CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

Transparent Abstractions for Scientific Computing (TASC)

James C. Sutherland

Associate Professor, Chemical Engineering

Matthew Might

Assistant Professor, Computer Science

Tony Saad

Research Associate - ICSE

Christoper Earl

Post-Doctoral Research Associate - ICSE

ALSTOM

Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH

SCI
www.sci.utah.edu

Berkeley
UNIVERSITY OF CALIFORNIA

BYU
BRIGHAM YOUNG
UNIVERSITY

NNSA
National Nuclear Security Administration

Orientation

- Uintah controls & allocates resources
  - *Coarse data & task parallel*

Application Code

- Implements physics kernels.
- Provide interaction between user input, framework & creation of physics kernels.

Uintah-X

- Coarse data parallelism.
- Coarse task parallelism.
- Fault-tolerant runtime system.
- Scalable IO.

CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

Orientation

- Uintah controls & allocates resources
  - *Coarse data & task parallel*
- When resources exceed available work, Uintah pushes resources "downward" to finer levels of parallelism
  - *Fine-grained task graphs within each Uintah task*
  - *Fine-grained data parallel (multicore/GPU)*

Application Code

- Implements physics kernels.
- Provide interaction between user input, framework & creation of physics kernels.

ExprLib

- Fine-grained task parallelism.
- Interface to various structured frameworks.
- Dynamic assembly of problem structure.

SpatialOps

- **Nebo** EDSL for computational kernels.
- Fine-grained data parallelism.

Uintah-X

- Coarse data parallelism.
- Coarse task parallelism.
- Fault-tolerant runtime system.
- Scalable IO.

Orientation

- Uintah controls & allocates resources
  - *Coarse data & task parallel*
- When resources exceed available work, Uintah pushes resources "downward" to finer levels of parallelism
  - *Fine-grained task graphs within each Uintah task*
  - *Fine-grained data parallel (multicore/GPU)*

Wasatch

ExprLib

SpatialOps

Uintah-X

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

Application Code

- Implements physics kernels.
- Provide interaction between user input, framework & creation of physics kernels.

ExprLib

- Fine-grained task parallelism.
- Interface to various structured frameworks.
- Dynamic assembly of problem structure.

SpatialOps

- **Nebo** EDSL for computational kernels.
- Fine-grained data parallelism.

Uintah-X

- Coarse data parallelism.
- Coarse task parallelism.
- Fault-tolerant runtime system.
- Scalable IO.

Orientation

- Uintah controls & allocates resources
  - *Coarse data & task parallel*
- When resources exceed available work, Uintah pushes resources "downward" to finer levels of parallelism
  - *Fine-grained task graphs within each Uintah task*
  - *Fine-grained data parallel (multicore/GPU)*

Wasatch

Arches

ExprLib

SpatialOps

Uintah-X

Uintah-X

CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

Application Code

- Implements physics kernels.
- Provide interaction between user input, framework & creation of physics kernels.

ExprLib

- Fine-grained task parallelism.
- Interface to various structured frameworks.
- Dynamic assembly of problem structure.

SpatialOps

- **Nebo** EDSL for computational kernels.
- Fine-grained data parallelism.

Uintah-X

- Coarse data parallelism.
- Coarse task parallelism.
- Fault-tolerant runtime system.
- Scalable IO.

Orientation

- Uintah controls & allocates resources
  - *Coarse data & task parallel*
- When resources exceed available work, Uintah pushes resources "downward" to finer levels of parallelism
  - *Fine-grained task graphs within each Uintah task*
  - *Fine-grained data parallel (multicore/GPU)*

Wasatch	Arches	Arches 2.0?
ExprLib		
SpatialOps		SpatialOps
Uintah-X	Uintah-X	Uintah-X

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

Application Code

- Implements physics kernels.
- Provide interaction between user input, framework & creation of physics kernels.

ExprLib

- Fine-grained task parallelism.
- Interface to various structured frameworks.
- Dynamic assembly of problem structure.

