Case Studies in Using a DSL and Task Graphs for Portable Reacting Flow Simulations

James C. Sutherland
Associate Professor - Chemical Engineering

Tony Saad
Assistant Professor - Chemical Engineering

Acknowledgments

BABAK GOSHAYESHI
Research Staff

CHRISTOPHER EARL
Postdoctoral Researcher
Now at LLNL

ABHISHEK BAGUSETTY
DEVIN ROBISON
MICHAEL BROWN
M.S. Students

MIKE HANSEN
JOSH MCCONNELL
Ph.D. Students

National Science Foundation
WHERE DISCOVERIES BEGIN
XPS award1337145

DE-NA0002375
DE-NA-000740
DE-SC0008998

Nebo (E)DSL: "Matlab for PDEs on Supercomputers"

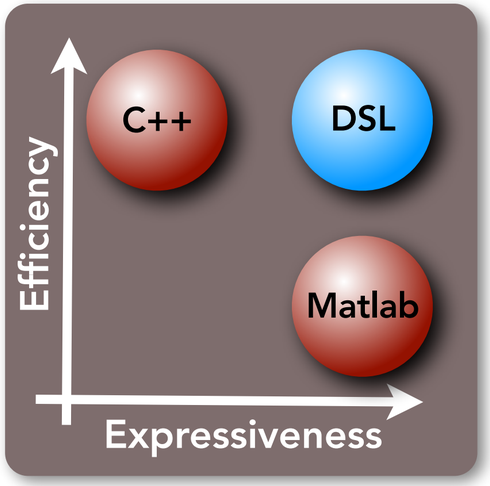
Field & stencil operations:

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

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

Can "chain" stencil operations where necessary.

- **Stencils**: >150 natively supported stencil operations (easily extensible)
- **cond**: "vectorized if"
- **Arbitrary composition** of operations
- **Masked assignment** (perform operations on a defined subset of points)
- **Portable**: same code works for CPU, multicore, GPU execution
- Embedded in C++ → "**plays well with others**"



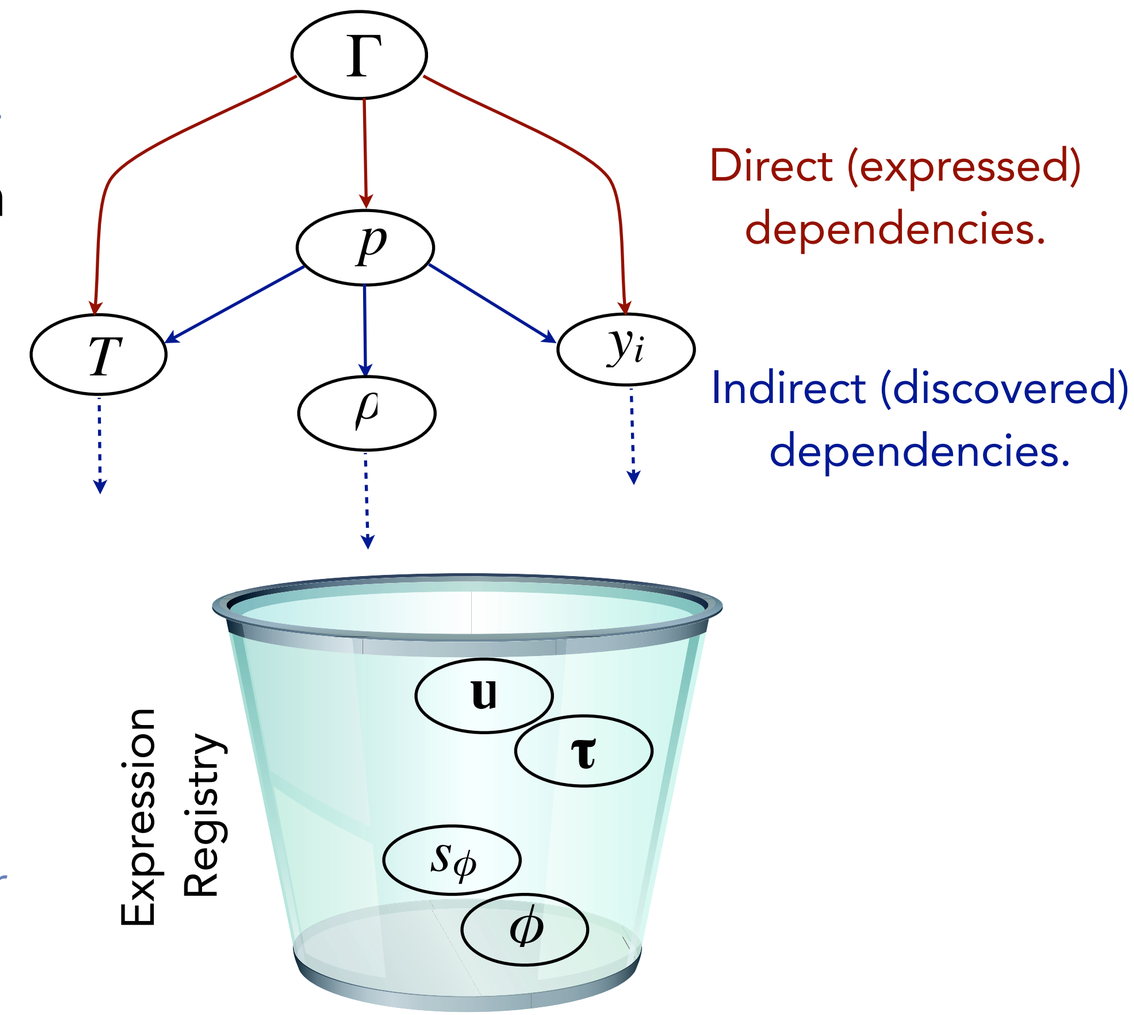
Auto-generate code for efficient execution on CPU, GPU, XeonPhi, etc. during compilation.

Earl, C., Might, M., Bagusetty, A., & Sutherland, J. C., Journal of Systems and Software (2016).

The Power of Task Graphs

- 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.*
  - *Robust scheduling algorithms are key.*

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

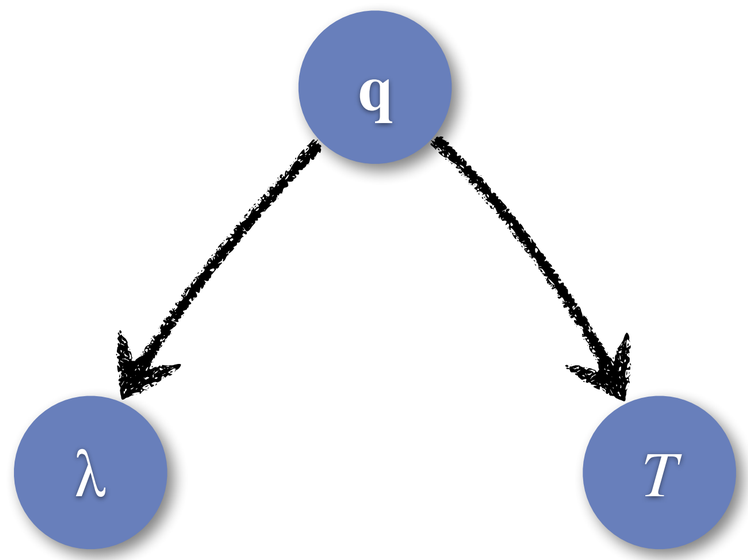


*Notz, Pawlowski, & Sutherland (2012). ACM Transactions on Mathematical Software, 39(1).

Changes in model form are naturally handled

Pure substance heat flux:

