# Efficient Abstractions for Exascale Software Design

JAMES C. SUTHERLAND
Associate Professor - Chemical Engineering
The University of Utah

DOE Awards
DE-NA0002375
DE-NA-000740

National Science Foundation
WHERE DISCOVERIES BEGIN
PetaApps award 0904631
XPS award1337145

# Acknowledgments


**Matt Might**
Associate Professor
School of Computing

**Chris Earl**
Post-Doctoral Research
Associate (now at LLNL)

**Tony Saad**
Senior Computational Scientist

**Devin Robison**
M.S. Student
Now at Fusion IO

**Abhishek Bagusetty**
M.S. Student
Now at U. Pittsburgh

**Amir Biglari**
Ph.D. Student

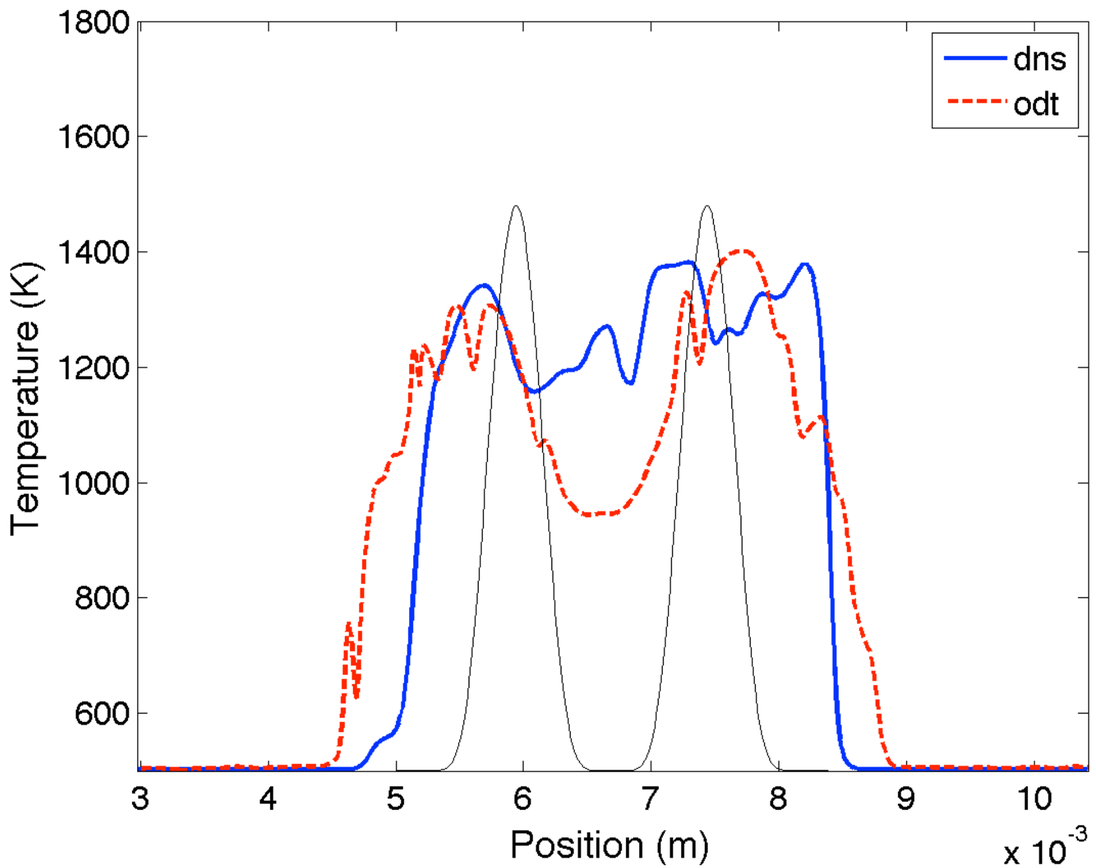
**Babak Goshayeshi**
Ph.D. Student
Now a Post-Doc

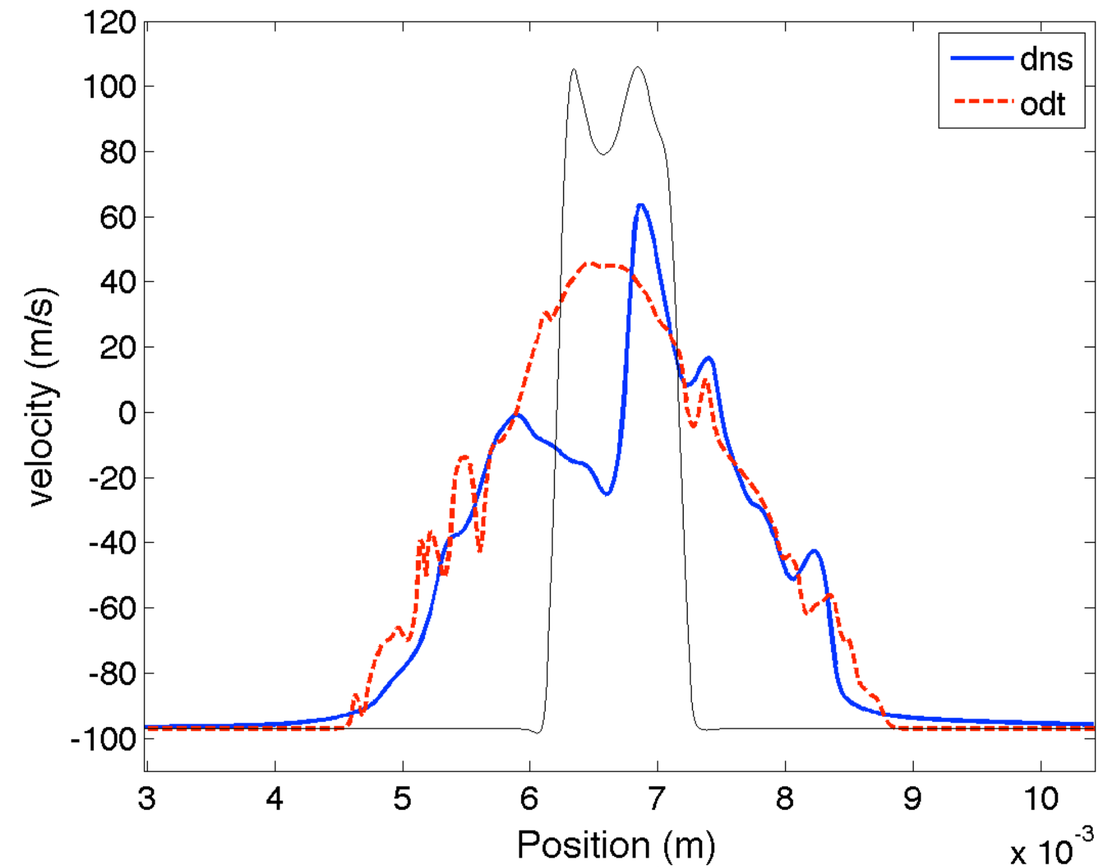
**Nathan Yonkee**
Ph.D. Student

US DOE/NNSA

National Science Foundation
WHERE DISCOVERIES BEGIN
NSF PetaApps award 0904631

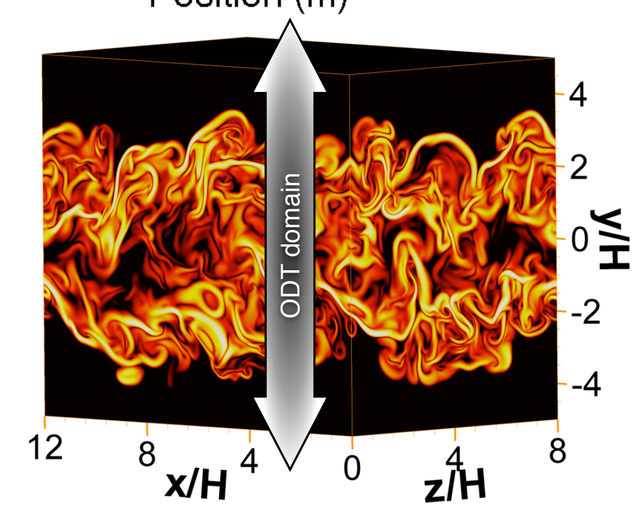
# One-Dimensional Turbulence (ODT)



