} w$$

- Ramifications for the FEM method:
  - Canonical 27-point stencil for structured hex
  - Full elemental diffusion operator (issues with diffusion operator monotonicity exists for aspect ratios greater than sqrt(2))
  - Classic Galerkin Method not regularly used due to the need for residual-based stabilization thus making most implementations a Petrov-Galerkin method

CVFEM/FEM Comparison

- Laminar back step (Re = 389 based on step height) comparing two implementations of CVFEM (pressure projection and monolithic) and monolithic FEM
  - Fourth order stabilization with time step scaling



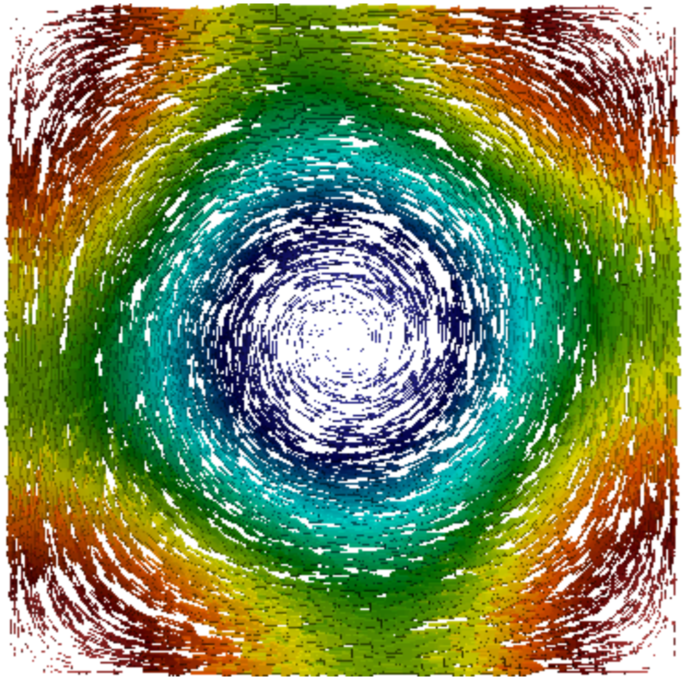


☐ Ux; vertical line is the 1-d comparison

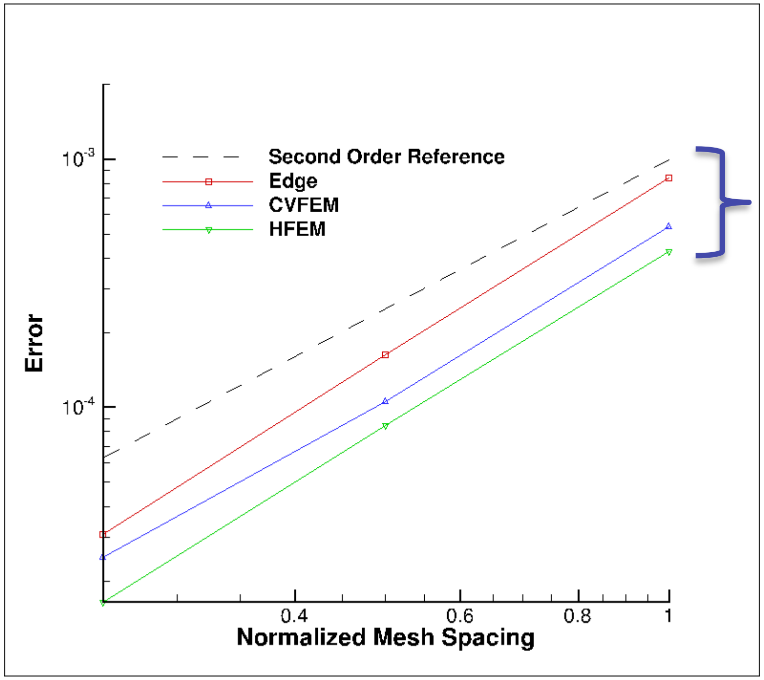
☐ comparison

Error Tradeoff

- Error disparity on “nice” mesh for a Steady Taylor Vortex MMS for each schemes are comparable
- Other attributes of the scheme, i.e., speed, robustness, time to solution, etc. are far more significant



☐ Steady TV; u_vel colored by pressure

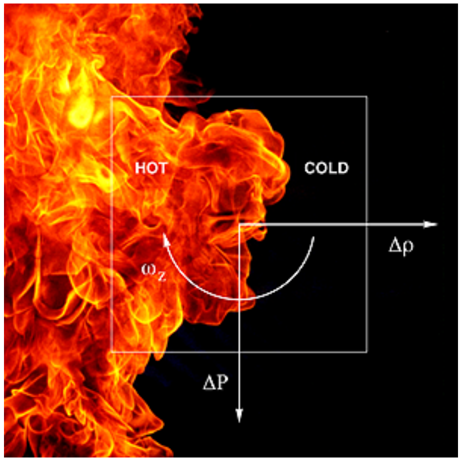


> 1 Order of magnitude speed disparity

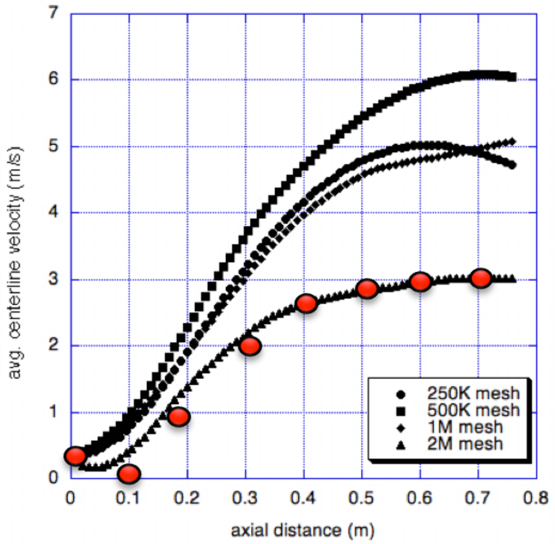
☐ Loo norms for three discretizations

Discretization Error vs Resolution

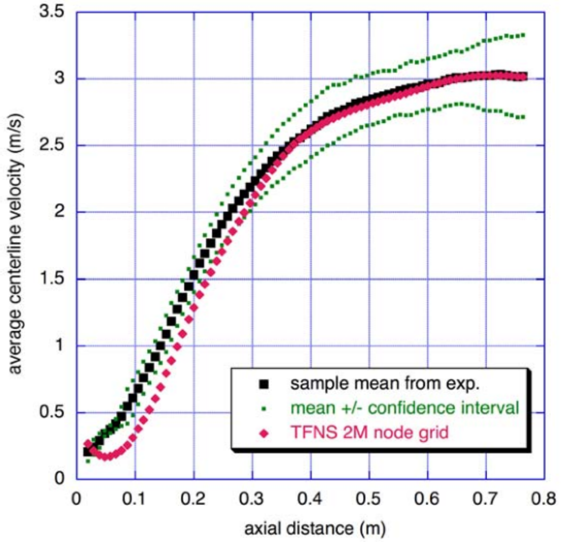
- Common Value System: The best numerical scheme is the one in which errors for a canonical code verification suite are smallest
- However, oftentimes the ability to resolve a physics scale is of prime importance



☐ Fire instability 101



☐ Core collapse as a Function of mesh resolution



☐ Data comparison

Coupling

- The traditional low Mach algorithm is an approximate projection algorithm in which splitting and pressure stabilization terms exist

$$\begin{bmatrix} A & G \\ D & 0 \end{bmatrix}\begin{bmatrix} u^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} f \\ b \end{bmatrix} + \begin{bmatrix} (I - A\tau)G\left(p^{n+1/2} - \alpha p^{n+1/2}\right) \\ \tau\left(L - \beta DG\right)p^{n-1/2} \end{bmatrix}$$

- α and β define incremental pressure/pressure-free and 2nd and 4th pressure stab

- Other approaches are possible including monolithic and flavors of operator split
- In general, there exists a trade space between time scale of interest and coupling approach

Algorithm	Speed factor
uvwp; Imp/Imp	3.4x
uvw_p; Imp;/Imp	1.2x
uvw_p; Imp/Exp	0.6x
u_v_w_p; Imp/Imp	1.0x
uvw_p; Exp/Exp	0.7x

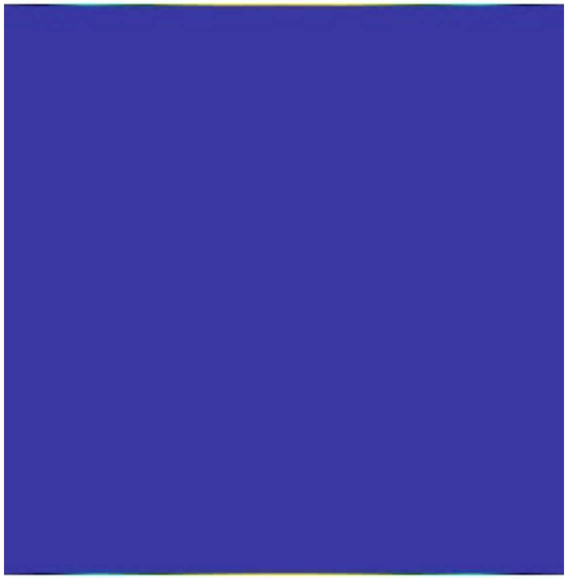
Coupling

- α and β define incremental pressure/pressure-free and 2nd and 4th

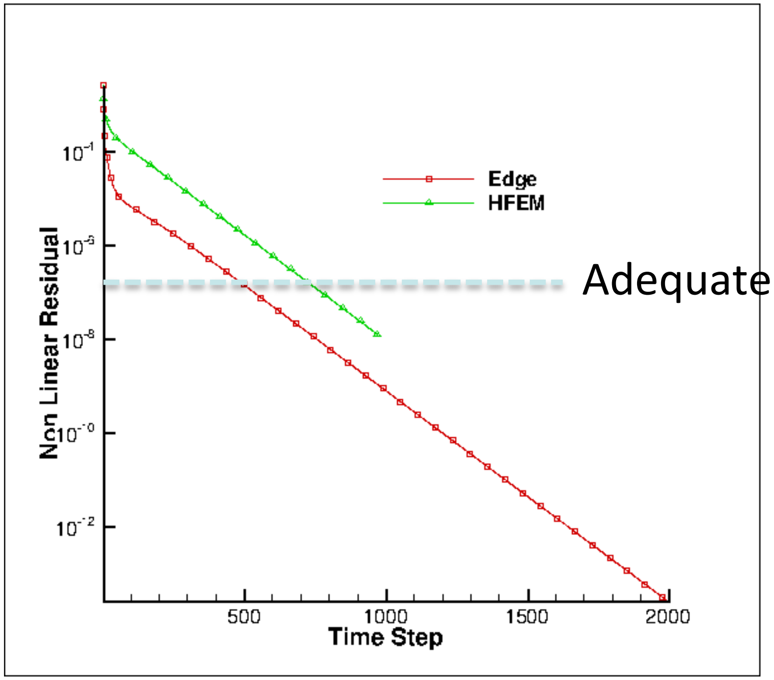
u_v_w_p; Imp/Imp	1.0x
uvw_p; Exp/Exp	0.7x

Algorithmic Convergence (1/2)

- Comparison between monolithic FEM and Edge-based FV
  - Edge-based scheme is operator split, approximate pressure projection
  - Monolithic FEM scheme is full analytical sensitivities with exception of nodal L2 grad(P) equation



❑ Steady TV (pressure)

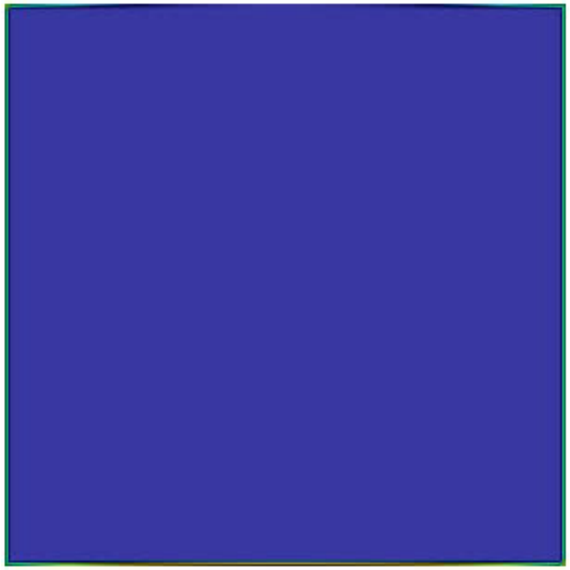


Adequate tolerar

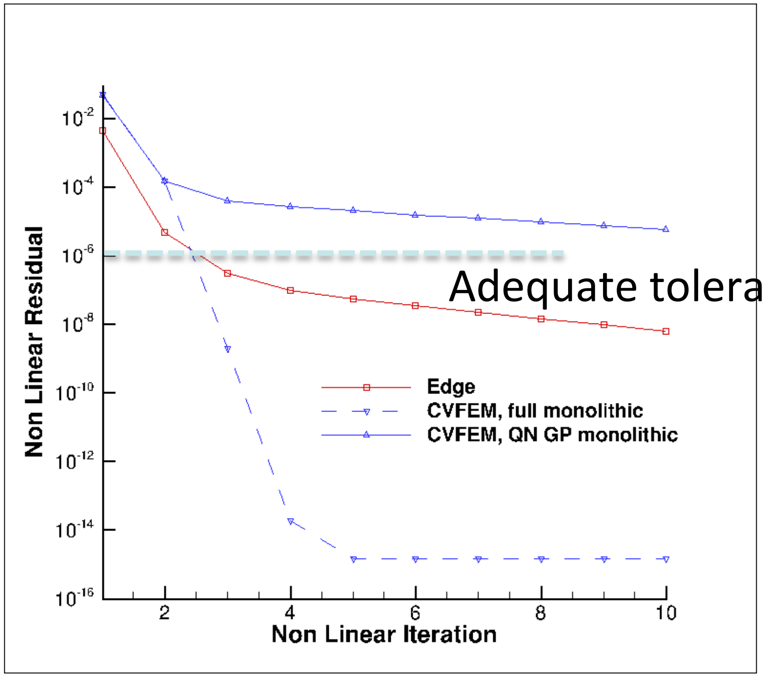
❑ Steady TV NLR

Algorithmic Convergence (2/2)

