UINTAH-X A Software Approach that leads towards Exascale

Martin Berzins

Chuck Hansen

James Sutherland

Matt Might

Valerio Pascucci

DOE: for CSAFE project (97-10), DOE NETL, DOE NNSA, DOE INCITE & ALCC
NSF for SDCI and PetaApps, XSEDE, Argonne Oak Ridge

www.uintah.utah.edu

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

Institute for CLEAN AND SECURE ENERGY THE UNIVERSITY OF UTAH

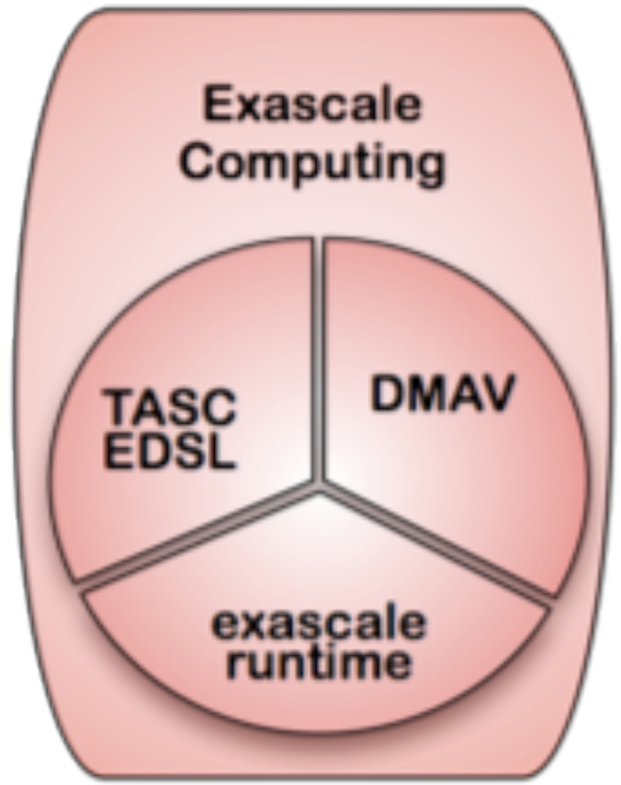
SCI www.sci.utah.edu

Berkeley UNIVERSITY OF CALIFORNIA

BYU BRIGHAM YOUNG UNIVERSITY

NNSA National Nuclear Security Administration

UINTAH-X A Software Approach that leads towards Exascale



CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

Institute for CLEAN AND SECURE ENERGY THE UNIVERSITY OF UTAH

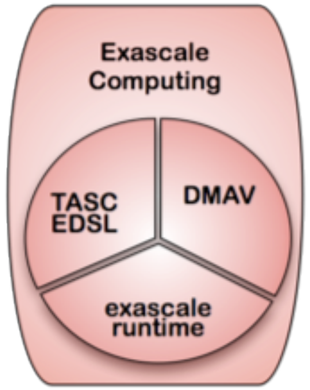
SCI www.sci.utah.edu

Berkeley UNIVERSITY OF CALIFORNIA

BYU BRIGHAM YOUNG UNIVERSITY

NNSA National Nuclear Security Administration

UINTAH-X –Advancing Progress towards Exascale Computing by CS contributions in:

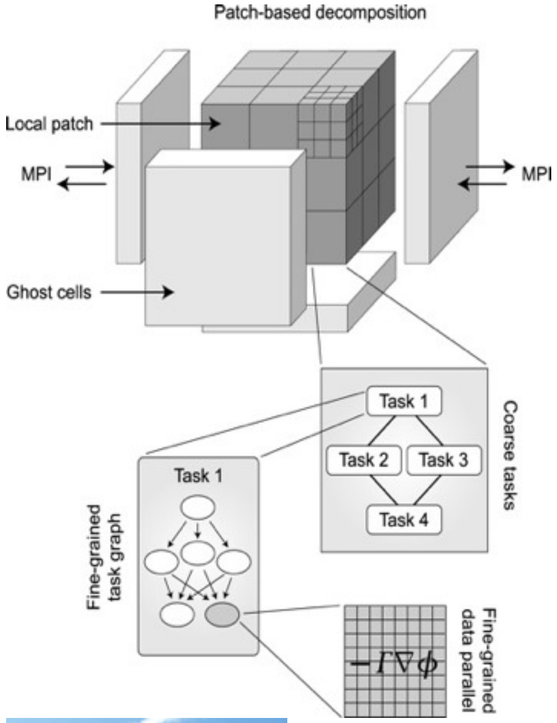


1. UINTAH-X RTS Runtime System (Martin)
Will show that a DAG asynchronous RTS will give scalable exascale solutions for the Alstom problem when combined with code from

2. TASC (WASATCH NEBO)- Domain Specific Language (Matt / James) - will generate efficient DAG extensible exascale code from high-level complex physics (Alstom) that in turn uses

3. Visus (PIDX) and Visit for I/O, Analytics & Vis (Chuck/Valerio) - performs in-situ visualization and efficient I/O of exascale solutions of e.g. Alstom Boiler.

4. Related Work: Synergistic work is e.g. fault tolerance formal methods for debugging, languages (Martin, James and Matt with NSF funding)



Uintah-X Architecture

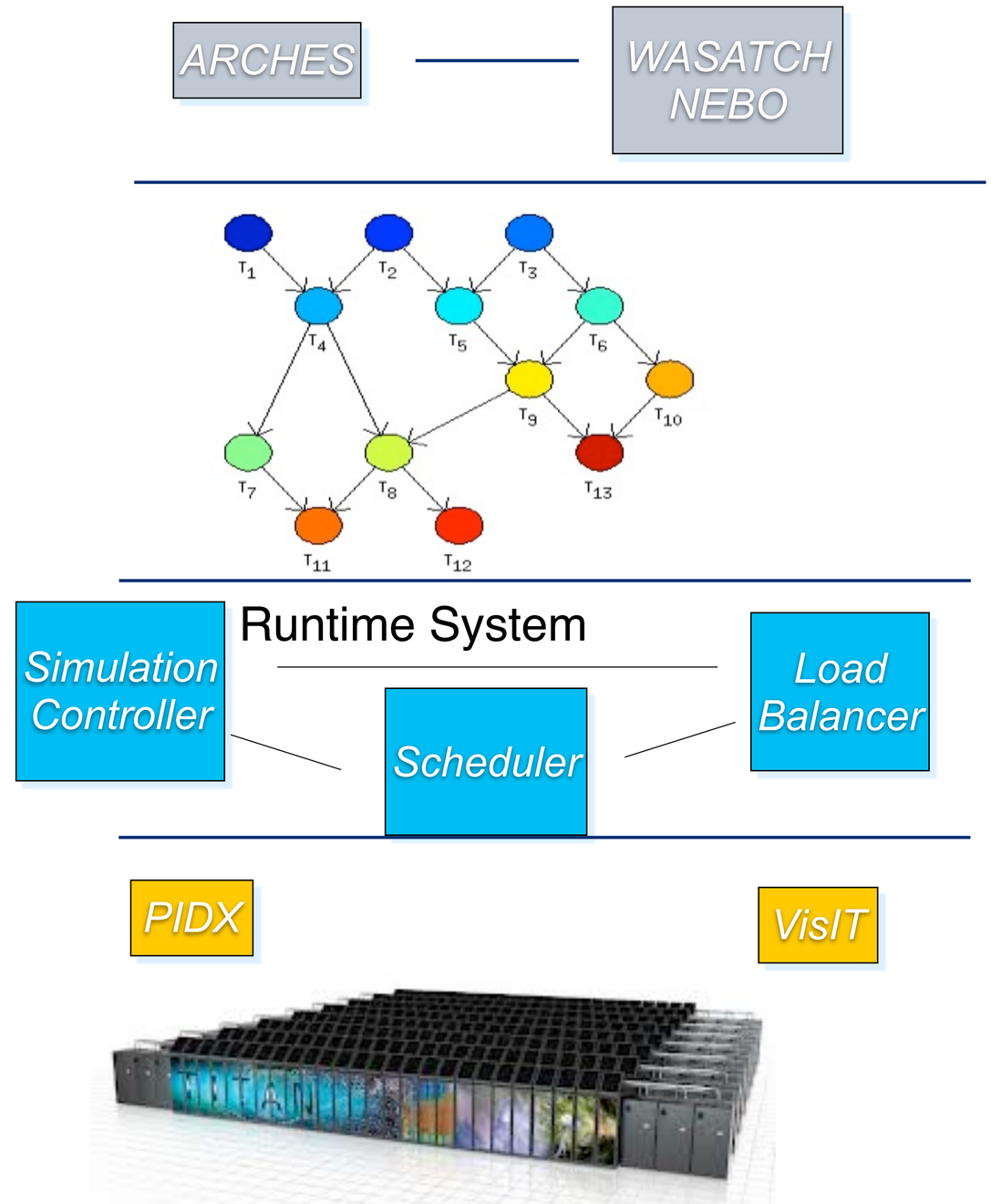
- **Problem Specification** via ARCHES or WASATCH/NEBO EDSL

- **Abstract task-graph** program executes on:

- **Runtime System** with:
  - Asynchronous out-of-order execution
  - Work stealing
  - Overlap communication & computation
  - Tasks running on cores and accelerators

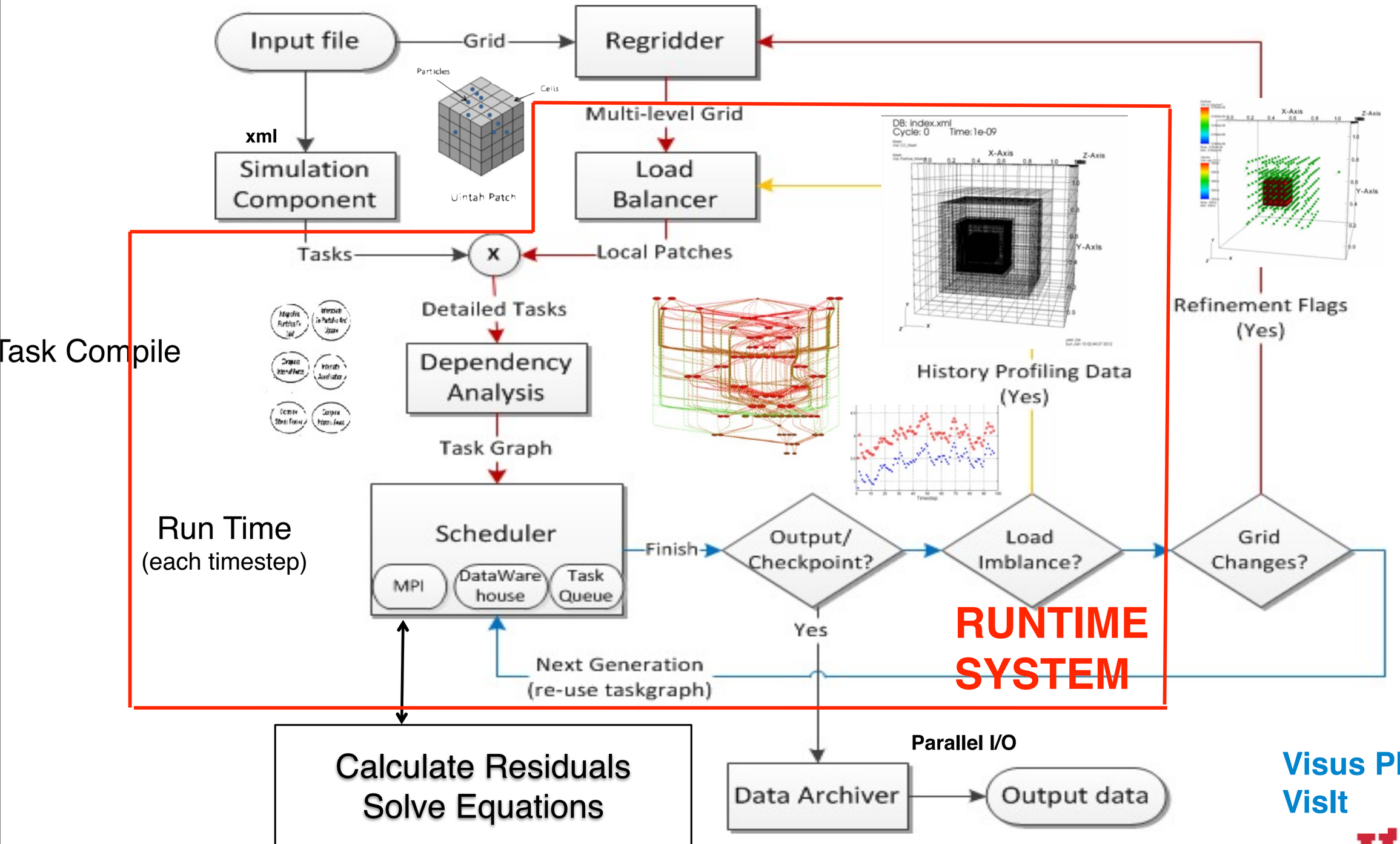
- **Scalable I/O via Visus PIDX**

Data Analysis and Visualization



"Exascale programming will require prioritization of critical-path and non-critical path tasks, adaptive directed acyclic graph scheduling of critical-path tasks, and adaptive rebalancing of all tasks……"[Brown et al. Exascale Report**]**