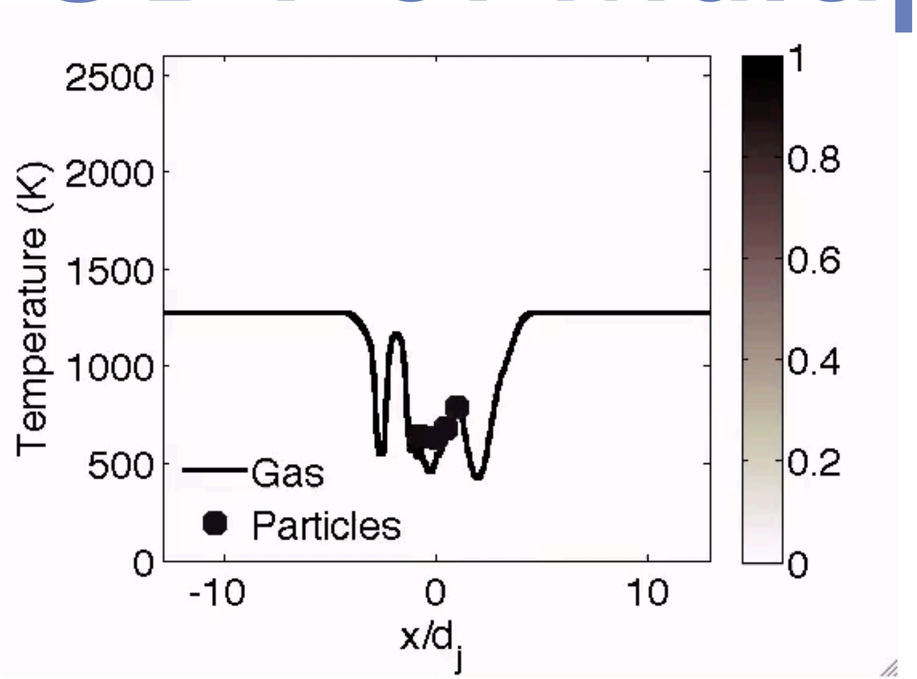



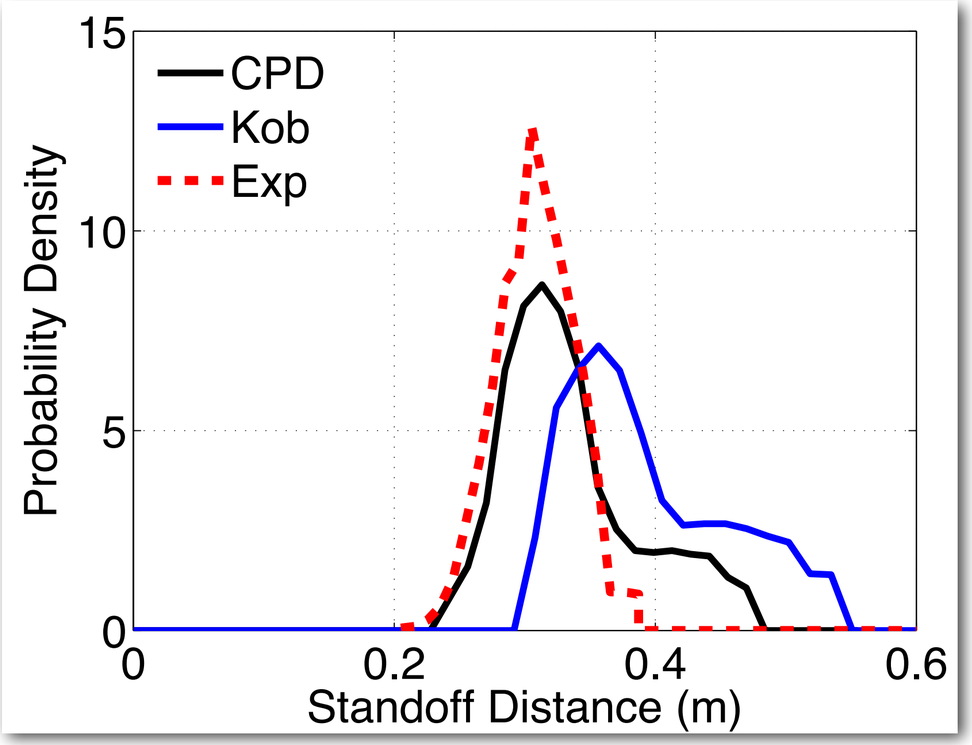

## Cost:

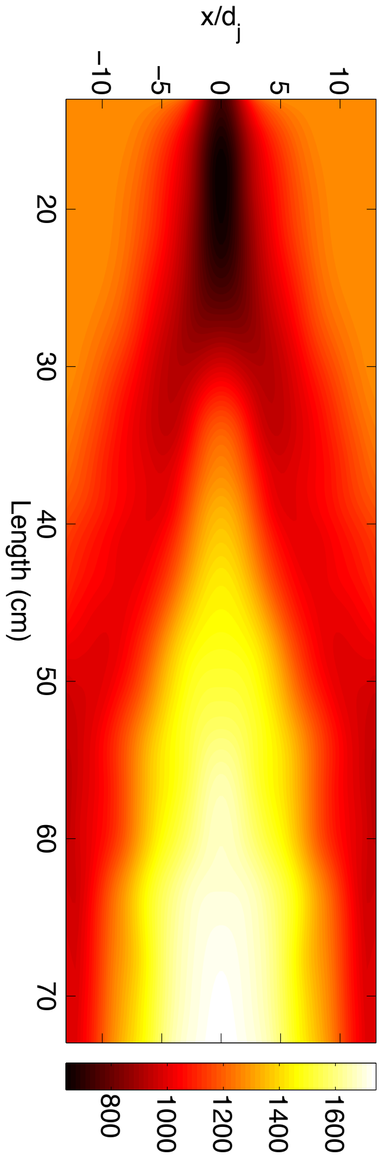
- ODT: ~600 CPU hours (~1.5 hours/realization, 400 realizations) - scales as $Re^{3/2}$.
- DNS: ~2 million hours - scales as $Re^{3}$.

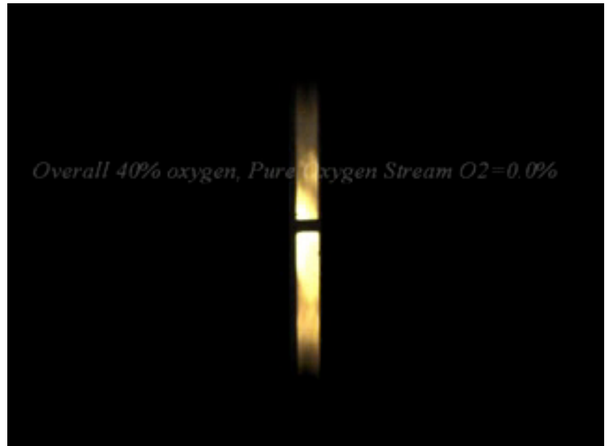


# ODT of Multiphase Reacting Flows



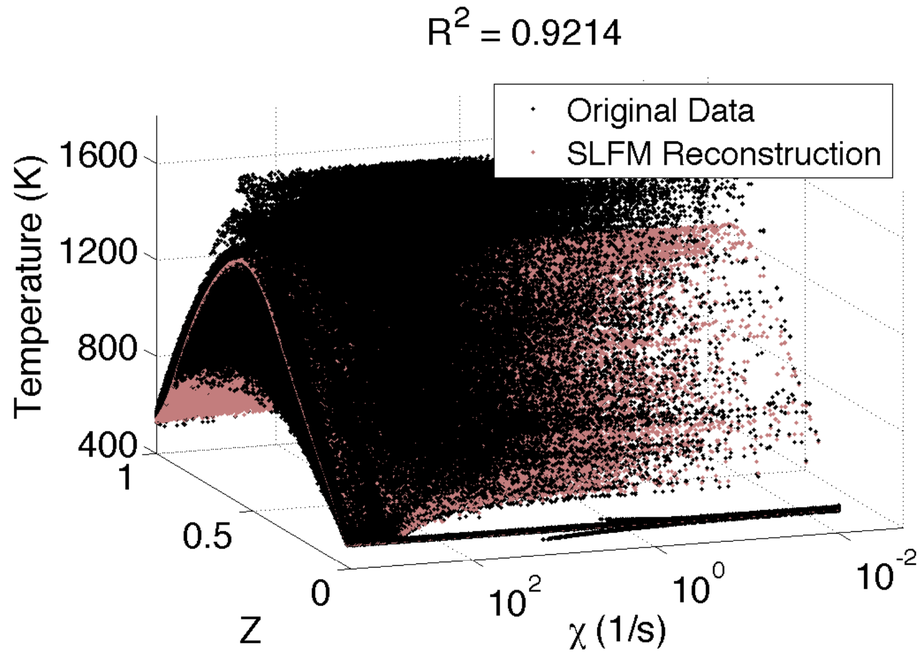



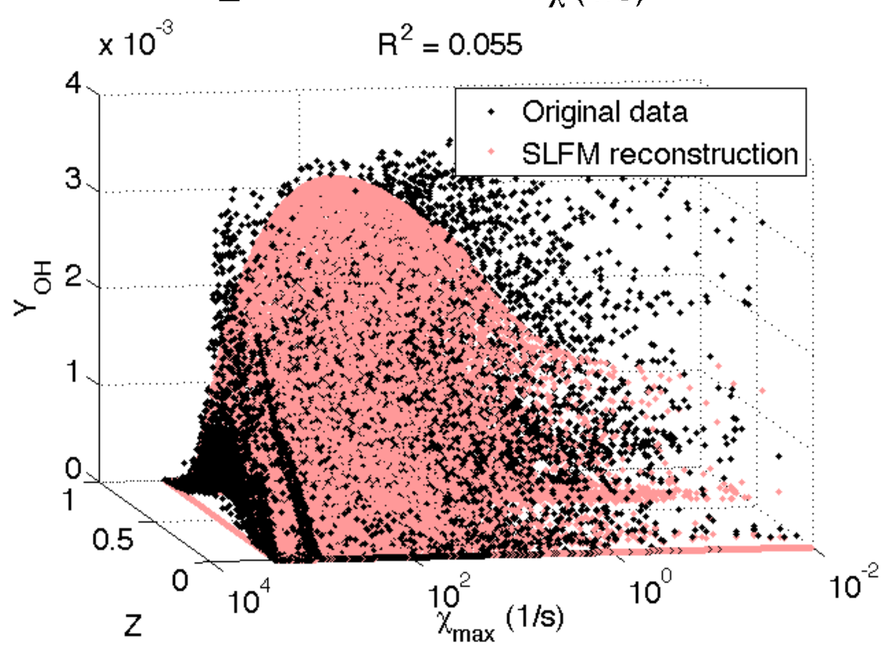



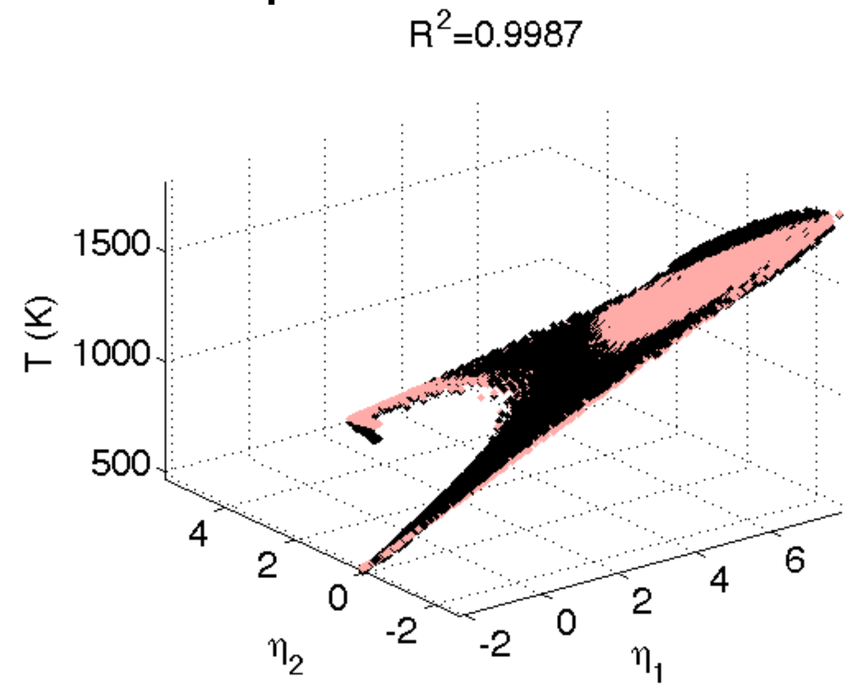
# Parameterizing Manifolds in Turbulent Combustion