SpatialOps

- **Nebo** EDSL for computational kernels.
- Fine-grained data parallelism.

Uintah-X

- Coarse data parallelism.
- Coarse task parallelism.
- Fault-tolerant runtime system.
- Scalable IO.

Orientation

- Uintah controls & allocates resources
  - *Coarse data & task parallel*
- When resources exceed available work, Uintah pushes resources "downward" to finer levels of parallelism
  - *Fine-grained task graphs within each Uintah task*
  - *Fine-grained data parallel (multicore/GPU)*

Wasatch	Arches	Arches 2.0?	ODT
ExprLib			ExprLib
SpatialOps		SpatialOps	SpatialOps
Uintah-X	Uintah-X	Uintah-X	

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

Application Code

- Implements physics kernels.
- Provide interaction between user input, framework & creation of physics kernels.

ExprLib

- Fine-grained task parallelism.
- Interface to various structured frameworks.
- Dynamic assembly of problem structure.

SpatialOps

- **Nebo** EDSL for computational kernels.
- Fine-grained data parallelism.

Uintah-X

- Coarse data parallelism.
- Coarse task parallelism.
- Fault-tolerant runtime system.
- Scalable IO.

CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

SpatialOps/Nebo: Embedded Domain Specific Language for HPC

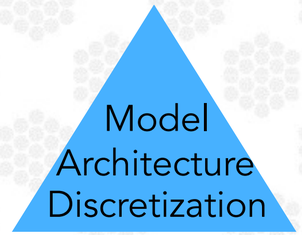
SpatialOps: the software project

Nebo: the EDSL subset of SpatialOps

Why an EDSL?

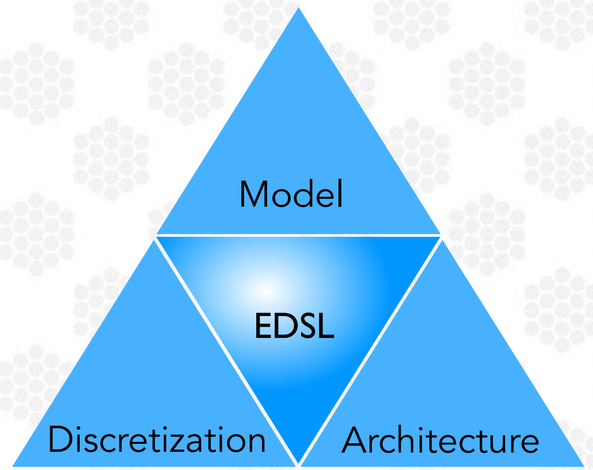
- Expressive syntax (matlab-style array operations)
  - *Programmer expresses intent (problem structure) - not implementation.*

CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER



Why an EDSL?

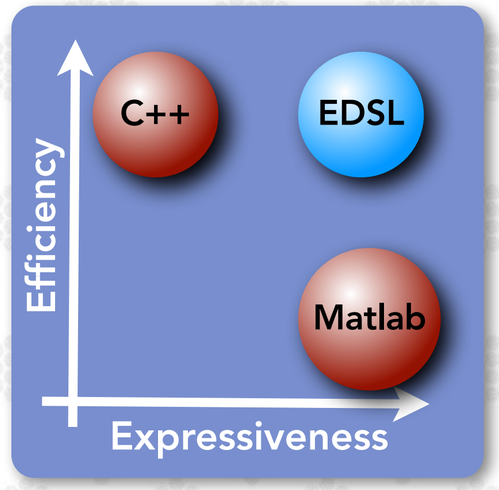
- Expressive syntax (matlab-style array operations)
  - *Programmer expresses intent (problem structure) - not implementation.*