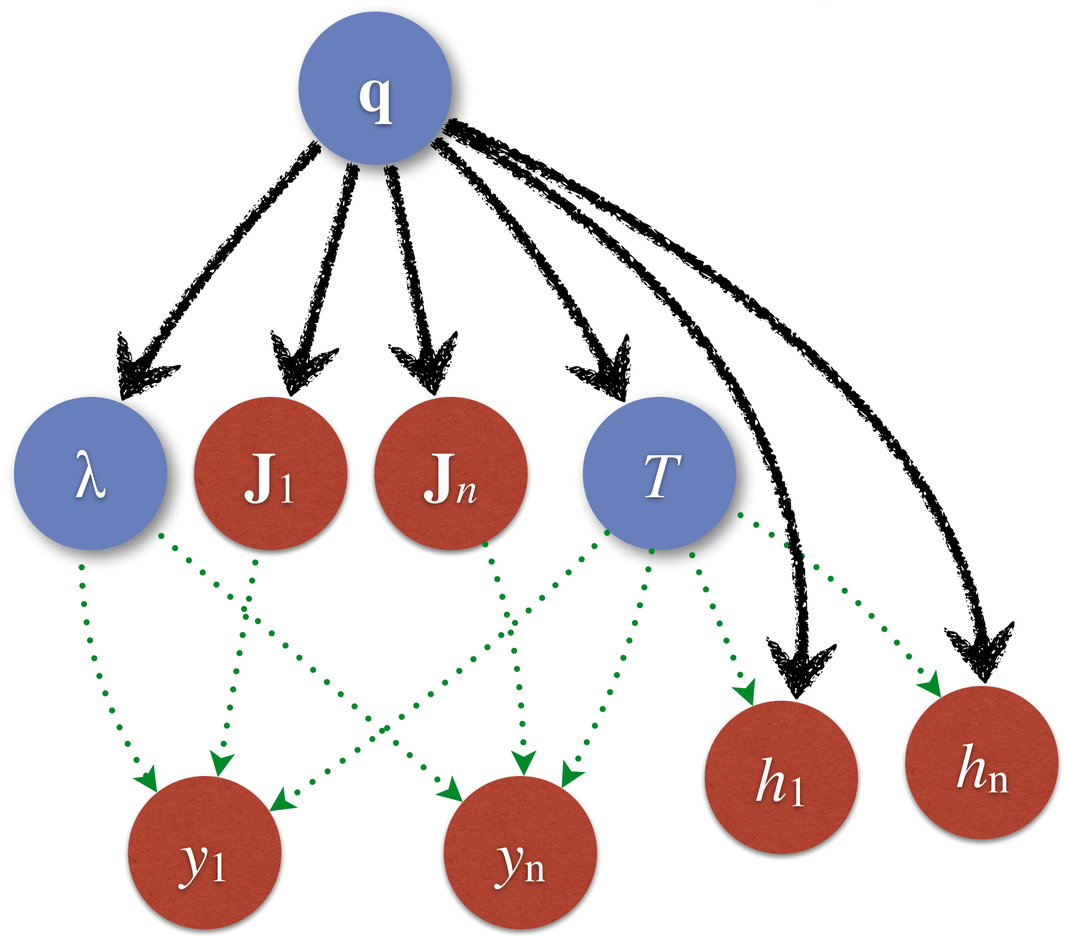
$$\mathbf{q} = -\lambda \nabla T$$



Changes in model form are naturally handled

Multi-species mixture heat flux:

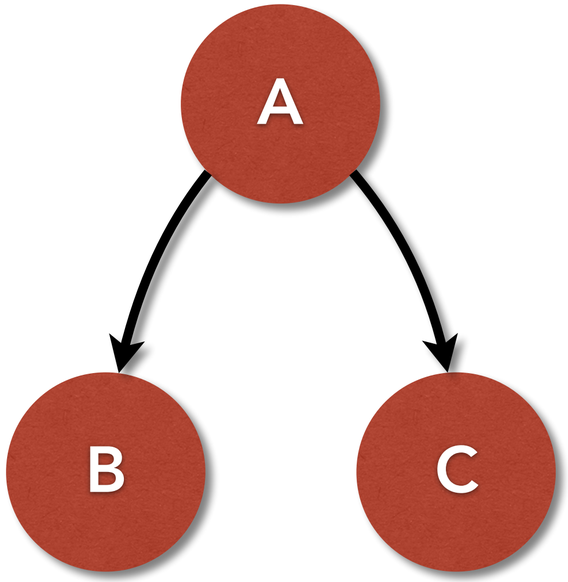
$$\mathbf{q} = -\lambda \nabla T + \sum_{i=1}^{n} h_i \mathbf{J}_i$$



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

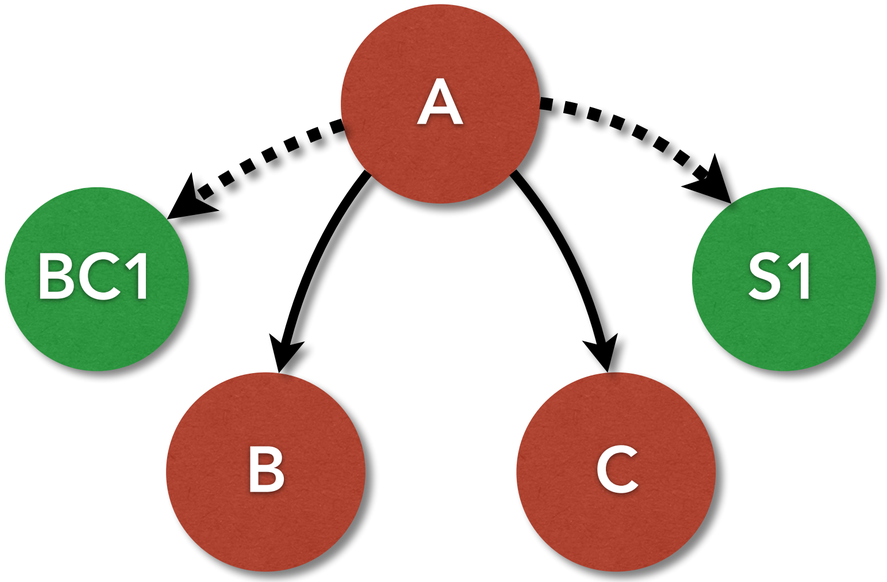
"Modifiers" — injecting new dependencies

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



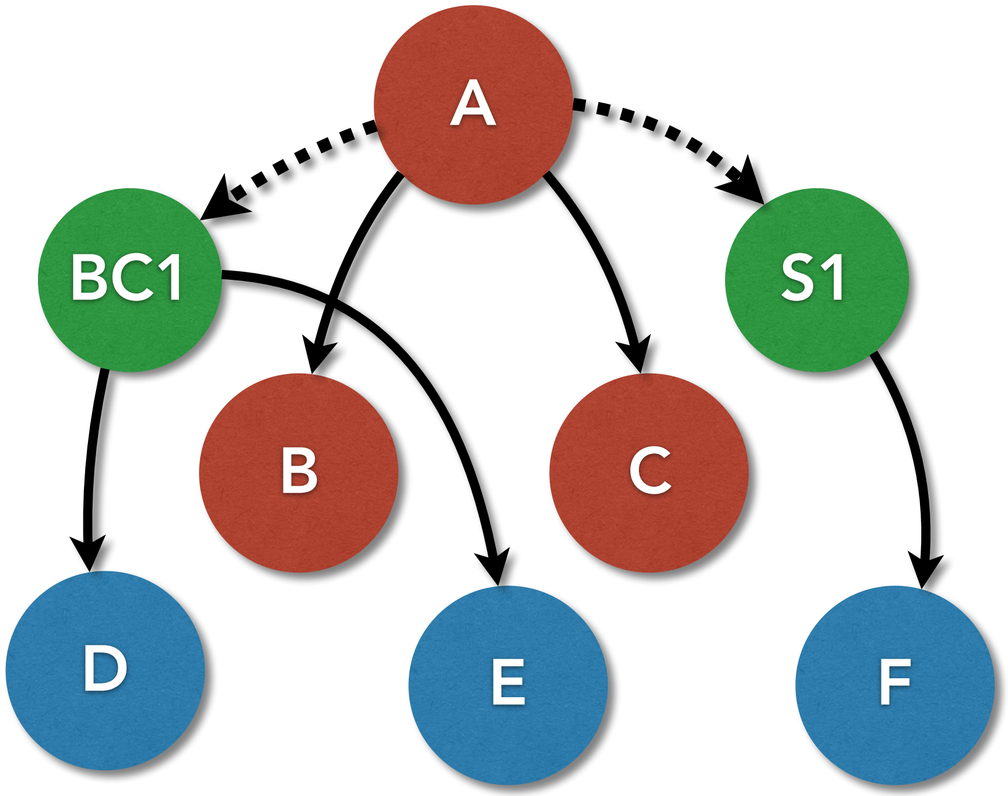
"Modifiers" — injecting new dependencies

- 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.



Example: PoKiTT

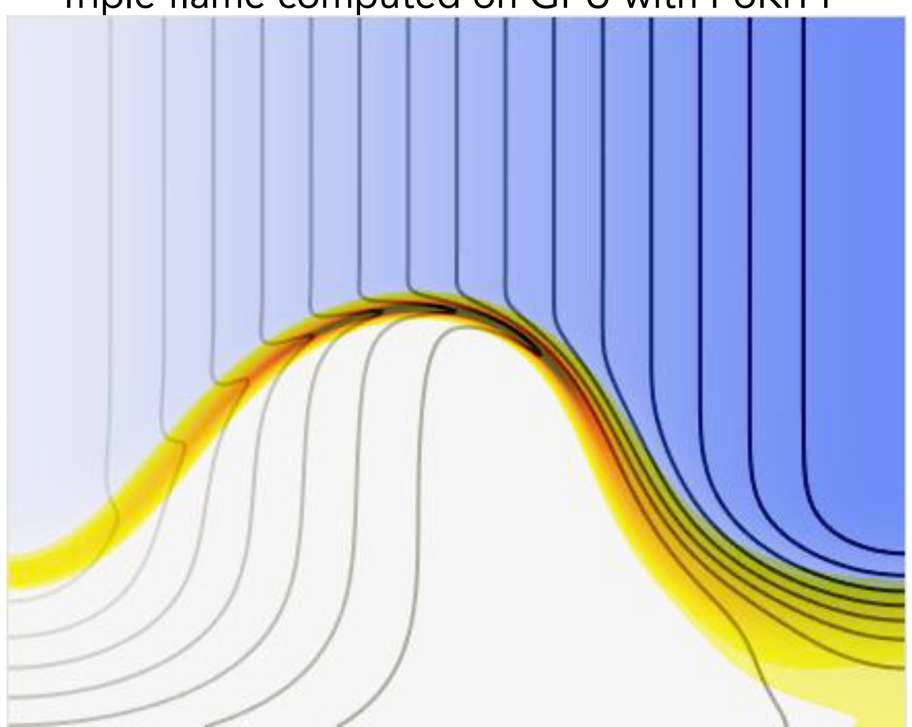
(**Po**rtable **Ki**netics, **T**hermodynamics & **T**ransport)