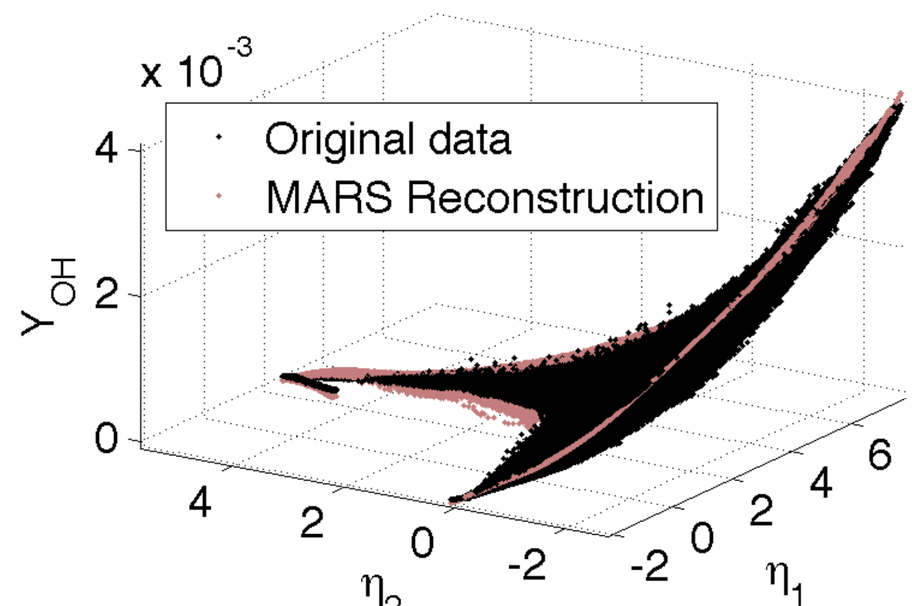
## Common parameterization



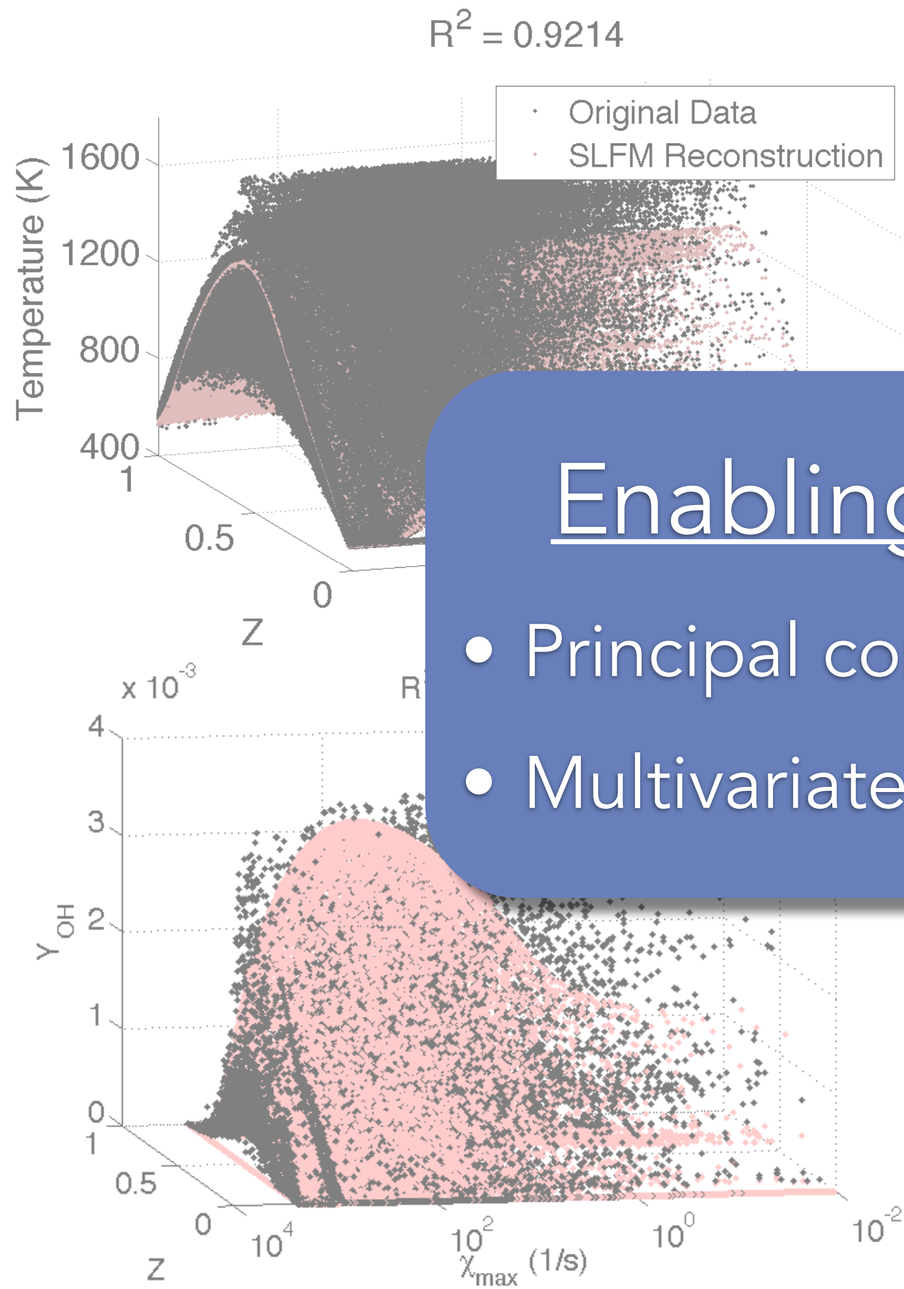

## PCA parameterization



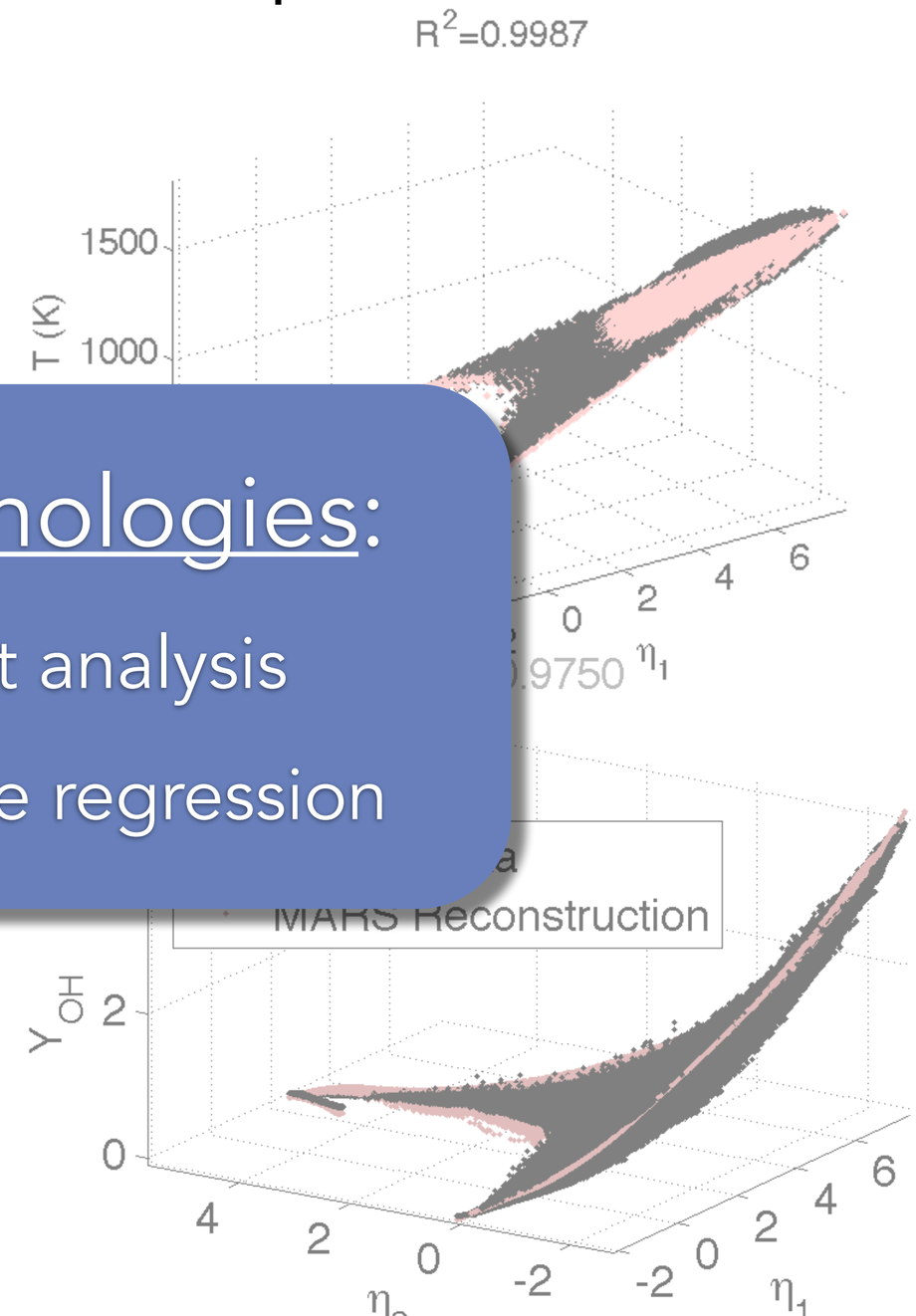



# Parameterizing Manifolds in Turbulent Combustion

## Common parameterization




## PCA parameterization

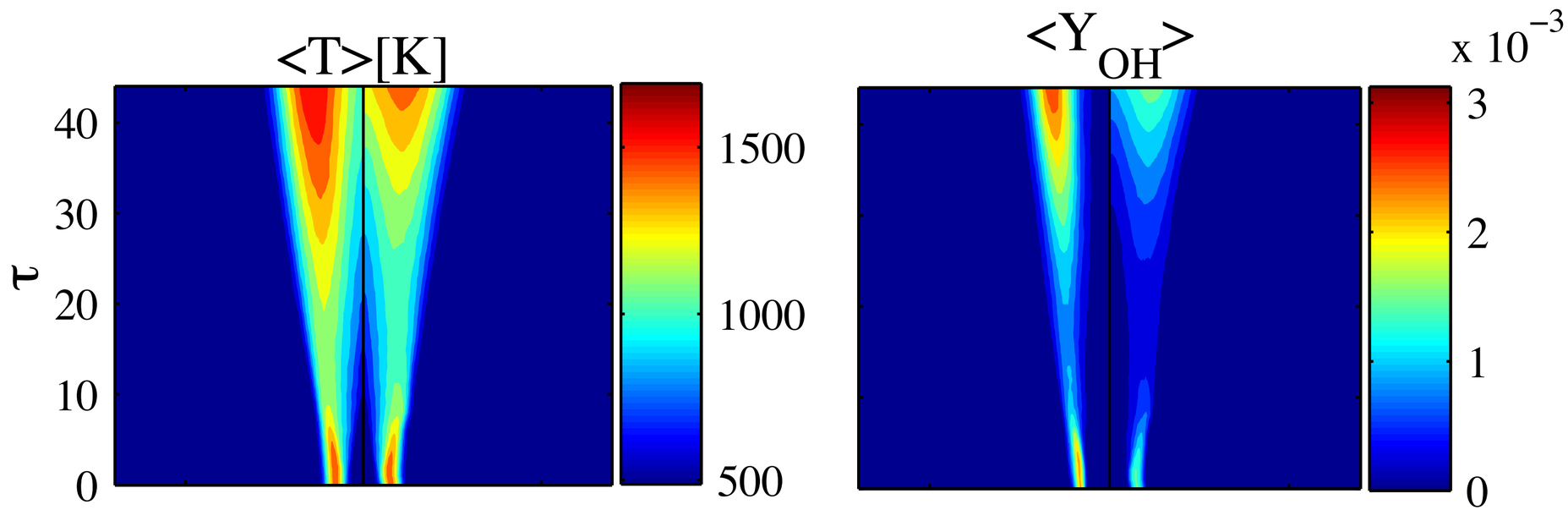



**Enabling technologies:**

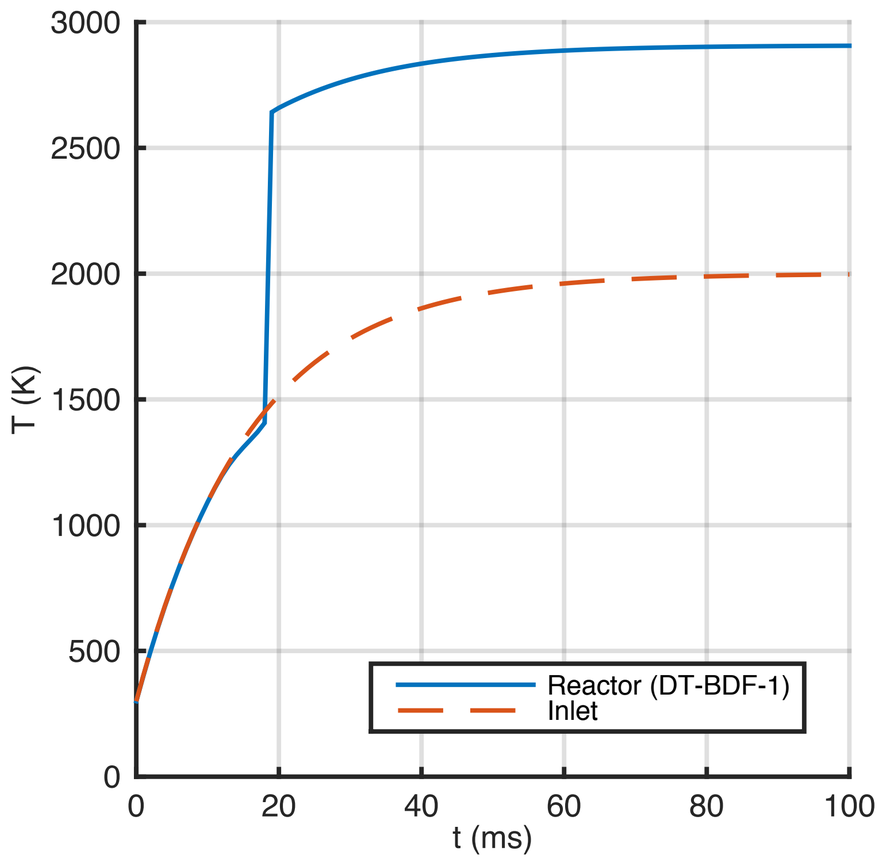
- Principal component analysis
- Multivariate adaptive regression

# Reduced cost while maintaining accuracy

PCA to identify model on 11-dimensional original system.
Truncation to two dimensions.