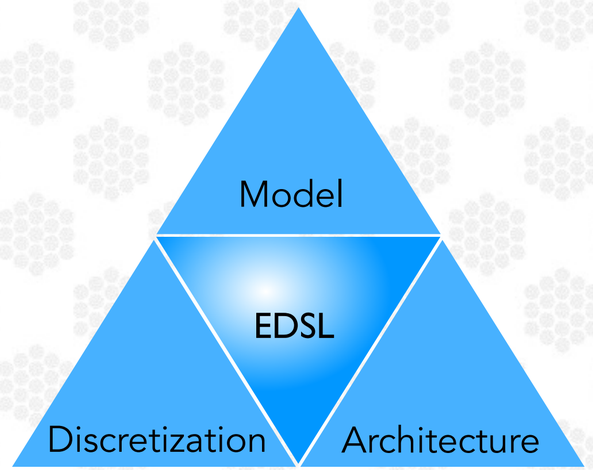


CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

Why an EDSL?

- Expressive syntax (matlab-style array operations)
  - *Programmer expresses intent (problem structure) - not implementation.*
- High performance
  - *Should match hand-tuned code in performance.*

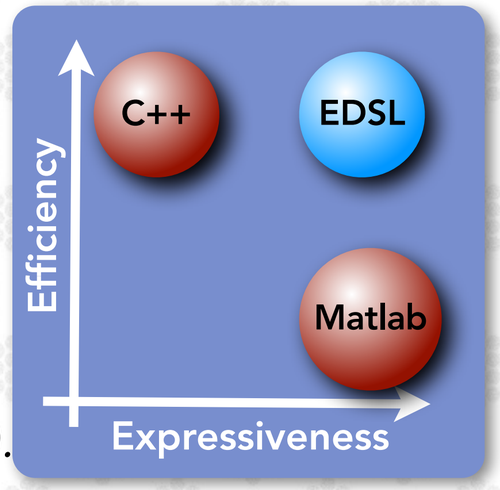




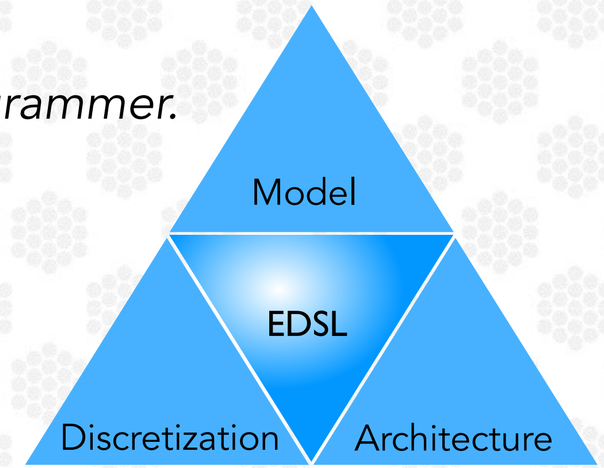
CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

Why an EDSL?

- Expressive syntax (matlab-style array operations)
  - *Programmer expresses intent (problem structure) - not implementation.*
- High performance
  - *Should match hand-tuned code in performance.*
- Extensible
  - *Insulate programmer from architecture changes (e.g. multicore → GPU → …).*
  - *EDSL "back-end" compiles into code for target architecture.*



- "Plays well with others"
  - *Allow programmer to write in C++ and inter-operate with EDSL.*
  - *Not an "all-or-none" approach.*
  - *Allows concurrent development of EDSL and application codes.*
  - *Adapt to appropriate data structures without impacting application programmer.*



Nebo Field Expressions

$$\vec{c} = \vec{a} + sin(\vec{b})$$

Manual C++

Thread 1

```
Field::const_iterator ia1 = a1.begin();
Field::const_iterator ib1 = b1.begin();
for(Field::iterator ic1 = c1.begin();
    ic1 != c1.end();
    ++ic1, ++ia1, ++ib1) {
  *ic1 = *ia1 + sin(*ib1);
};
```

⋮

Thread *n*

```
...
Field::const_iterator ian = an.begin();
Field::const_iterator ibn = bn.begin();
for(Field::iterator icn = cn.begin();
    icn != cn.end();
    ++icn, ++ian, ++ibn) {
  *icn = *ian + sin(*ibn);
};
```

Nebo

```
c <<= a + sin(b);
```

- Data parallel handled internally.
- Thread deployment (resizable threadpool).
- GPU deployment.
- Compile-time guarantee of field compatibility for given operations.