- Comparison between monolithic CVFEM and Edge-based FV
  - Edge-based scheme is operator split, approximate pressure projection
  - Monolithic scheme is full analytical sensitivities with and without nodal L2 grad(P) equation



☐ Convective TV (velocity mag)



Adequate tolerance

☐ Convecting TV NLR

Open Literature Works

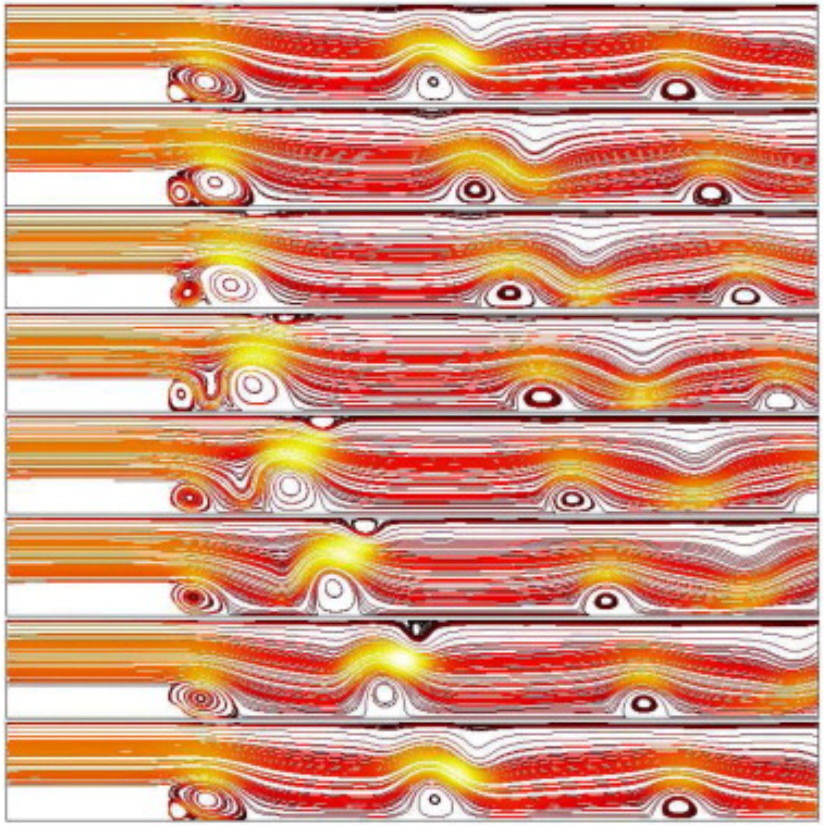
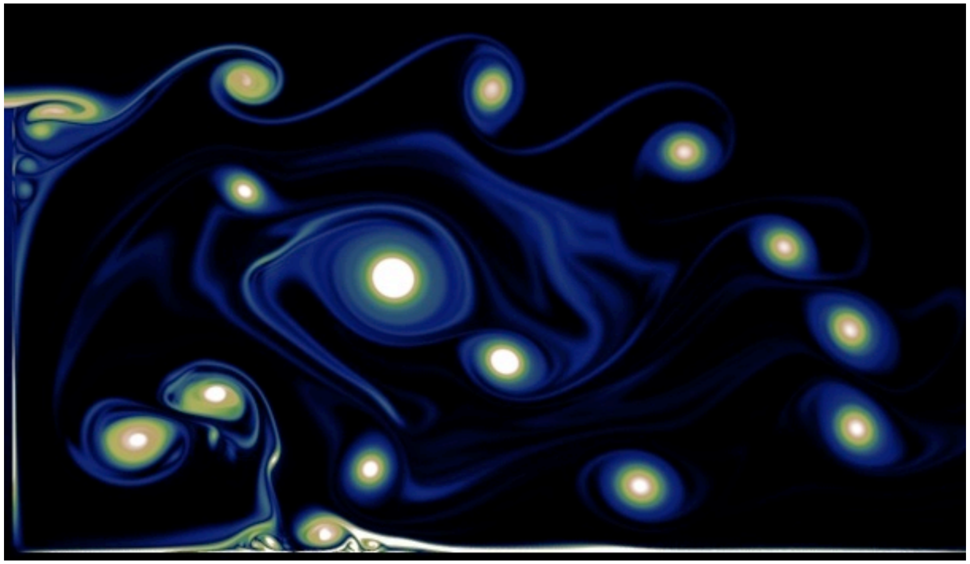


Fig. 18. Periodic evolution of streamlines from $t = 3.04$ s to $t = 3.11$ s.

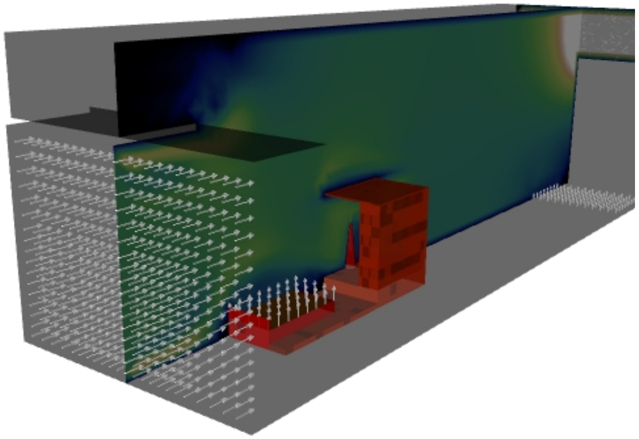
- ❑ Hachem et al. JCP 229:23, 2010; monolithic stabilized FEM (40k tri)

- ❑ Re 45k turbulent back step

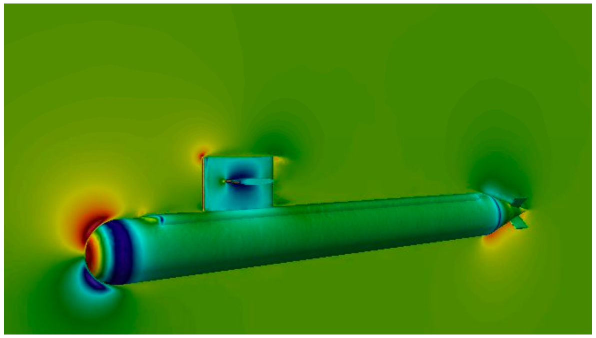


- ❑ Domino; approximate pressure projection with KE preserving operators (8000k tri elements)

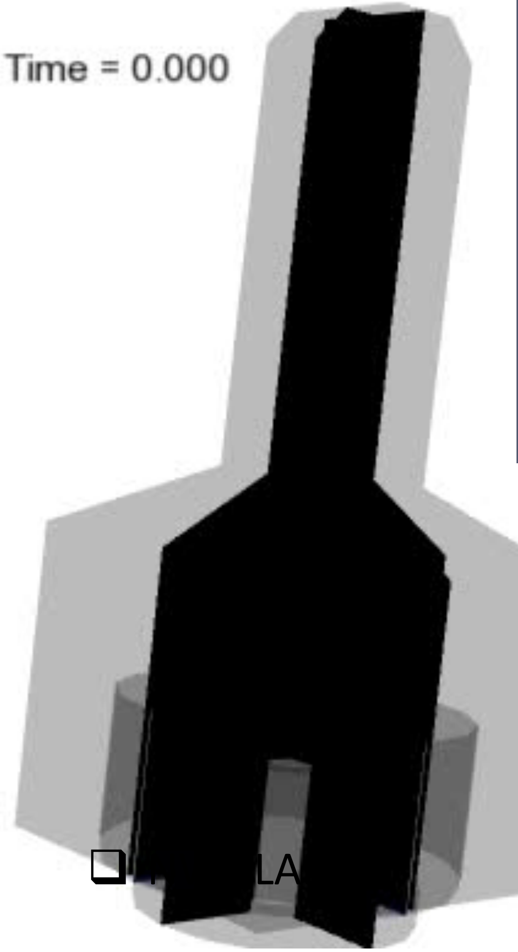
Edge-Based Scheme Deployed



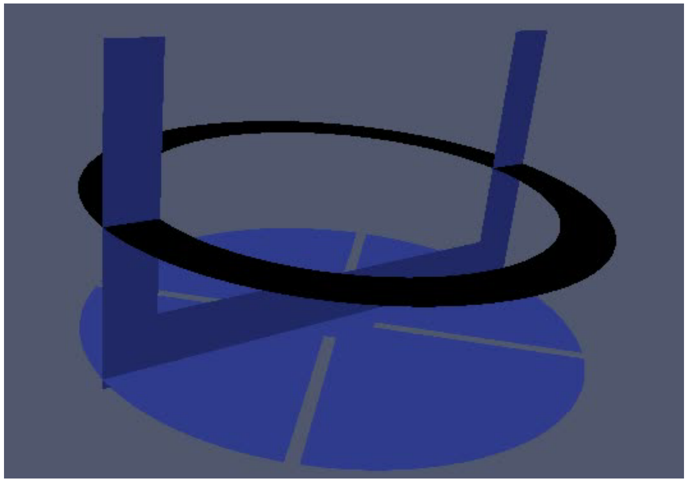
- ❑ NW XTF



- ❑ WFO Tet-based sub



- ❑ NW FLAME



- ❑ LDRD Waste reprocessing

- ❑ WFO Wind energy apps

Code Consolidation

The first quarter of the milestone effort centered on code consolidation to use the expression-based system provided within the SIERRA/FD *Aria* module

- Partial differential equations are a collection of terms that are composed of expressions with a list of ***pre-requisites***
- Expressions are a C++ abstraction with pure virtual compute_values() and compute_sensitivities()
- Code base allows for easy implementation of analytical Jacobians since the developer only needs to code derivatives of the expression wrt its set of prerequisites
- Analytical sensitivities are chain-ruled
- An implementation of expressions that wrap Automatic Differentiation (AD) via Trilinos Sacado exists

Aria::Expression 101

- Consider the power of the Expression system by outlining the CVFEM advection operator

Momentum: $\int \frac{\partial \overline{\rho}\tilde{u}_i}{\partial t} \mathrm{d}V + \int \overline{\rho}\,\tilde{u}_i \tilde{u}_j n_j\, dS = \int \left(\overline{\sigma}_{ij} - \sigma_{u_i u_j} \right) n_j dS + \int \left(\overline{\rho} - \rho_o \right) g_i \mathrm{d}V$

Walking down the “expression dependence tree”

$$\int \overline{\rho}\,\tilde{u}_i \tilde{u}_j n_j\, dS \approx \sum_{ip} \dot{m}_{ip} \tilde{u}_{j,ip} = \sum_{ip} \dot{m}_{ip} \tilde{u}_{j,ip}$$

$$\dot{m}_{ip} = \left[\left(\rho \tilde{u}_j \right)_{ip} + G_j \tau p - \tau \frac{\partial p}{\partial x_j} \right] n_{j,ip} dS$$

$$\rho = \frac{pM}{RT} \qquad \tilde{u}_{j,ip} = N_{ip,k} \tilde{u}_{j,k}$$

***Mass flow rate prerequisites*:**

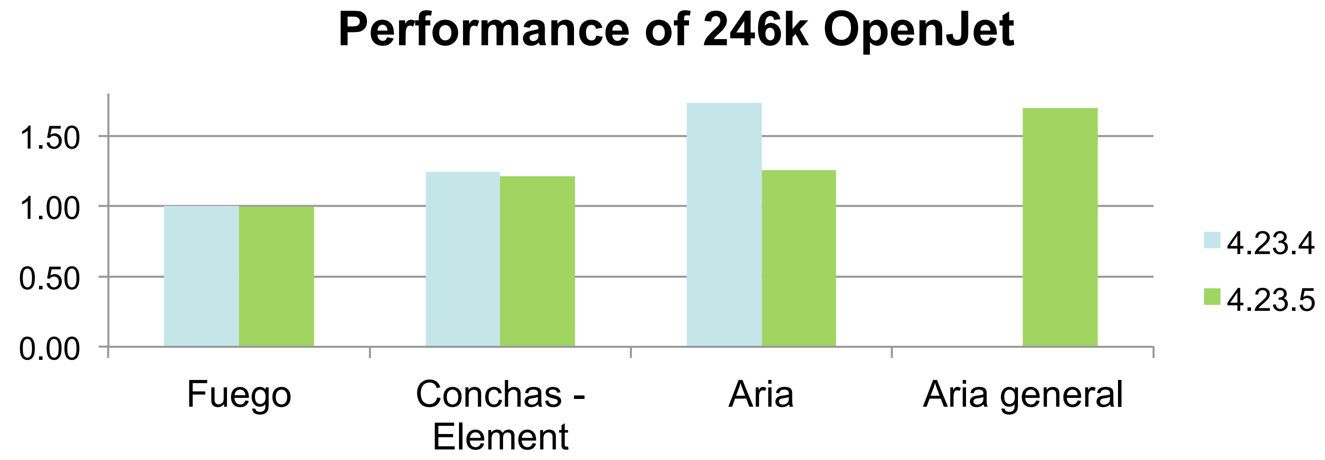
$$\dot{m}_{ip} = f\left(\rho, \tilde{u}_j, \tau, p, \frac{\partial}{\partial x_j}, N\right)$$