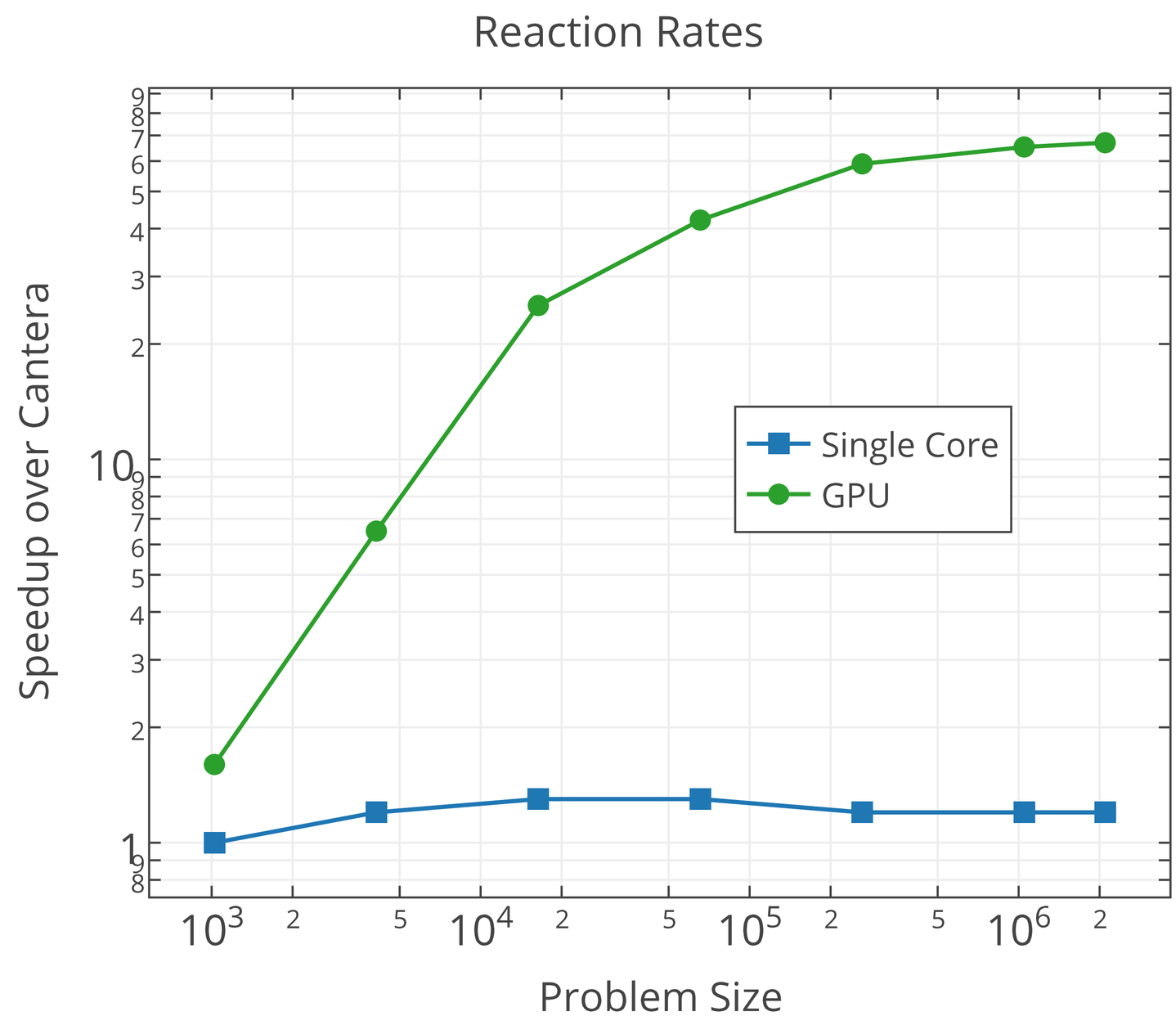
$$\rho \frac{\partial y_i}{\partial t} = -\nabla \cdot \mathbf{J}_i + s_i$$

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

- Detailed kinetics
- Mixture-averaged transport
- Detailed thermodynamics

Triple flame computed on GPU with PoKiTT





Yonkee & Sutherland, SIAM Journal on Scientific Computing (2016)

Example: PoKiTT

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

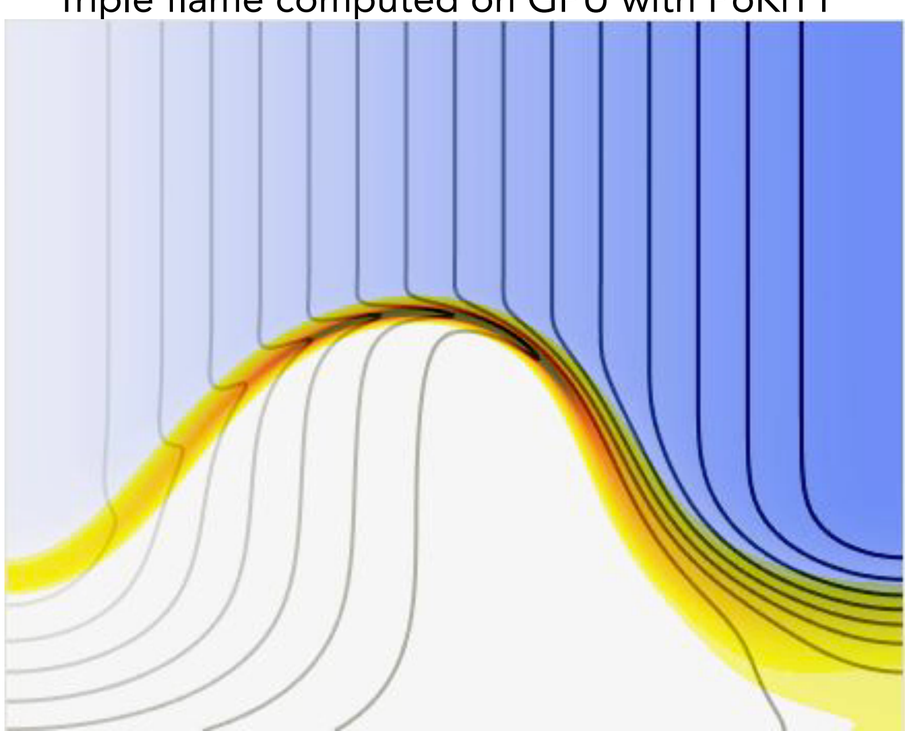
$$\rho \frac{\partial y_i}{\partial t} = -\nabla \cdot \mathbf{J}_i + s_i$$

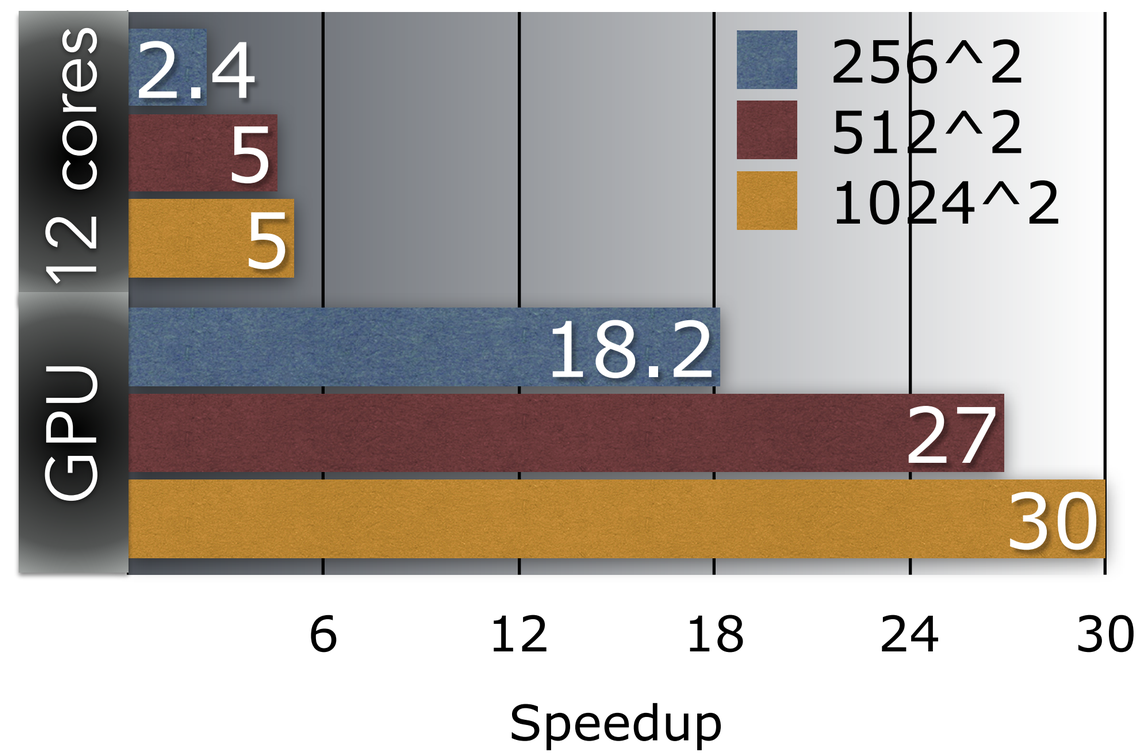
$$\rho \frac{\partial h}{\partial t} = -\nabla \cdot \mathbf{q}_i$$