# Time Integration Methods for Stiff, Nonlinear Systems




**Dual time-stepping**

| |
|---|
| Robust nonlinear solver for explosive chemistry. |
| Equally fast as Newton's method at small Δt. |
| Stable for any Δt. |
| Gives choice of resolution despite ignition/extinction. |
| Significant performance gains with our adaptive Δσ scheme. |

<u>Heptane/Air (654 spec., 4846 rxns)</u>

$\Delta t_{max}$ Newton = 0.1 μs

$\Delta t_{process}$ = 1 ms

est. 2000x speedup

# HPC is becoming more challenging

Physics problems that we are tackling are increasingly complex.

Hardware architectures are increasingly complex, uncertain, and even divergent!

Cost of software rewrites measured in tens of millions of dollars!

| *Year* | *Machine* | *Peak Speed* | *Cores* | *Programming model* | *Cost ($)* | *Footprint* | |
|---|---|---|---|---|---|---|---|
| 1997 | ASCI Red | 1 TFLOPS | 9,298 | MPI (distributed memory) | 55 M | 1,600 $ft^2$ |  |
| 2012 | Sequoia | 16 PFLOPS | 1,572,864 (98,304x16) | MPI (+threads) (mixed) | ~250 M | 3,000 $ft^2$ |  |
| 2012 | Titan | 17 PFLOPS | 299,000 CPU (18,688x16) 50,233,344 GPU | MPI + CUDA + threads (mixed) | 97 M | 4,350 $ft^2$ |  |
| 2014 | Xeon Phi | 1 TFLOPS | ~50 | "traditional" | ~3 K | your foot |  |
| 2014 | NVidia GPU | ~3 TFLOPS | 4,992 | CUDA | ~3 K | your foot |  |

# Taming the complexity beast…

## Goals:

- Efficiently use complex modern architectures.
- Enhance programmer productivity by insulating the programmer from details.

## Enabling technologies:

### Task-graph:

- MPI communication
- Threaded task scheduling
- Allows overlap of computation with communication
- Automatic memory management (fields are where you need them, when you need them)

### Domain-Specific Language

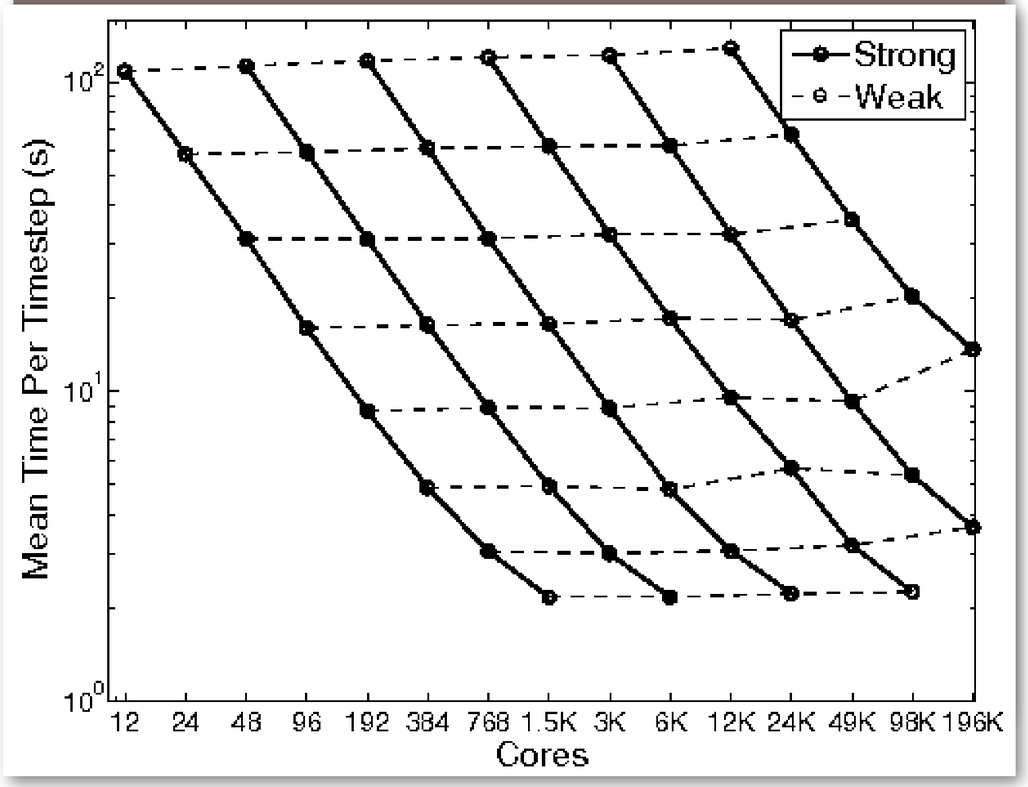
- Array & stencil operations
- GPU & multithread execution

# Hierarchical Parallelization

- Uintah framework[1]
  - *Domain decomposition data parallelism*
  - *Task parallelism & coarse-grained DAG*
  - *Scales to largest capability machines.*

**Distributed task-graph:**

- MPI communication
- Coarse-grained, threaded task scheduling




1. Berzins, M., Meng, Q., Schmidt, J., & Sutherland, J. C., DAG-based software frameworks for PDEs. In Euro-Par 2011: Parallel Processing Workshops (pp. 324–333). Springer.

# Hierarchical Parallelization

- Uintah framework[1]
  - *Domain decomposition data parallelism*
  - *Task parallelism & coarse-grained DAG*
  - *Scales to largest capability machines.*
- "Expression Library" [2]
  - *"fine-grained" task graph for PDE assembly.*
  - *tasks consist of stencil & field operations*
- Nebo EDSL[3]
  - *GPU & multithreaded execution*
  - *Matlab-style syntax*
- Utilize resources at "high" level & "push down" as parallelism runs out

**Distributed task-graph:**

- MPI communication
- Coarse-grained, threaded task scheduling

**On-node task-graph**

- memory management & fine-grained task scheduling.
- thread-pools
- GPU management

**EDSL**

- array & stencil operations
- GPU & multithread execution

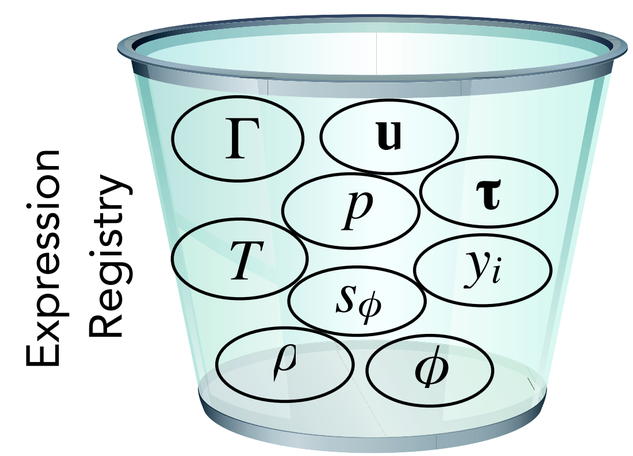

1. Berzins, M., Meng, Q., Schmidt, J., & Sutherland, J. C., DAG-based software frameworks for PDEs. In Euro-Par 2011: Parallel Processing Workshops (pp. 324–333). Springer.

2. Notz, P. K., Pawlowski, R. P., & Sutherland, J. C., Graph-Based Software Design for Managing Complexity and Enabling Concurrency in Multiphysics PDE Software. ACM TOMS (2012).

3. Earl, C., Might, M., Bagusetty, A., & Sutherland, J. C., Nebo: An efficient, parallel, and portable domain-specific language for numerically solving partial differential equations. Journal of Systems and Software, to appear.

# A Simple Example of DAGs

- Register all expressions
  - *Each "expression" calculates one or more field quantities.*
  - *Each expression advertises its direct dependencies.*

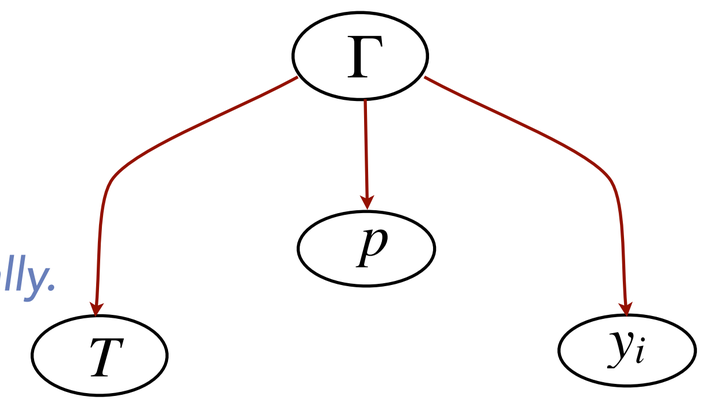



*Notz, Pawlowski, & Sutherland (2012). ACM Transactions on Mathematical Software, 39(1).

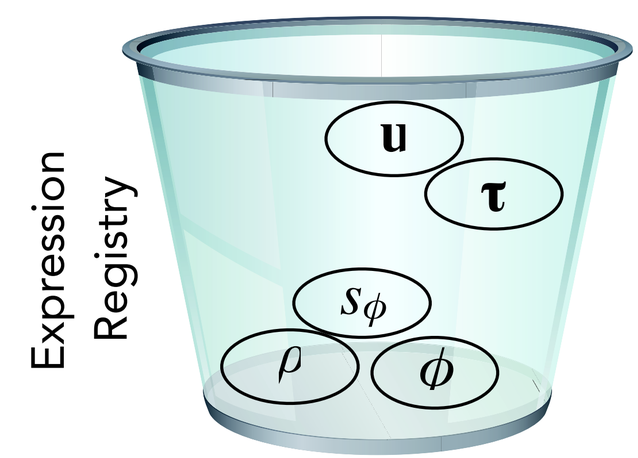
# A Simple Example of DAGs

- Register all expressions
  - *Each "expression" calculates one or more field quantities.*
  - *Each expression advertises its direct dependencies.*
- Set a "root" expression; construct a graph
  - *All dependencies are discovered/resolved automatically.*
  - *Highly localized influence of changes in models.*
  - *Not all expressions in the registry may be relevant/used.*

$$\Gamma = \Gamma(T, p, y_i)$$




Direct (expressed) dependencies.

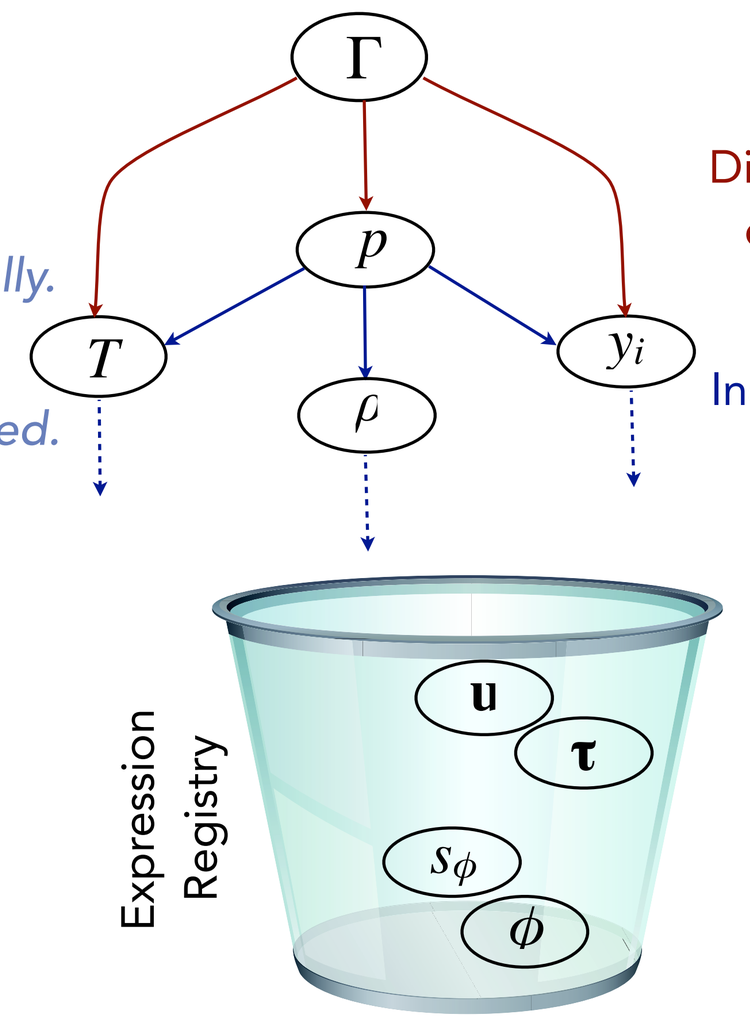



*Notz, Pawlowski, & Sutherland (2012). ACM Transactions on Mathematical Software, 39(1).