ARCHES or WASATCH/NEBO

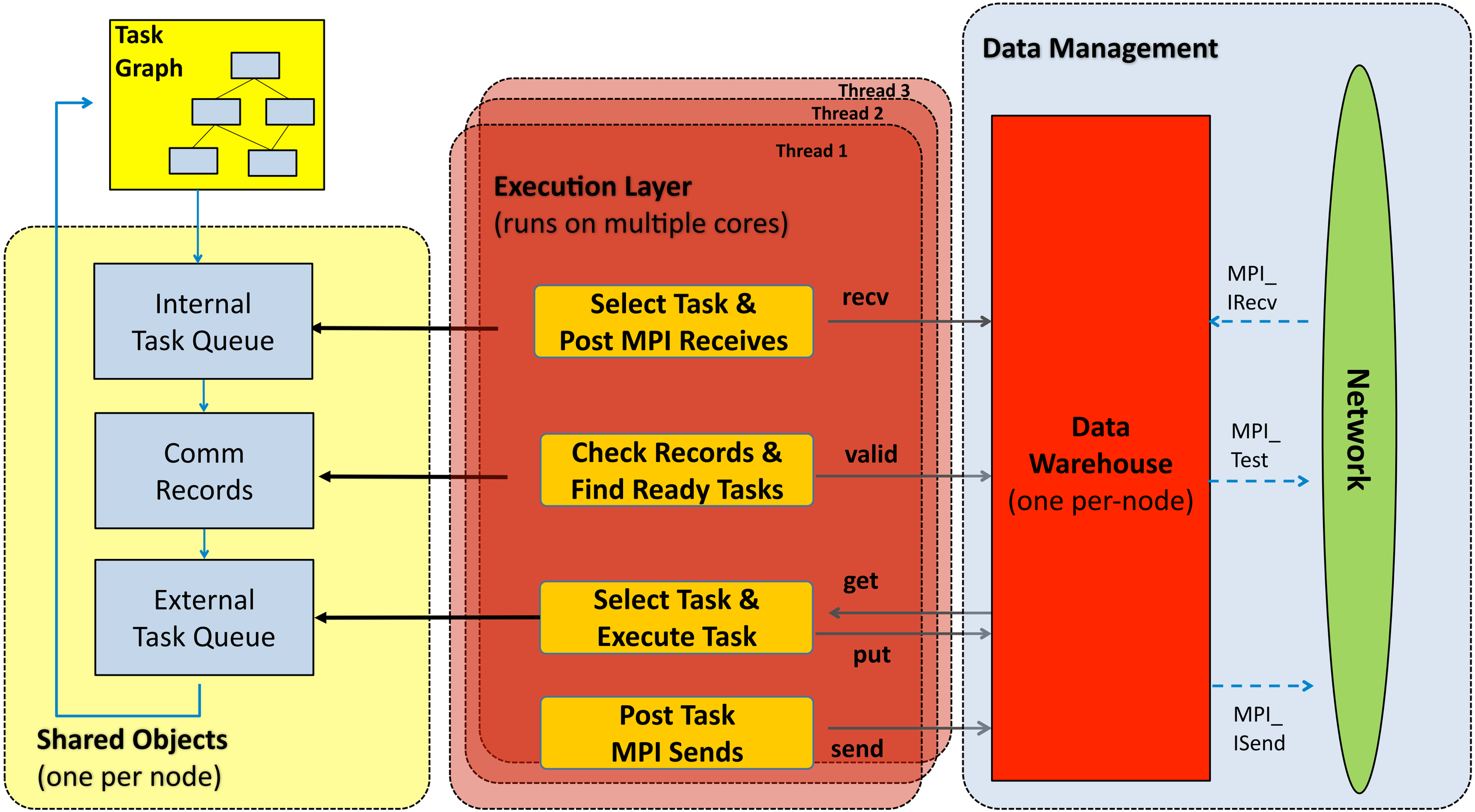


Visus PIDX
VisIt

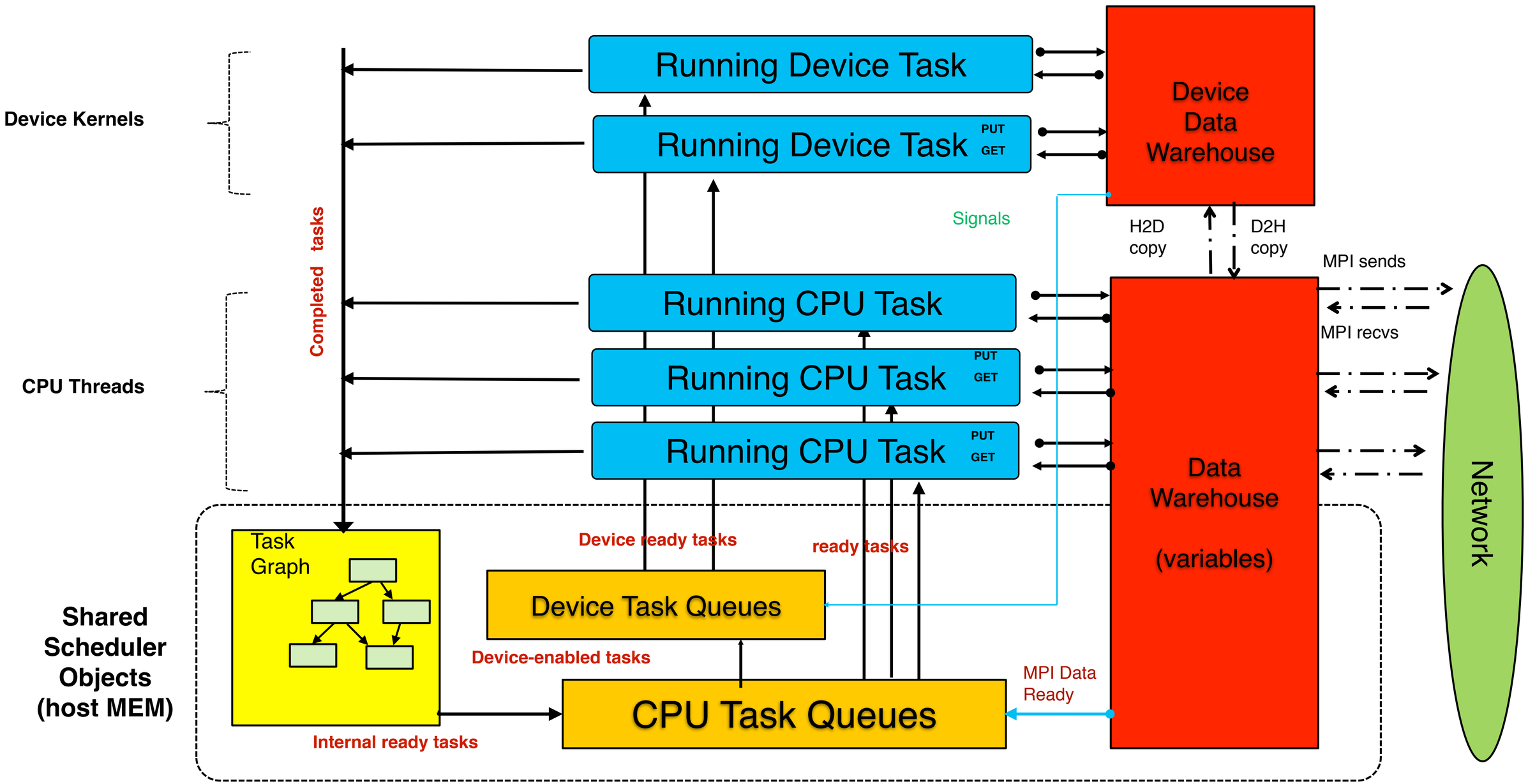
THE UNIVERSITY OF UTAH

UINTAH-X ARCHITECTURE

Uintahx Runtime System

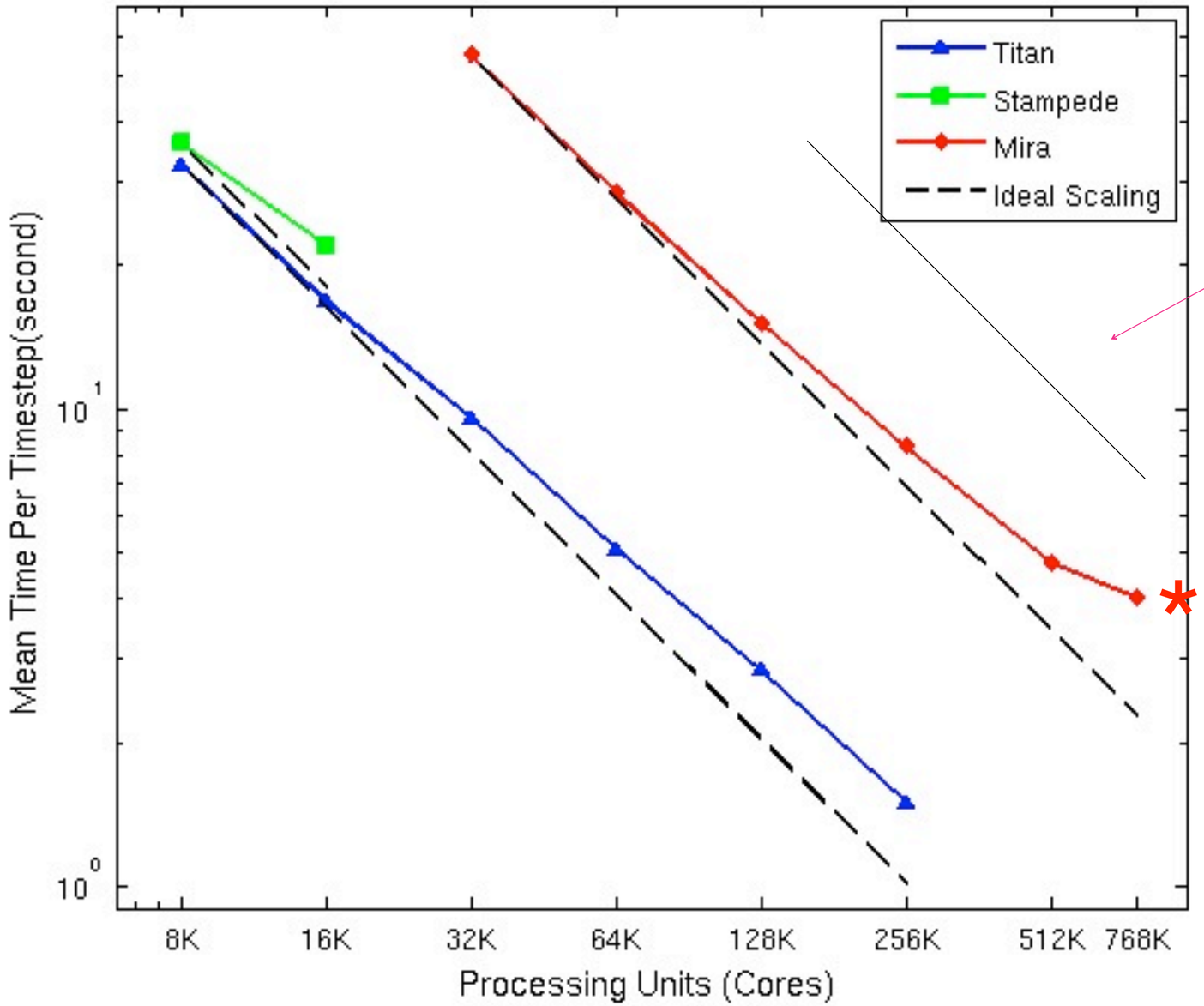


Unified Heterogeneous Scheduler & Runtime



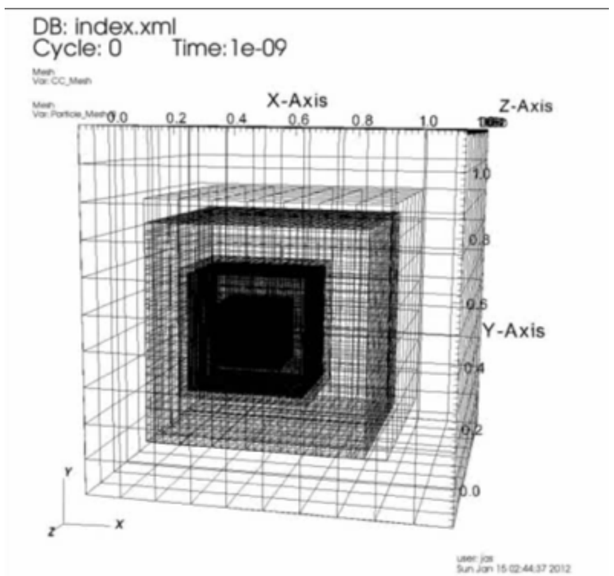
THE UNIVERSITY OF UTAH

MPM AMR ICE Strong Scaling



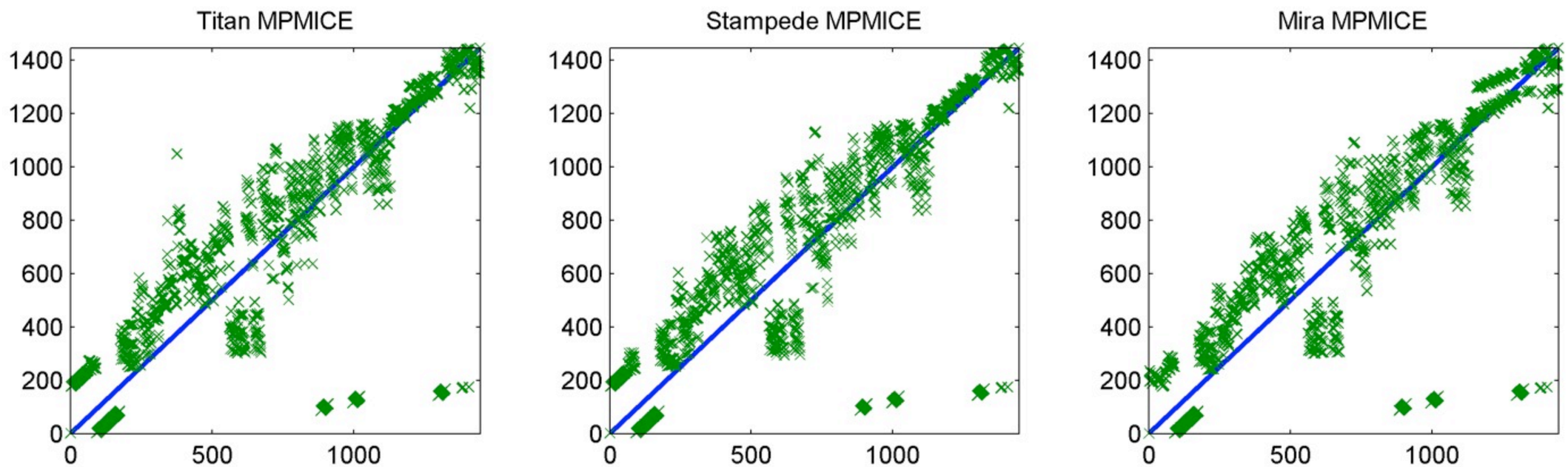
Debugging what should have been this run caused DDT to crash Mira

* one mesh patch for every 3 cores



Complex fluid-structure interaction problem with adaptive mesh refinement, see SC13 paper
NSF funding.