- Detailed kinetics
- Mixture-averaged transport
- Detailed thermodynamics

- 32 PDEs
- 256^2 grid points
- 8 million timesteps
- 8 days on 1 GPU (~5 months on 1 CPU core)

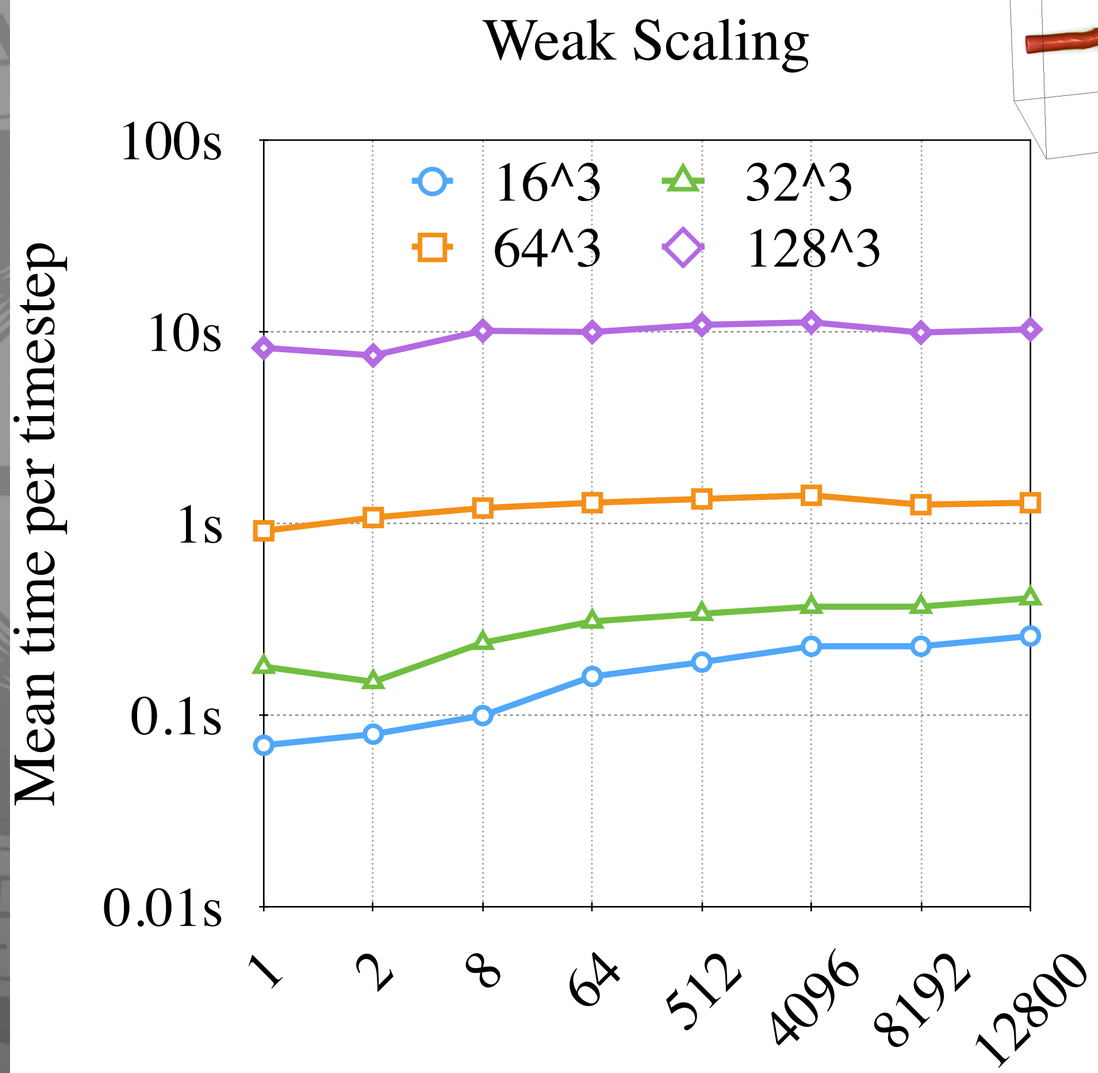
Triple flame computed on GPU with PoKiTT

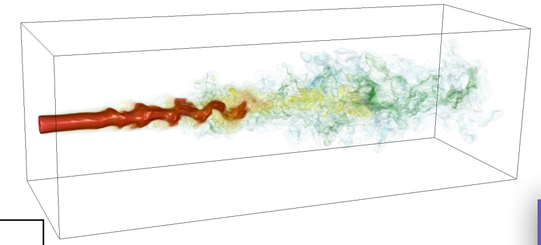


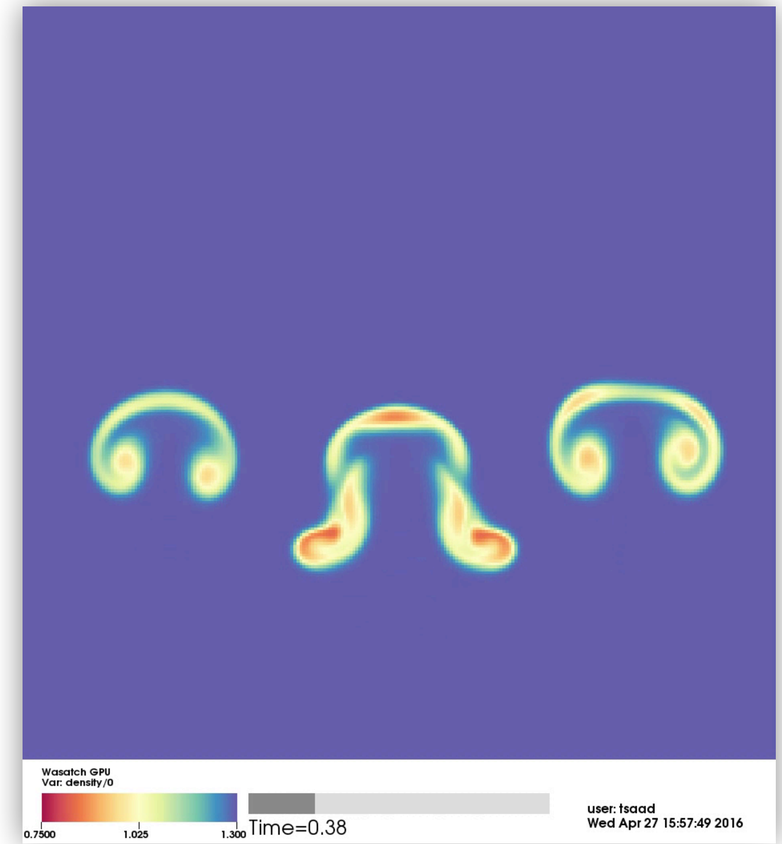


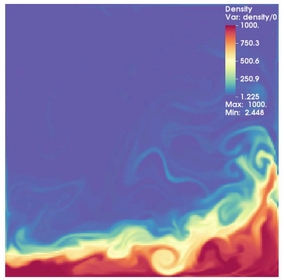
Yonkee & Sutherland, SIAM Journal on Scientific Computing (2016)