… and so on

- Note that adding a new, e.g., density model does not change the mass flow rate expression as sensitivities are automatically propagated

Expression Price Tag

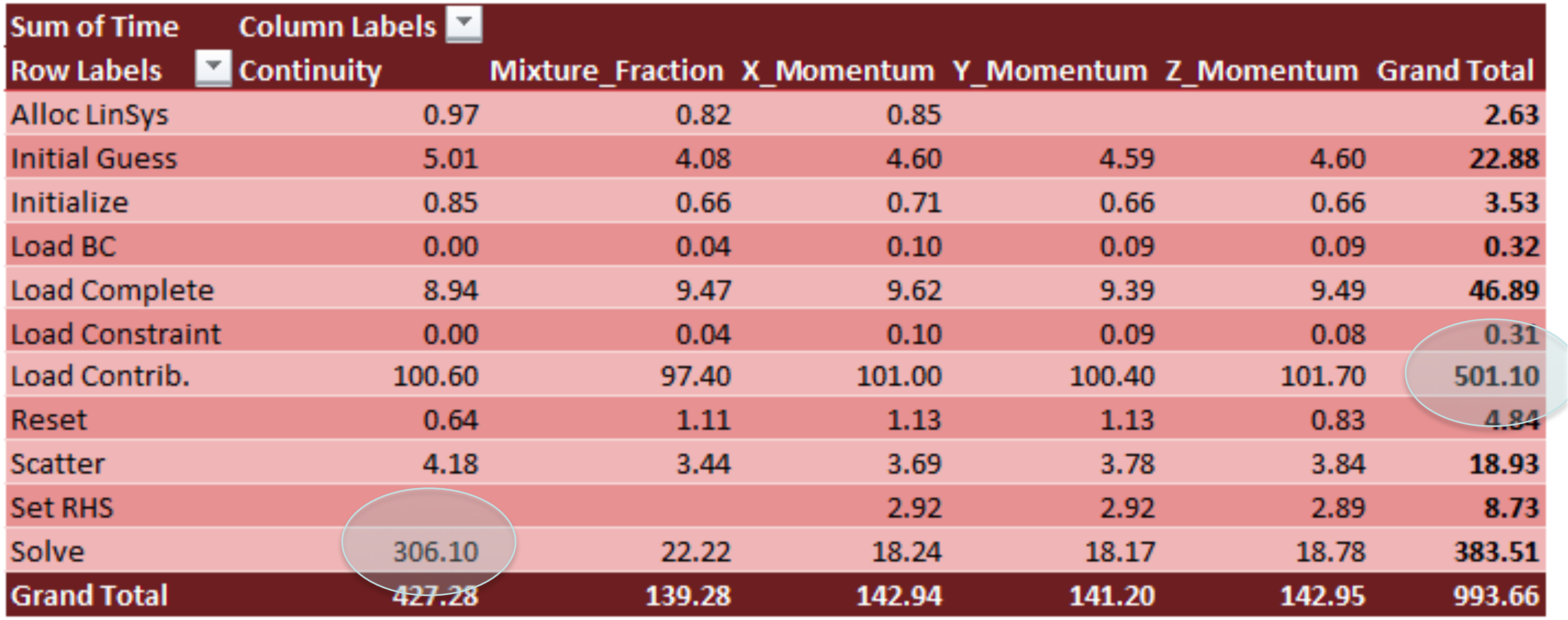
- Can one afford true generality? Consider the evolution of code performance using the Aria::Expression infrastructure



- In the context of a uvw_p pressure projection scheme, the Expression infrastructure was
  - Was NOT competitive when using the most general composed set of expressions
  - Was competitive when these expressions were hand written to exploit known sensitivities

Evaluation of Current Code Timings

- Consider a typical mixture fraction-based LES for a transient buoyant plume simulation



Sum of Time	Column Labels ▾					
Row Labels ▾	**Continuity**	**Mixture_Fraction**	**X_Momentum**	**Y_Momentum**	**Z_Momentum**	**Grand Total**
Alloc LinSys	0.97	0.82	0.85			**2.63**
Initial Guess	5.01	4.08	4.60	4.59	4.60	**22.88**
Initialize	0.85	0.66	0.71	0.66	0.66	**3.53**
Load BC	0.00	0.04	0.10	0.09	0.09	**0.32**
Load Complete	8.94	9.47	9.62	9.39	9.49	**46.89**
Load Constraint	0.00	0.04	0.10	0.09	0.08	**0.31**
Load Contrib.	100.60	97.40	101.00	100.40	101.70	**501.10**
Reset	0.64	1.11	1.13	1.13	0.83	**4.84**
Scatter	4.18	3.44	3.69	3.78	3.84	**18.93**
Set RHS			2.92	2.92	2.89	**8.73**
Solve	306.10	22.22	18.24	18.17	18.78	**383.51**
Grand Total	**427.28**	**139.28**	**142.94**	**141.20**	**142.95**	**993.66**

- ❑ Solve and assembly time dominates

Decision on Code Consolidation

- Over the past two years, Expression-based code simulations have reduced from >> 10x too slow to well under 2x
- However, the L2 Milestone team reached the consensus that the Expression-based infrastructure would not meet the immediate Low Mach project performance needs
- In a given approximate projection simulation, roughly 20% of the simulation time is spent within application code
- The majority of simulation time is spent in matrix scatter and solves
  - It is this overhead, which is shared across most all implicit codes, that was addressed in this milestone
- However, with this knowledge, we can transition the lessons learned to the production Expression-based code

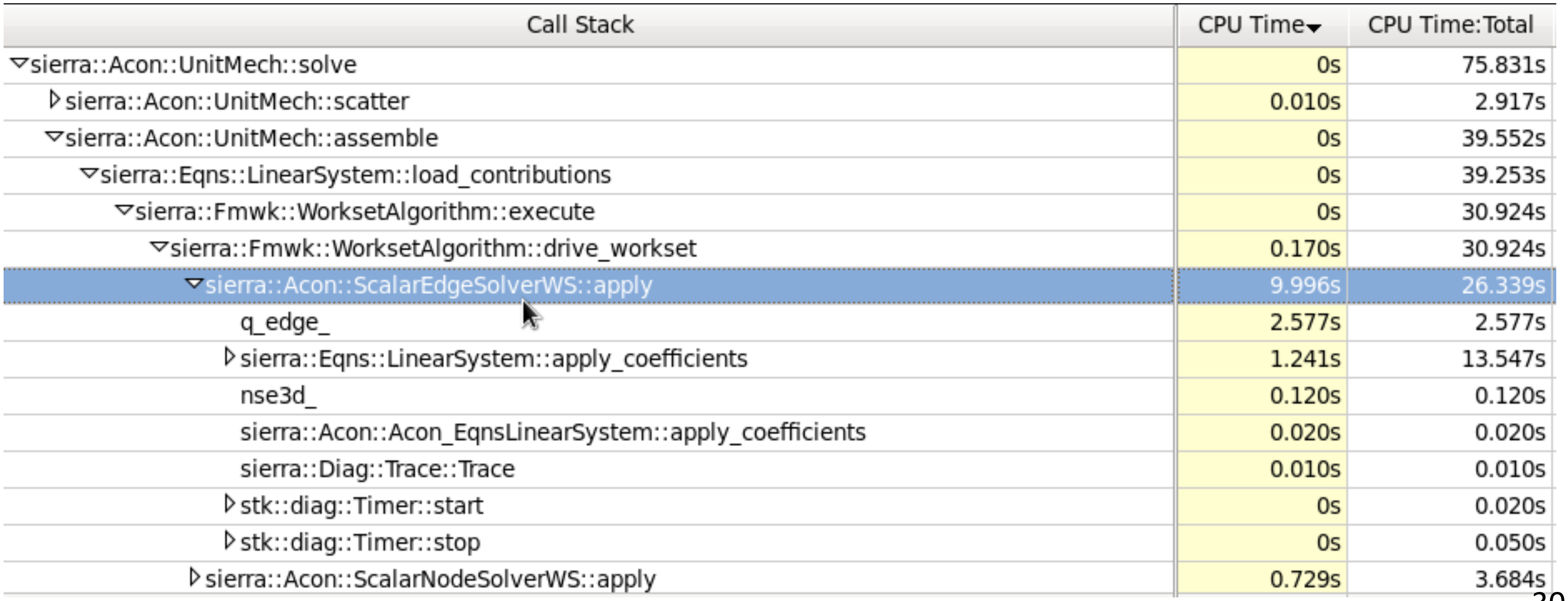
Possible Bottlenecks to Evaluate

Evaluation of the following common interfaces

- The Finite Element Interface (FEI)
  - Path: Algorithms can code directly to ePetra or FETI
- The role of the Workset Algorithm
  - Path: Evaluate different approaches for gathers; drop notion of supplemental scatter algorithm
- Trilinos Solver performance
  - Path: Code our own Sierra/FD written solvers
- ePetra Assembly
  - Path a: Compare ePetra to FETI
  - Path b: Wrap ePetra by FETI to gain access to Trilinos/ML

Performance Tools

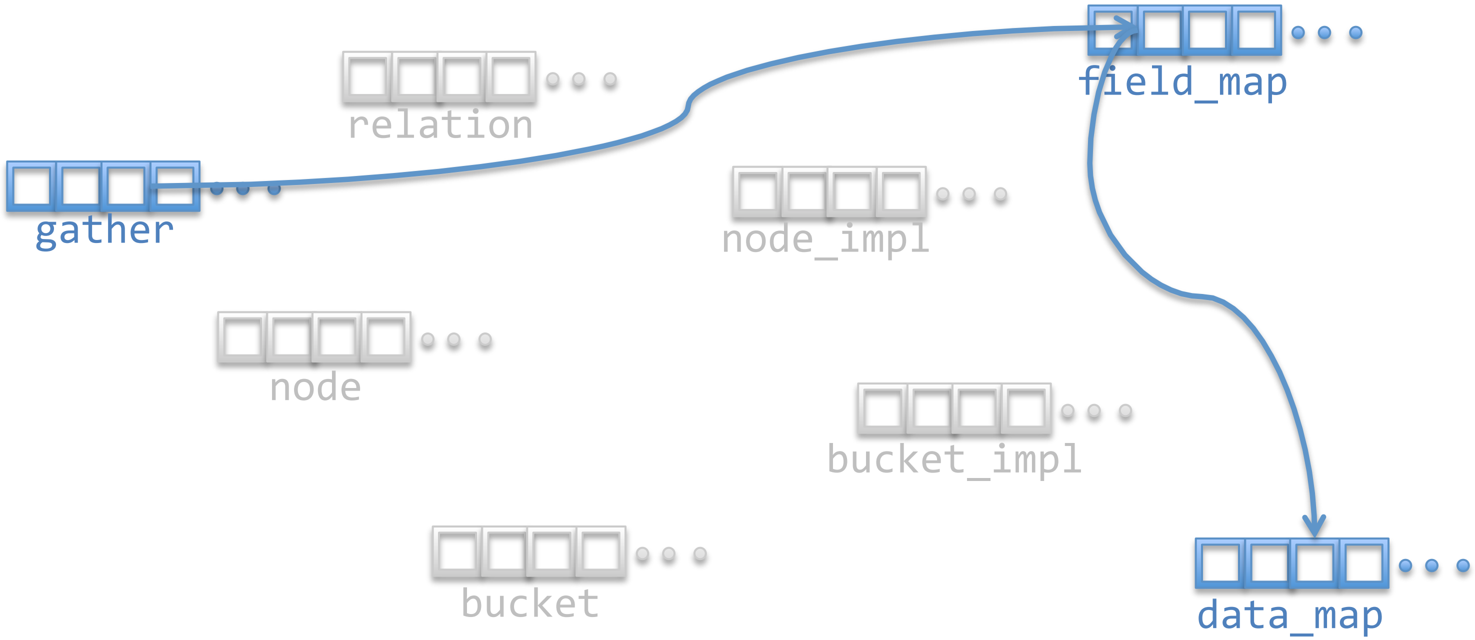
- VTune represents a tool to evaluate timings for call stack as well as memory usage effectiveness
- For example, it provided an ability to see inefficient execution and lots of cache misses in ScalarEdgeSolverWS



Call Stack	CPU Time▾	CPU Time:Total
▽sierra::Acon::UnitMech::solve	0s	75.831s
▷sierra::Acon::UnitMech::scatter	0.010s	2.917s
▽sierra::Acon::UnitMech::assemble	0s	39.552s
▽sierra::Eqns::LinearSystem::load_contributions	0s	39.253s
▽sierra::Fmwk::WorksetAlgorithm::execute	0s	30.924s
▽sierra::Fmwk::WorksetAlgorithm::drive_workset	0.170s	30.924s
▽sierra::Acon::ScalarEdgeSolverWS::apply	9.996s	26.339s
q_edge_	2.577s	2.577s
▷sierra::Eqns::LinearSystem::apply_coefficients	1.241s	13.547s
nse3d_	0.120s	0.120s
sierra::Acon::Acon_EqnsLinearSystem::apply_coefficients	0.020s	0.020s
sierra::Diag::Trace::Trace	0.010s	0.010s
▷stk::diag::Timer::start	0s	0.020s
▷stk::diag::Timer::stop	0s	0.050s
▷sierra::Acon::ScalarNodeSolverWS::apply	0.729s	3.684s