Scalability is at least partially achieved by not executing tasks in order e.g. AMR fluid-structure interaction



Straight line represents given order of tasks Green X shows when a task is actually executed.
Above the line means late execution while below the line means early execution took place. More "late" tasks than "early" ones as e.g.
TASKS: 1 2 3 4 5 1 4 2 3 5

Early **Late execution**

Linear Solves arises from Navier –Stokes Equations

$$\frac{\partial \rho}{\partial t} + \nabla.\rho u = 0,$$

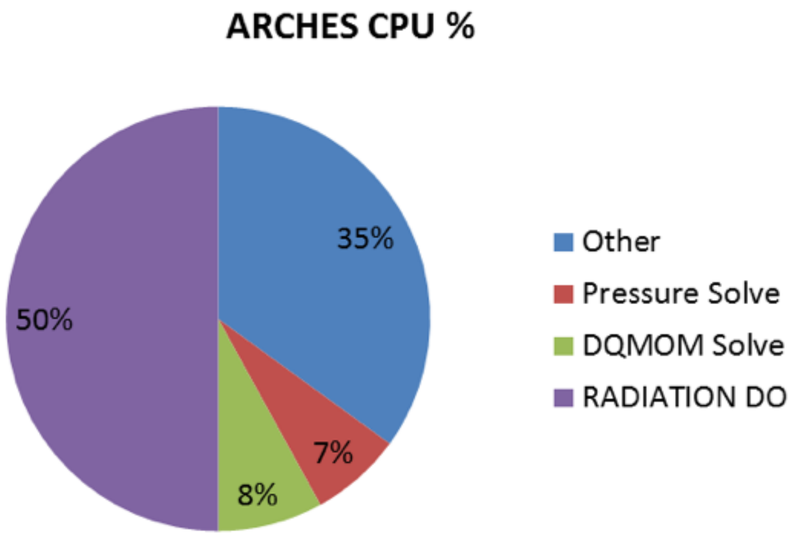
Full model includes turbulence, chemical reactions and radiation

where ρ is density, u is velocity vector and p is pressure

$$\frac{\partial \rho u}{\partial t} = F - \nabla p, \quad \text{where} \quad F = -\nabla.\rho uu + \nu \nabla^2 u + \rho g$$

Arrive at pressure Poisson equation to solve for *p*

$$\nabla^2 p = R, \quad \text{where } R = \nabla.F + \frac{\partial^2 p}{\partial t^2}$$



Use Hypre Solver distributed by LLNL
Many linear solvers inc. Preconditioned Conjugate Gradients on regular mesh patches used
Multi-grid pre-conditioner used
Careful adaptive strategies needed to get scalability
CCGrid13 paper.

One radiation solve per timestep

THE UNIVERSITY OF UTAH

Linear Solves arises from Navier –Stokes Equations

$$\frac{\partial \rho}{\partial t} + \nabla.\rho u = 0,$$

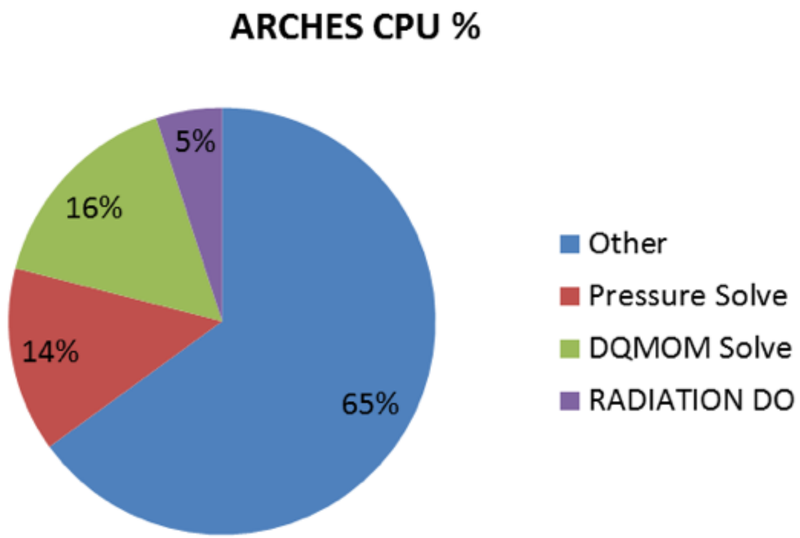
Full model includes turbulence, chemical reactions and radiation

where ρ is density, u is velocity vector and p is pressure

$$\frac{\partial \rho u}{\partial t} = F - \nabla p, \quad \text{where} \quad F = -\nabla.\rho uu + \nu \nabla^2 u + \rho g$$

Arrive at pressure Poisson equation to solve for *p*

$$\nabla^2 p = R, \quad \text{where } R = \nabla.F + \frac{\partial^2 p}{\partial t^2}$$



Use Hypre Solver distributed by LLNL
Many linear solvers inc. Preconditioned Conjugate Gradients on regular mesh patches used
Multi-grid pre-conditioner used
Careful adaptive strategies needed to get scalability
CCGrid13 paper.

One radiation solve
Every 10 timesteps

THE UNIVERSITY OF UTAH

NVIDIA AMGX Linear Solvers on GPUs

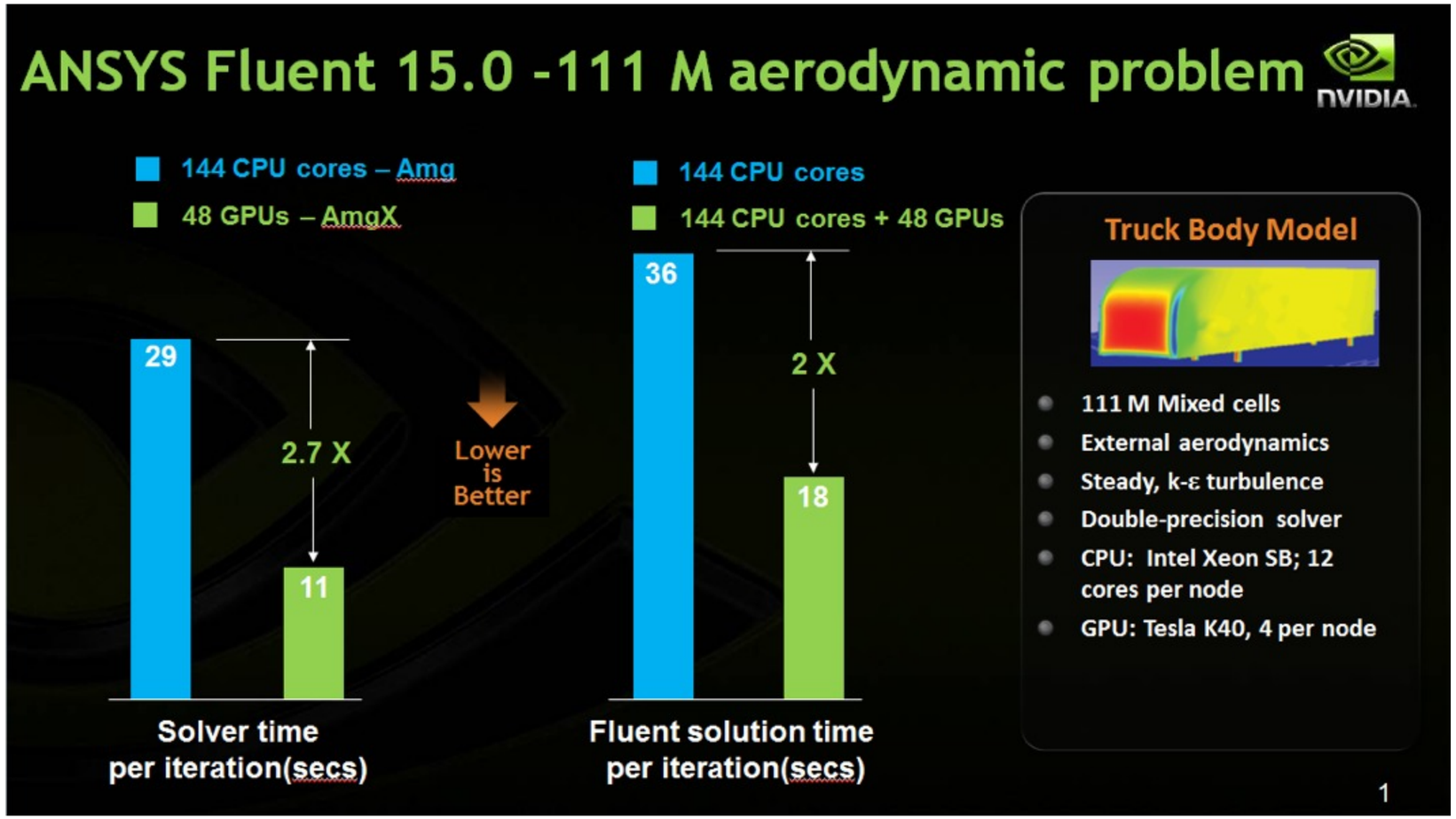
- Fast, scalable iterative gpu linear solvers for packages e.g.,
- Flexible toolkit provides GPU accelerated Ax = b solver
- Simple API for multiple apps domains.
- Multiple GPUs (maybe thousands) with scaling

Key Features

Ruge-Steuben algebraic MG
Krylov methods: CG, GMRES, BiCGStab, Smoothers and Solvers: Block- Jacobi, Gauss-Seidel, incomplete LU,

Flexible composition system
Scalar or coupled block Systems, MPI support
OpenMP support,
Flexible and high level C API,

Free for non-commercial use
Utah access via Utah CUDA COE.



THE UNIVERSITY OF UTAH

Uintah-X Parting Thoughts

- Uintah DAG-based approach
  - Solve wide range of problems with different communication requirements: Fluid-Structure interaction(MPMICE), Combustion(Arches), Radiation(RMCRT)
  - Excellent scaling on many Peta-scale supercomputers: Kraken, Titan, Stampede, Mira..
  - Ability to use heterogeneous resources:
    - GPU and MIC
- Schedule tasks dynamically according to run-time MPI communications costs
- Future work:
  - Task stealing across CPU/device queues
  - Improve GPU/MIC memory management

THE UNIVERSITY OF UTAH

Wasatch Expressions Example

Energy equation

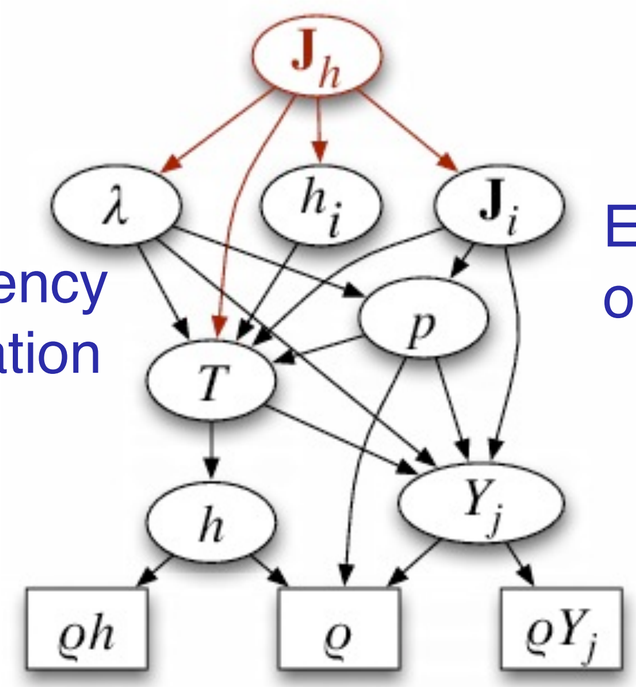
$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e \underline{u}) + \nabla.\underline{J}_h + terms = 0$$

Enthalpy diffusive flux

$$\underline{J}_h = -\lambda(T, Y_j)\nabla T - \sum_{i=1}^{n} h_i \underline{J}_i$$

$$\underline{J}_i = -\sum_{j=1}^{ns} D_{ij}(T, Y_j)\nabla Y_j - D_i^T(T, Y_j)\nabla T$$

Expression Tree

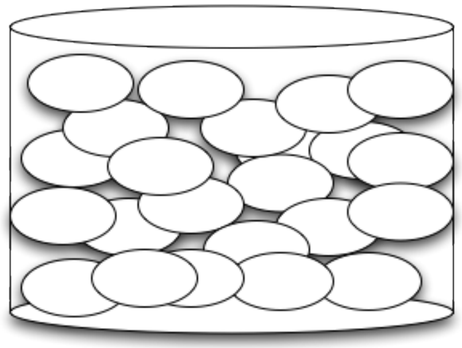


Dependency specification

Execution order

THE UNIVERSITY OF UTAH

Wasatch Expressions Example



Create individual expressions and register them.

Energy equation

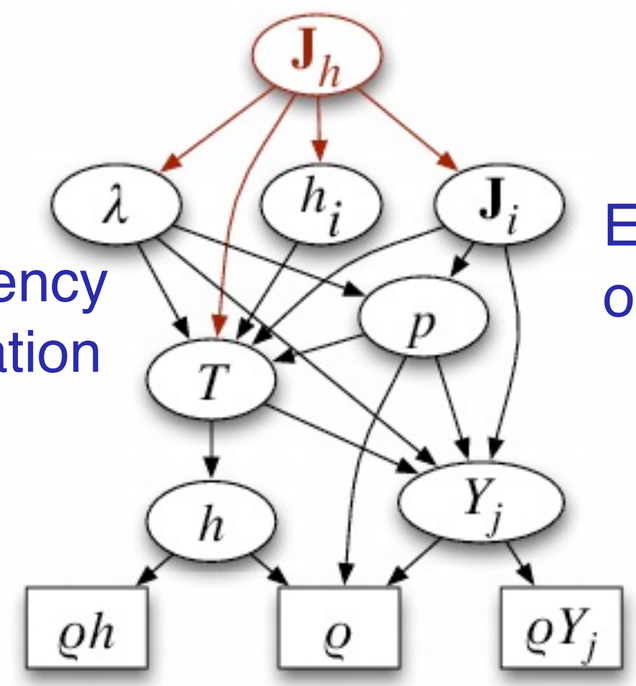
$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e \underline{u}) + \nabla.\underline{J}_h + terms = 0$$

Enthalpy diffusive flux

$$\underline{J}_h = -\lambda(T, Y_j)\nabla T - \sum_{i=1}^{n} h_i \underline{J}_i$$

$$\underline{J}_i = -\sum_{j=1}^{ns} D_{ij}(T, Y_j)\nabla Y_j - D_i^T(T, Y_j)\nabla T$$

Expression Tree



Dependency specification

Execution order

THE UNIVERSITY OF UTAH

Wasatch Expressions Example

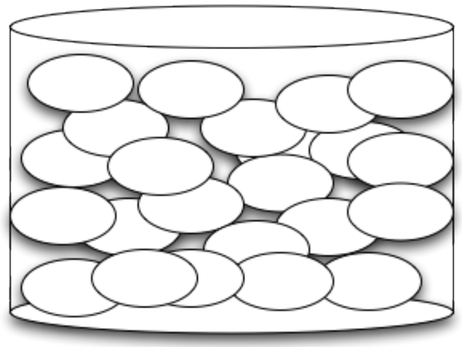
Energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e \underline{u}) + \nabla.\underline{J}_h + terms = 0$$