Chained Stencil Operations

$$\begin{aligned}\phi &= -\nabla \cdot \mathbf{q} \\ &= \nabla \cdot (-\lambda \nabla T)\end{aligned}$$

```
phi <<= divX( interpx(lambda) * gradX(temperature) )
      + divY( interpy(lambda) * gradY(temperature)
      + divZ( interpz(lambda) * gradZ(temperature);
```

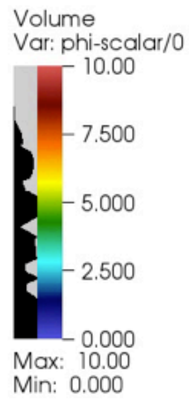
```
// field type inference:
typedef FaceTypes<FieldT>::XFace  XFluxT;
typedef FaceTypes<FieldT>::YFace  YFluxT;
typedef FaceTypes<FieldT>::ZFace  ZFluxT;
```

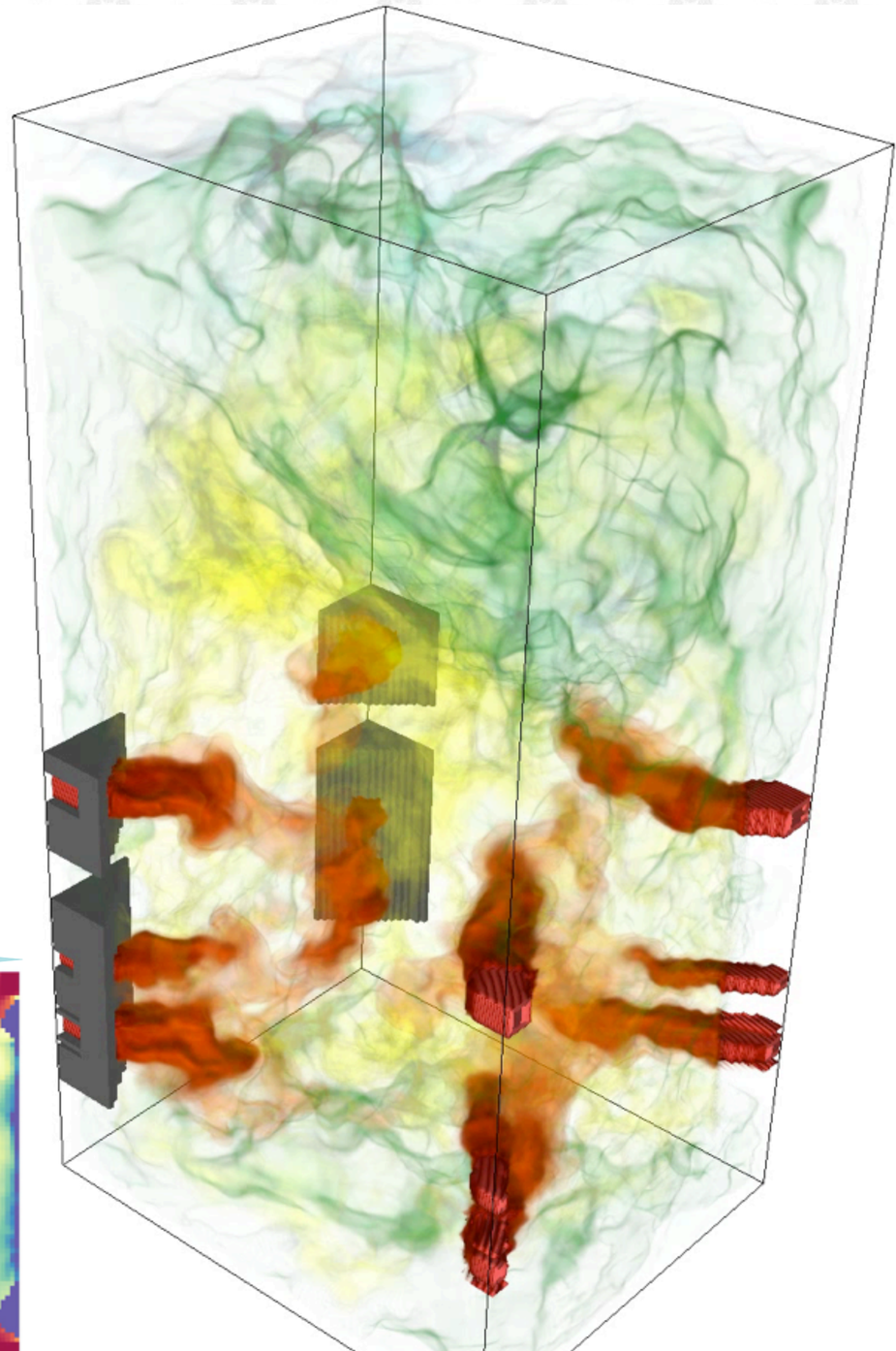
```
// operator type inference:
typedef OpTypes<FieldT>::DivX  DivX;
typedef OpTypes<FieldT>::DivY  DivY;
typedef OpTypes<FieldT>::DivZ  DivZ;
```