# A Simple Example of DAGs

- Register all expressions
  - *Each "expression" calculates one or more field quantities.*
  - *Each expression advertises its direct dependencies.*
- Set a "root" expression; construct a graph
  - *All dependencies are discovered/resolved automatically.*
  - *Highly localized influence of changes in models.*
  - *Not all expressions in the registry may be relevant/used.*

$$\Gamma = \Gamma(T, p, y_i)$$




Direct (expressed) dependencies.

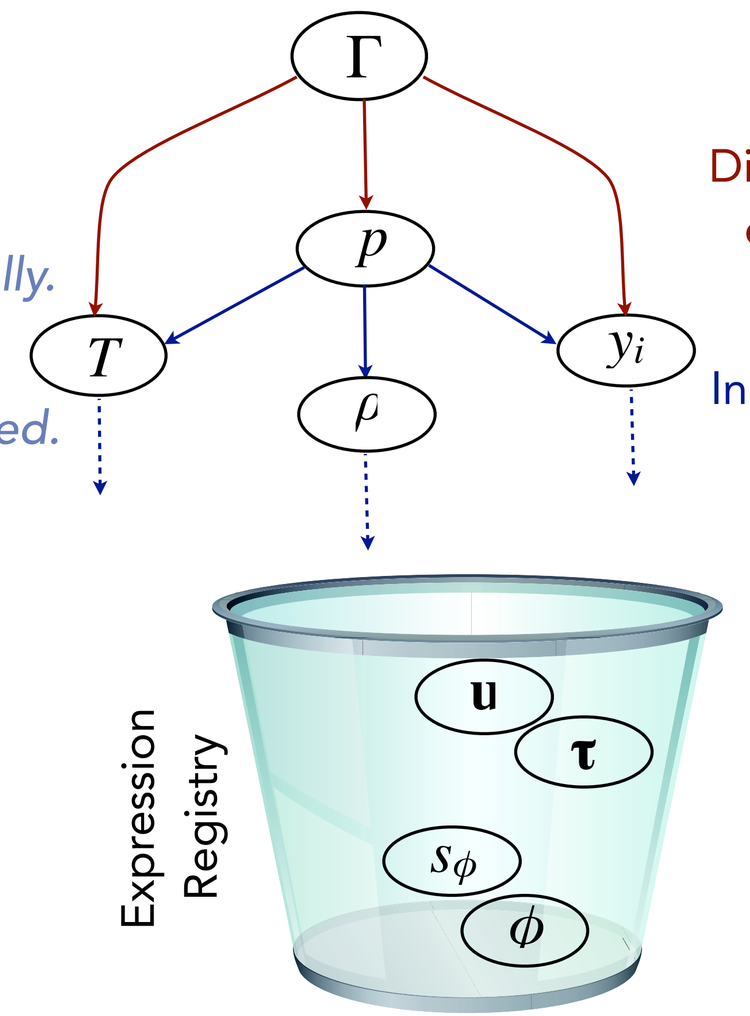
Indirect (discovered) dependencies.

*Notz, Pawlowski, & Sutherland (2012). ACM Transactions on Mathematical Software, 39(1).

# A Simple Example of DAGs

- Register all expressions
  - *Each "expression" calculates one or more field quantities.*
  - *Each expression advertises its direct dependencies.*
- Set a "root" expression; construct a graph
  - *All dependencies are discovered/resolved automatically.*
  - *Highly localized influence of changes in models.*
  - *Not all expressions in the registry may be relevant/used.*
- From the graph:
  - *Deduce storage requirements & allocate memory (externally to each expression).*
  - *Automatically schedule evaluation, ensuring proper ordering.*
    - Asynchronous execution is critical! (overlap communication & computation)
  - *Robust scheduling algorithms are key.*

$$\Gamma = \Gamma(T, p, y_i)$$




Direct (expressed) dependencies.

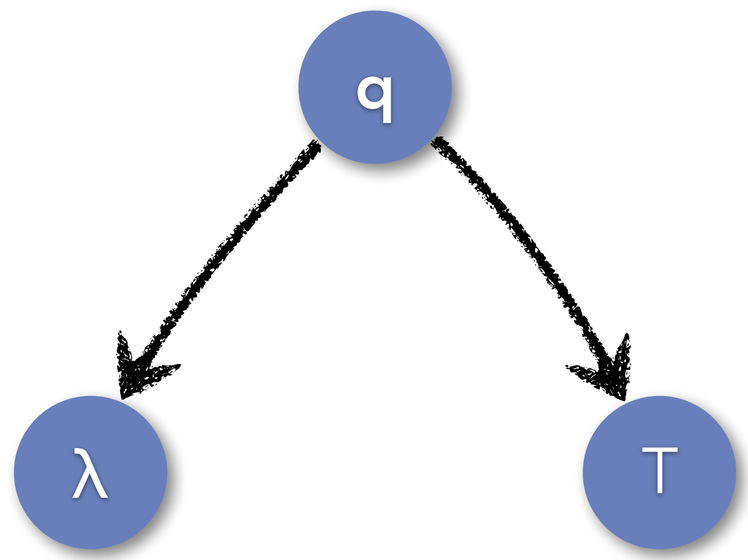
Indirect (discovered) dependencies.

*Notz, Pawlowski, & Sutherland (2012). ACM Transactions on Mathematical Software, 39(1).

# Changes in model form are naturally handled

Pure substance heat flux:

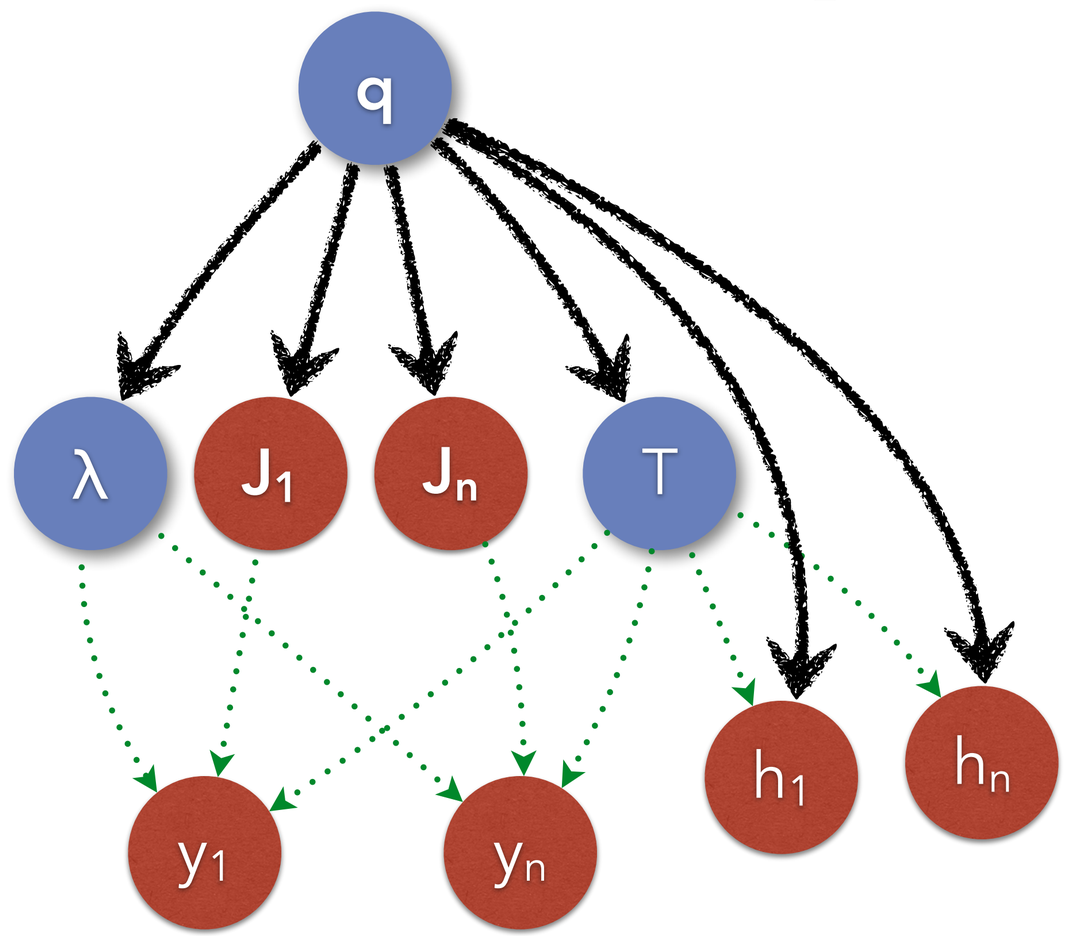
$$\mathbf{q} = -\lambda \nabla T$$



# Changes in model form are naturally handled

Multi-species mixture heat flux:

$$\mathbf{q} = -\lambda \nabla T + \sum_{i=1}^{n} h_i \mathbf{J}_i$$




No complex logic changes in code when model are added/changed.

# DSL: "Matlab for PDEs on Supercomputers"

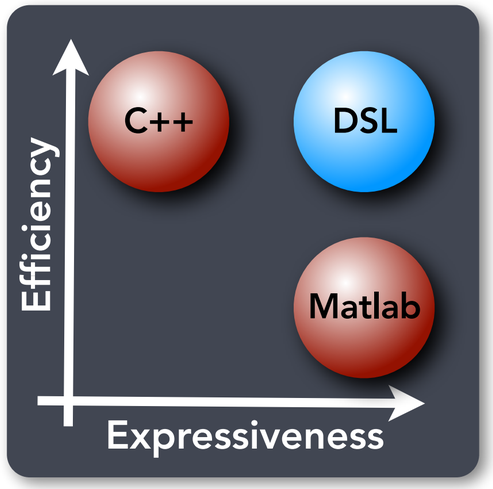
Field & stencil operations:

(and much more)

$$\text{rhs} = -\frac{\partial}{\partial x}(J_x + \mathcal{C}_x) - \frac{\partial}{\partial y}(J_y + \mathcal{C}_y) - \frac{\partial}{\partial z}(J_z + \mathcal{C}_z)$$

```
rhs <<= -divOpX( xConvFlux + xDiffFlux )
        -divOpY( yConvFlux + yDiffFlux )
        -divOpZ( zConvFlux + zDiffFlux );
```

Can "chain" stencil operations.




- Embedded in C++ to avoid two-stage compilation & improve portability.
- CPU (serial, multicore), GPU, Xeon Phi backends.
- Cross-platform memory pools (CPU/GPU).
- 70+ natively supported discrete operators (easily define new ones).
- Strongly typed fields & operators.
- Masks - allow operations on field subsets.
- conditional operations (vectorized 'if').
- Field can live in multiple locations (GPU, CPU) *simultaneously*.

Auto-generate code for efficient execution on CPU, GPU, XeonPhi, *etc.* during compilation.