Enthalpy diffusive flux

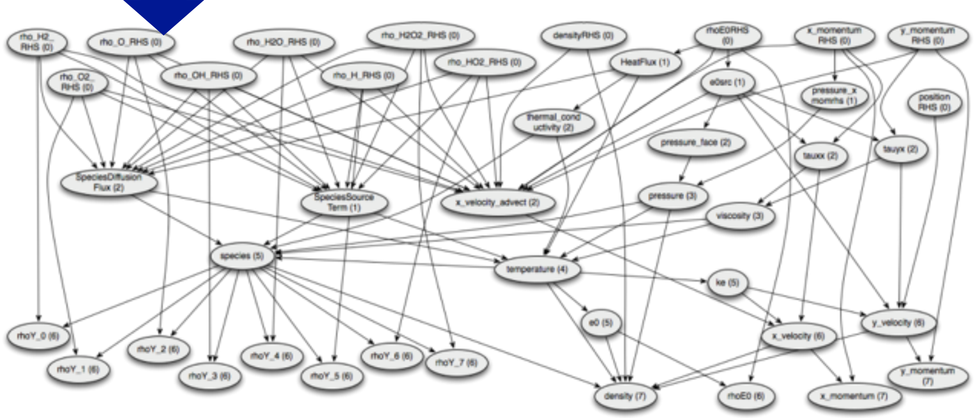
$$\underline{J}_h = -\lambda(T, Y_j)\nabla T - \sum_{i=1}^{n} h_i \underline{J}_i$$

$$\underline{J}_i = -\sum_{j=1}^{ns} D_{ij}(T, Y_j)\nabla Y_j - D_i^T(T, Y_j)\nabla T$$



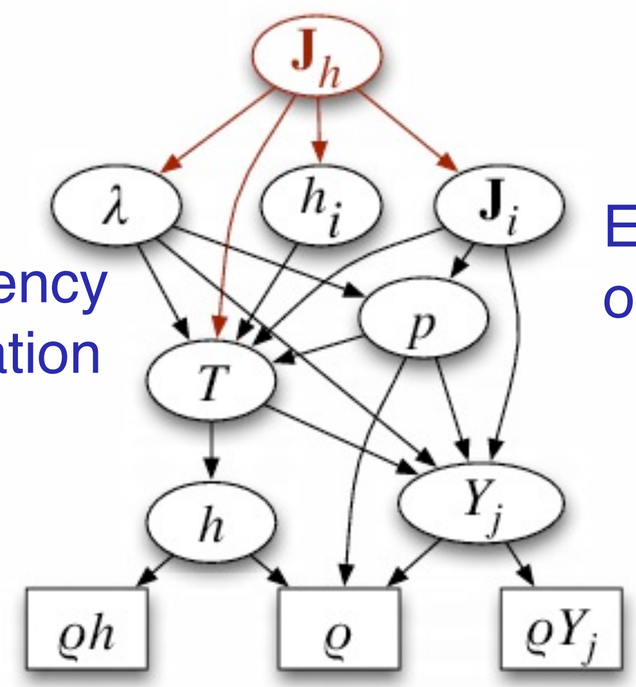
Create individual expressions and register them.

Expression Tree



Request calculation of any # of fields. The graph to accomplish this is automatically created!

Dependency specification



Execution order

THE UNIVERSITY OF UTAH

Wasatch Expressions Example

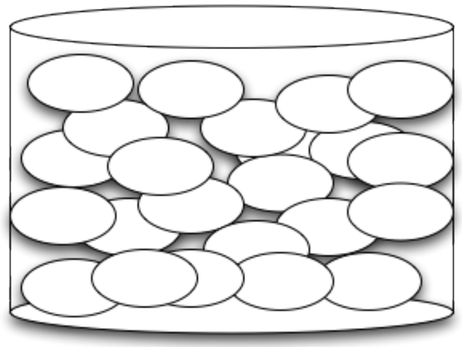
Energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e \underline{u}) + \nabla.\underline{J}_h + terms = 0$$

Enthalpy diffusive flux

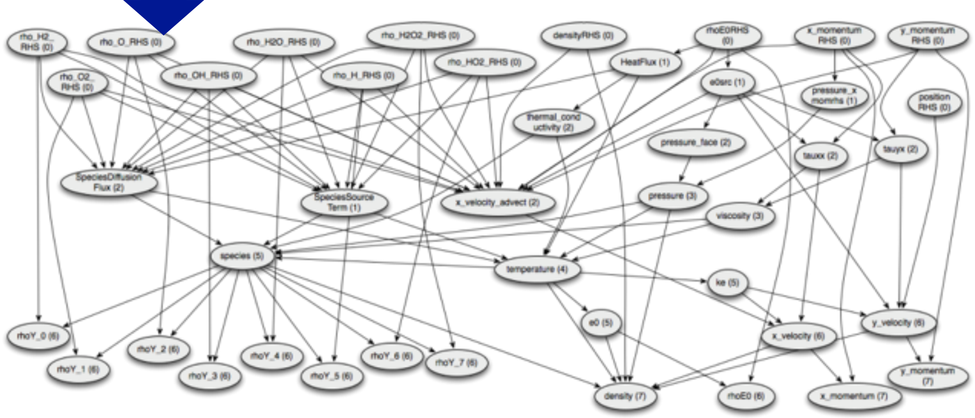
$$\underline{J}_h = -\lambda(T, Y_j)\nabla T - \sum_{i=1}^{n} h_i \underline{J}_i$$

$$\underline{J}_i = -\sum_{j=1}^{ns} D_{ij}(T, Y_j)\nabla Y_j - D_i^T(T, Y_j)\nabla T$$



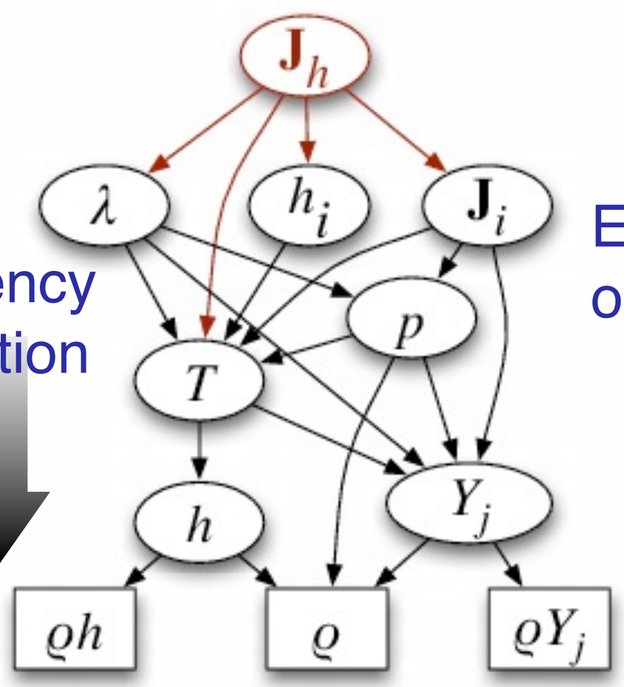
Create individual expressions and register them.

Expression Tree



Request calculation of any # of fields. The graph to accomplish this is automatically created!

Dependency specification



Execution order

THE UNIVERSITY OF UTAH

Wasatch Expressions Example

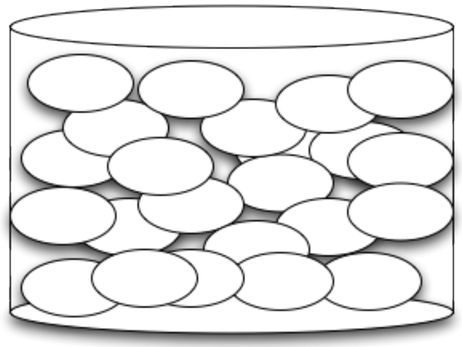
Energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e \underline{u}) + \nabla.\underline{J}_h + terms = 0$$

Enthalpy diffusive flux

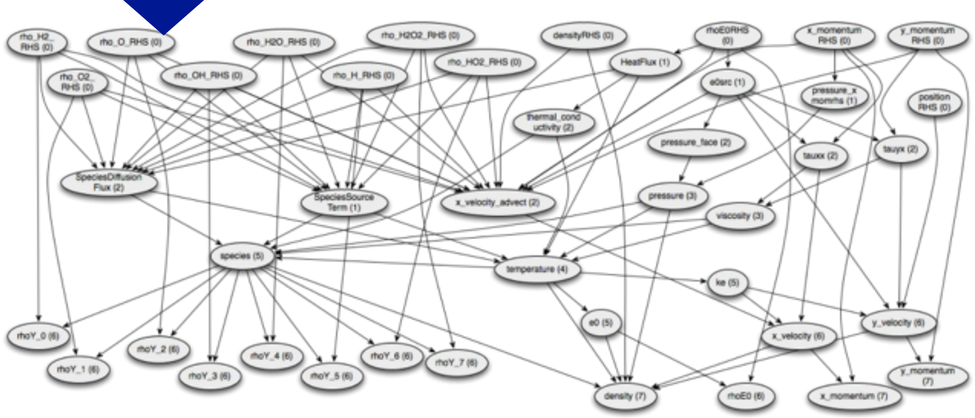
$$\underline{J}_h = -\lambda(T, Y_j)\nabla T - \sum_{i=1}^{n} h_i \underline{J}_i$$

$$\underline{J}_i = -\sum_{j=1}^{ns} D_{ij}(T, Y_j)\nabla Y_j - D_i^T(T, Y_j)\nabla T$$



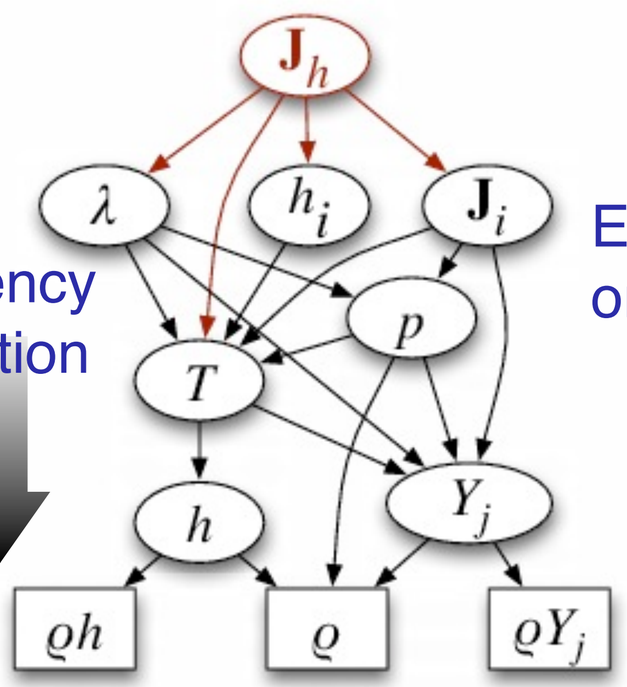
Create individual expressions and register them.

Expression Tree



Request calculation of any # of fields. The graph to accomplish this is automatically created!

Dependency specification



Execution order

Nebo: Embedded Domain Specific Language for HPC

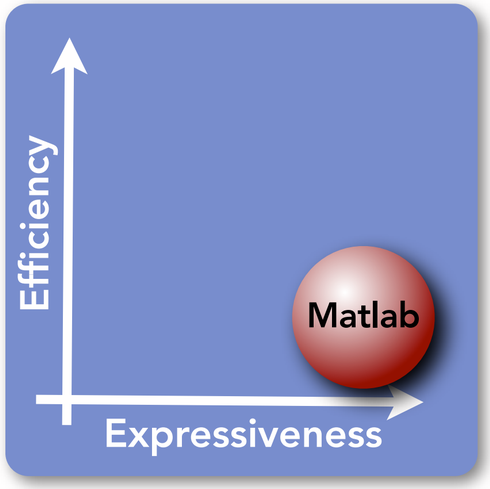
Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Why an EDSL?

Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Why an EDSL?

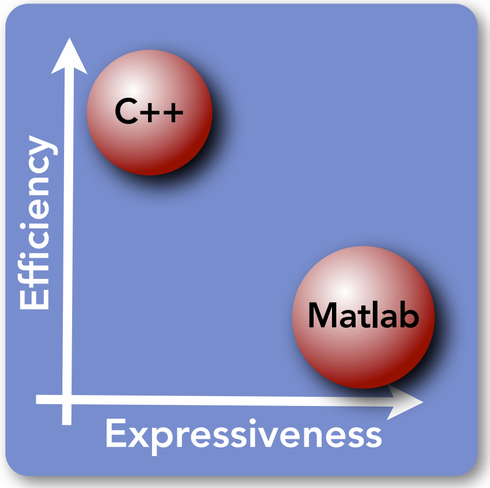
- Expressive syntax (matlab-style array operations)
  - *Programmer expresses intent (problem structure) - not implementation.*



Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Why an EDSL?