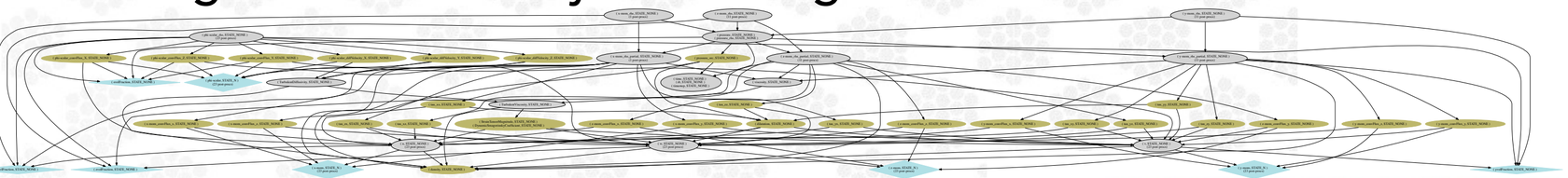
- One inlined grid loop, no temporaries.
- Better performance & scalability than without chaining.
- Compile-time consistency checking (field-operator and field-field compatibility).
- Runtime consistency checks for ghost cell validity.

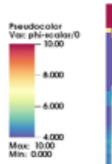
Performance on CFD problems

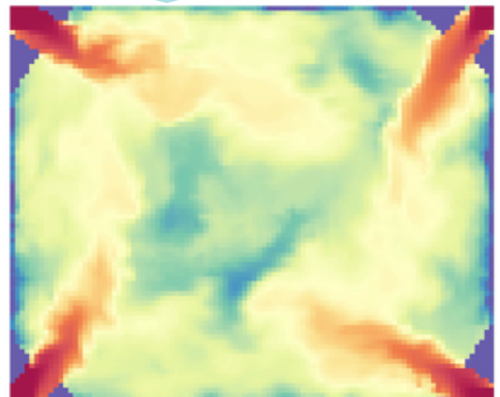
- Wasatch is 3-4x faster when using the EDSL in Uintah on the mini boiler problem.
  - *Pure LES without scalar transport or other complicating factors.*
  - *Consistent with prior results showing up to 5x & 10x faster than sibling codes in Uintah on a Taylor-Green vortex problem.*
- Scalable to 260K processes.
- This week, we introduced a new Nebo CPU backend that resulted in 20-30% performance gains immediately felt through all of Wasatch.