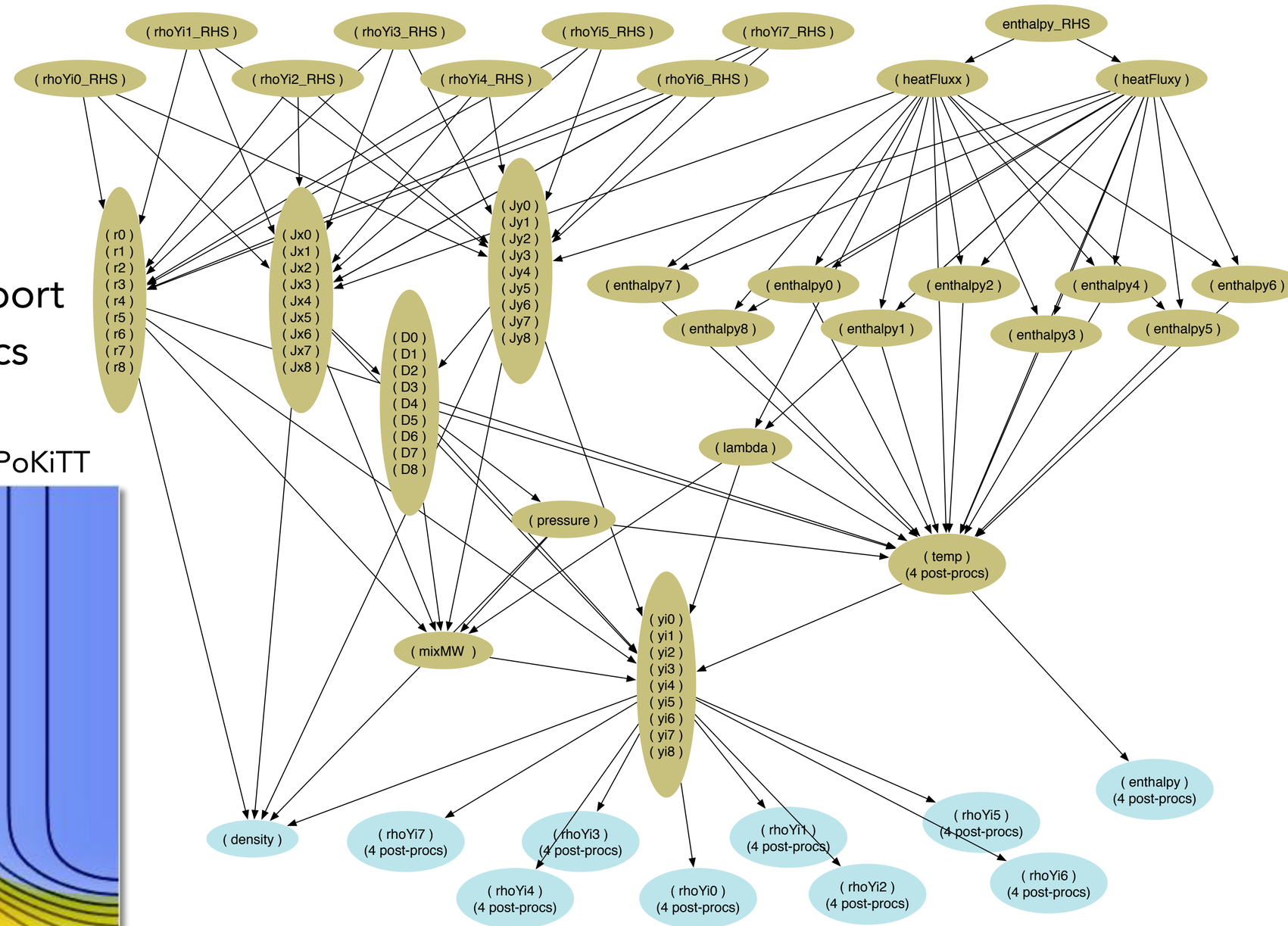
# Real example: PoKiTT

(**Po**rtable **Ki**netics, **T**hermodynamics & **T**ransport)

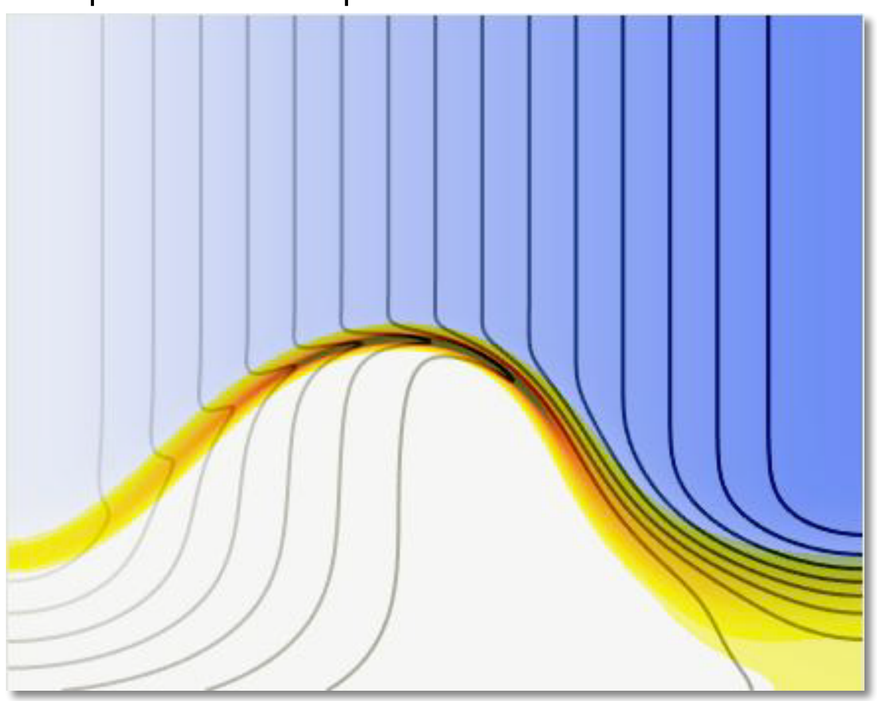
$$\rho \frac{\partial y_i}{\partial t} = -\nabla \cdot \mathbf{J}_i + s_i$$

$$\rho \frac{\partial h}{\partial t} = -\nabla \cdot \mathbf{q}_i$$

- Detailed kinetics
- Mixture-averaged transport
- Detailed thermodynamics




Triple flame computed on GPU with PoKiTT



# Real example: PoKiTT

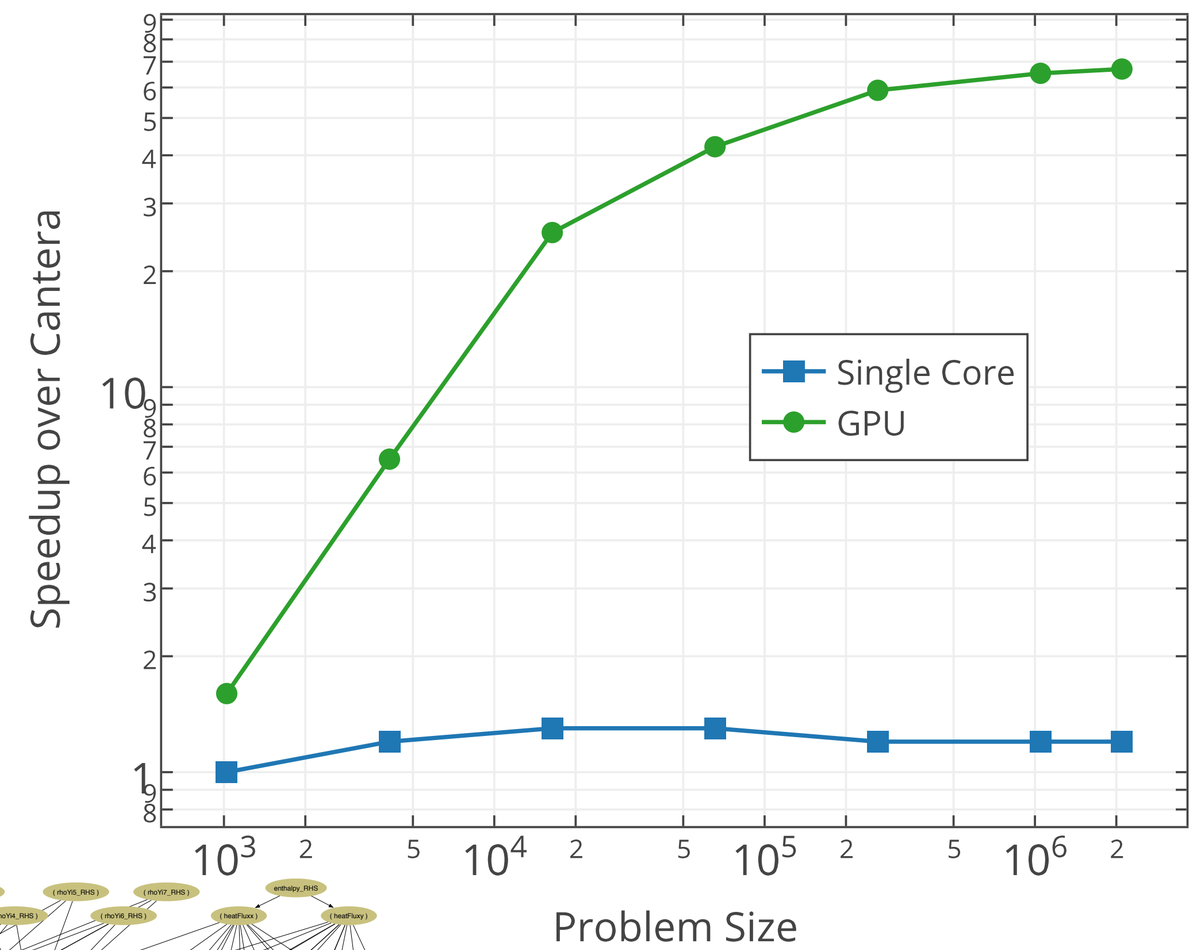
(**Po**rtable **Ki**netics, **T**hermodynamics & **T**ransport)

$$\rho \frac{\partial y_i}{\partial t} = -\nabla \cdot \mathbf{J}_i + s_i$$

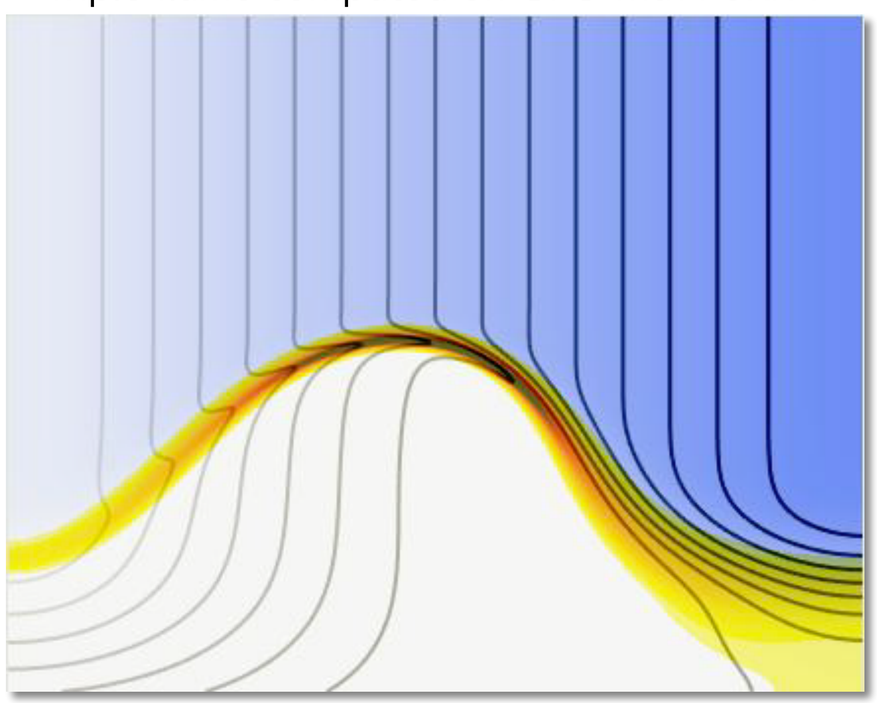
$$\rho \frac{\partial h}{\partial t} = -\nabla \cdot \mathbf{q}_i$$