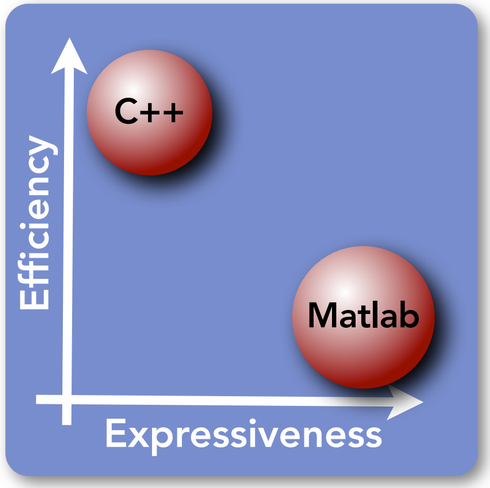
- Expressive syntax (matlab-style array operations)
  - *Programmer expresses intent (problem structure) - not implementation.*
- High performance
  - *should match hand-tuned code in performance*

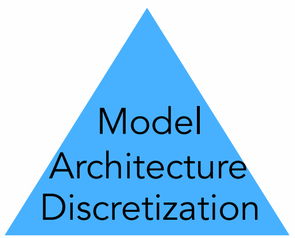


Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

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*

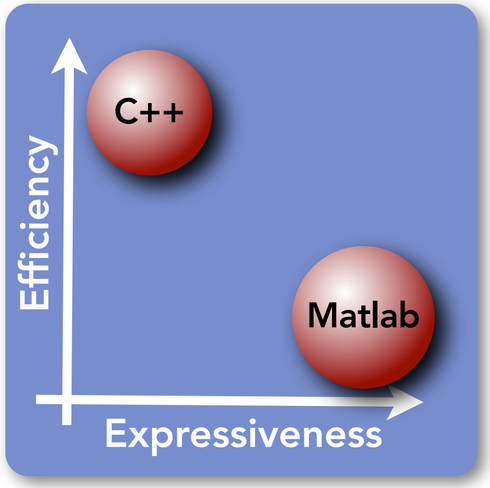


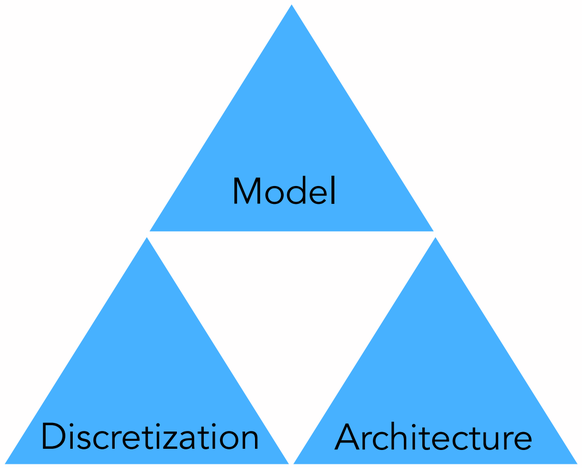


Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

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*

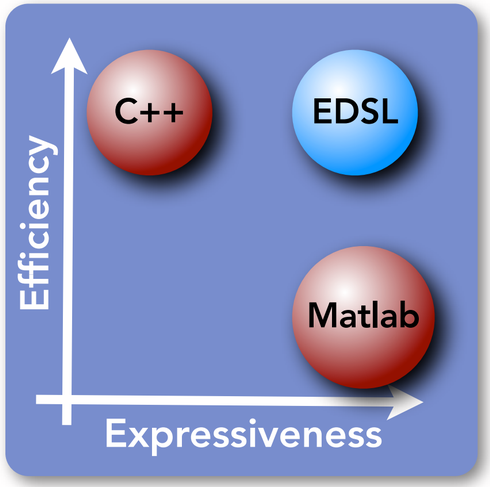


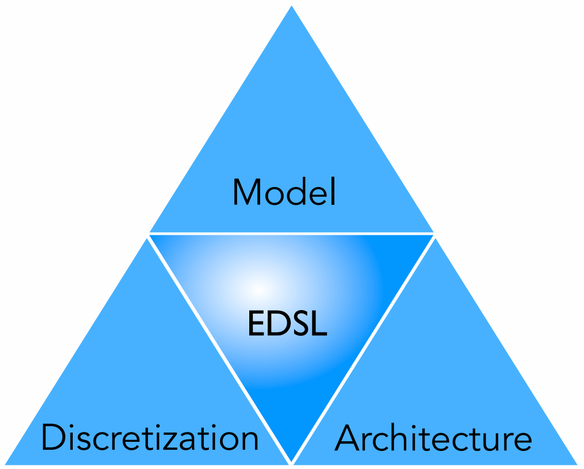


Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

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*

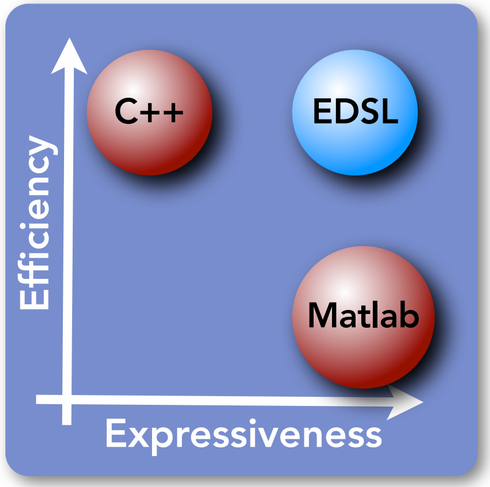


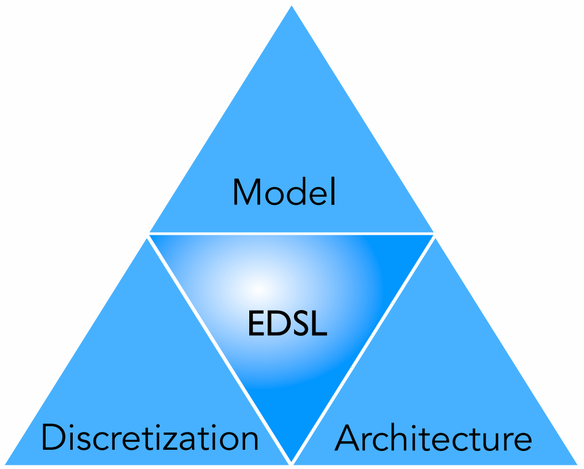


Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

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.*

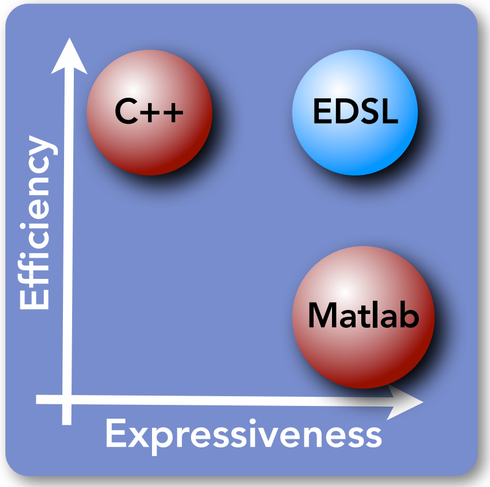




Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

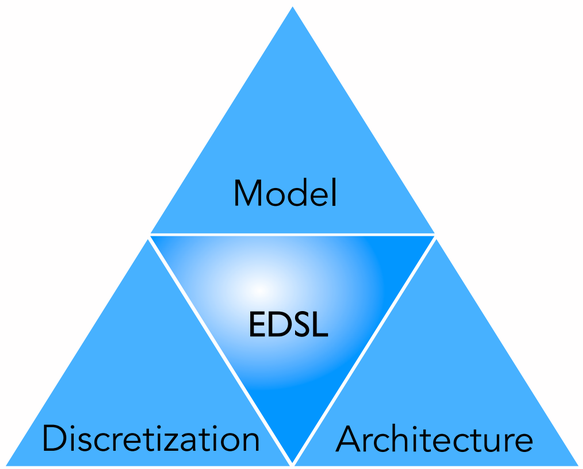
Why an EDSL?

- 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.*



C++ Template Metaprogramming

- embedded in C++
- compile-time reasoning about the problem



Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Chained Stencil Operations

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

Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Chained Stencil Operations

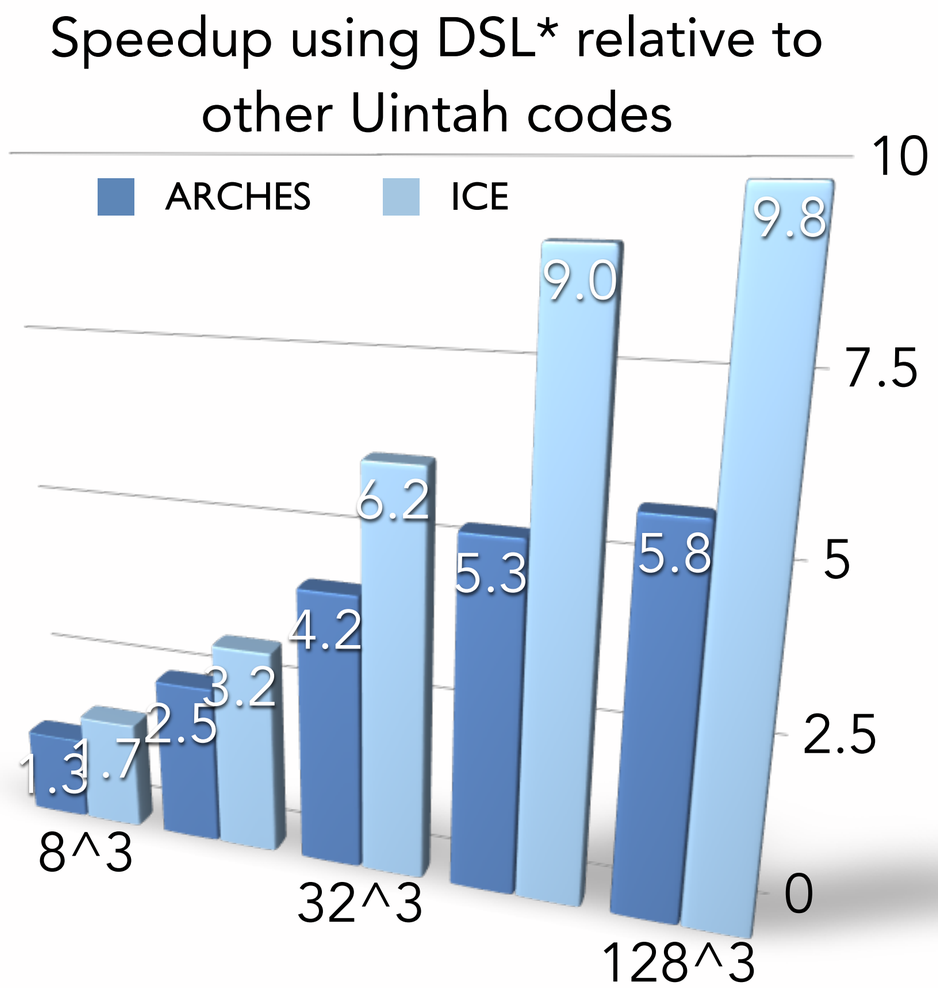
$$\begin{aligned} \phi &= -\nabla \cdot \mathbf{q} \\ &= \nabla \cdot (-\lambda \nabla T) \end{aligned}$$

```
phi <<= divx( interpx(lambda) * gradx(temperature) )
      + divy( interpy(lambda) * grady(temperature)
      + divz( interpz(lambda) * gradz(temperature);
```

- One inlined grid loop, no temporaries.
- Better parallel performance than without chaining.
- Compile-time consistency checking (field-operator and field-field compatibility).

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Performance & Scalability

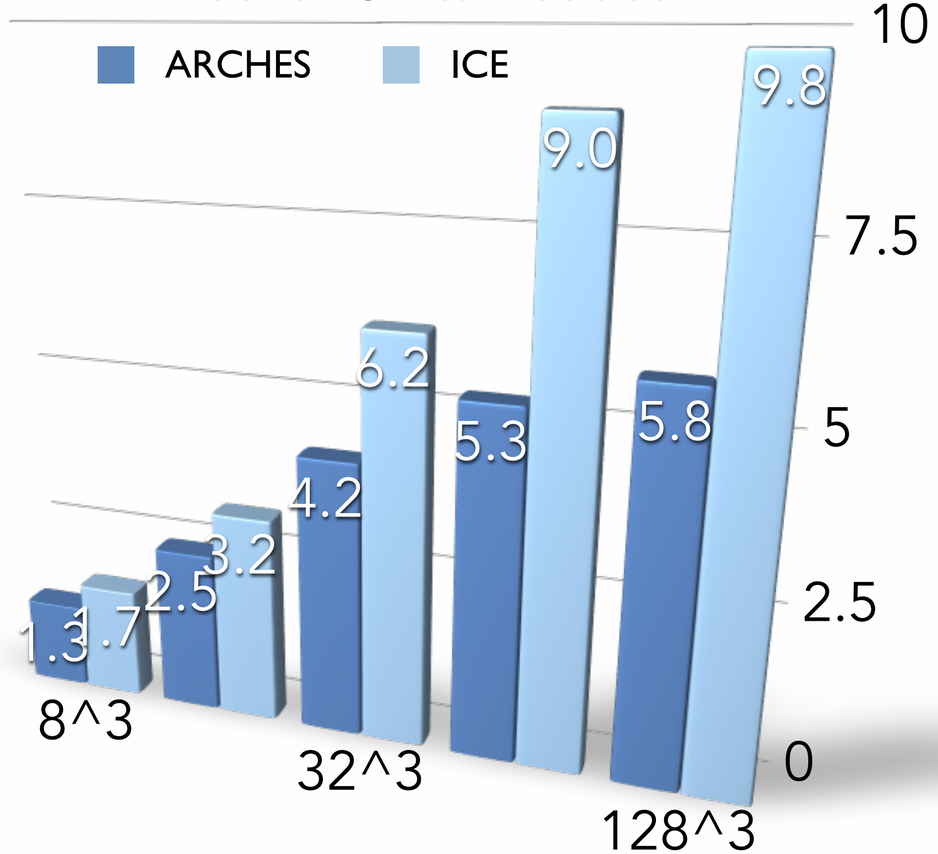


*Comparison to ICE and ARCHES, sister codes in Uintah, on a 3D Taylor-Green vortex problem. Run on a single processor.