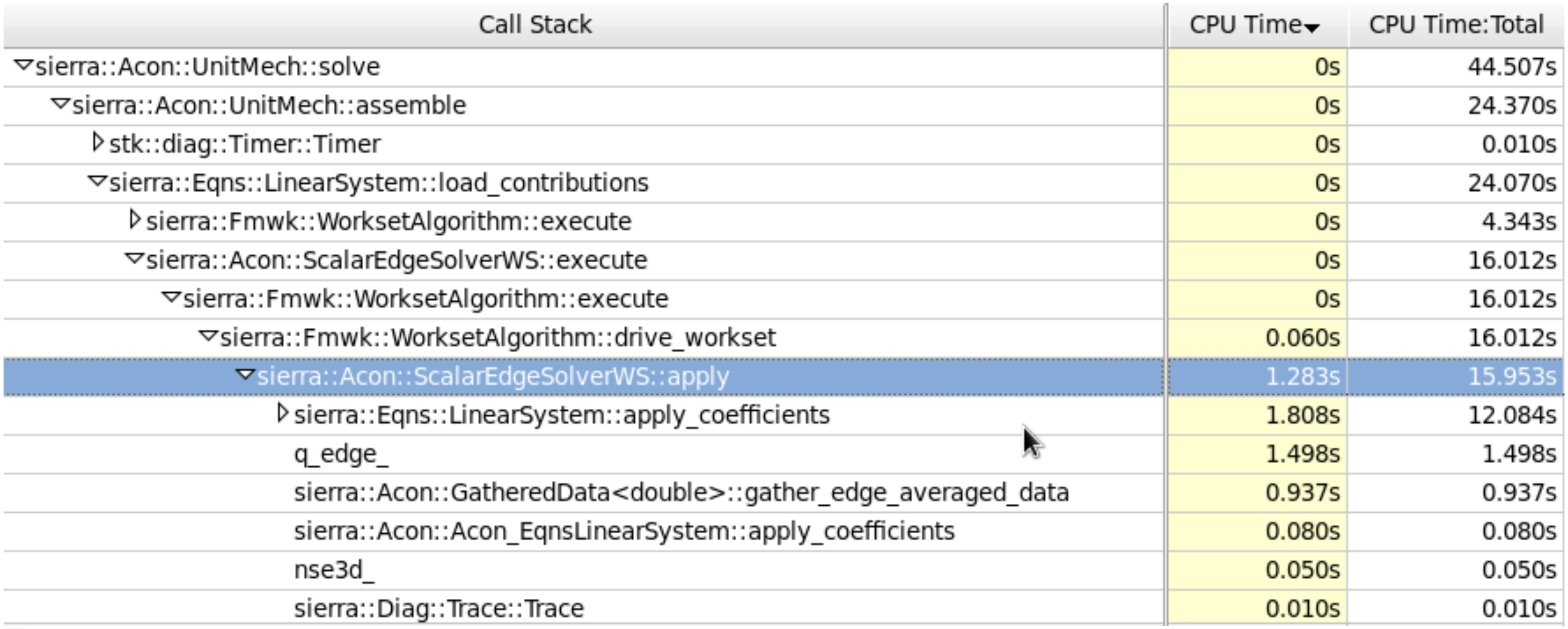
Fast Edge Gathers

- Fast Gathers use caching to eliminate hopping through intermediates:



New Fast Gathers

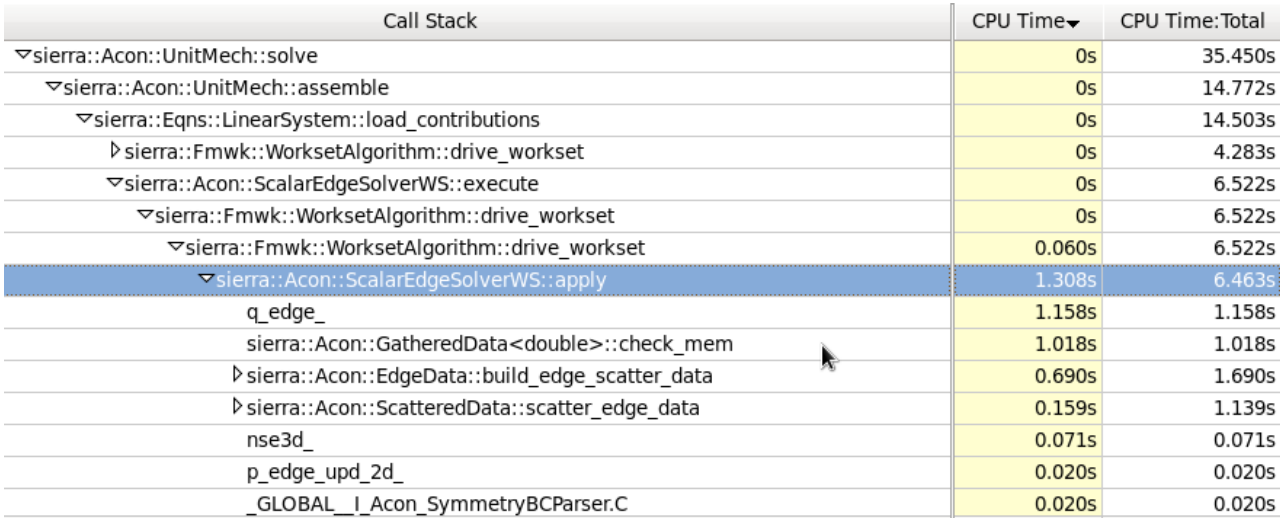
- New gathers have fewer instructions, fewer memory hops and fewer cache misses. Gathers dropped from 10s to <1 s
- Total “solve” time down from 75s to 44s



Call Stack	CPU Time▾	CPU Time:Total
▽sierra::Acon::UnitMech::solve	0s	44.507s
▽sierra::Acon::UnitMech::assemble	0s	24.370s
▷stk::diag::Timer::Timer	0s	0.010s
▽sierra::Eqns::LinearSystem::load_contributions	0s	24.070s
▷sierra::Fmwk::WorksetAlgorithm::execute	0s	4.343s
▽sierra::Acon::ScalarEdgeSolverWS::execute	0s	16.012s
▽sierra::Fmwk::WorksetAlgorithm::execute	0s	16.012s
▽sierra::Fmwk::WorksetAlgorithm::drive_workset	0.060s	16.012s
▽sierra::Acon::ScalarEdgeSolverWS::apply	1.283s	15.953s
▷sierra::Eqns::LinearSystem::apply_coefficients	1.808s	12.084s
q_edge_	1.498s	1.498s
sierra::Acon::GatheredData<double>::gather_edge_averaged_data	0.937s	0.937s
sierra::Acon::Acon_EqnsLinearSystem::apply_coefficients	0.080s	0.080s
nse3d_	0.050s	0.050s
sierra::Diag::Trace::Trace	0.010s	0.010s

New Fast Scatters

- Similar approach used for scatters to the linear system
- Scatters dropped from ~12s to 1.1s. Total solve time down from 75s to 35s
- New scatters only help with locally owned rows. At high core counts (>1024) total scatter time is still expensive. We're working to resolve this



Call Stack	CPU Time▾	CPU Time:Total
▽sierra::Acon::UnitMech::solve	0s	35.450s
▽sierra::Acon::UnitMech::assemble	0s	14.772s
▽sierra::Eqns::LinearSystem::load_contributions	0s	14.503s
▷sierra::Fmwk::WorksetAlgorithm::drive_workset	0s	4.283s
▽sierra::Acon::ScalarEdgeSolverWS::execute	0s	6.522s
▽sierra::Fmwk::WorksetAlgorithm::drive_workset	0s	6.522s
▽sierra::Fmwk::WorksetAlgorithm::drive_workset	0.060s	6.522s
▽sierra::Acon::ScalarEdgeSolverWS::apply	1.308s	6.463s
q_edge_	1.158s	1.158s
sierra::Acon::GatheredData<double>::check_mem	1.018s	1.018s
▷sierra::Acon::EdgeData::build_edge_scatter_data	0.690s	1.690s
▷sierra::Acon::ScatteredData::scatter_edge_data	0.159s	1.139s
nse3d_	0.071s	0.071s
p_edge_upd_2d_	0.020s	0.020s
_GLOBAL__I_Acon_SymmetryBCParser.C	0.020s	0.020s

Profiling Communication at Scale

- mpiP to profile MPI communication
  - Bug in Intel 11.x compilers was a set back
  - Manually calling the linker works
  - On-going work
- Also trying to use CrayPat
  - bugs in CrayPat too
  - At this point CrayPat is not usable
- The bulk of information is still based on detailed code timers in the context of strong and weak scaling

Common Interface Results

Findings for interface evaluation are as follows

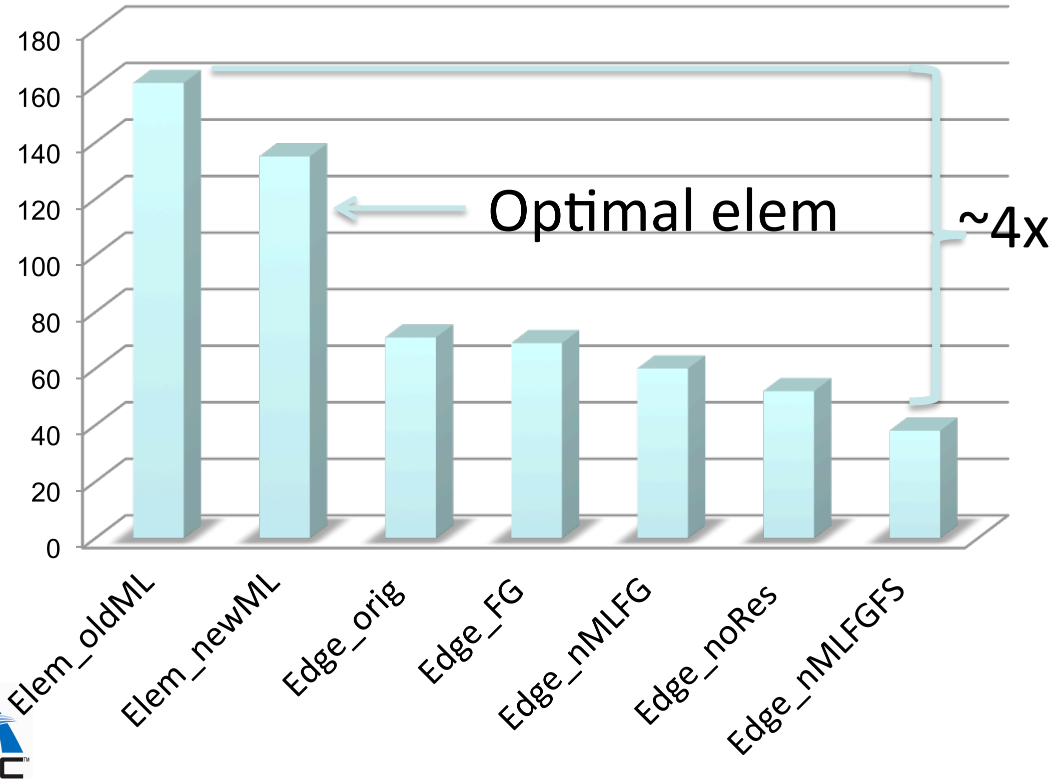
- The Finite Element Interface is not generally the bottleneck
- Trilinos solvers might benefit from performance work
- ePetra is more expensive that FETI

	Control	Implementation 1	Implementation 2
FEI (apply_coeff(), etc.)	+	−	−
ePetra	+	+	−
Trilinos	+	+	−
Fmwk::WsAlg	+	−	+

3D Open jet	total	18:59.434	Almost exactly the same	14:41.768 (1.3x)
	Matrix_Solve	6:05.089		6:14.132 (0.98x)
	Matrix_Assemble	10:05.652		6:01.331 (1.7x)
	Load_Complete	1:48.018		40.388 (1.5x)
3D DC	apply_coeff()	6.79	5.75 (1.17x)	

Edge-based Timing History

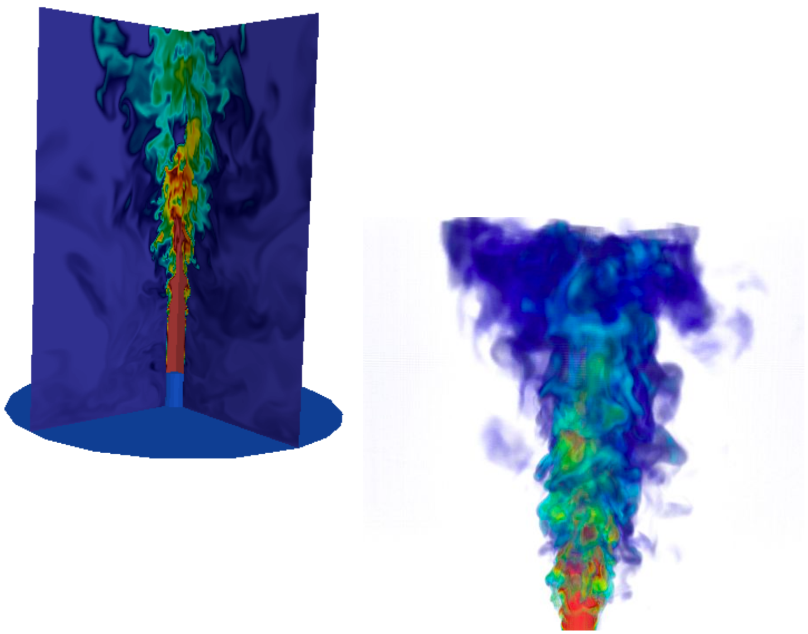
- History of Edge-based timing compared to Element-based scheme for the mixture fraction-based open jet simulation (17 million element; 128 core)