Titan: Hybrid Low Mach Algorithm





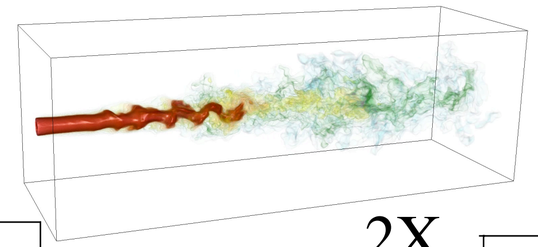


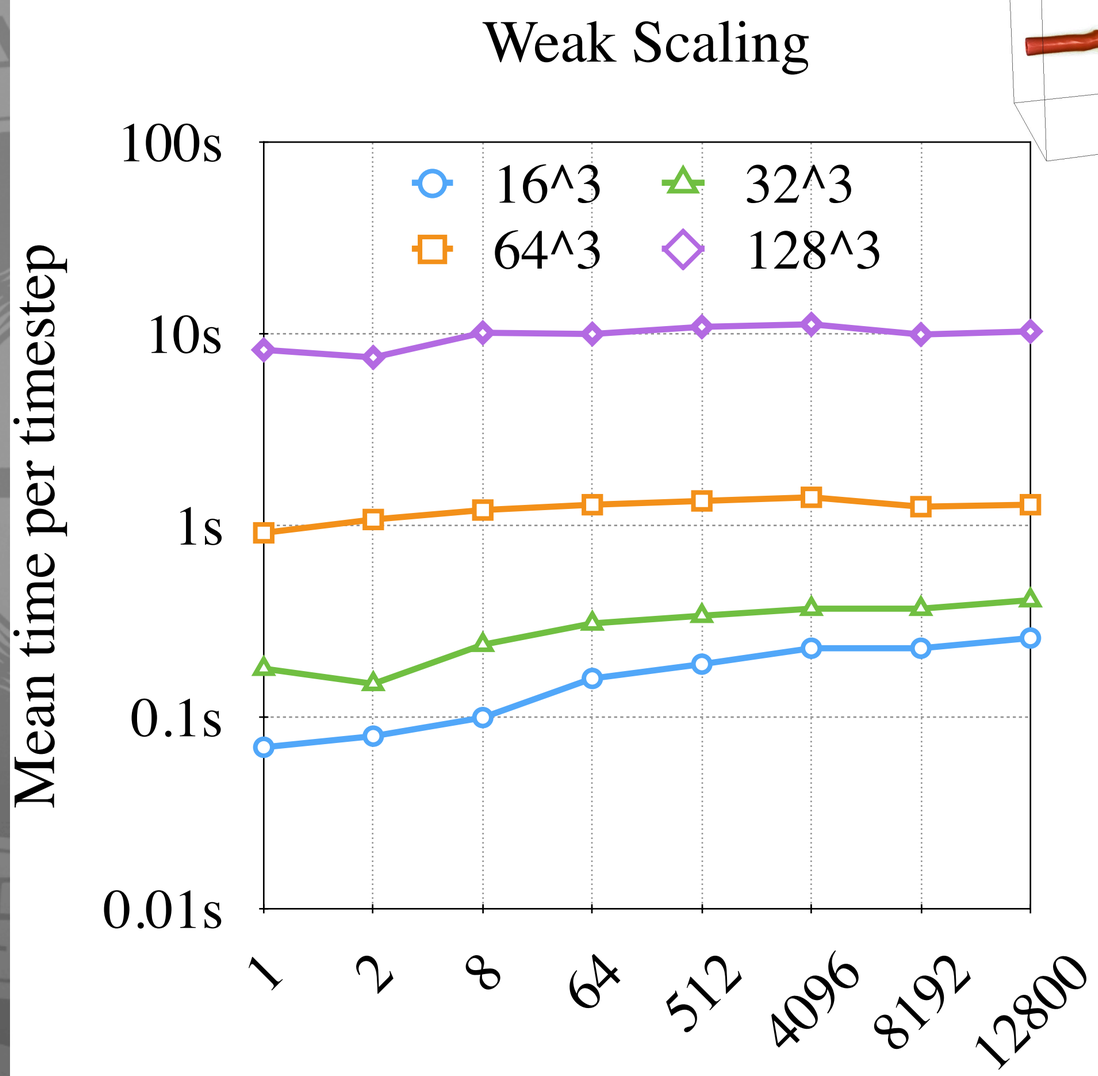


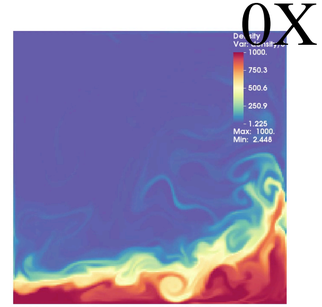
Everything on GPU except Poisson solve on CPU.

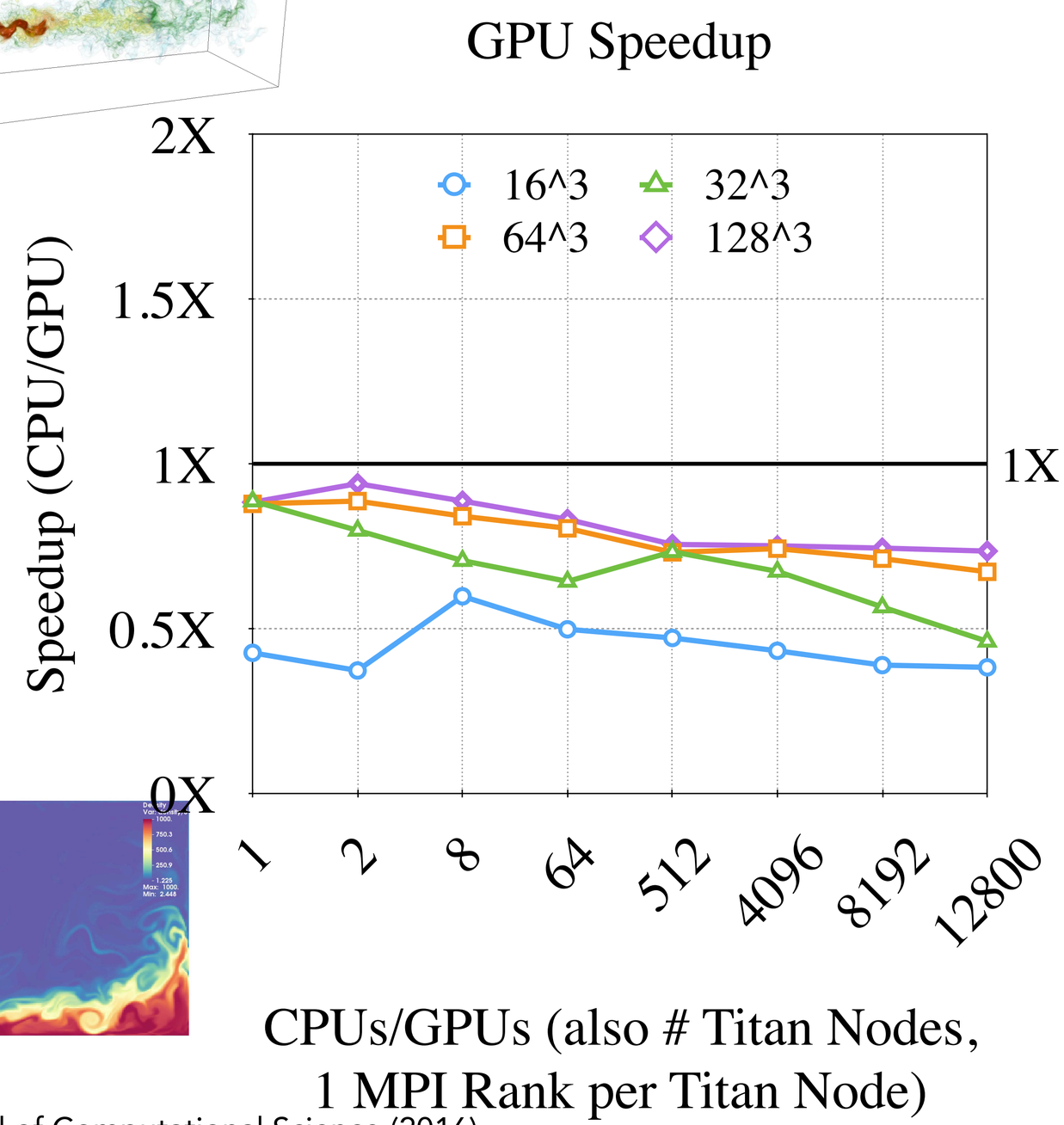
Saad, T., & Sutherland, J. C., Journal of Computational Science (2016)