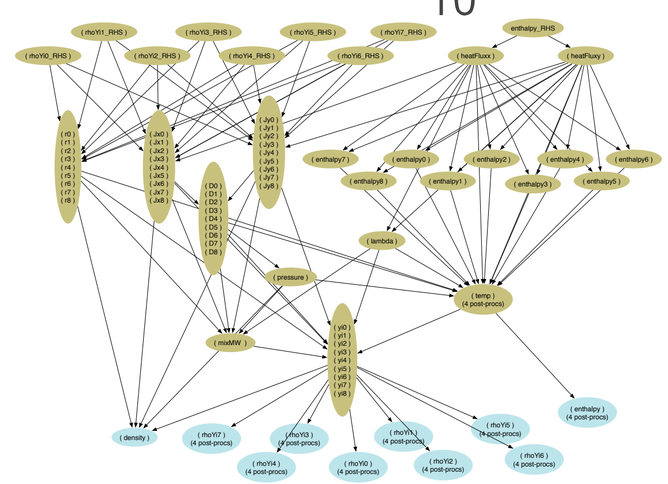
- Detailed kinetics
- Mixture-averaged transport
- Detailed thermodynamics

Reaction Rates



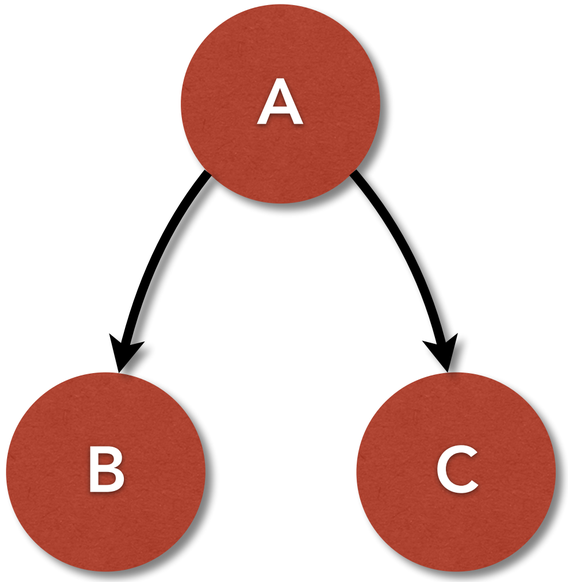

Triple flame computed on GPU with PoKiTT





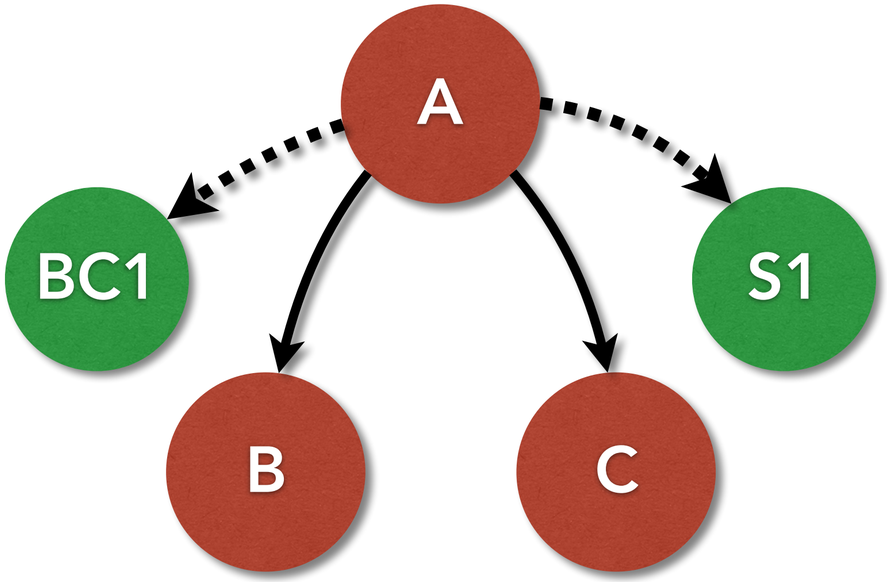
# "Modifiers" — injecting new dependencies

- Motivation:
  - ***Boundary conditions***: *modify a subset of the computed values.*
  - ***Multiphase coupling***: *add source terms to RHS of equations.*



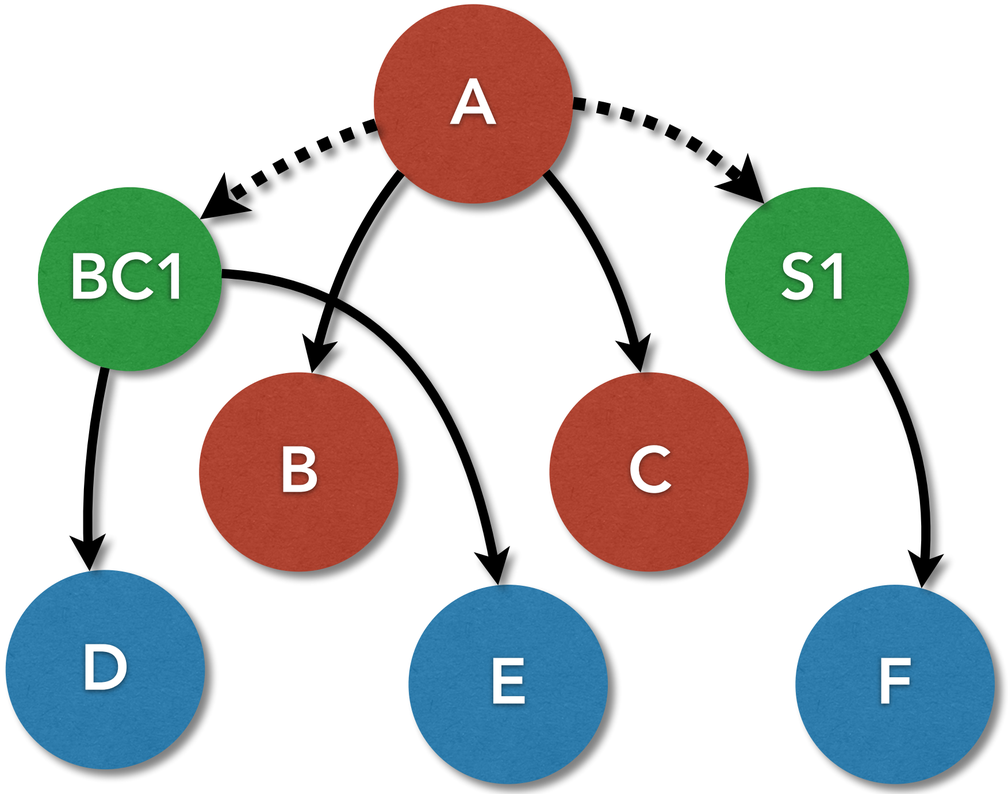
# "Modifiers" — injecting new dependencies

- Motivation:
  - ***Boundary conditions***: *modify a subset of the computed values.*
  - ***Multiphase coupling***: *add source terms to RHS of equations.*
- Modifiers allow "push" rather than "pull" dependency addition.
- Modifiers are deployed after the node they are attached to, and are provided a handle to the field just computed.



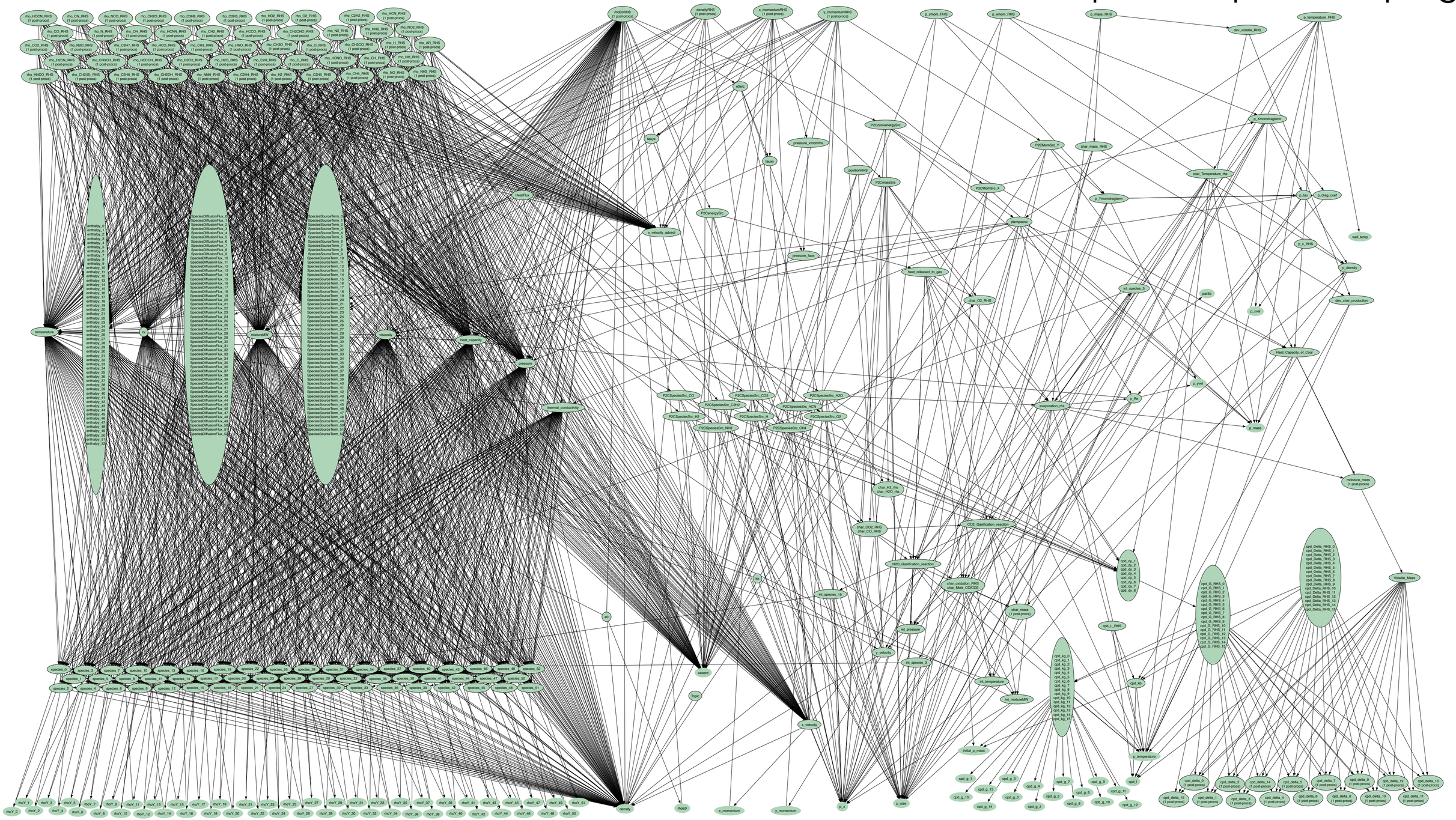
# "Modifiers" — injecting new dependencies

- Motivation:
  - ***Boundary conditions***: *modify a subset of the computed values.*
  - ***Multiphase coupling***: *add source terms to RHS of equations.*
- Modifiers allow "push" rather than "pull" dependency addition.
- Modifiers are deployed after the node they are attached to, and are provided a handle to the field just computed.
- Modifiers can introduce new dependencies to the graph.