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

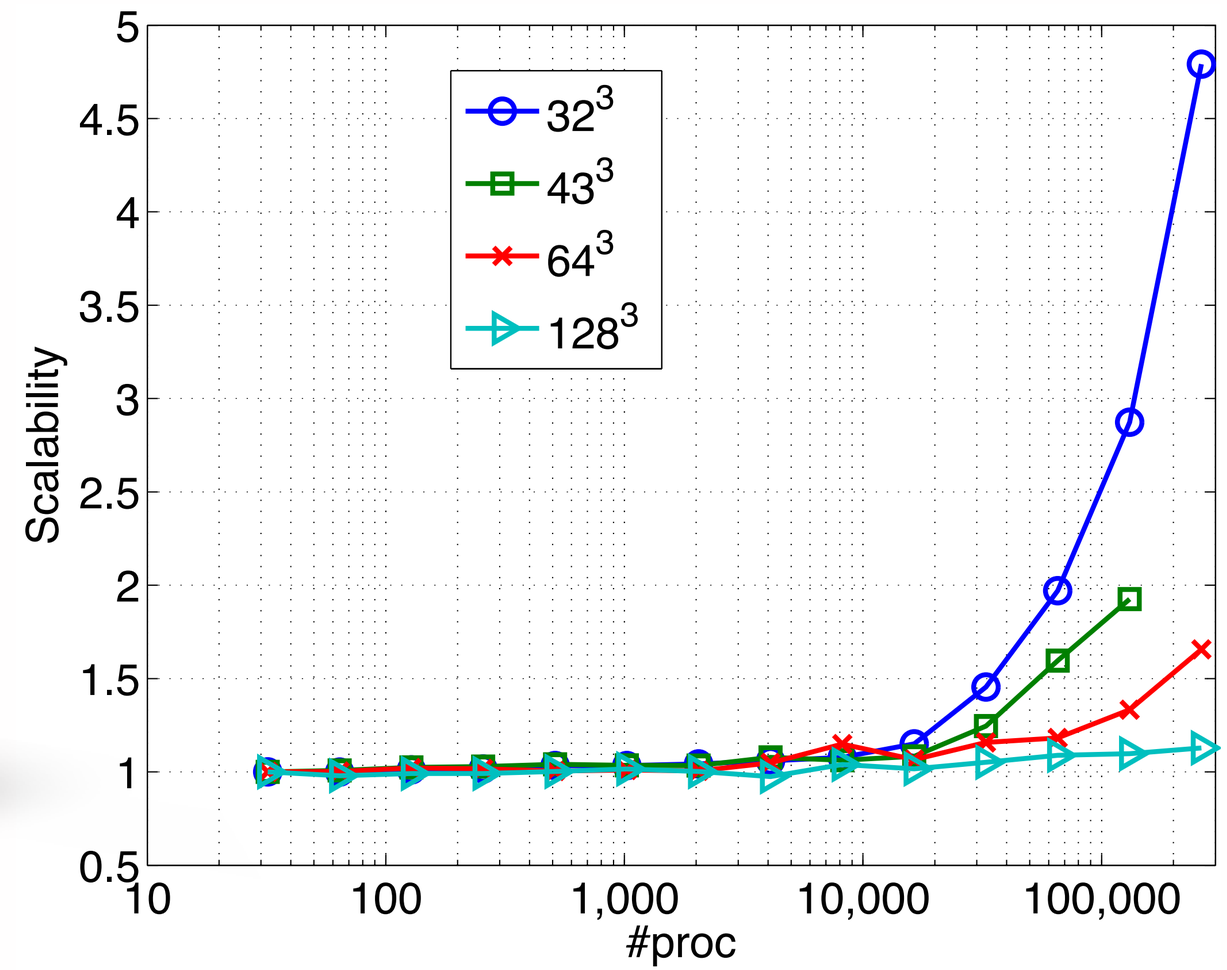
Performance & Scalability

Speedup using DSL* relative to other Uintah codes



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

Weak scaling on Titan

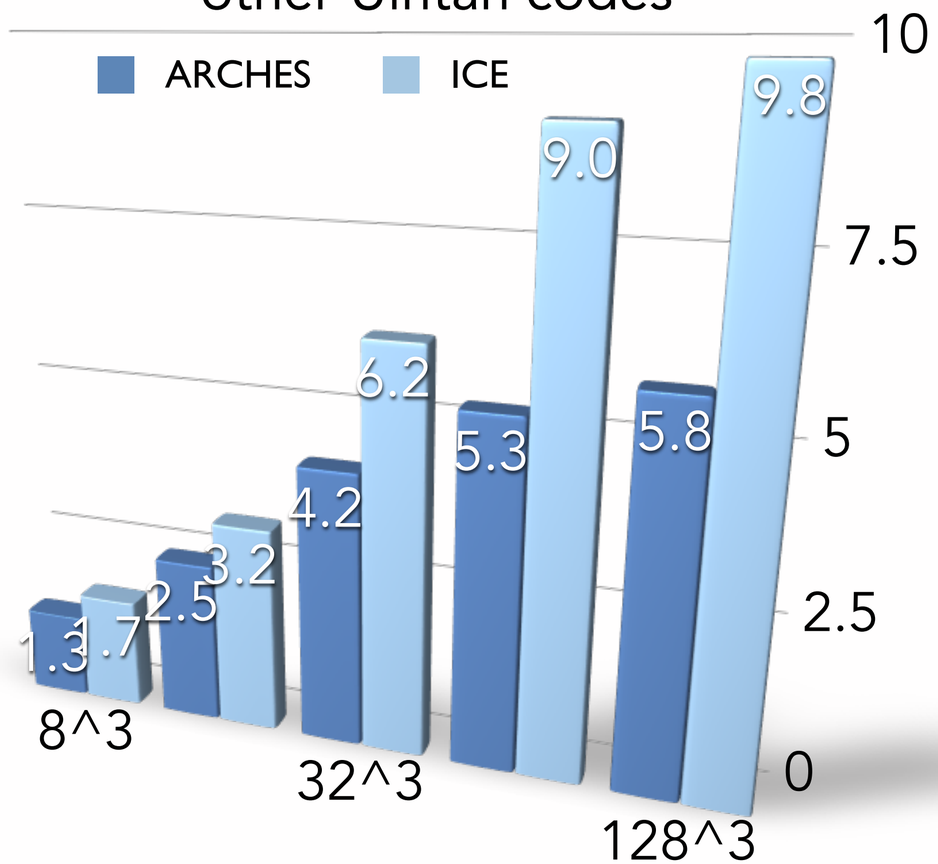


"1" indicates perfect weak scaling

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

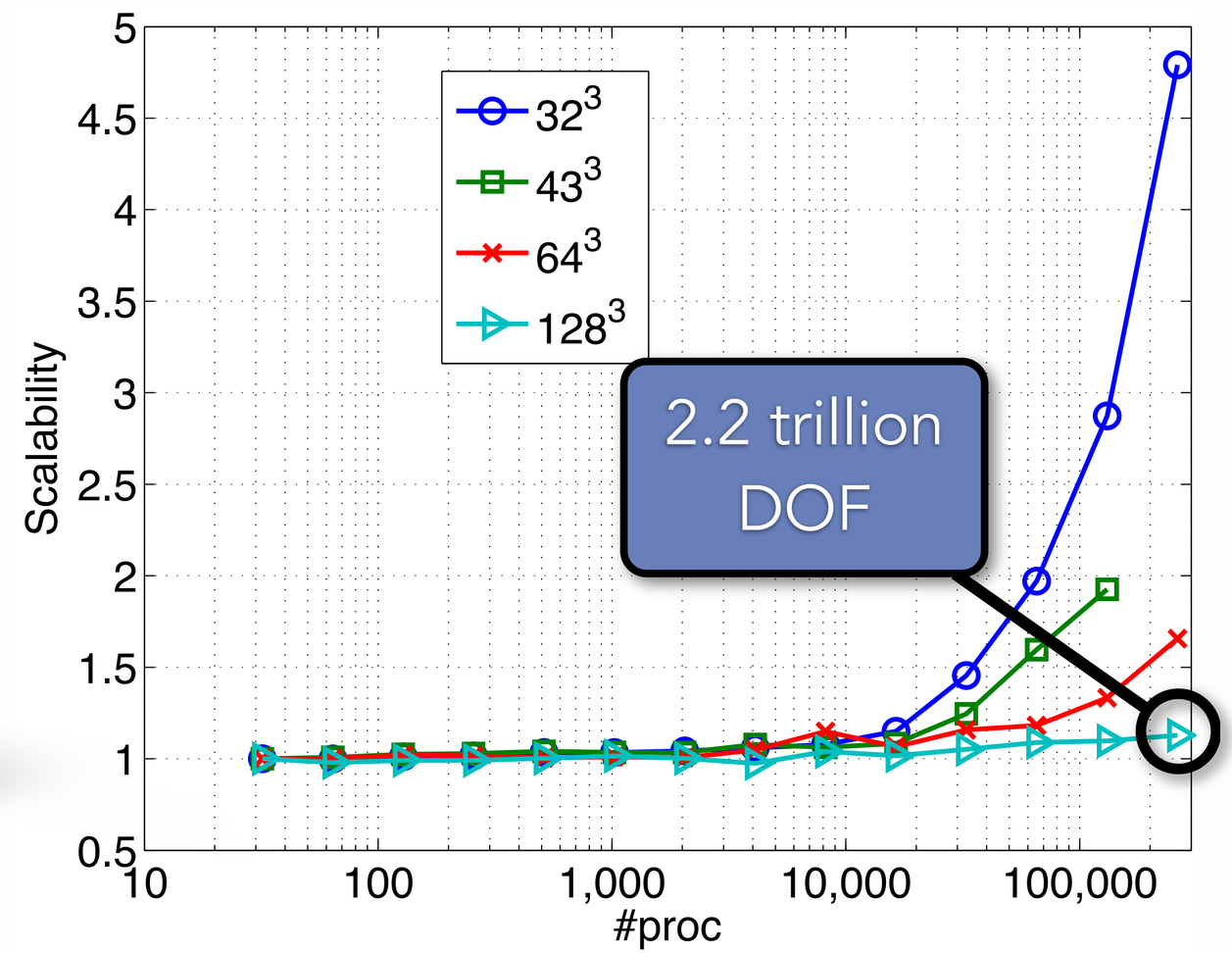
Performance & Scalability

Speedup using DSL* relative to other Uintah codes



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

Weak scaling on Titan



"1" indicates perfect weak scaling

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU Performance

$$\nabla \cdot (\lambda \nabla T)$$

Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU Performance

$$\nabla \cdot (\lambda \nabla T)$$

```
phi <<= divX( -interpX(lambda) * gradX(T) )
      + divY( -interpY(lambda) * gradY(T) )
      + divZ( -interpZ(lambda) * gradZ(T) );
```

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU Performance

$$\nabla \cdot (\lambda \nabla T)$$

```
phi <<= divX( -interpX(lambda) * gradX(T) )
      + divY( -interpY(lambda) * gradY(T) )
      + divZ( -interpZ(lambda) * gradZ(T) );
```

	2	4	6	8	10	12	GPU
64x64x64	1.3	2.4	2.6	3.3	3.3	3.3	13.8
128x128x128	1.9	3.9	4.9	6.8	7.8	6.0	26.0

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU Performance

$$\nabla \cdot (\lambda \nabla T)$$

```
phi <<= divX( -interpX(lambda) * gradX(T) )
      + divY( -interpY(lambda) * gradY(T) )
      + divZ( -interpZ(lambda) * gradZ(T) );
```

	2	4	6	8	10	12	GPU
64x64x64	1.3	2.4	2.6	3.3	3.3	3.3	13.8
128x128x128	1.9	3.9	4.9	6.8	7.8	6.0	26.0

$$-\nabla \cdot (\rho \phi \mathbf{u} + \mathbf{J}_\phi)$$

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU Performance

$$\nabla \cdot (\lambda \nabla T)$$

```
phi <<= divX( -interpX(lambda) * gradX(T) )
      + divY( -interpY(lambda) * gradY(T) )
      + divZ( -interpZ(lambda) * gradZ(T) );
```

	2	4	6	8	10	12	GPU
64x64x64	1.3	2.4	2.6	3.3	3.3	3.3	13.8
128x128x128	1.9	3.9	4.9	6.8	7.8	6.0	26.0

$$-\nabla \cdot (\rho \phi \mathbf{u} + \mathbf{J}_\phi)$$

```
rhs <<= -divOpX_( xConvFlux_ + xDiffFlux_ )
        -divOpY_( yConvFlux_ + yDiffFlux_ )
        -divOpZ_( zConvFlux_ + zDiffFlux_ );
```

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU Performance

$$\nabla \cdot (\lambda \nabla T)$$

```
phi <<= divX( -interpX(lambda) * gradX(T) )
      + divY( -interpY(lambda) * gradY(T) )
      + divZ( -interpZ(lambda) * gradZ(T) );
```

	2	4	6	8	10	12	GPU
64x64x64	1.3	2.4	2.6	3.3	3.3	3.3	13.8
128x128x128	1.9	3.9	4.9	6.8	7.8	6.0	26.0

$$-\nabla \cdot (\rho \phi \mathbf{u} + \mathbf{J}_\phi)$$

```
rhs <<= -divOpX_( xConvFlux_ + xDiffFlux_ )
        -divOpY_( yConvFlux_ + yDiffFlux_ )
        -divOpZ_( zConvFlux_ + zDiffFlux_ );
```

	2	4	6	8	10	12	GPU
64x64x64	1.8	2.9	2.9	3.4	3.6	3.6	16.3
128x128x128	2.0	3.6	5.0	6.5	6.1	4.8	13.5

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU - pointwise calculations

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

Multicore & GPU - pointwise calculations

```
delG <<= 16.0 * PI / 3.0 * molecularVolume * molecularVolume *
        surfaceEng * surfaceEng * surfaceEng / (KB * KB *
        temperature * temperature * log(superSat) *
        log(superSat)) + KB * temperature * log(superSat) –
        surfaceEng * pow(36.0 * PI * molecularVolume *
        molecularVolume, 1.0 / 3.0);

iC <<= 32.0 * PI / 3.0 * molecularVolume * molecularVolume *
      surfaceEng * surfaceEng * surfaceEng / (KB * KB * KB *
      temperature * temperature * temperature * log(superSat) *
      log(superSat) * log(superSat));

z <<= sqrt( delG / (3.0 * PI * KB * temperature * iC * iC));

kF <<= diffusionCoef * pow(48.0 * PI * PI * molecularVolume * iC, 1.0 / 3.0);

N1 <<= NA * eqConc * superSat;

result <<= cond(superSat > 1.0,  z * kF * N1 * N1 * exp( – delG / KB / temperature))
                    (0.0);
```

	2	4	6	8	10	12	GPU
64x64x64	1.8	2.3	3.6	4.9	5.9	5.5	27.3
128x128x128	1.9	3.6	5.2	6.0	7.3	6.6	37.4

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

(E)DSL Parting Thoughts

Institute for CLEAN AND SECURE ENERGY
THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING
COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

(E)DSL Parting Thoughts

- Hierarchical parallelization is key for scalability
  - *Domain decomposition (SIMD)*
    - should allow a node 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.