Titan: Hybrid Low Mach Algorithm



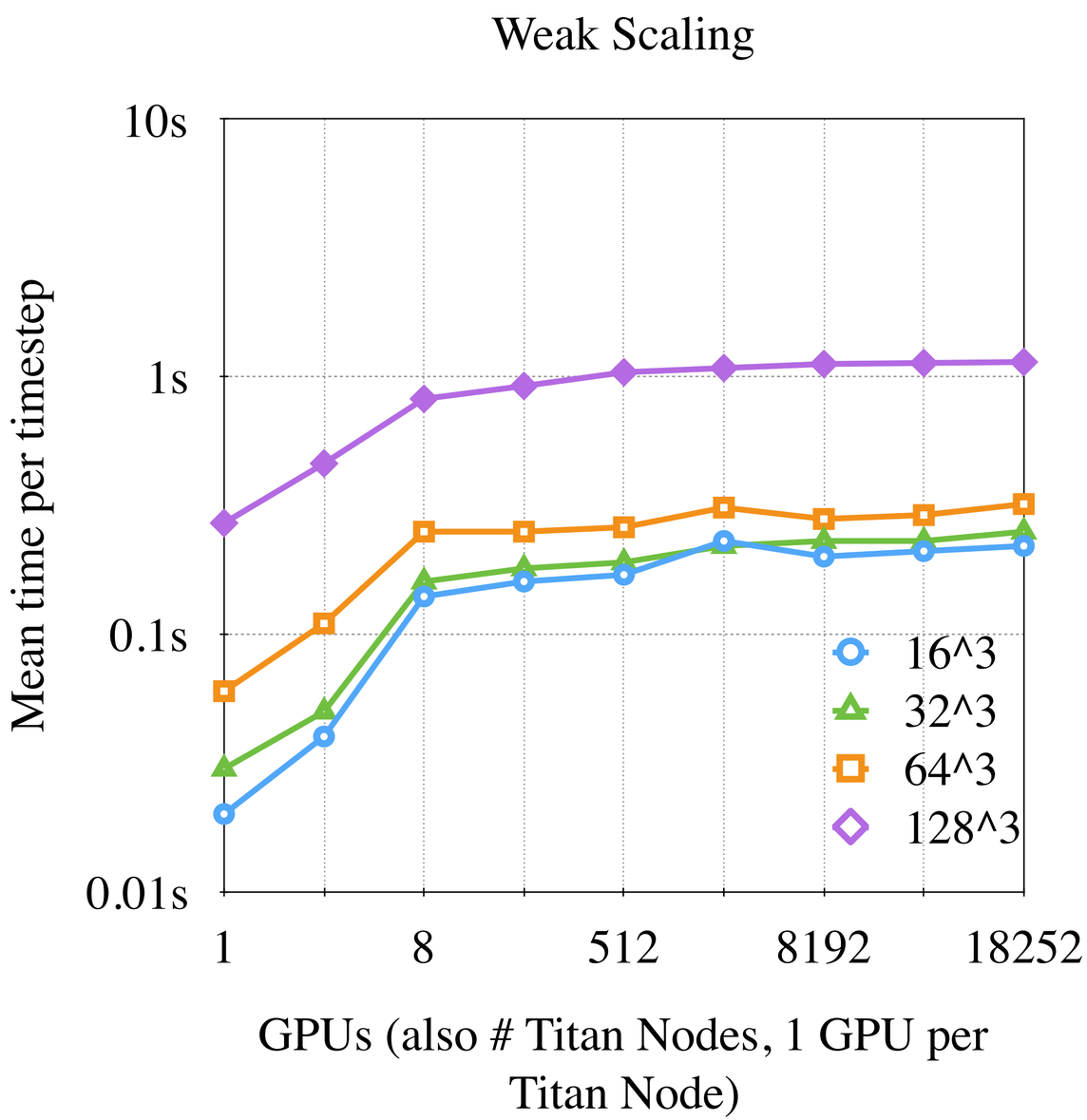


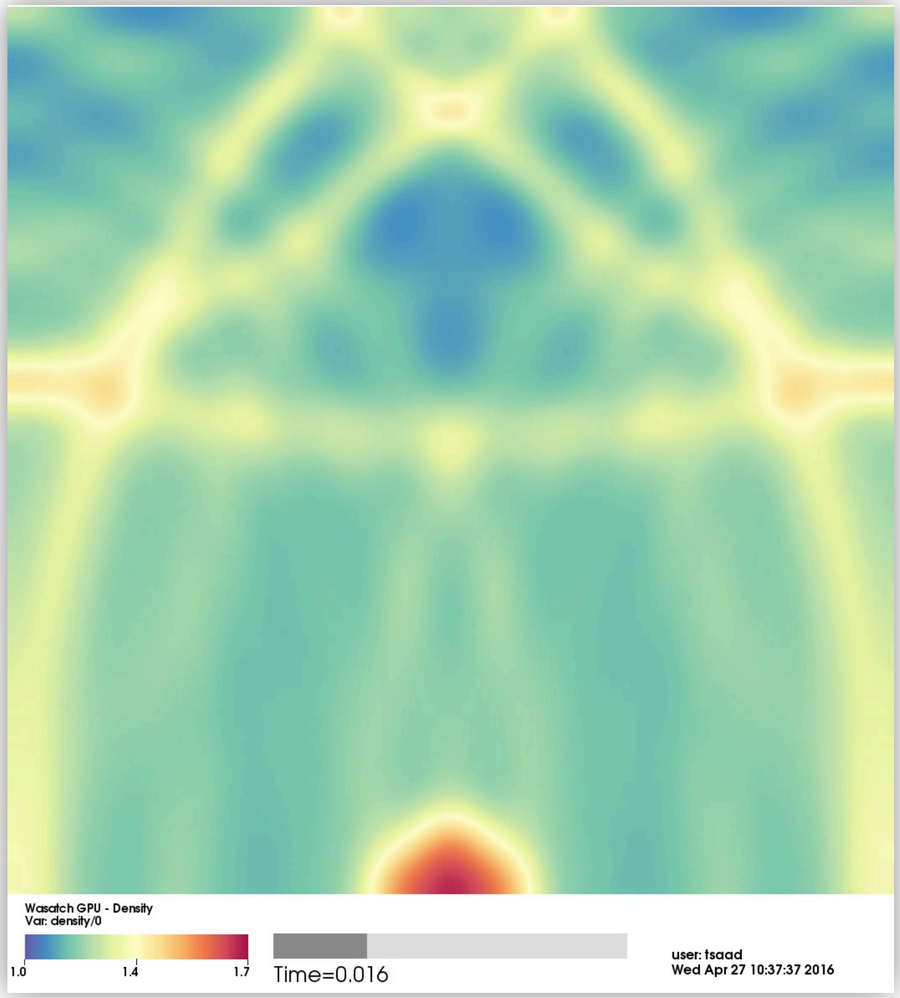




Saad, T., & Sutherland, J. C., Journal of Computational Science (2016)