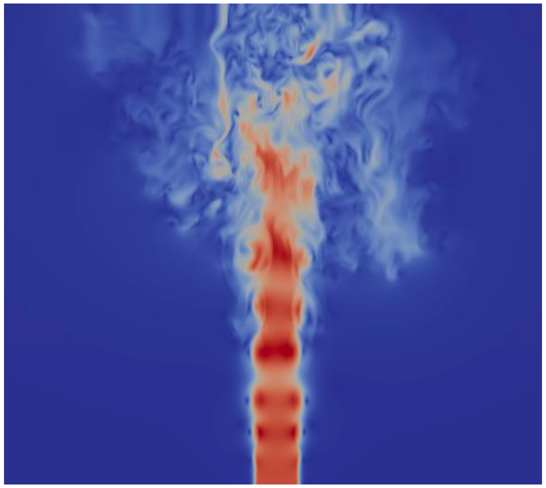
- ❑ Elem
  - CVFEM code base
- ❑ Edge
  - EBVC code base
- ❑ oldML
  - Default ML
- ❑ nML
  - ML settings from 2008
- ❑ FG
  - Fast gather
- ❑ noRes
  - No reset of cont equation
- ❑ FS
  - Fast scatter

Test Problem of Interest

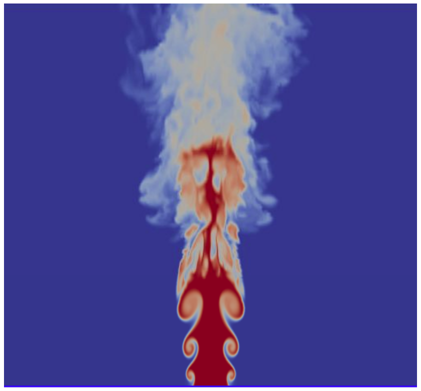
- The three dimensional test problem of interest that has been used for this milestone effort is a turbulent open jet (Re = 6,600) of Abdel et al. (1997)



- Re = 6,600 3D mesh (VR mixture fraction)



- 140 million element Re = 6,600 turbulent jet (velocity)



- Mixture fraction

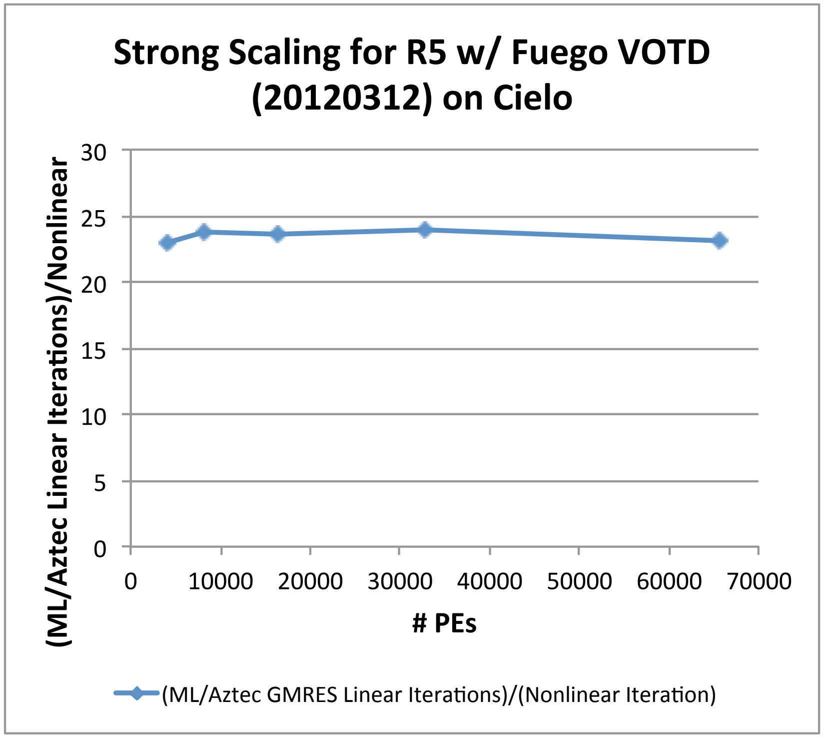
Scaling Studies

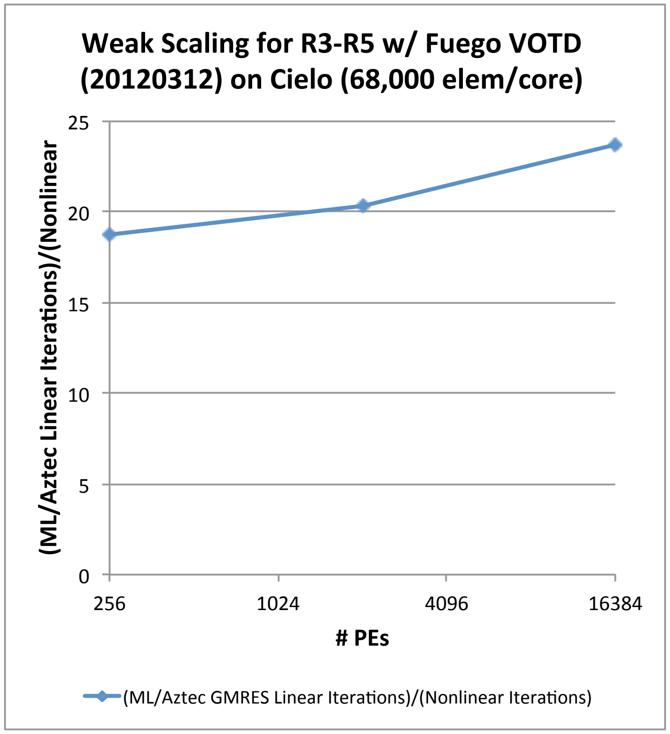
- Cielo scaling studies for mixture fraction-based turbulent open jet problem (Re=6,600)
- Sequence of meshes:
  - R3 (17.5 million elements; 64 – 4096 cores)
  - R4 (140 million elements; 512 – 16384 cores)
  - R5 (1.12 billion elements; 4096 – 65536 cores)
- Linear Solve options
  - Continuity: GMRES/ML
  - Scalars: GMRES/SGS
- Element-based algorithmic studies: R3 – R5
  - Internal code name "Fuego"
- Edge-based algorithmic studies: R4
  - Global ID size impediment
  - Internal code name "Conchas"

Scaling Study Summary

- 1.12 billion elements on 65,536 Cielo cores
    - Largest and most cores SNL low Mach run to date
    - Likely, largest in the Low Mach community based on personal poll including Utah, Stanford, Minnesota
- GMRES/ML multigrid preconditioner performed very well and effectively controlled growth in linear solve iterations for pressure Poison solve
    - Continuity linear solve scaled ideally to 16,384 cores
        - 65,536 cores still faster than 32,000 cores
        - Cielo scaled better than Red Sky; maintained strong scaling to ~68,000 element/core
- Element workset setup + LHS/R scale ideally
- Scaling of overall matrix assembly needs improvement as it is currently suboptimal

ML Algorithmic Scaling Performance



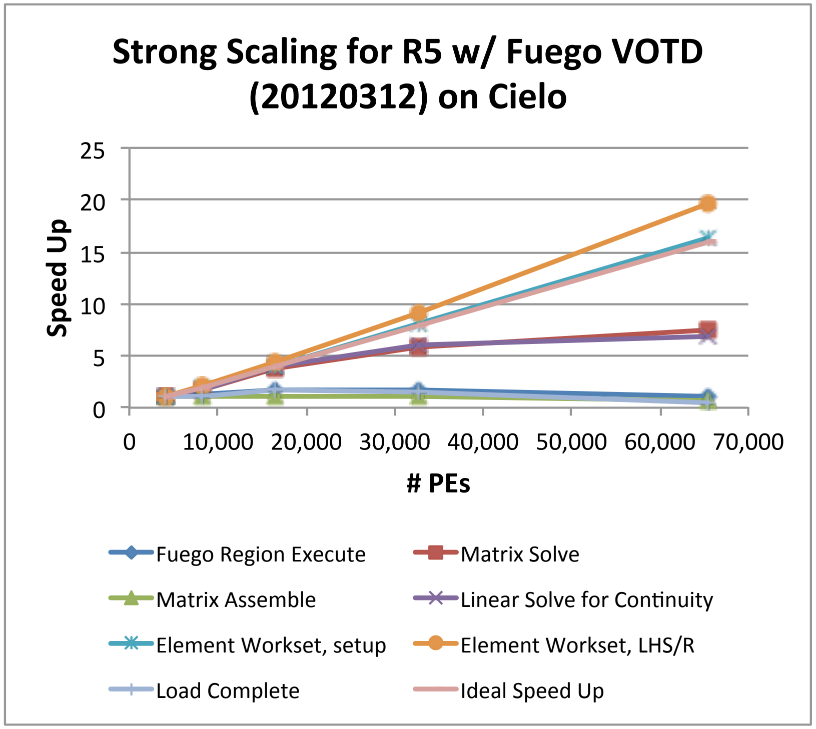


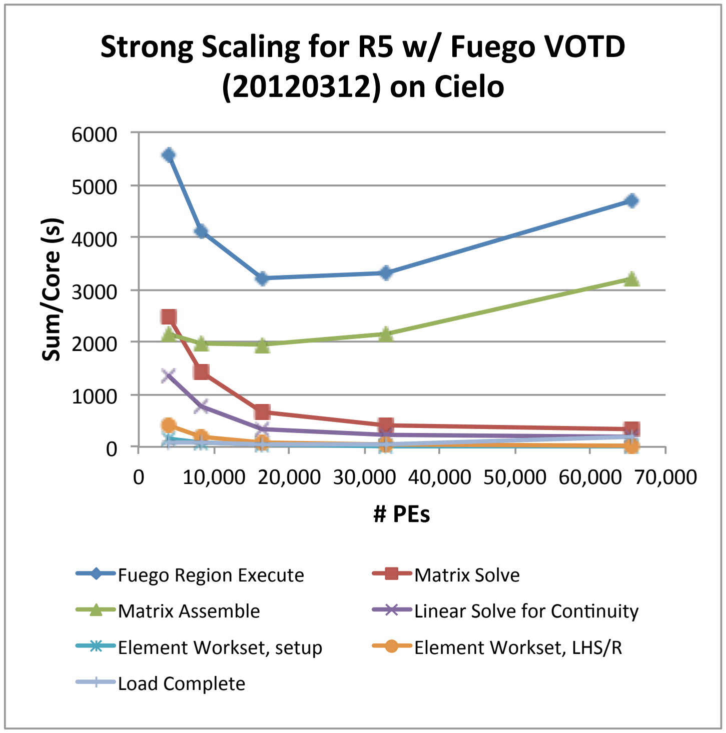
- ☐ Strong scaling for R5 mesh
- ☐ Weak scaling

Key MultiLevel Solver Settings

- Load-balancing coarse grid matrices
  - Repartition: max min ratio
    - Max allowable matrix row imbalance, above which repartitioning occurs
  - Repartition: partitioner = Zoltan
    - Load-balancing algorithm
    - Recursive coordinate bisection (RCB) from Zoltan
- Smoothers; Coarsening
  - Smoother: type = Chebyshev
    - Sparse matrix-vector multiply kernel
- Aggregation: threshold = 0.02
  - During coarse grid formation, ignores connections between fine DOFs with weak mutual influence
  - Aids in representing coefficient jumps, mesh anisotropy on coarse levels.
- General
  - Eigen-analysis: type = power-method
    - Estimate of largest eigenvalue of $D^{-1}A$
    - Required by smoother and prolongator construction