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

(E)DSL Parting Thoughts

- Hierarchical parallelization is key for scalability
  - *Domain decomposition (SIMD)*
    - should allow a node 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*

Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

(E)DSL Parting Thoughts

- Hierarchical parallelization is key for scalability
  - *Domain decomposition (SIMD)*
    - should allow a node 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***
  - *EDSLs allow **seamless transition** of a code base*
  - *template metaprogramming pushes work from run-time to compile-time for more efficiency*

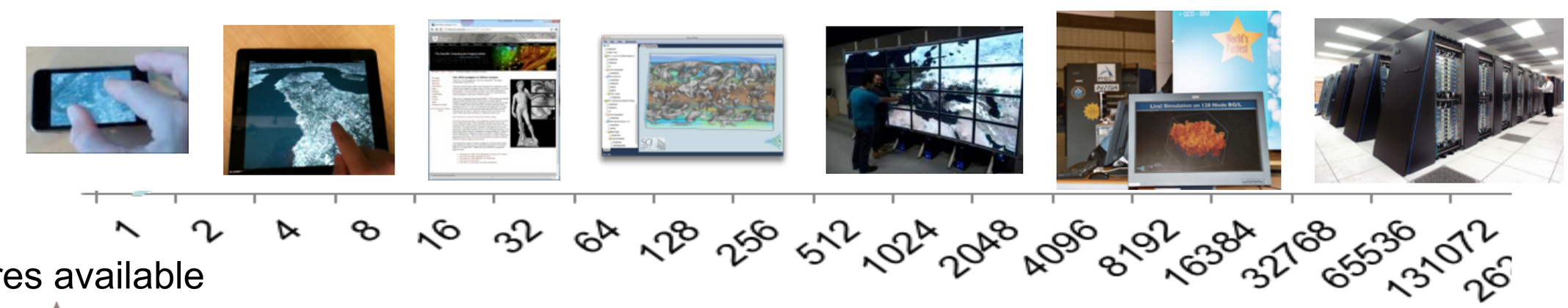
Institute for CLEAN AND SECURE ENERGY | THE UNIVERSITY OF UTAH
DEPARTMENT OF CHEMICAL ENGINEERING | COLLEGE OF ENGINEERING | THE UNIVERSITY OF UTAH

We build our data management solution on most advanced technology in big data streaming analytics and visualization

Live demonstration at SC12 & 13:

- ~4TB per time step (100s of PF 3D timesteps generated on Intrepid)
- Steaming live from ANL visualization cluster
- Interactive, immersive, analysis and visualization

Infrastructure that scales gracefully with available hardware resources

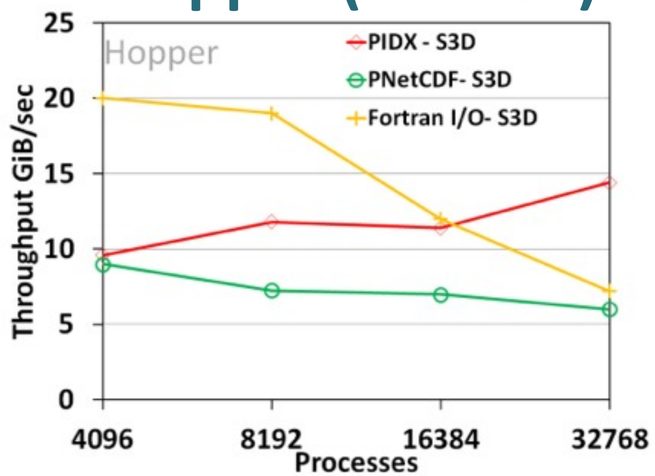


Cores available

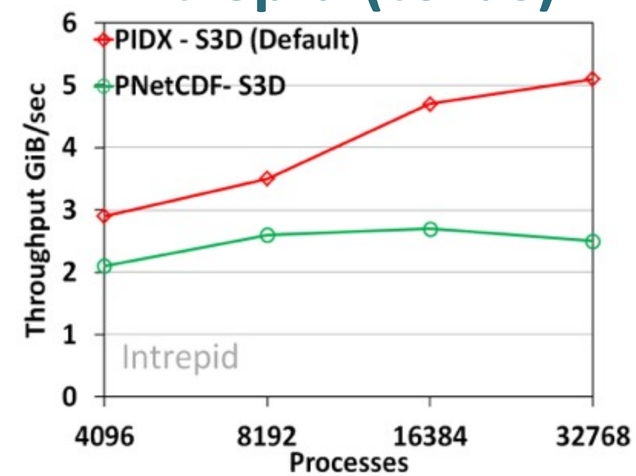
CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

High Performance Data Movements for Real-Time Monitoring of Large Scale Simulations

Hopper (fat tree)

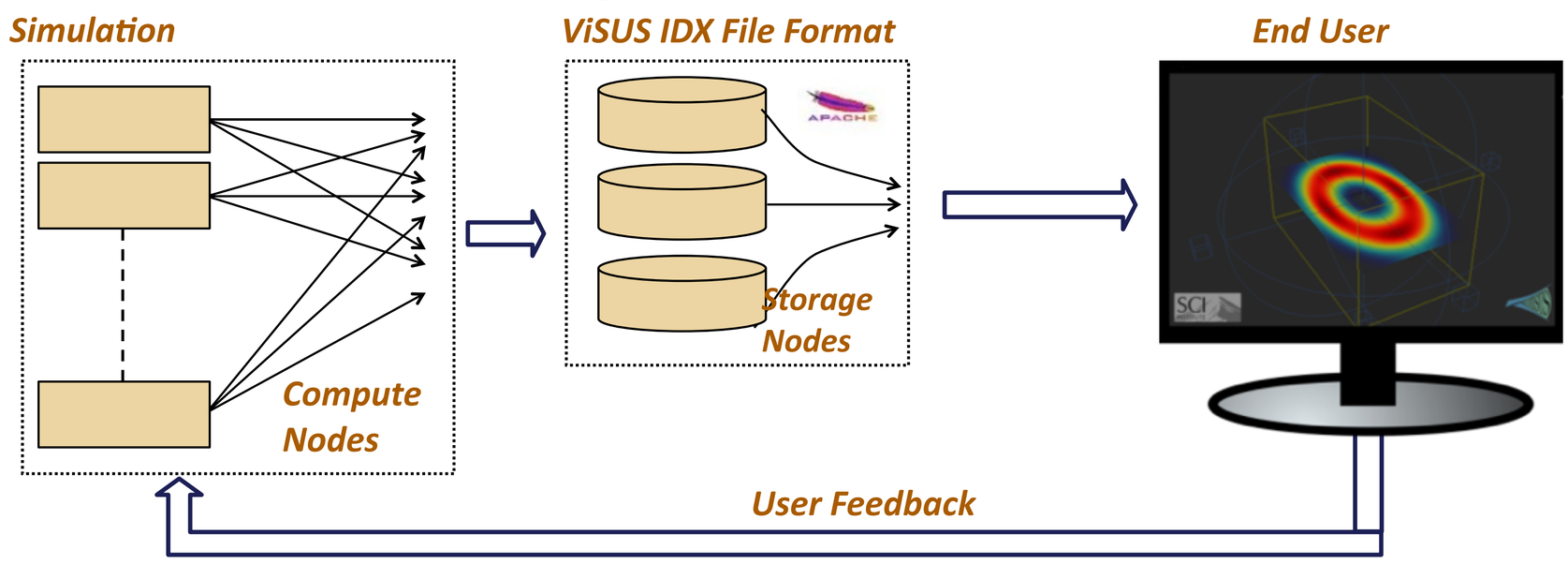


Intrepid (torus)



Scale simulation dumps to 130K cores with better performance than state of the art libraries while enabling real-time, remote visualization

Real-time Data Analysis and Remote Visualization

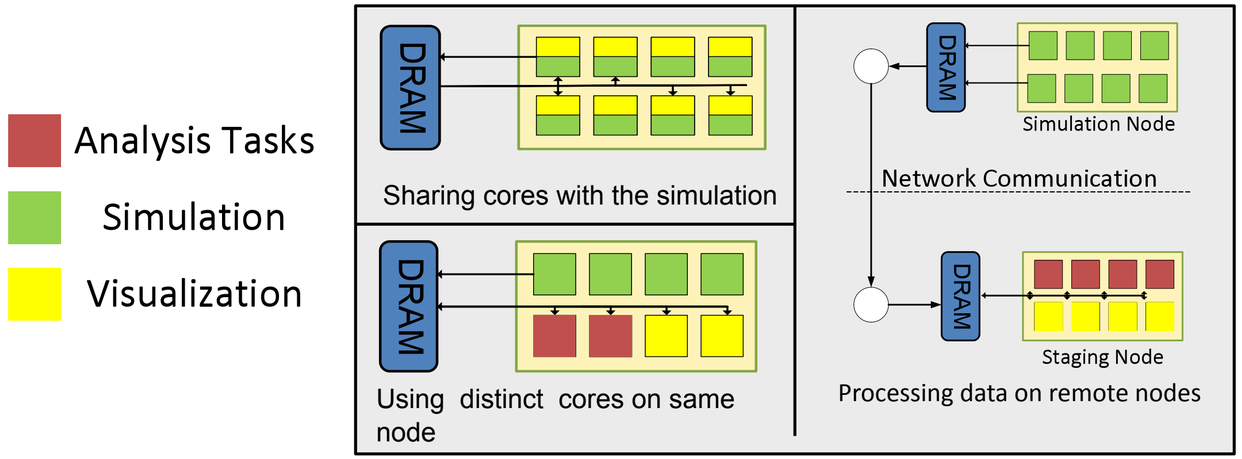


CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

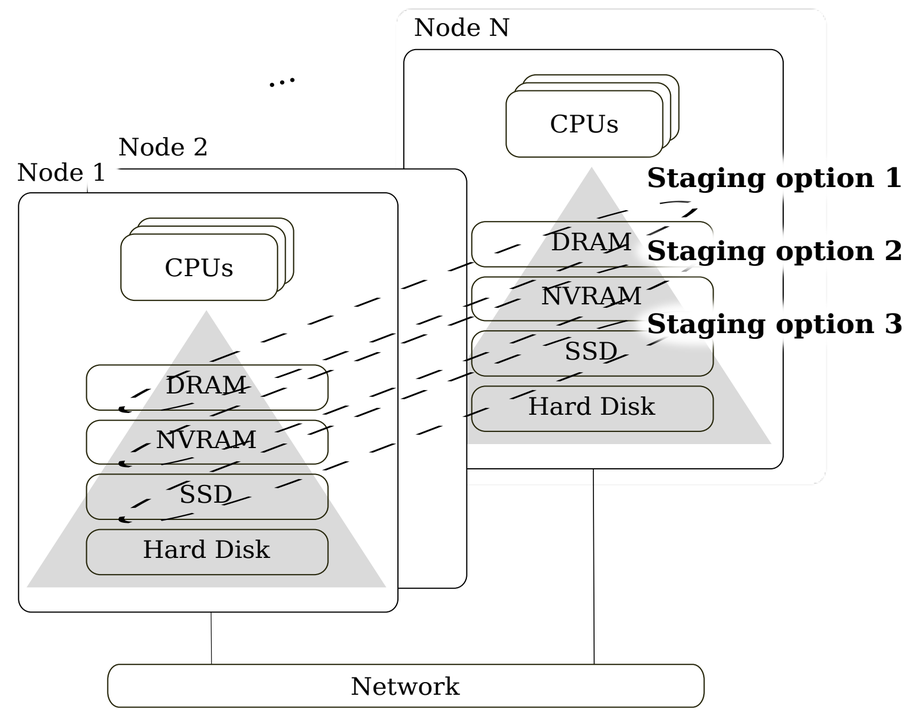
[SC12a] **Efficient Data Restructuring and Aggregation for IO Acceleration in PIDX**

Flexible DMAV Architecture that Allows Exploiting the "Possible" Exascale Hardware Available

- **Location of the compute resources**
  - –Same cores as the simulation
  - –Dedicated cores
  - –Dedicated nodes
  - –External resource



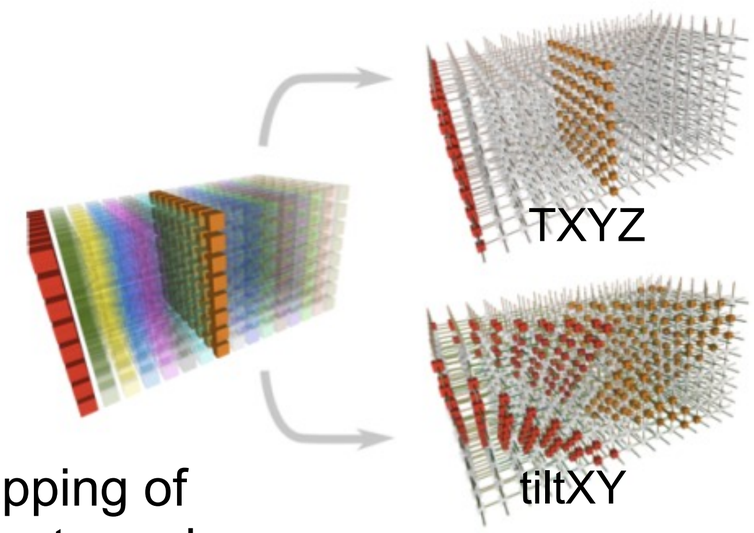
- **Data access, placement, and persistence**
  - –Shared data structures
  - –Shared memory
  - –Non-volatile on node storage (NVRAM)
  - –Network transfer



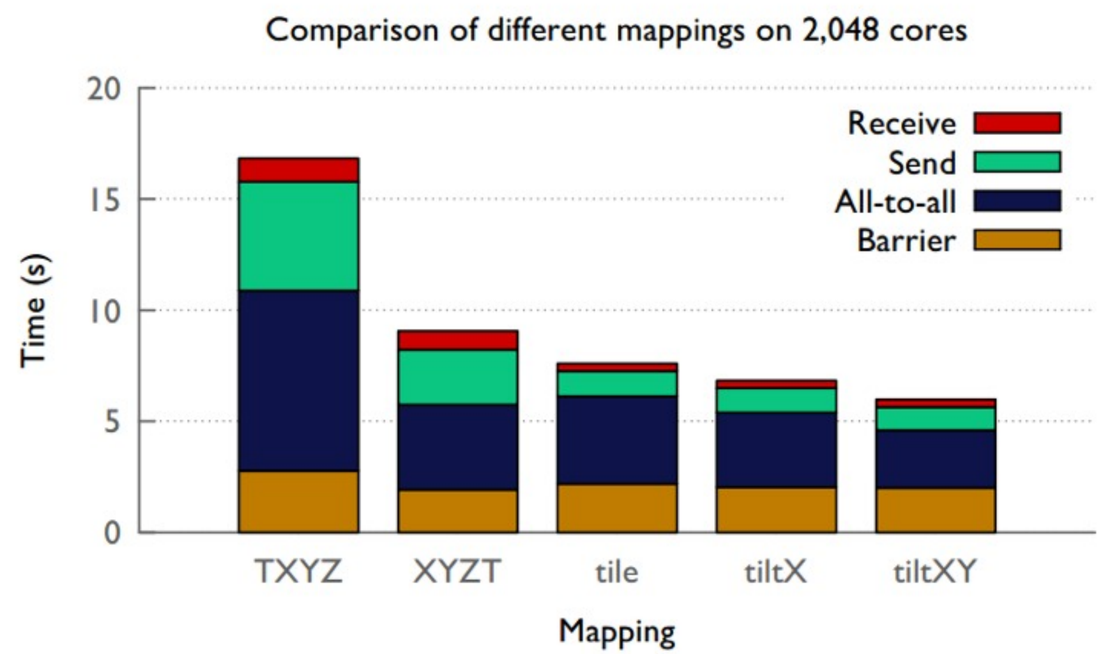
- **Synchronization and scheduling**
  - –Execute synchronously with simulation every n^{th} simulation time step
  - –Execute asynchronously