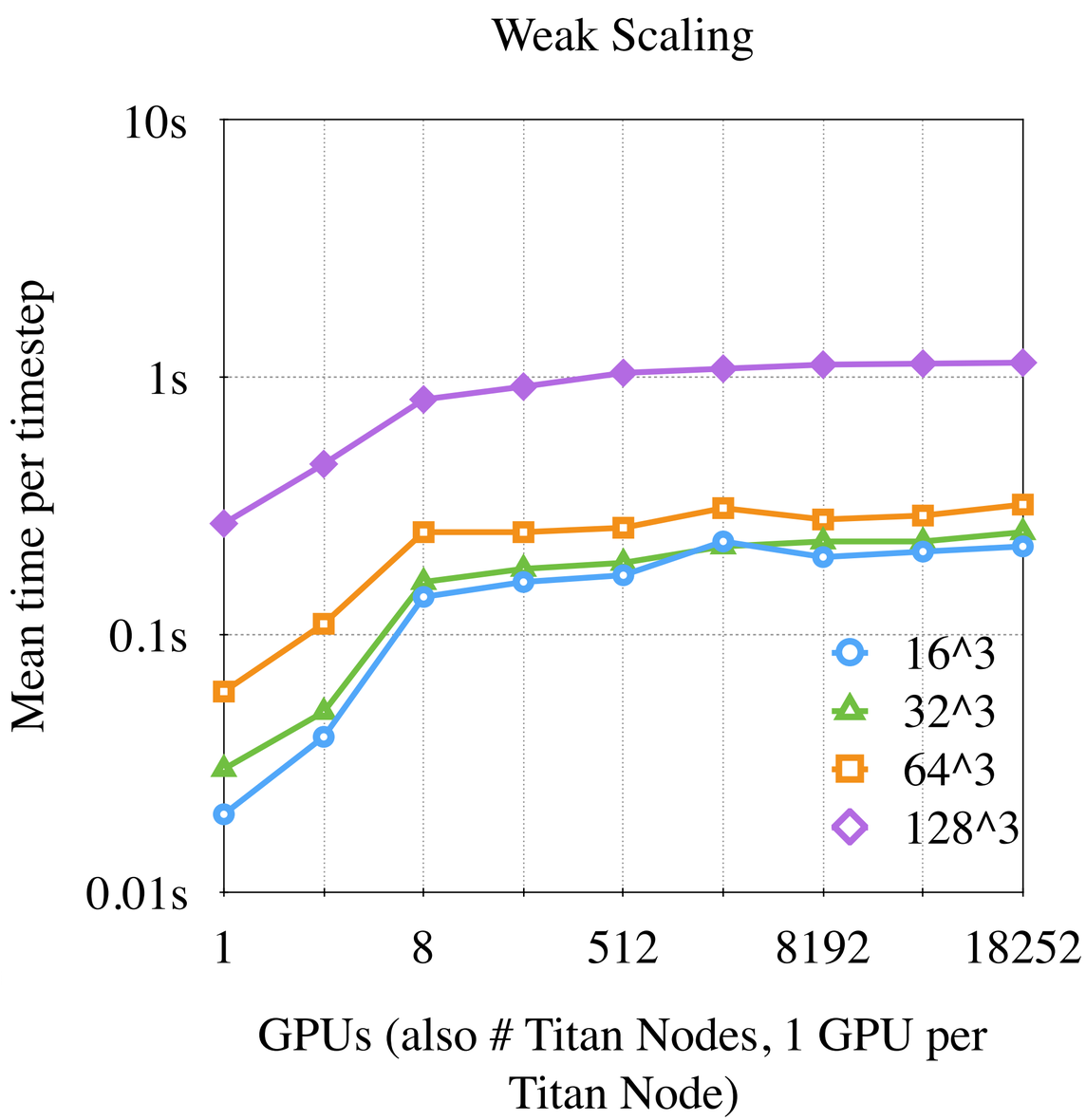
Titan: Compressible Algorithm

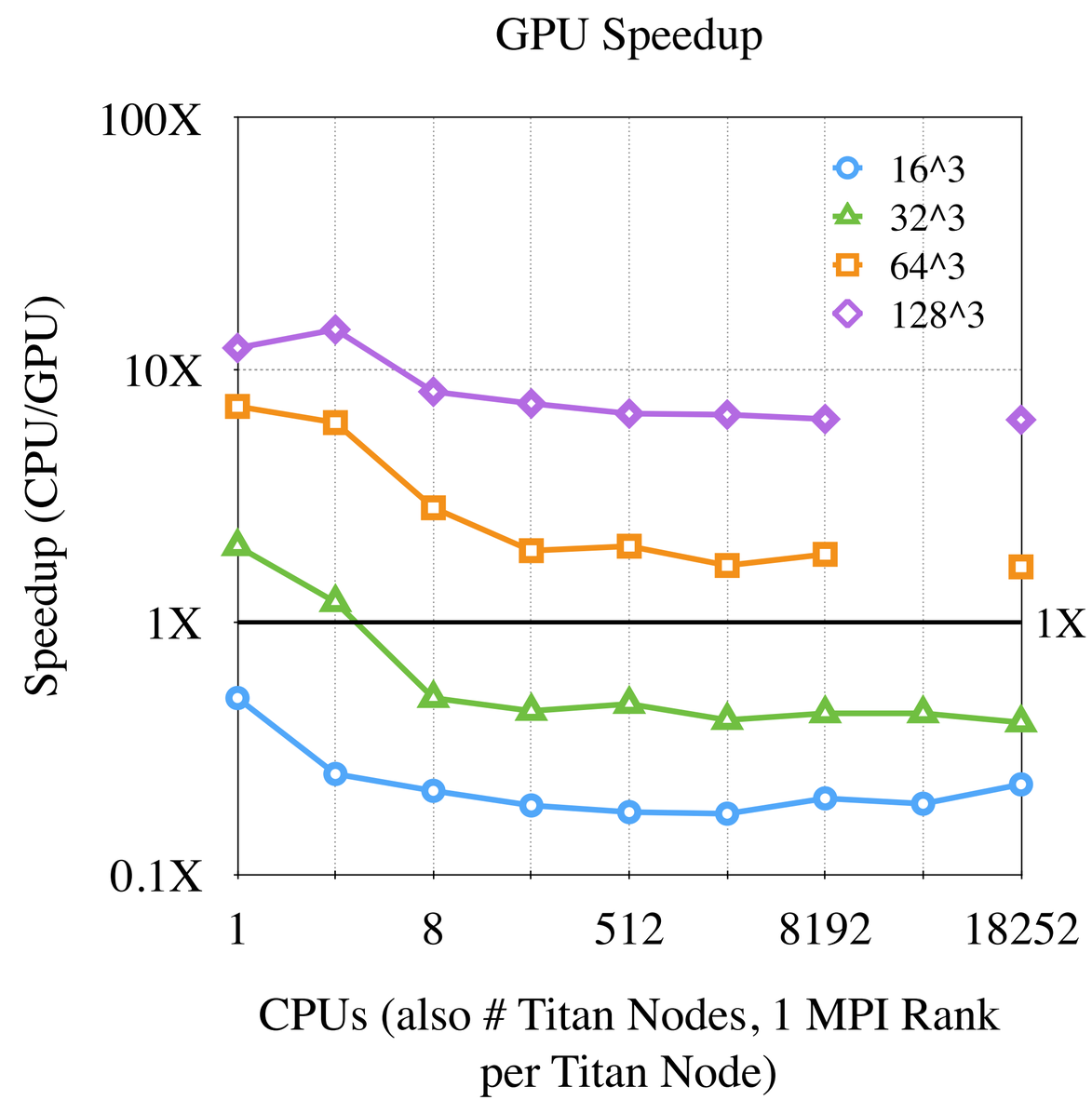




Saad, T., & Sutherland, J. C., Journal of Computational Science (2016)

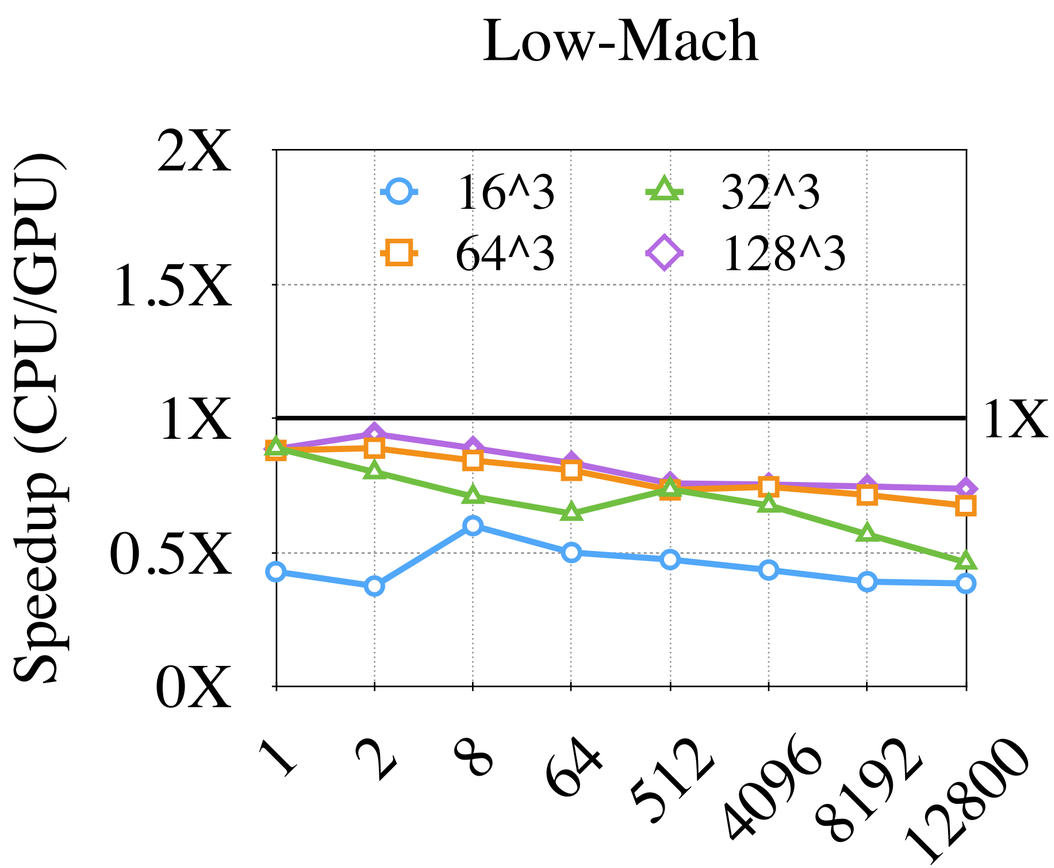
Titan: Compressible Algorithm

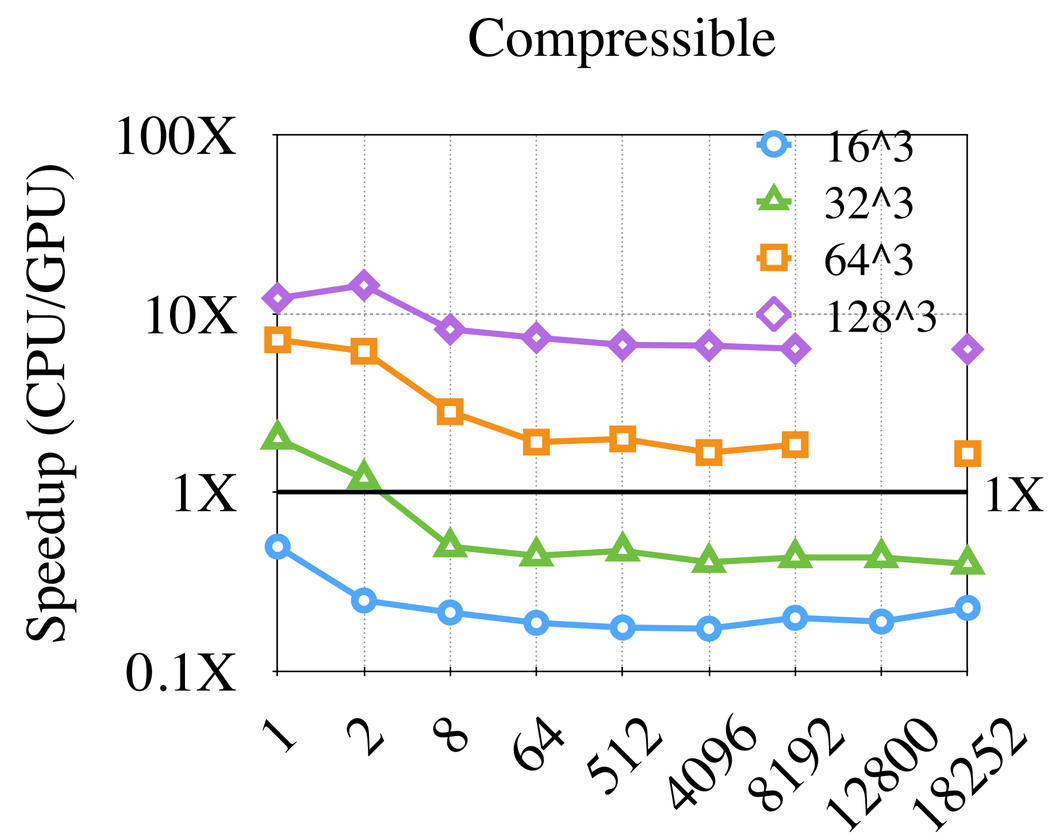




Saad, T., & Sutherland, J. C., Journal of Computational Science (2016)