R5 Element-based Strong Scaling

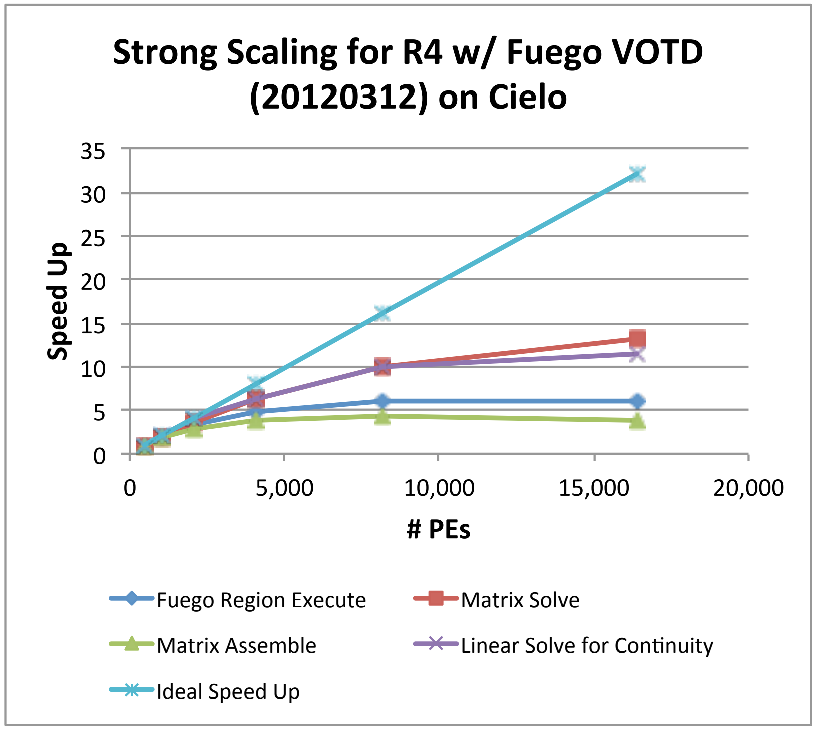


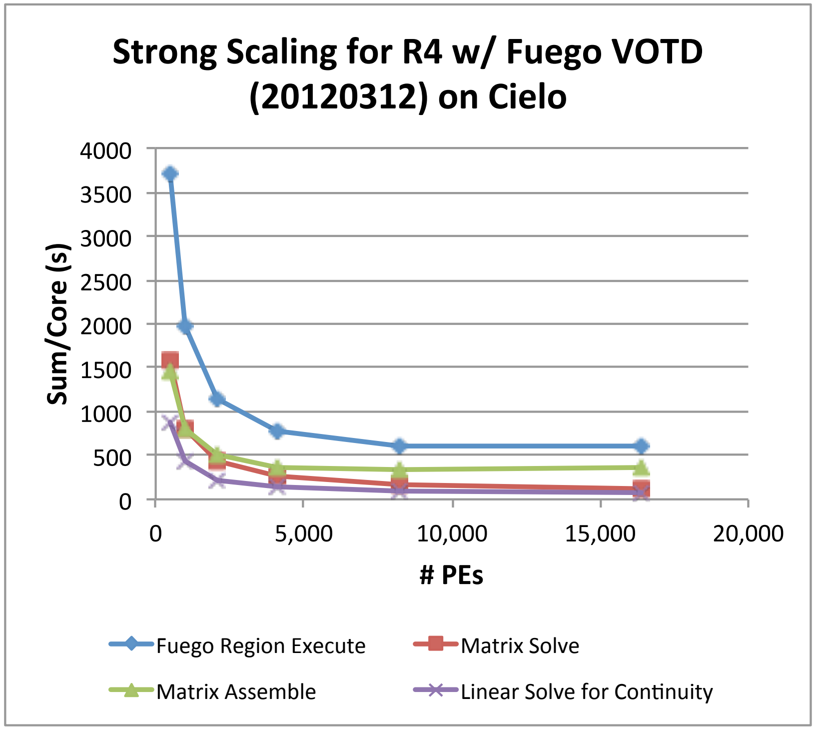


☐ Base for speed up is 4096 cores

☐ Time per core

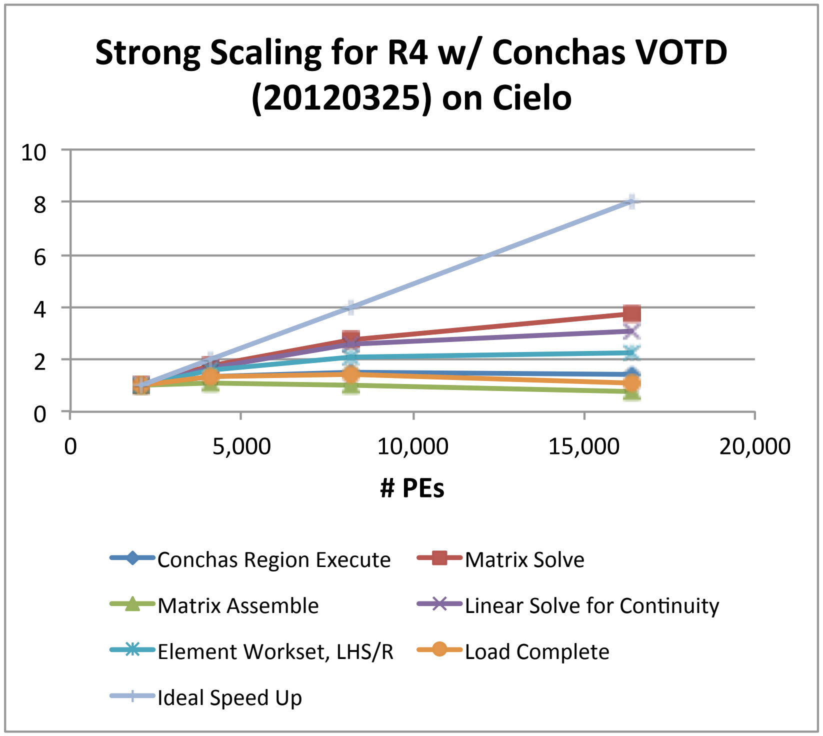
R4 Element-based Strong Scaling

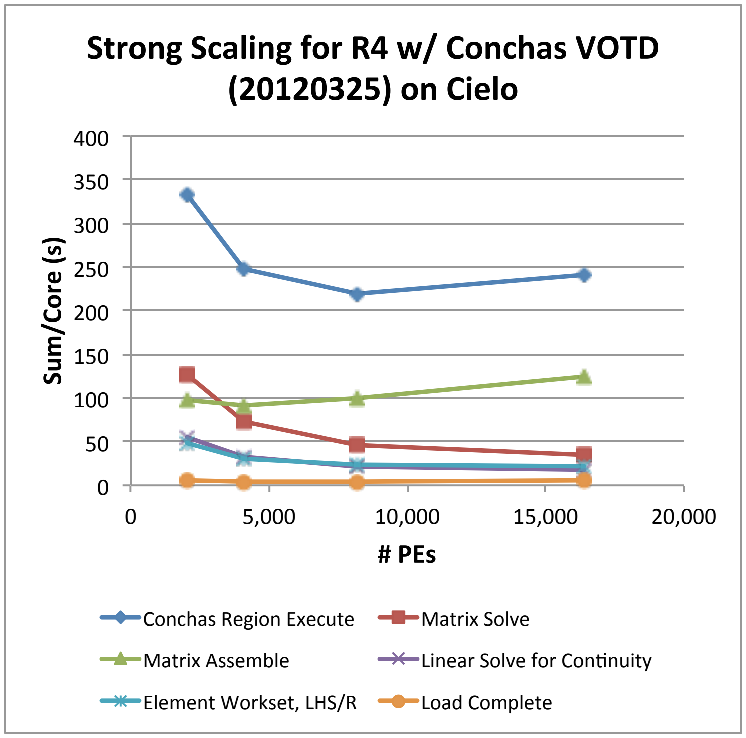




- ☐ Base for speed up is 512 cores
- ☐ Time per core

R4 Edge-based Strong Scaling

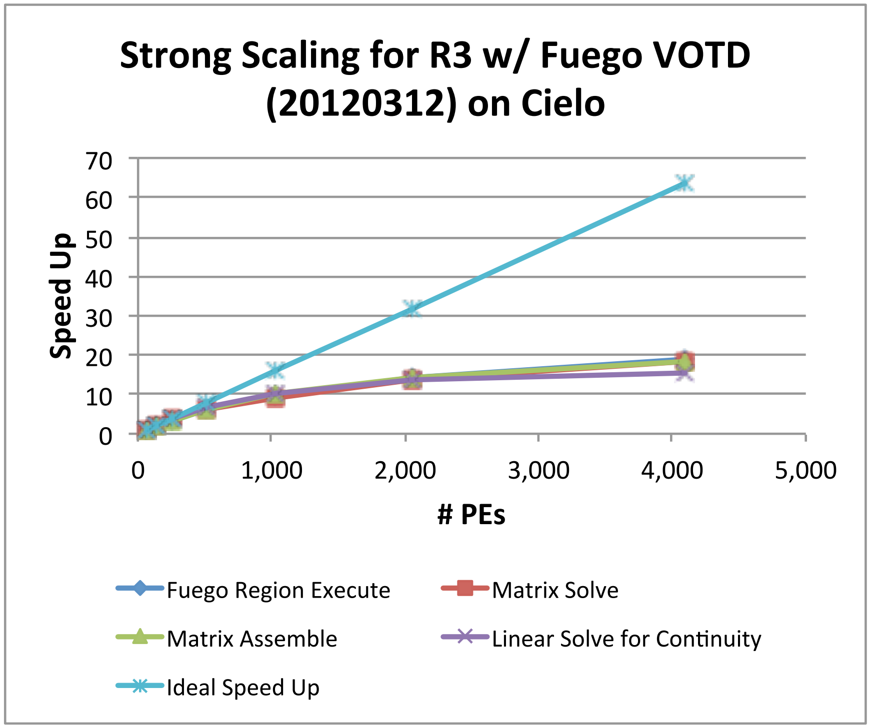


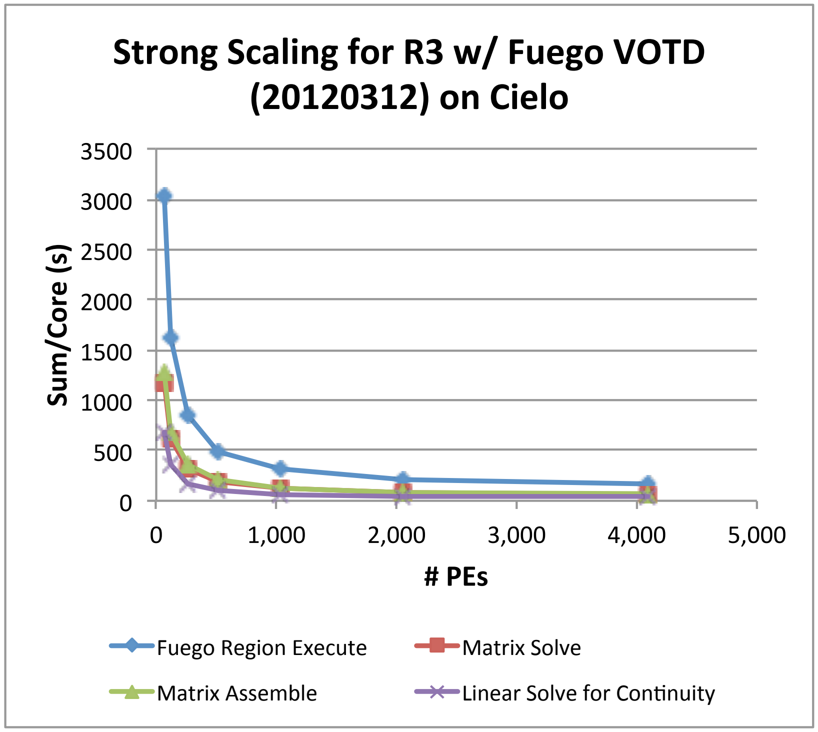


☐ Base for speed up is 2048 cores

☐ Time per core

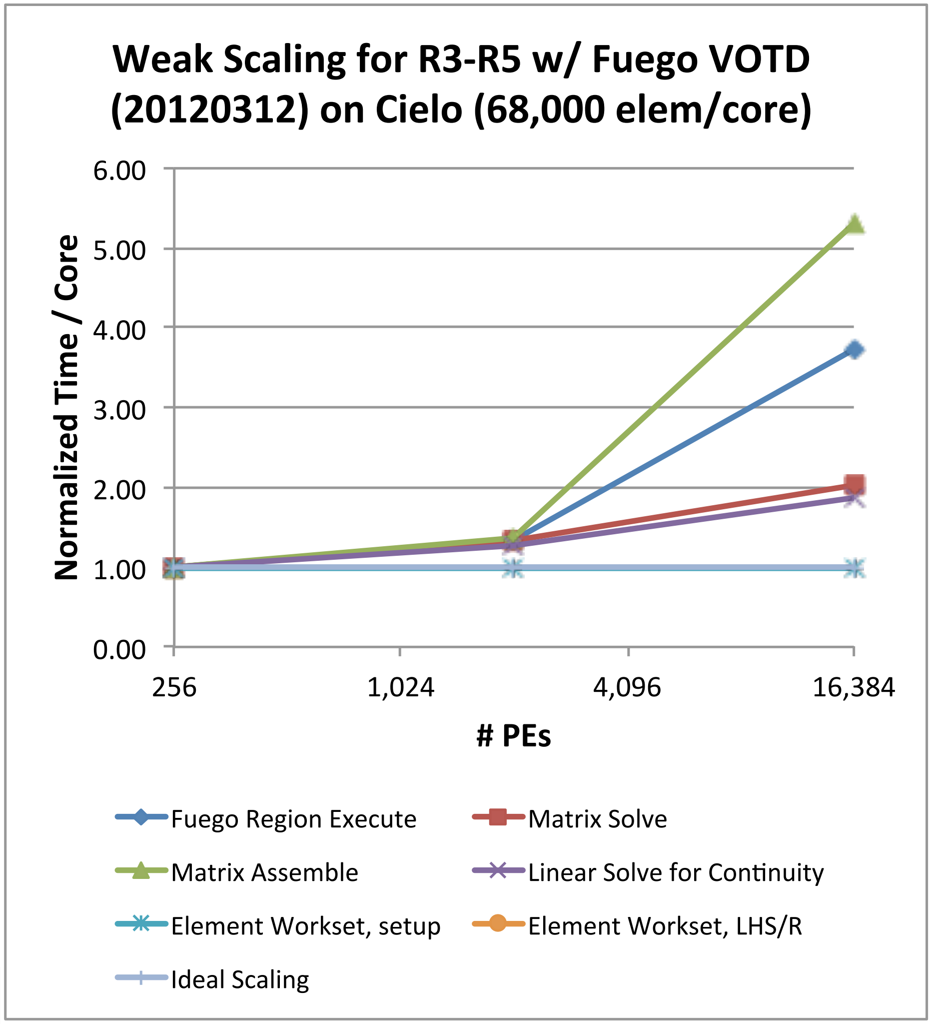
R3 Element-based Strong Scaling



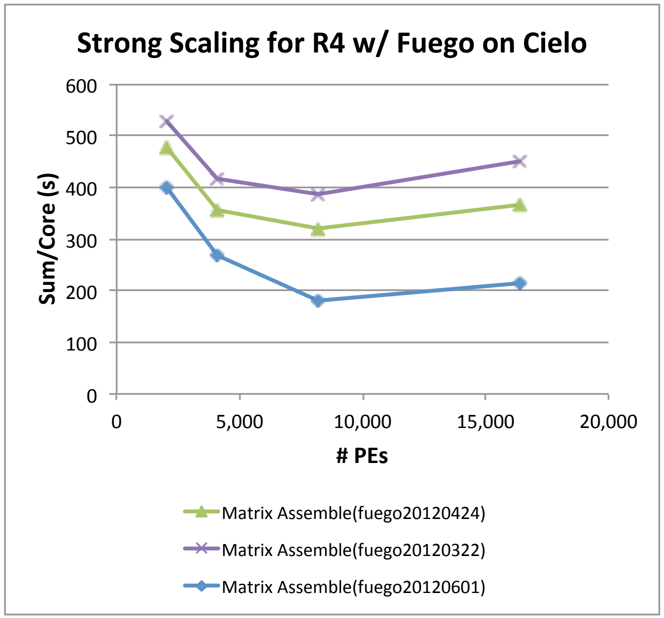


- ☐ Base for speed up is 64 cores
- ☐ Time per core

R3-R5 Element-based Weak Scaling



- ☐ Time per code normalized by 256-core simulation time
- ☐ Scaling of overall matrix assembly is in need of improvement as ideal scaling is expected
- ☐ Matrix solves also are non-optimal