# Coal combustion

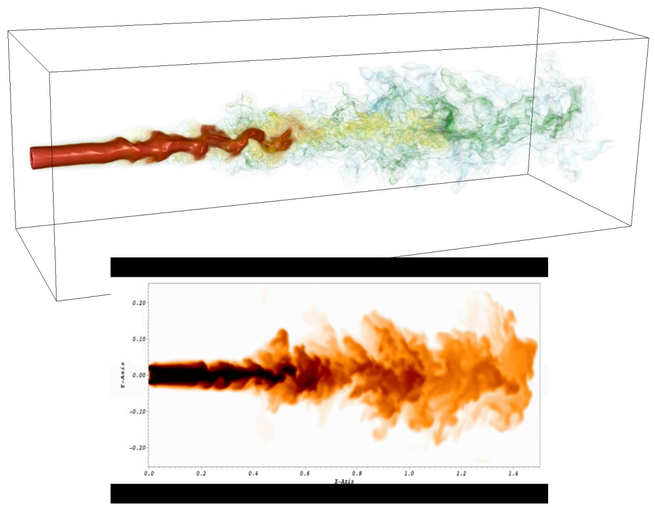
- 55 PDEs
- ~35 ODEs per particle
- Complex interphase coupling

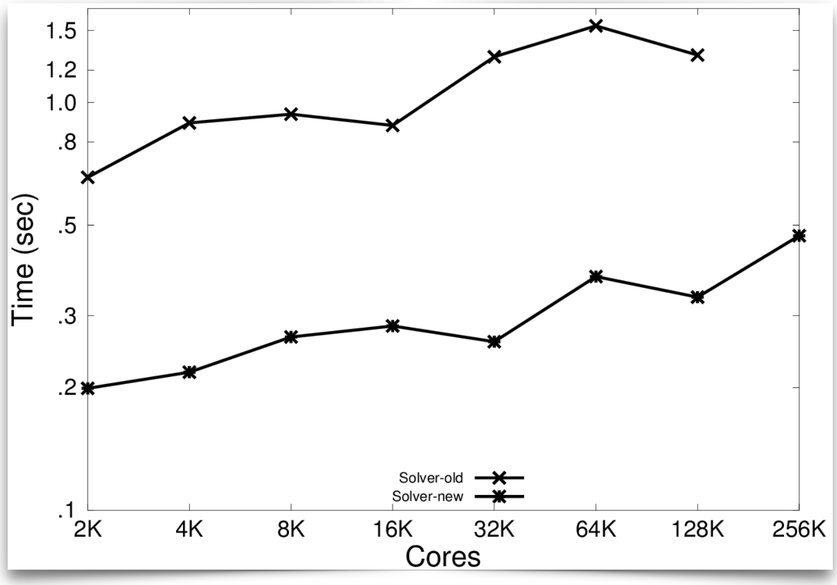


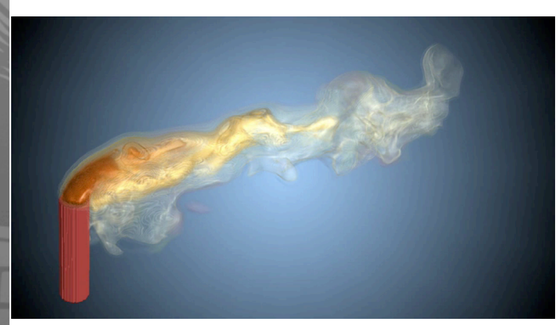
Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH

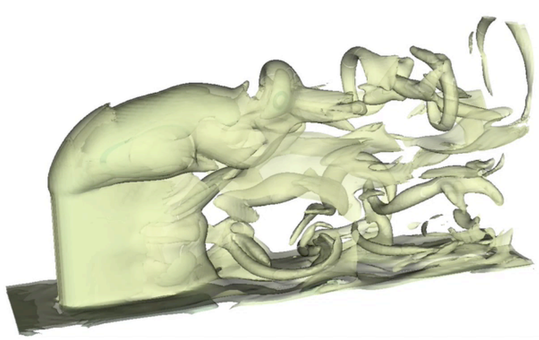
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

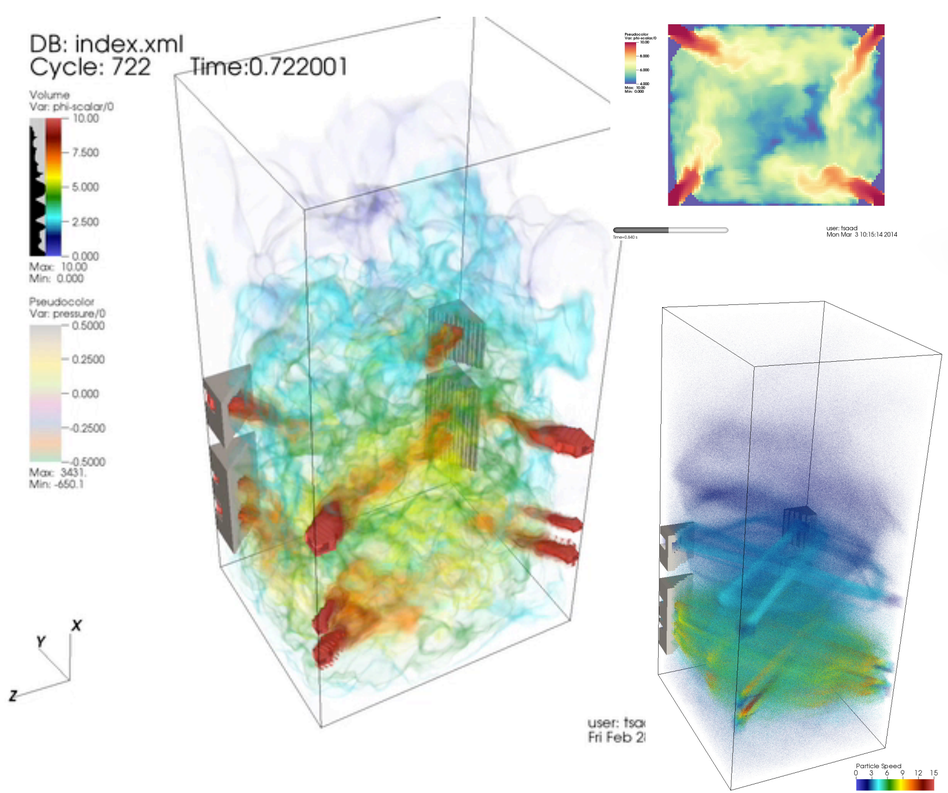
# Wasatch: flexible, efficient multiphysics solver

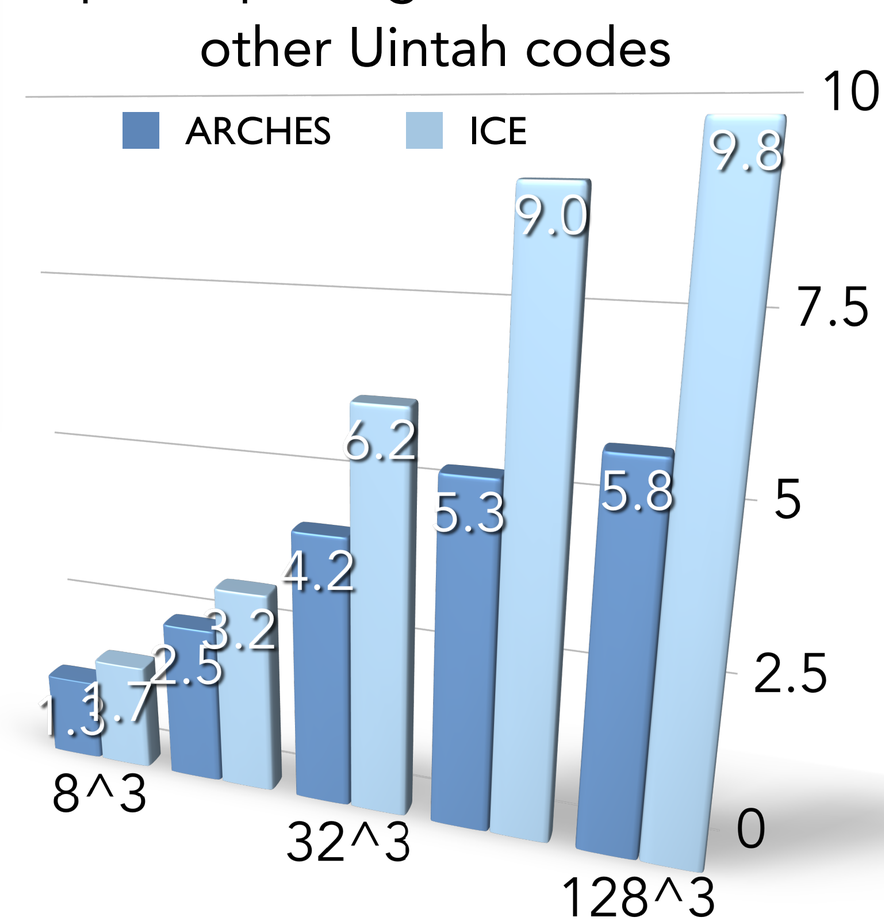






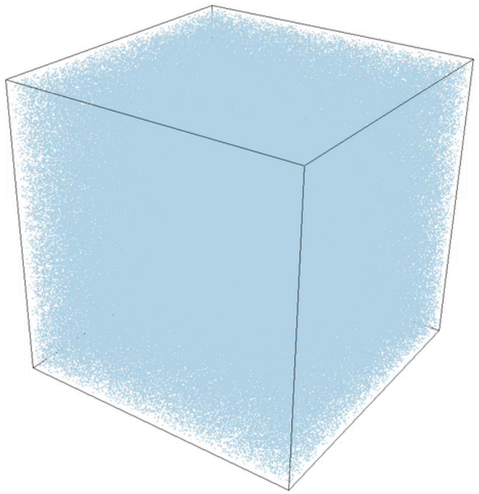


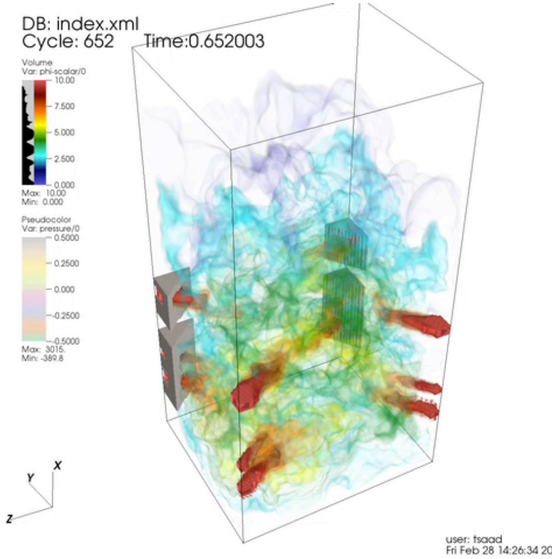




*Comparison to ICE and ARCHES, sister codes in Uintah, on a 3D Taylor-Green vortex problem. Run on a single processor.

# Conclusions

- Reduced-order modeling is very valuable.
  - *Can guide high-cost, high-fidelity models.*
  - *Allows parametric studies.*
  - *Tractable on entry-level rather than enterprise-level computing systems.*
- The current/coming hardware revolution will cause pain. How much pain depends on:
  - *How good your crystal ball is; OR*
  - *How well you can insulate your application from hardware changes.*
- Graph-based decomposition of the problem handles complexity nicely.
  - *Expose and exploit task & data parallelism.*
  - *Automatically handle data movement (inter-node and intra-node)*
  - *Overlap communication & computation*
- EDSL enables rapid development of portable, performant code.
  - *insulate developer from platform-specific implementation.*






National Science Foundation
WHERE DISCOVERIES BEGIN
PetaApps award 0904631
XPS award1337145

DE-NA0002375
DE-NA-000740