CARBON CAPTURE
MULTIDISCIPLINARY
SIMULATION CENTER

Multicore & GPU Performance

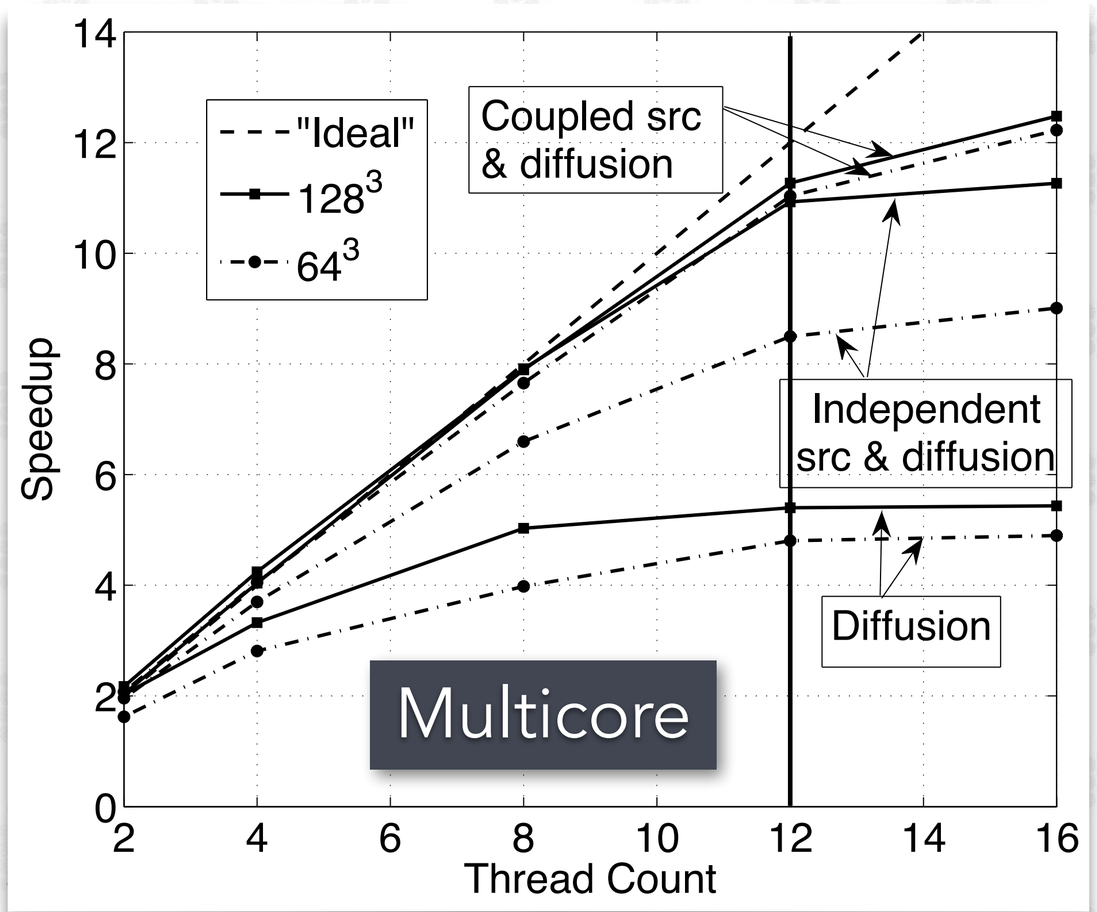
Test: mockup of a diffusion-reaction problem.