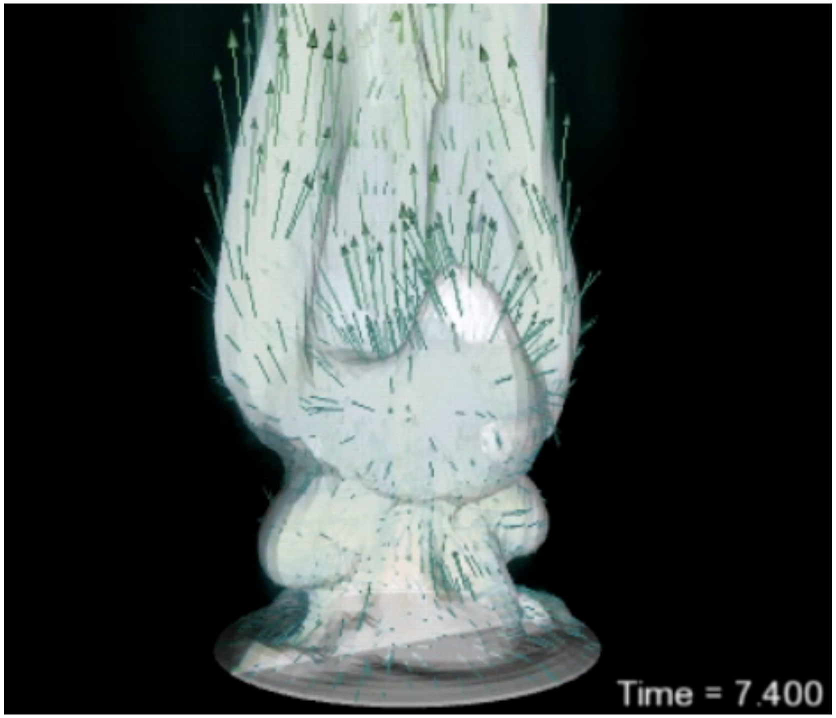
- ☐ Performance enhancement

Capability Class Mesh Resolutions

Level	Code	Machine/Proc	Time mm:ss	Nnode	Nelem	Exo (Gzip) File Size
R1	-	-		278.9K	273.7K	
R1R2	Encore votd	blade/4	21:06	2.2M	2.2M	
R1R2	Encore migrate	blade/4	15:20			
R1R2	Percept	blade/4	00:26			
R2R3	Encore votd	blade/4	*out-of-mem*	17.6M	17.5M	1.2G
R2R3	Encore migrate	blade/4	*out-of-mem*			
R2R3	Percept	blade/4	02:02			
R3R4	Percept	blade/4	*out-of-mem*	140.4M	140.14M	9G (2.4G)
R3R4	Percept	Redsky/2K	00:06.4			
R4R5	Percept	Redsky/2K/ 512/4*	00:26	1.12G	1.12G	72G (21G)

Helium Plume V&V

- Goal: Compare a element and edge-based Helium plume simulation based on the data set of O'Hern et al.
  - See simulation study of Tieszen, Domino and Black (SAND2005-3210)



☐ 1 meter helium plume performance test case

Algorithm	time
Elem_PPE	303:25
Edge_PPE	59:48
Edge_PPE_noReset	46:43

☐ Overall simulation time comparison

☐ Fastest edge alg 6.5x faster

Summary of Accomplishments

- Strong and weak scaling studies have been performed on meshes ranging from 17 million to 1.12 billion elements on core counts up to 65,536
  - This scale is unprecedented in the unstructured low Mach community based on personal interactions with Utah, Minnesota and Stanford
- Various code design principles have been evaluated in the context of understanding the trade space between general code and specific code
  - Really, this speaks to a particular research need in the area of monolithic couplings for general, new applications
- The team has deployed a new numerical scheme, the edge-based low Mach discretization, that has been shown to be second order accurate and almost ~4x faster than the current element-based approximate projection method

Lessons Learned

“If you do not test ***it***, ***it*** will break”

“If it is not in the rtest suite, it is broken”

  - The SIERRA code project strives to provide multi-physics, capability class computing with an effort towards high performing code, however, we rarely test at scale and have just begun the implementation of germane performance tests (although potentially not yet fully representative)

- Performance testing requires running simulations at all scales with a variety of tools (code timers, VTune for cache effectiveness, etc.)
- High performing teams are made by combining a set of skilled staff aligned to a specific problem of interest acting as one unit
- Routine low Mach fluids productions runs can be noted in the ~200 million element range on~4-8k cores, however, at > 1 billion elements, the process halts due to mesh size and I/O impediments

Proposed Path Forward

- Scalability issues remain and should be addressed very soon
  - The team suggests a scalability epic that is shared between apps, algs and stk
- Edge-based scheme looks very promising and almost ready for transition to production NW runs
  - More representative meshes and applications are desired to make sure that the transition is smooth
- Hardening of Trilinos
  - ePetra can learn from FETI
  - Templated Trilinos call stack performance will be addressed in CSAR CCM portfolio (Domino)
- Deployment of a capability computing-based rtest suite that is run somewhat regularly

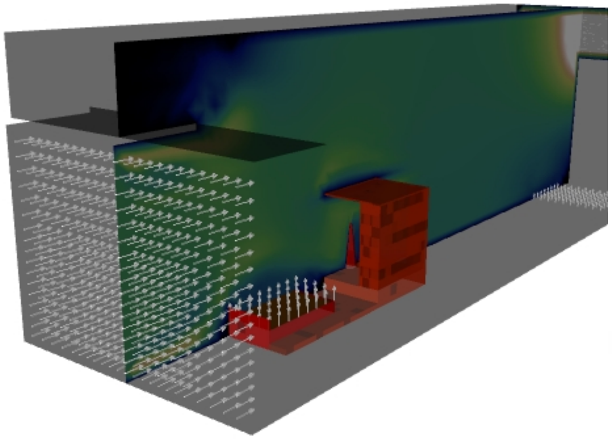
Cited Works

- Abdel et al, 1997; LDA measurements in the turbulent round jet; Mech. Re. Comm. 24:277-288
- Almgren et al., 2000; Approximate projection methods: part 1. Inviscid analysis, SIAM J. Sci. Comp., 22:1139-1159
- Brown et al., 2001; Accurate projection methods for the incompressible NS equations, J. Comp. Phys., 168:464-499
- Domino, 2006 and 2008; Towards verification of formal time accuracy for a family of approximate projection methods using MMS, Proceedings of the 2006 Summer Program, Stanford CTR, 163-177; A comparison of various equal order interpolation methodologies using MMS, Proceedings of the 2008 Summer Program, Stanford CTR, 97-111
- Hachem, 2010; Stabilized FEM for incompressible flows with high Reynolds number, J. Comp. Phys., 229(23):8643-8665
- Kim and Moin, 1987; Application of a fractional-step method to incompressible NS equations, J. Comp Phys., 162:411-428
- Moen et al, 2006; Fuego theory manual; section 3.9.1
- Schneider and Raw, 1997; Control volume finite element method for heat transfer and fluid flow using collocated variables, Part I; Num. Heat Trans. 11:363-390

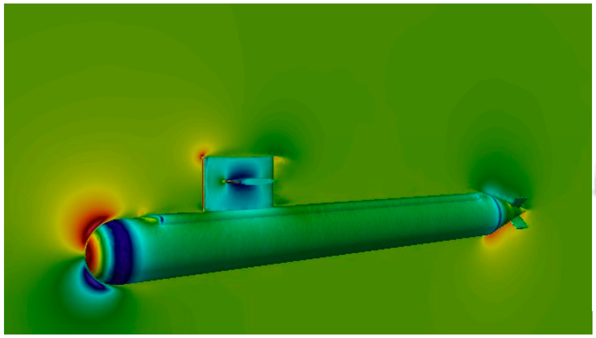
Extra Slides

- The slides that follow include potential topics of discussion.
- Note that all movies that have been attached are here in image form
- The abstract of the presentation is also included