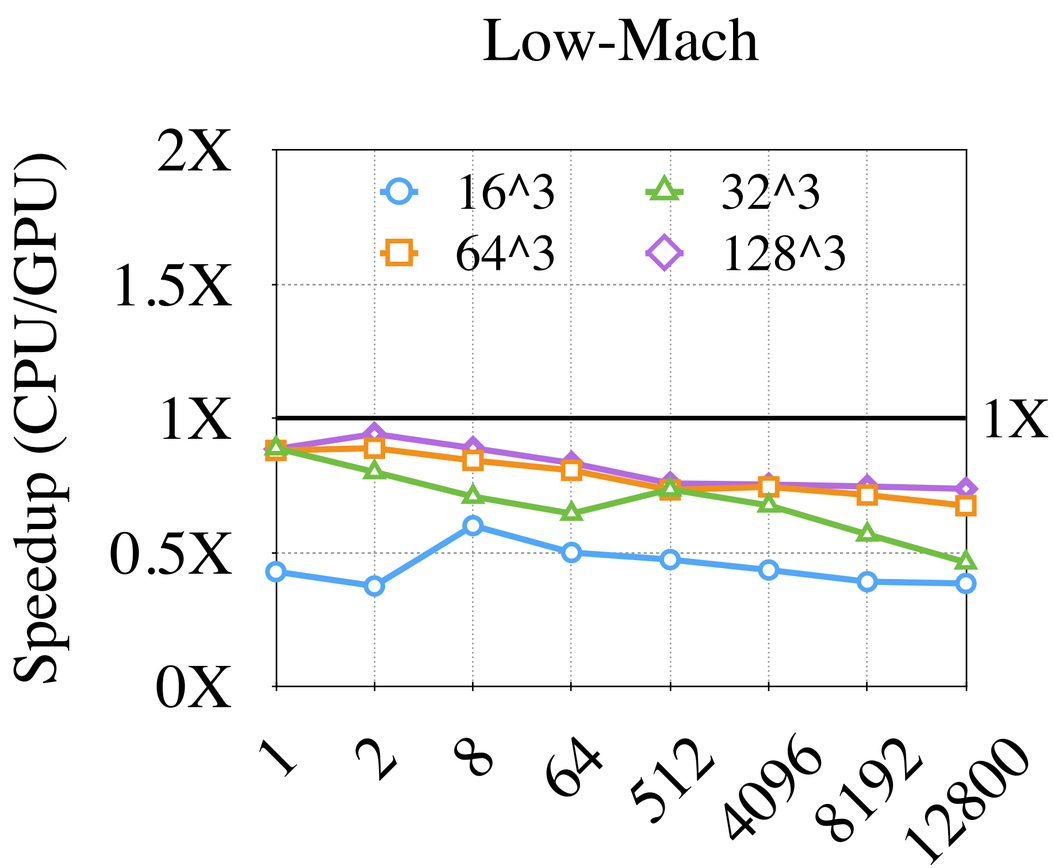
What next?

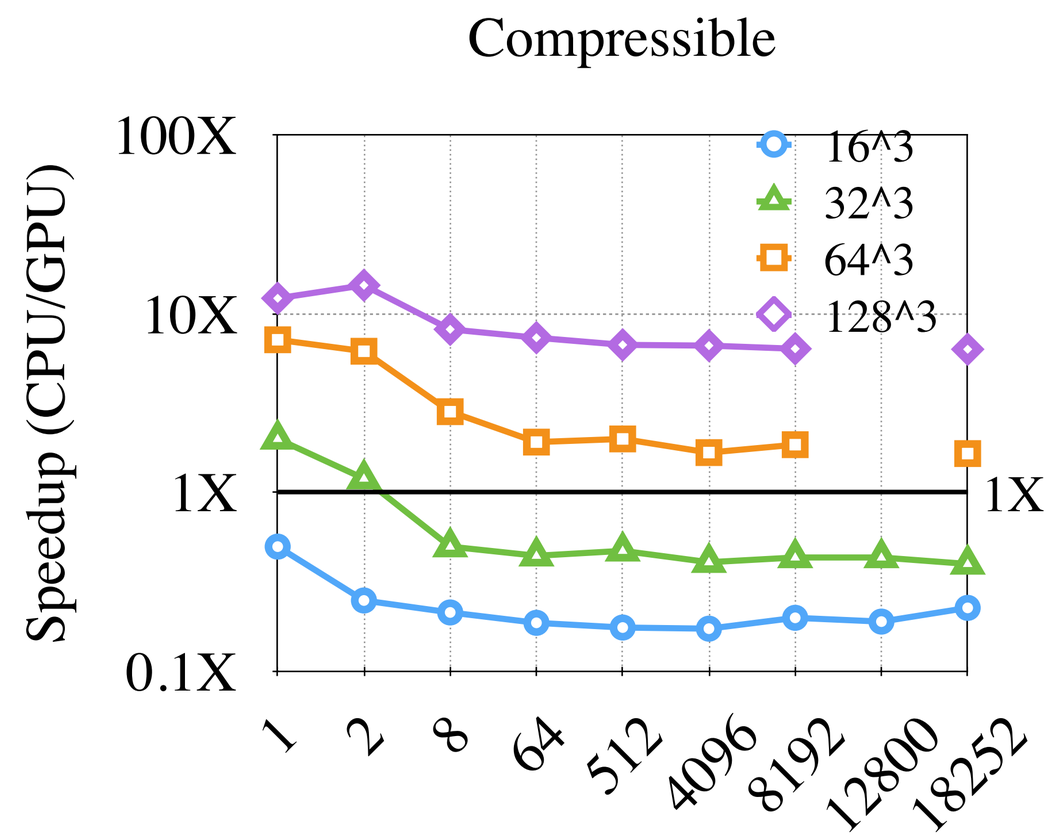




- Wait for linear solvers to get us to many-GPU systems?
  - *Even when these arrive, it puts a lot of demand on black-box linear solvers to achieve scalability & performance.*

What next?





- Wait for linear solvers to get us to many-GPU systems?
  - *Even when these arrive, it puts a lot of demand on black-box linear solvers to achieve scalability & performance.*
- Consider alternative algorithms?

Point-implicit algorithms:

- High arithmetic intensity
- Communication patterns are the same as explicit codes (ghost/halo-updates)
- Well-suited for reacting flow calculations.

$$\left[\mathrm{I} - \Delta\sigma \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}\right] \frac{\Delta \boldsymbol{u}}{\Delta \sigma} = \boldsymbol{h}(\boldsymbol{u})$$

Local residual

Local Jacobian matrix

Computational kernel

- Residual (right-hand side) evaluation
- Pointwise Jacobian evaluation
- Local linear solves
- Local eigenvalue decompositions

Matrix assembly must be efficient and extensible to complex, multiphysics problems

Example: Highly nonlinear, parameterized ODE systems

- Detailed chemical kinetics
  - *Analytical Jacobian in PoKiTT w/ Nebo for GPU*
  - *Dense matrix formed w/primitives and sparse transformation*
- Simple convective heat transfer
  - *Single-element Jacobian combined with sparse transform*
- Finite mixing time
  - *Scalar Jacobian matrix*

Right-hand side:

$$\underbrace{\boldsymbol{K}}_{\text{kinetics source terms}} + \underbrace{\boldsymbol{Q}}_{\text{convective heat transfer}} + \underbrace{\boldsymbol{T}}_{\text{mixing/flow}}$$

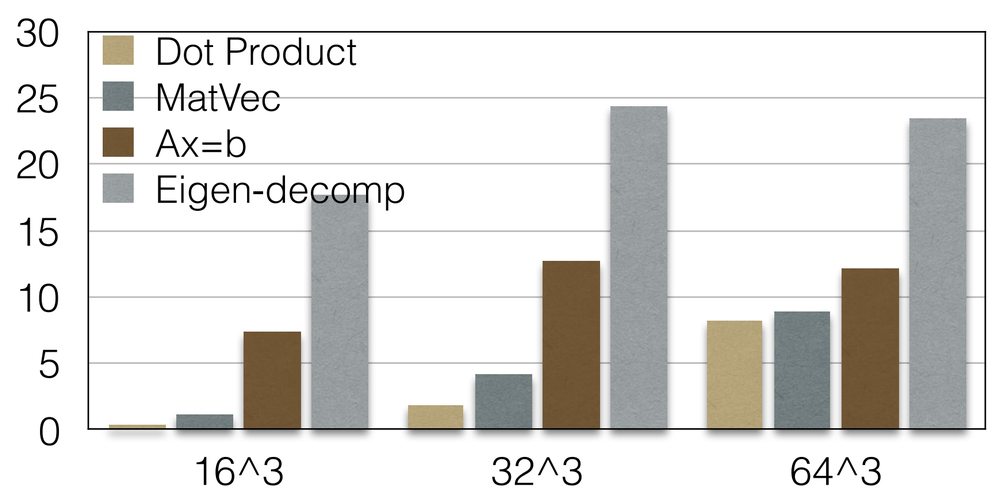
Jacobian:

$$\left[\underbrace{\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{V}}}_{\text{Full matrix (dense submat)}} + \underbrace{\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{V}}}_{\text{1-element (sparse)}}\right] \underbrace{\frac{\partial \boldsymbol{V}}{\partial \boldsymbol{U}}}_{\text{2N-elements (sparse)}} - \underbrace{\frac{1}{\tau}\mathrm{I}}_{\text{scalar matrix}}$$

C++ code:

```
( dKdV + dqdV ) * dVdU - invT
```

GPU Speedup - 16x16 Matrix



Conclusions

- Robust abstractions are needed to facilitate portable & performant applications on upcoming architectures.
  - *DAG-based software design allows flexibility needed for multiphysics codes on heterogeneous platforms.*
  - *(E)-DSLs can provide convenient, portable & performant abstractions for HPC applications*
- The Algorithm-Hardware collision:
  - *Scalable GPU linear solvers are needed for traditional algorithms to be viable on new architectures.*
  - *Alternative algorithms may be needed with higher arithmetic intensity*
    - higher-order
    - point-implicit?

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

National Science Foundation
WHERE DISCOVERIES BEGIN
XPS award1337145

DE-NA0002375
DE-NA-000740
DE-SC0008998