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

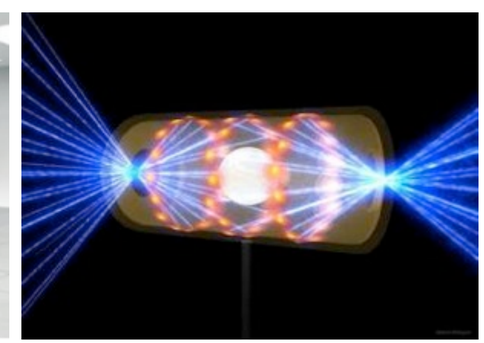
Systematic Exploration of the Efficiency and Cost of Data Movements for In-situ Data Structures



Different mapping of data structure to nodes



Sequoia (LLNL)



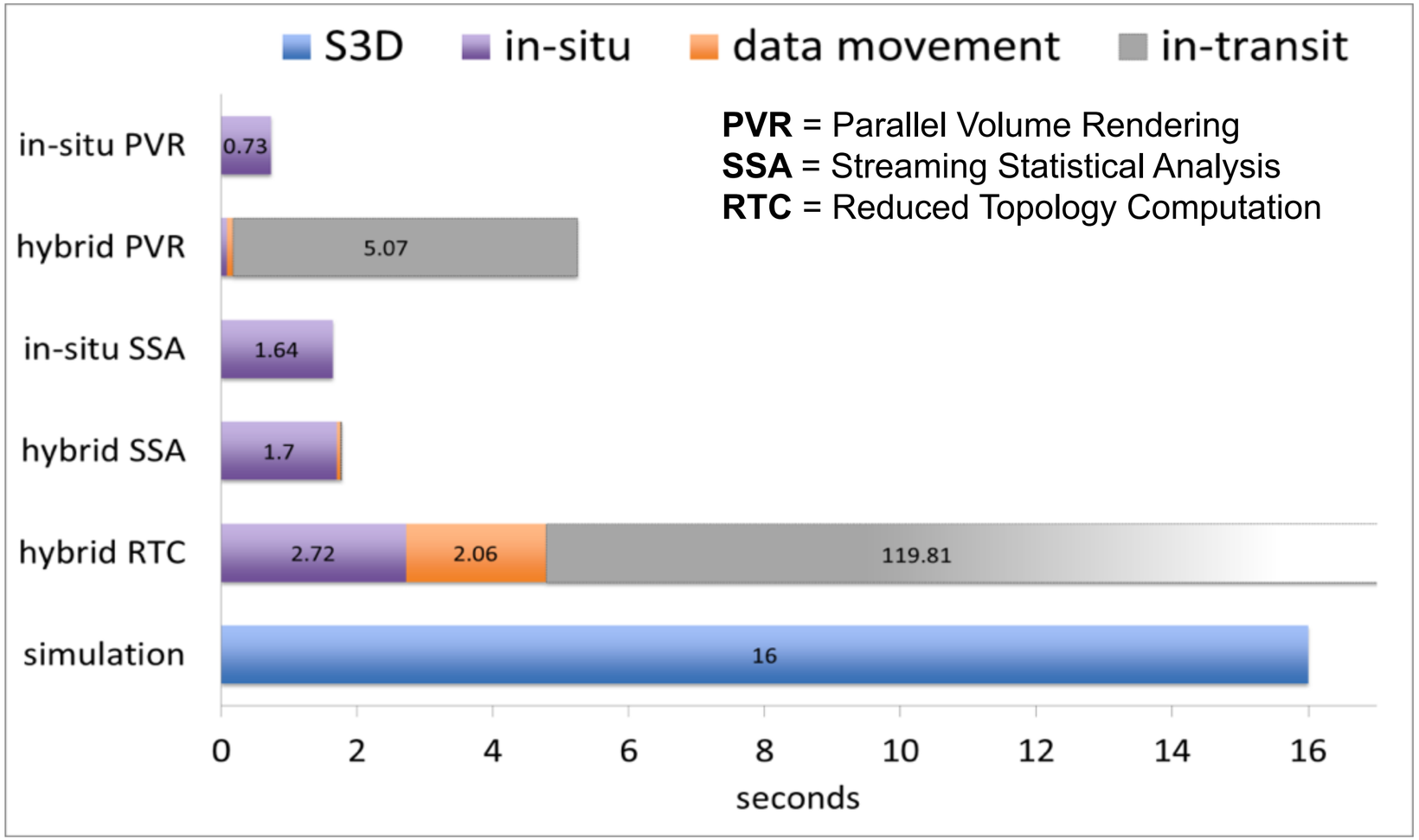
PF3D Simulation

[SC12a] **Mapping Applications with Collectives over Sub-Communicators on Torus Networks**

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

For Analysis and Visualization Algorithms We Determine the Effectiveness of In-Situ Task Allocations

in-situ + in-transit workflows
enable matching algorithms with available architectures



- 4896 cores total (4480 simulation/in situ; 256 in transit; 160 task scheduling/data movement)
- Simulation size: 1600x1372x430 ; All measurements are per simulation time step

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

[SC12b] **Combining In-Situ and In-Transit Processing to Enable Extreme-Scale Scientific Analysis**

Study of High Dimensional Space of Input, Tunable, and Output Parameters on For (auto)Tuning

Independent Variables (Input)	Parameters (Tunable)	Dependent Variables (Output)
n: Initial set of nodes	**n': nodes Participating in Aggregation**	Network throughput
V: Variable count	**A: aggregator count**	I/O throughput
D: Data size/resolution	**a_f: aggregation factor**	Combined throughput
M : Machine specs/memory	**F: File counts**	

Intrepid: BlueGene/P
Peak performance :
557 Tera-flops/sec
RAM :
2GiB / Node (4 cores)
File System: GPFS
Batch/Job Allocation :
Contiguous allocation of nodes

Hopper: Cray XE6
Peak performance:
1.28 Peta-flops/sec
RAM:
36GiB/Node (24 cores)
File System : Lustre
Batch/Job Allocation: Non-contiguous allocation of nodes

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

[SC13a] **Characterization and Modeling of PIDX Parallel I/O for Performance Optimization**

Study of High Dimensional Space of Input, Tunable, and Output Parameters on For (auto)Tuning

Independent Variables (Input)	Parameters (Tunable)	Dependent Variables (Output)
n: Initial set of nodes	**n': nodes Participating in Aggregation**	Network throughput
V: Variable count	**A: aggregator count**	I/O throughput
D: Data size/resolution	**a_f: aggregation factor**	Combined throughput
M : Machine specs/memory	**F: File counts**	

Intrepid: BlueGene/P
Peak performance :
557 Tera-flops/sec
RAM :
2GiB / Node (4 cores)
File System: GPFS
Batch/Job Allocation :
Contiguous allocation of nodes

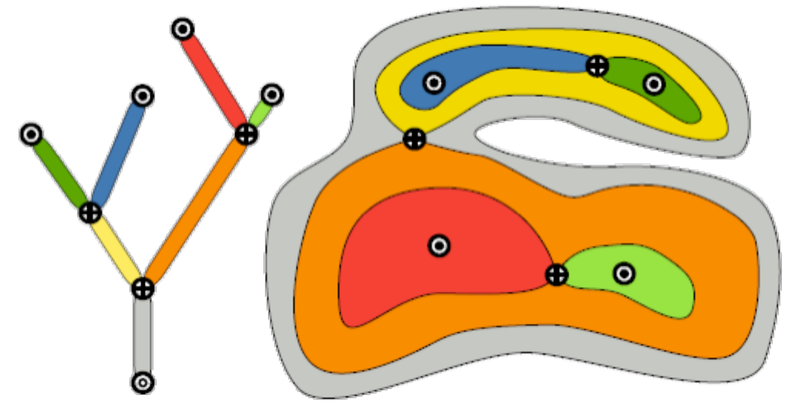
Hopper: Cray XE6
Peak performance:
1.28 Peta-flops/sec
RAM:
36GiB/Node (24 cores)
File System : Lustre
Batch/Job Allocation: Non-contiguous allocation of nodes

CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

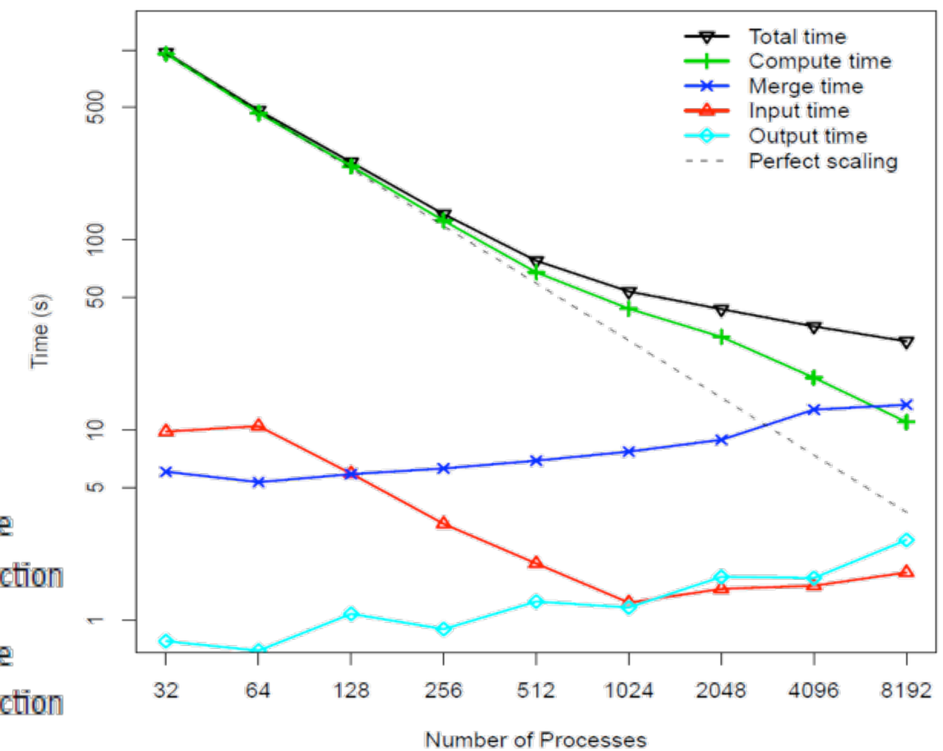
[SC13a] **Characterization and Modeling of PIDX Parallel I/O for Performance Optimization**

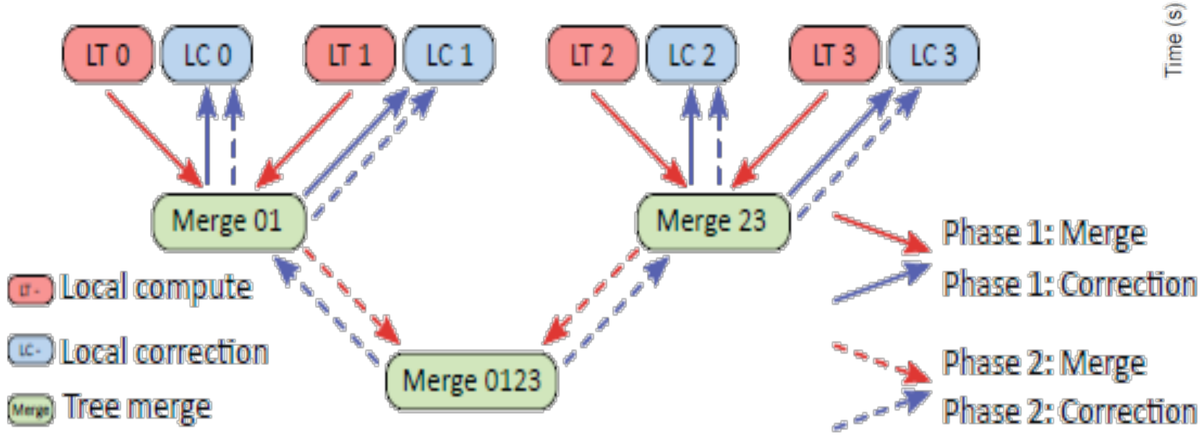
We Evaluate the Feasibility of Exascale *In-Situ* Feature Extraction and Tracking and its Impact on Power Consumption

- Analysis algorithms vs simulation:
  - different communication/instruction profile/memory access patterns
  - different communication patterns
- Develop power models with machine independent characteristics (Byfl and MPI)
- Validate power model using empirical studies with instrumented platforms
- Extrapolate to Titan



Total & Component Time For Jet Mixture Fraction

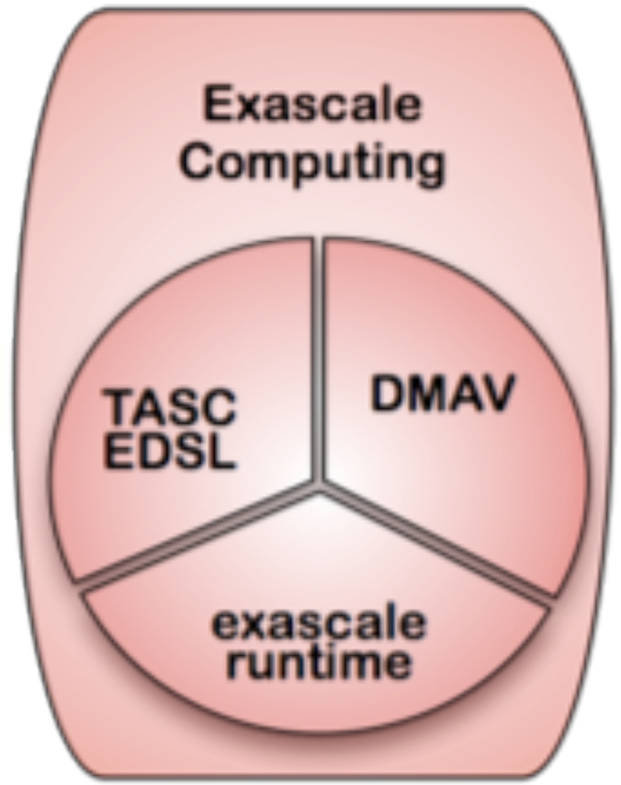




CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

[SC13b] **Exploring Power Behaviors and Trade-offs of In-situ Data Analytics**

UINTAH-X A Software Approach that leads towards Exascale



CARBON CAPTURE MULTIDISCIPLINARY SIMULATION CENTER

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