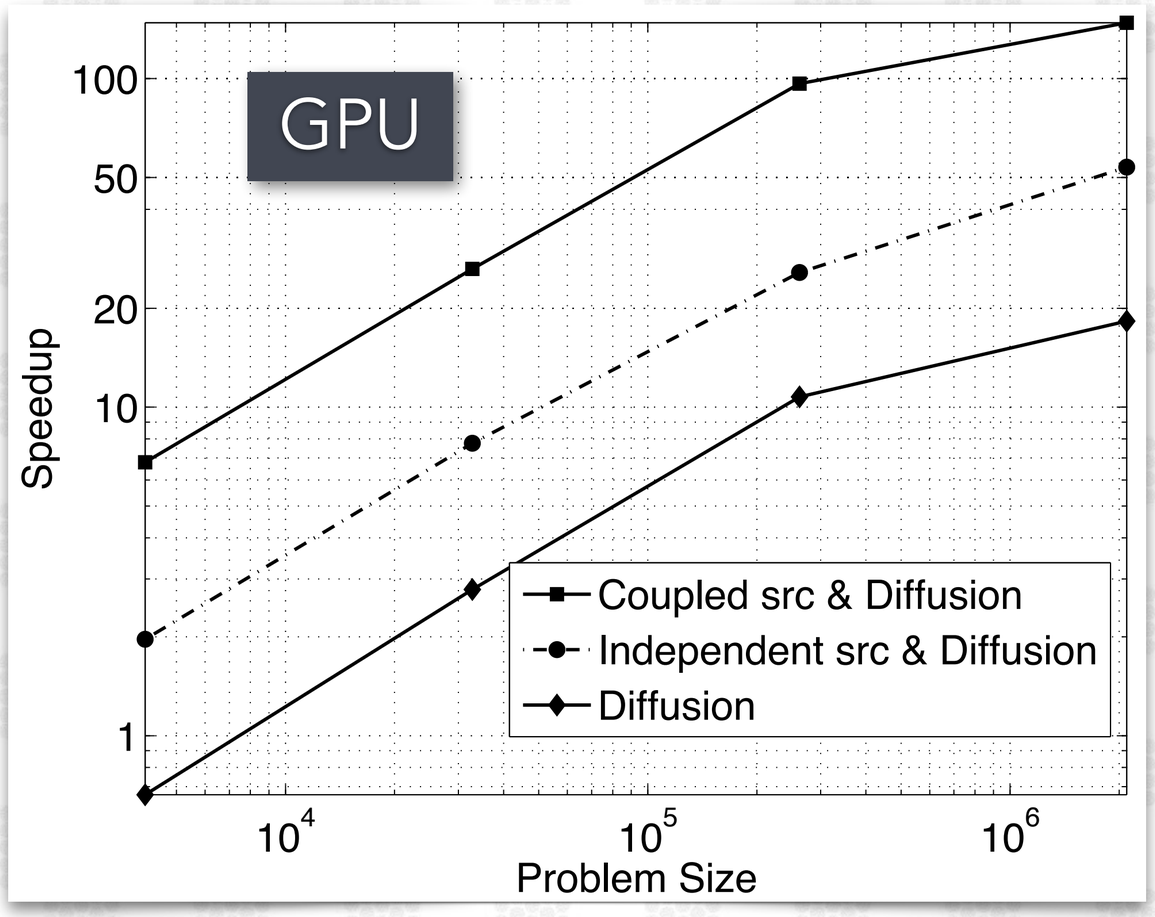
- Easily dial in the number of equations (30 here).
- Diffusion is an inexpensive stencil calculation.
- Reaction is an expensive point-wise calculation.

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

$$\mathbf{J}_i = -\Gamma \nabla \phi_i$$

$$s_i = f(\phi_i) \text{ or } s_i = f(\phi_j)$$





Parting Thoughts

- Hierarchical parallelization allows for flexible usage of available resources:
  - *Domain decomposition (SIMD)*
    - Should allow a process to do computation on "interior" while waiting on communication from neighbors.
  - *Task decomposition (MIMD)*
    - Decompose the solution into a DAG that can be scheduled asynchronously.
  - *Vectorized parallel (SIMD)*
    - Break grid operations across multicore, GPU, etc.
- DAG representation is a scalable abstraction that:
  - *Handles problem complexity gracefully.*
  - *Provides convenient separation of the problem's structure from the data.*
  - *Allows sophisticated scheduling algorithms to optimize scalability & performance.*
- (E)DSLs are very useful
  - *Future-proofing: separate intent from implementation.*
  - ***E**DSLs allow seamless transition of a code base and leverage existing compilers.*
  - *Template metaprogramming pushes work from run-time to compile-time for more efficiency.*

Wasatch

ExprLib

SpatialOps

Uintah-X