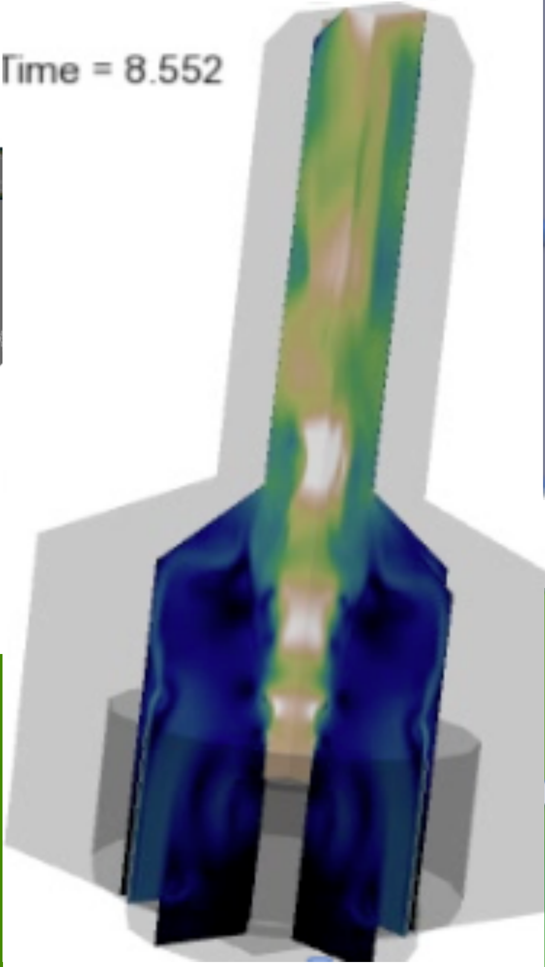
Novel Edge-Based Scheme Deployed



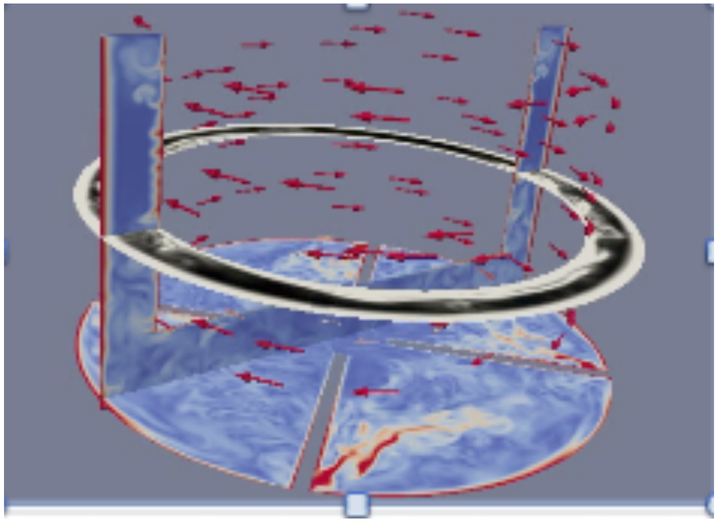
☐ NW XTF



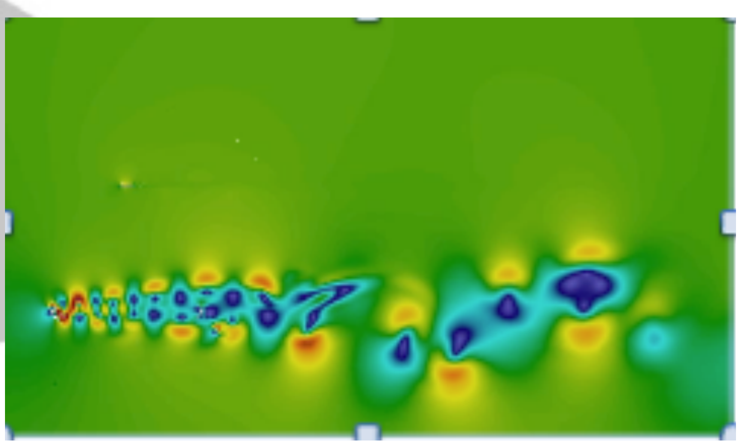
☐ WFO Tet-based sub



☐ NW FLAME



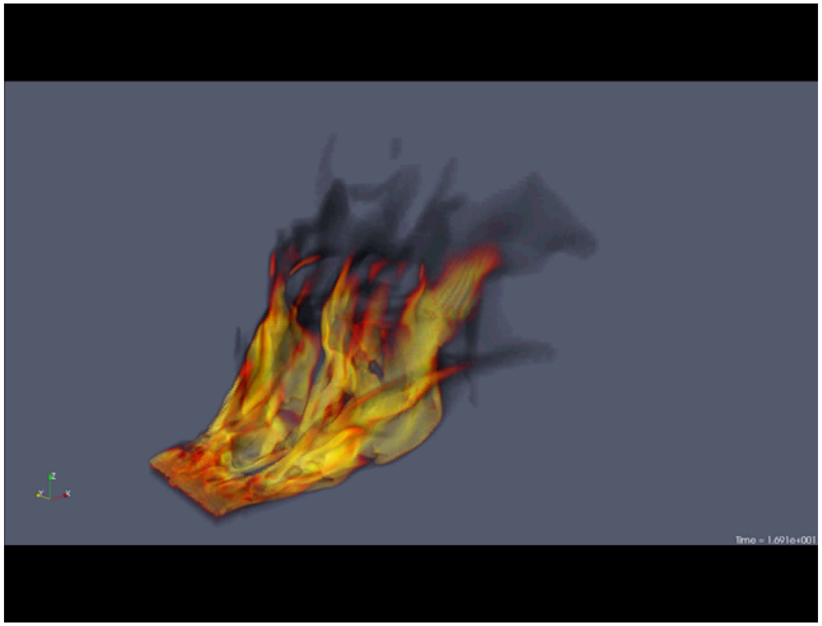
LDRD Waste reprocessing



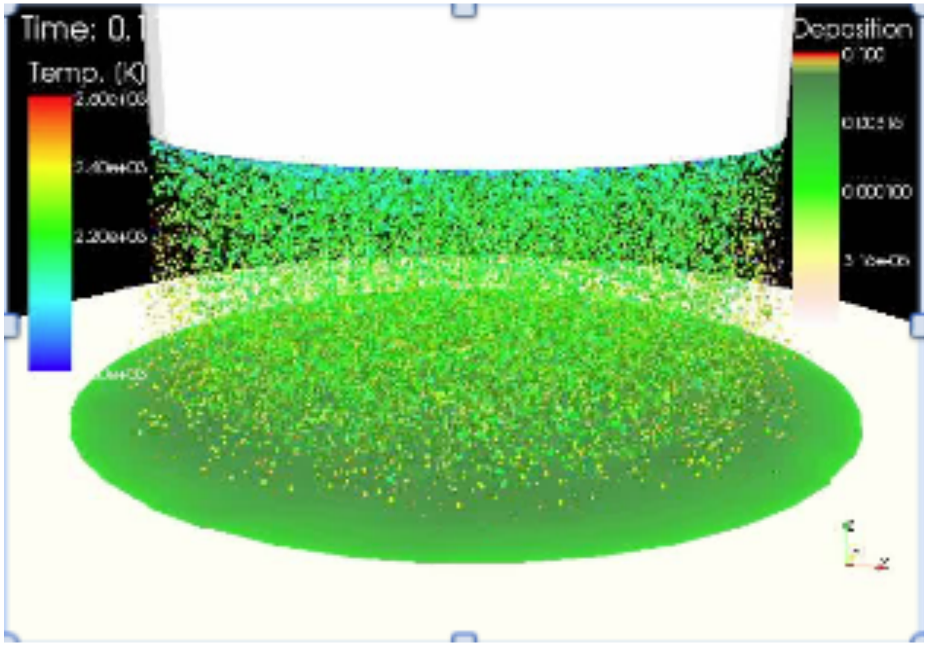
☐ WFO Wind energy apps

Simulation Capability

- Hydrocarbon JP-8 pool fire



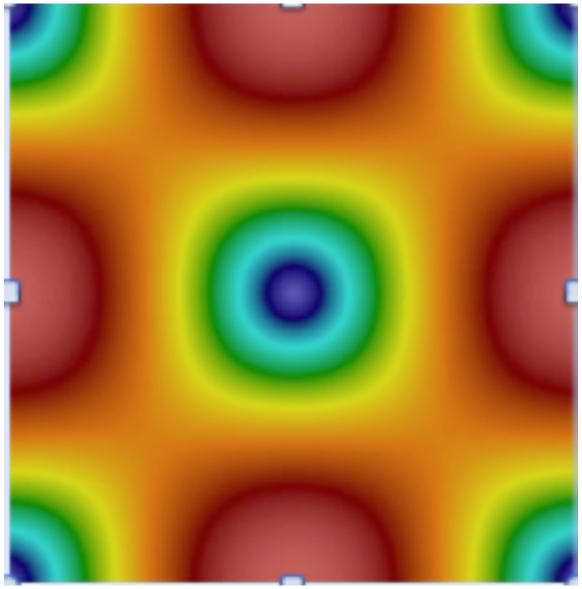
- Aluminum propellant fire



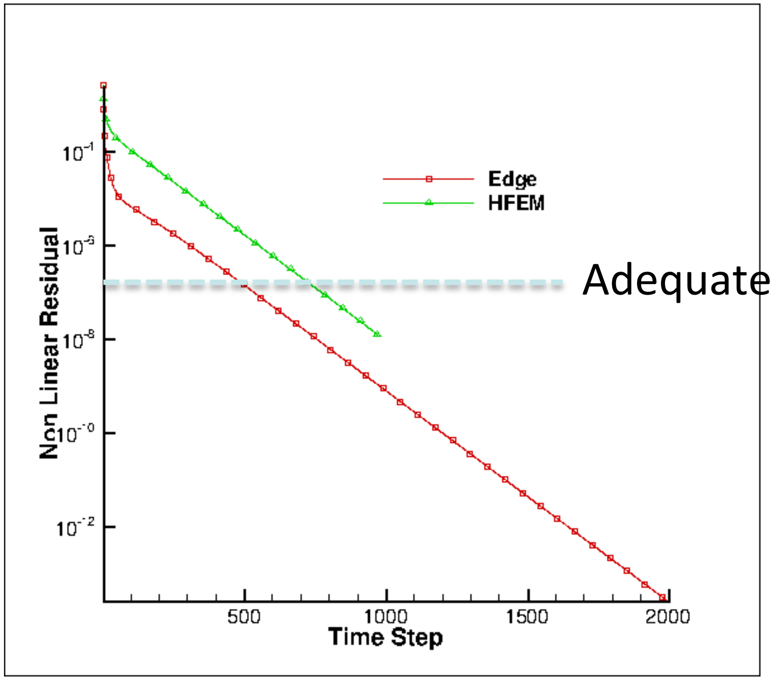
- ☐ Multi-physics pool fire simulation that supported the W76-1 qualification effort
- ☐ Multi-physics propellant fire simulation that is supporting the W88 effort

Algorithmic Convergence (1/2)

- Comparison between monolithic FEM and Edge-based FV
  - Edge-based scheme is operator split, approximate pressure projection
  - Monolithic FEM scheme is full analytical sensitivities with exception of nodal L2 grad(P) equation



☐ Steady TV (pressure)

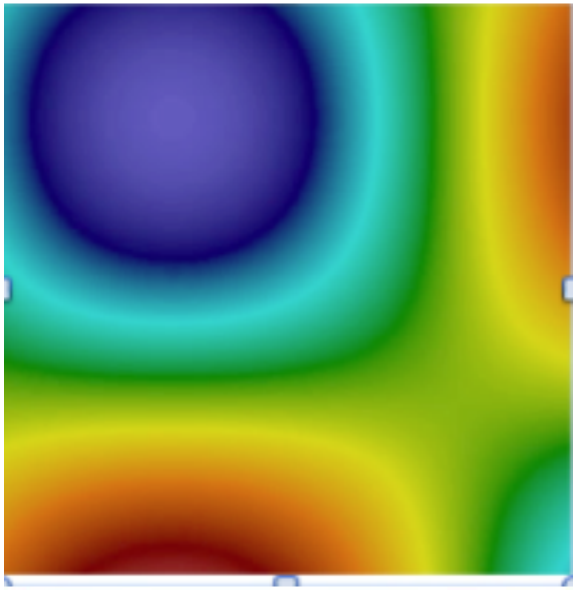


Adequate tolerar

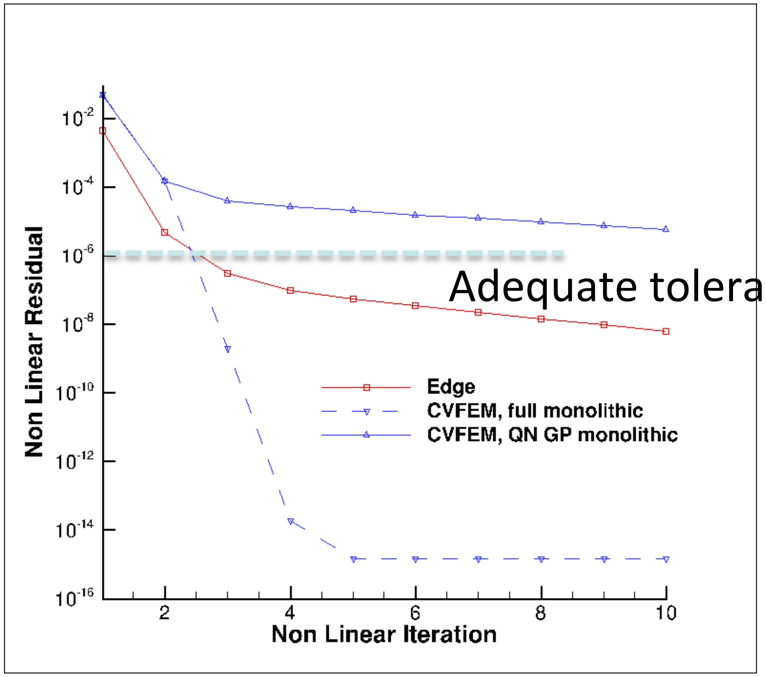
☐ Steady TV NLR

Algorithmic Convergence (2/2)

- Comparison between monolithic CVFEM and Edge-based FV
  - Edge-based scheme is operator split, approximate pressure projection
  - Monolithic scheme is full analytical sensitivities with and without nodal L2 grad(P) equation



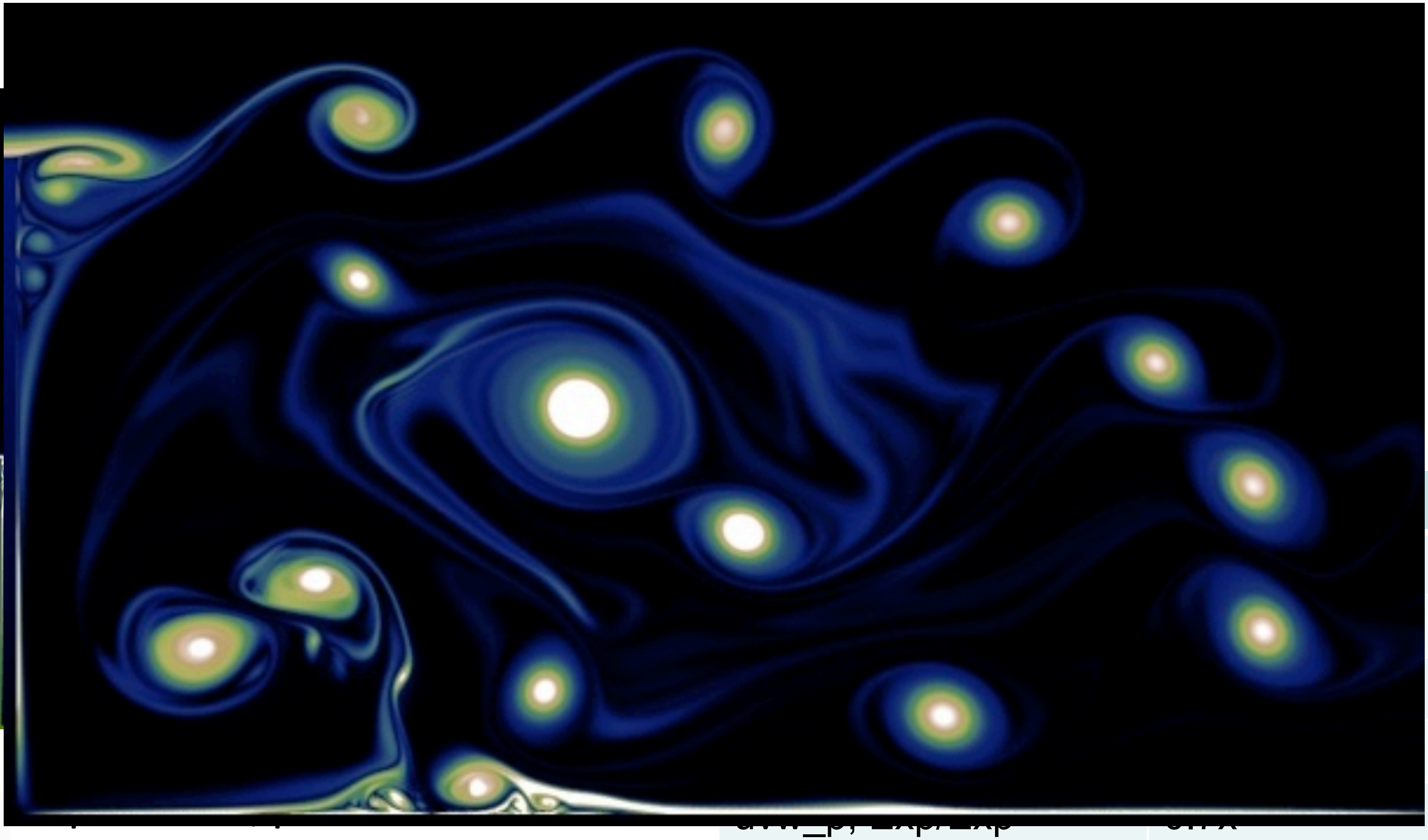
☐ Convective TV (velocity mag)



Adequate tolerance

☐ Convecting TV NLR

Coupling



2nd and 4th

Fast Gather/Scatters; FETI

	Date of Code	Trilinos/FEI gmres/SGS	Trilinos +Fast Gather	FETI BiCGStab/ Jacobi (+FG)	Trilinos + Fast Gather + Fast Scatter
Case: edgeOpenJet, 35K elem, 75 timestesp					
Development version	1/24/12	72.00		59.00	
Speedup (X factor)		1.00		1.22	
Add Fast Gathers pre-stk-migration	3/27/12		62.00	48.50	54.60
Speedup (X factor)			1.16	1.48	1.32
Fast Gather + post-stk-migration	4/10/12		61.00	47.30	53.60
Speedup (X factor)			1.18	1.52	1.34

Accessing Field Data w/ Fmwk

- Framework gathers require 7 memory hops at each node: