An Applications-Driven path from Terascale to Exascale (?) with the Asynchronous Many Task (AMT) Uintah Framework

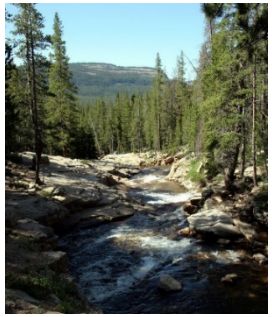
Martin Berzins
Scientific Imaging and Computing Institute
University of Utah

1. Foundations: ASCI and Uintah, runtimes and programming models
2. ASC(I) Impacts: apps codes, visualization, runtime, institutional
3. ASC Impact: Moving Uintah to AMT – a detonation example
4. PSAAP2: large scale combustion with radiation
5. Moving to Exascale - Kokkos AMTs and a feasibility study on the Sunway TiahuLight

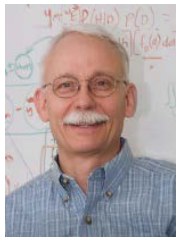
Good design decisions made over the last 20 years make possible exascale?

Uintah Background and Acknowledgements

DOE NSF People

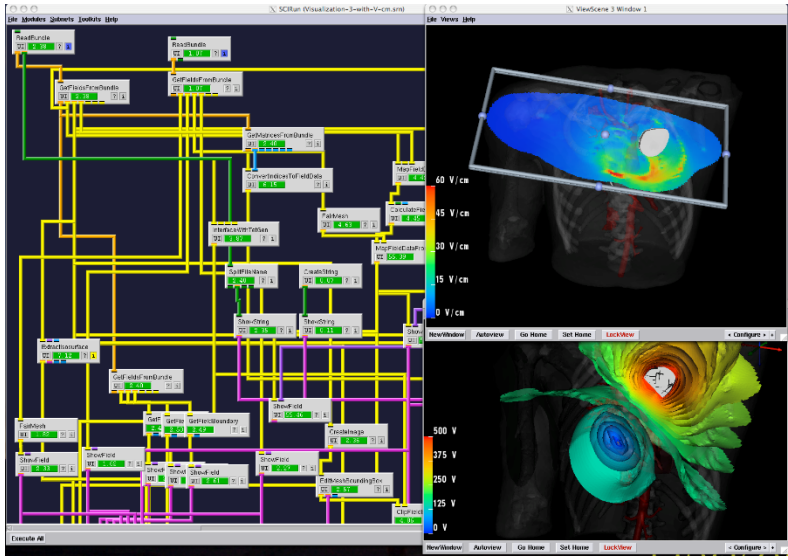


- DOE ASC Strategic Academic Alliance Program 1998 -2010
- ALCC and Directors Discretionary time awards
- INCITE (4 awards 700M cpu hours in total)
- Argonne , Oak Ridge and NNSA Facilities
- **NNSA PSAAP2 center funding 2014-2020**
- Argonne A21 exascale early science program
- Sandia Kokkos group and Livermore Hypre Group
- NSF software funding and Peta-Apps 2007- 2015
- NSF XSEDE TACC Blue Waters computer time and facilities
- The 50 or so people on Uintah and its related projects, since 2003 particularly The Uintah "wizards" Steve Parker, Justin Luitjens, Qingyu Meng and Alan Humphrey .
- NNSA PSAAP2 Co PIs Dave Pershing, Phil Smith Valerio Pascucci



Uintah development timeline

Task Based approach by Steve Parker
Originated in SCIRun problem solving (workflow) environment for large scale biomedical problems .
Simple programming model- separation Physics and Computer Science
Developed independently of Charm++ and Sarkar.



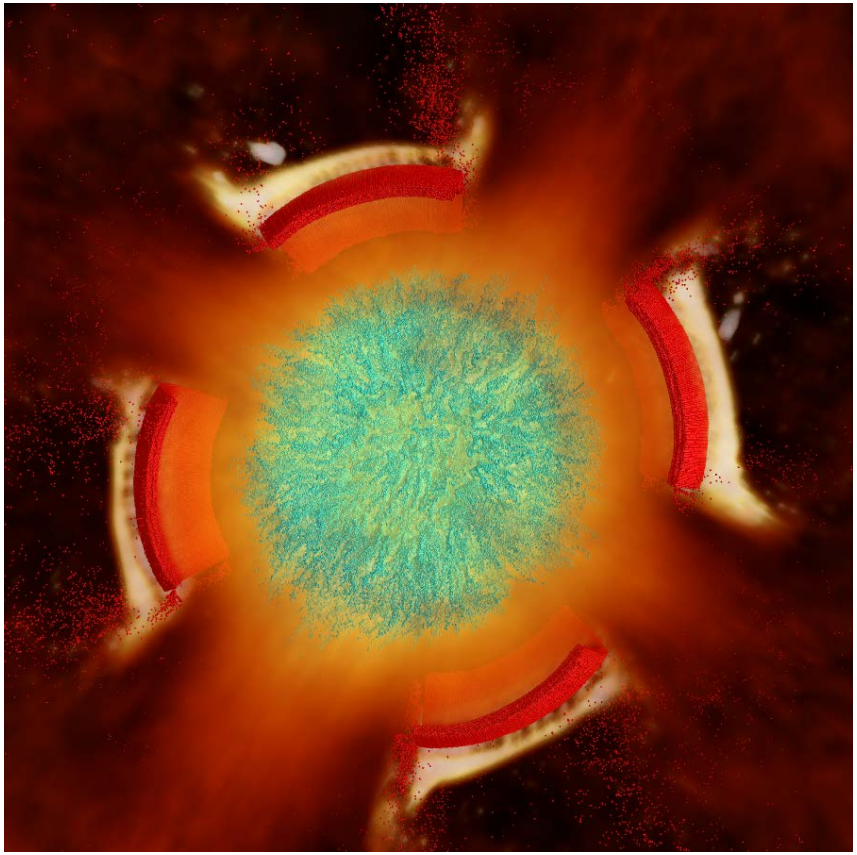
- 1998-2007 CSAFE ASCI Center- static execution of task graphs, complex multiphysics . Steve goes to NVIDIA ☹.
- 2008-2010 CSAFE Full physics, AMR for fluid-structure
- 2010-2015 Adaptive asynchronous. out-of-order task execution
- 2014- PSAAP2 Center Turbulent Combustion - full scalability on Titan Mira, Blue Waters – moving to exascale portability?

THE UNIVERSITY OF UTAH

CSAFE: Uintah Arches MPM-ICE Algorithm

- ICE is a cell-centered finite volume method for Navier Stokes equations extended version of LANL work.
- MPM Material Point Method is a PIC-like method for solids that uses particles and a background mesh. Disney snow in “frozen”
- Arches is an LES combustion code

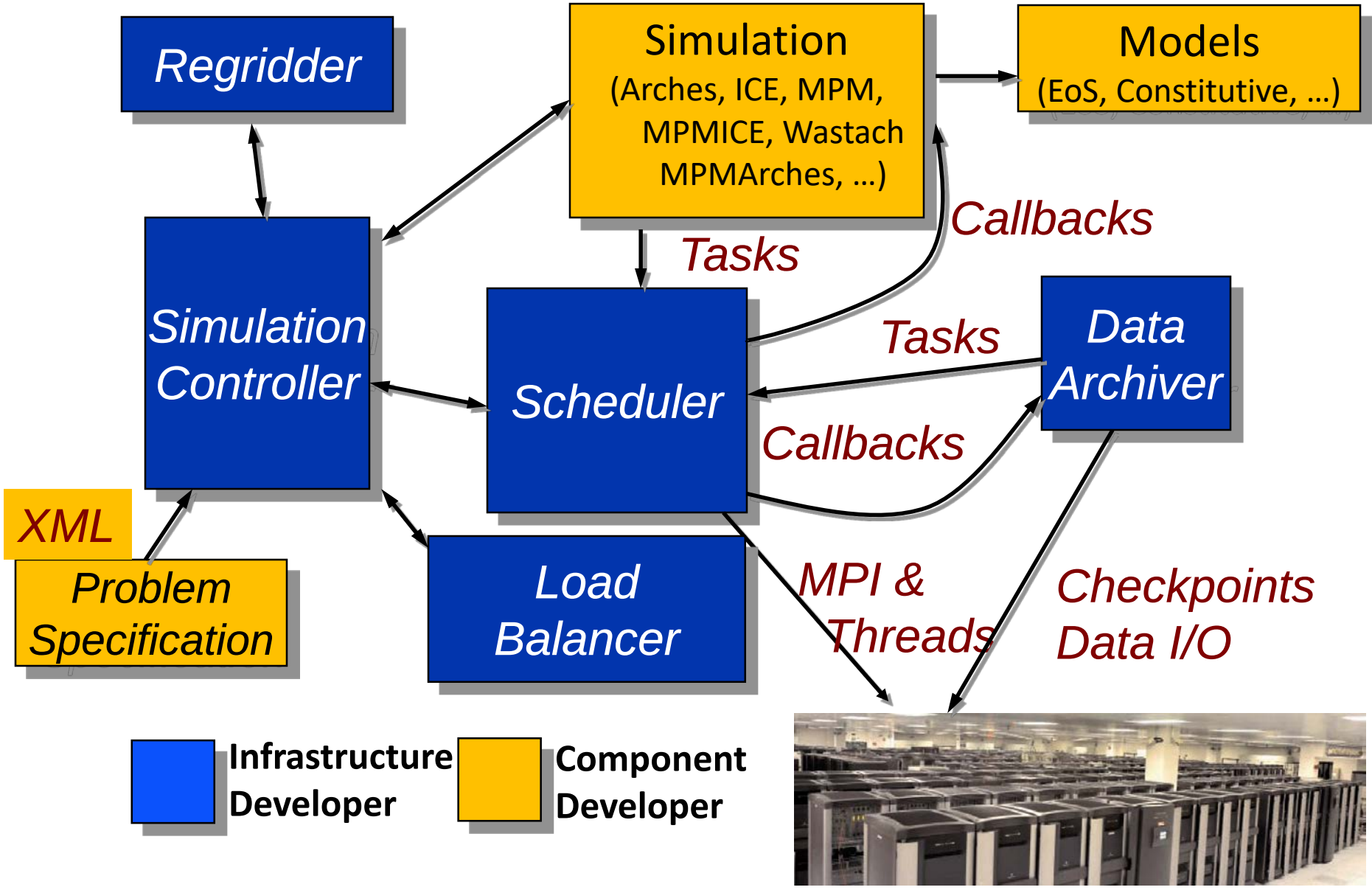
Uintah was originally designed for simulation of fires and explosions,



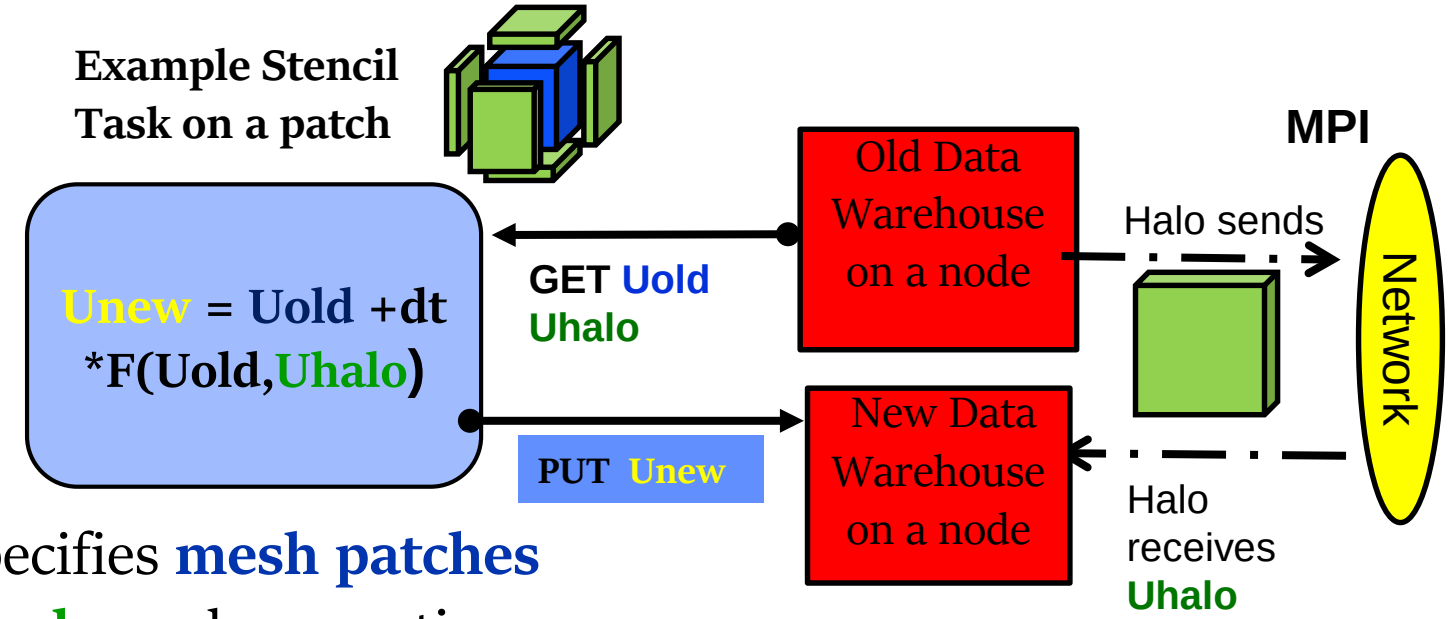
Container with PBX explosive

THE UNIVERSITY OF UTAH

Uintah Components Static Task-Based Approach

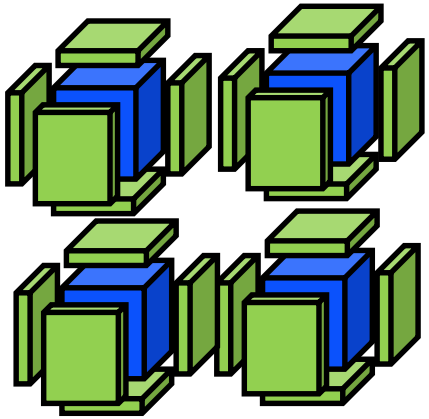


Uintah Programing Model for Stencil Timestep [Parker 1998]



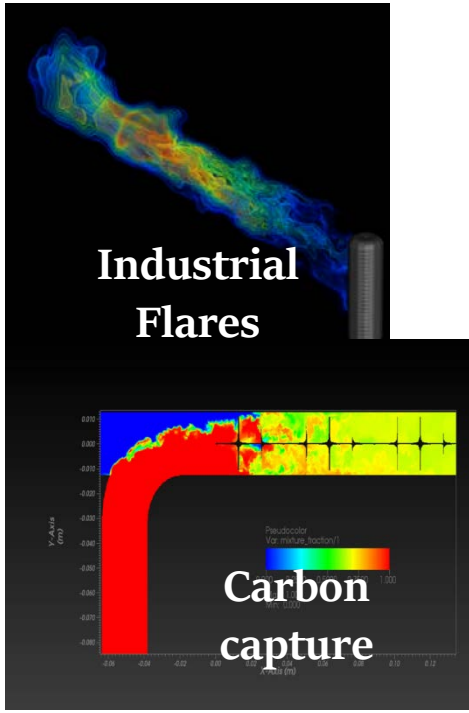
User specifies **mesh patches**
halo levels and connections

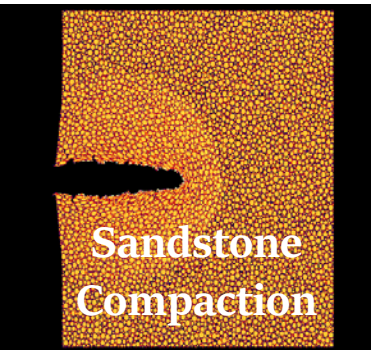
Clean separation between physics and CS runtime system
Problems specified 15 years ago run on todays architectures with one significant change as we go to exascale

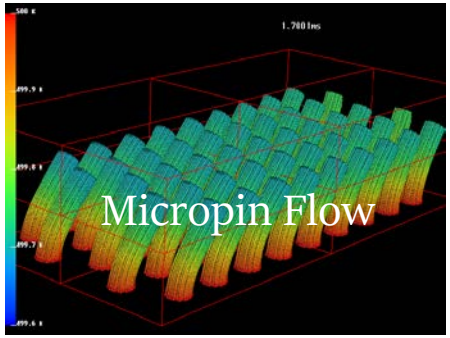


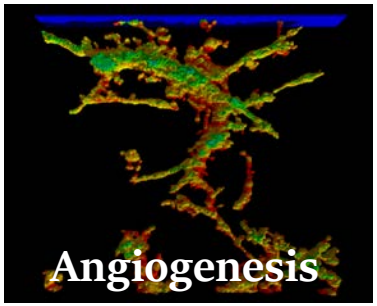
Uintah Apps Codes Example Impacts

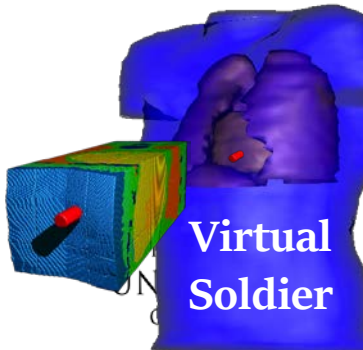
- **Arches:** industrial flares John Zink, ultra Low Nox: Chevron Fives, CO2 mineralization Calera Corp, LES with REI consulting, Mitsubushi Heavy Industries low Nox, General Electric Boilers + many universities. Radiation and LES models
- **ICE:** semiconductor devices, flow over cities, accidental detonations, turbulence , reactive models Air Force
- **MPM:** fundamental analysis. Army Research Lab Center in Materials Modeling (Utah leads) , novel battery models with silicon, penetration and fracture models for oil industry , Darpa heart injuries, angiogenesis.





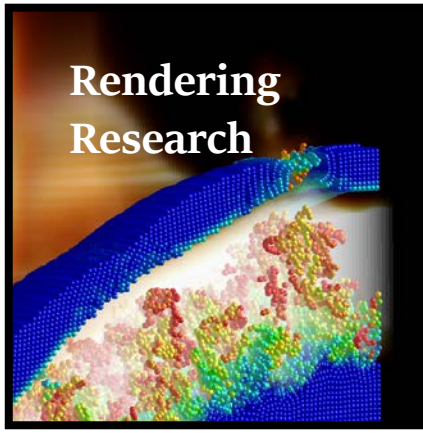


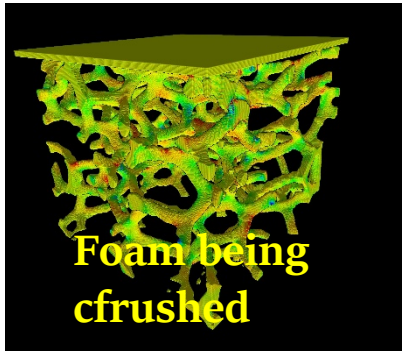


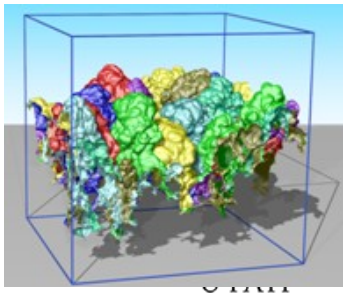


CS, Visualization, Utah Institutional and Lab Impacts

- Utah - three large multidisciplinary institutes SCI, ICSE, EGI. ASC and related funding is a key enabler of research and the bedrock of SCI and ICSE
- ASC has enabled the visualization and graphics research that has helped SCI become the highest ranked visualization research activity.
- Research in software and Asynchronous Many Task Runtimes has made possible automated parallel solution of complex engineering problems
- SCI led by Chris Johnson has grown to 200 people with 20 faculty with flow of people to DOE.
- Labs: Continued movement of SCI and ICSE research, software, students and post-docs to Labs.





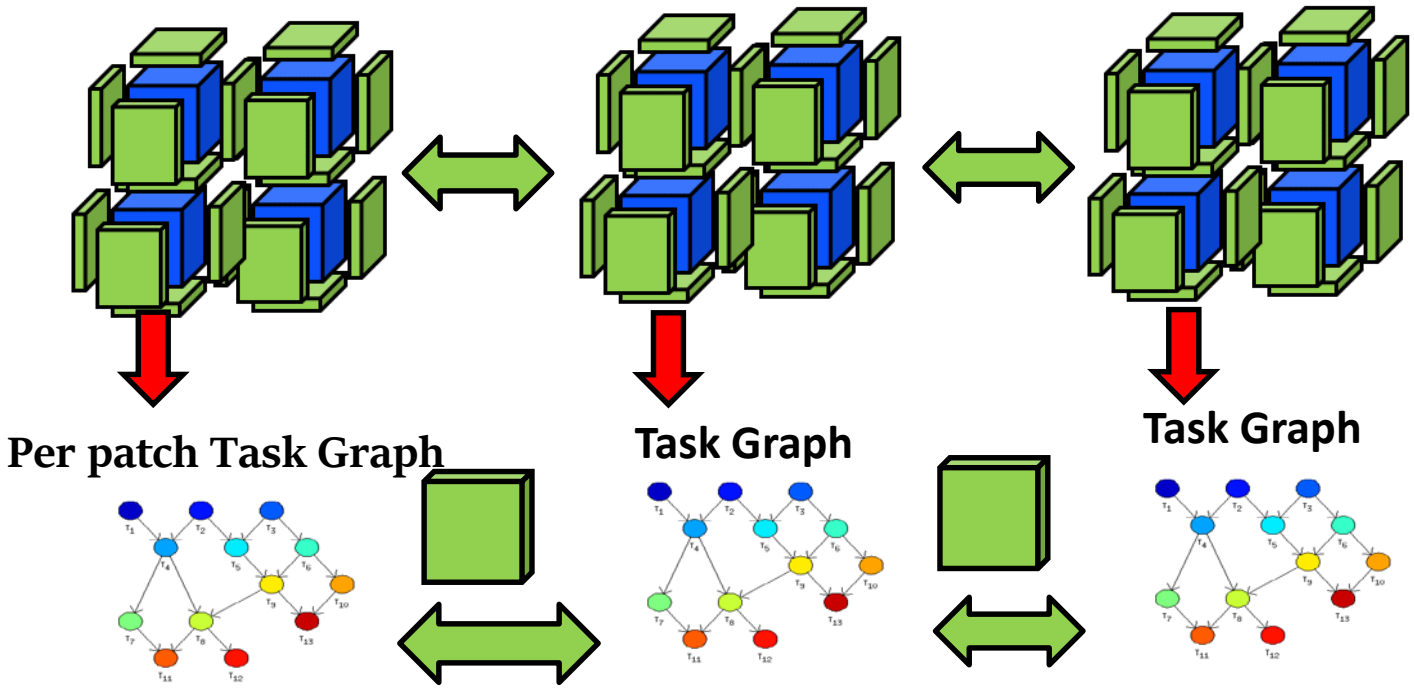


Topology of Combustion

Uintah Asynchronous Many Task (AMT) Approach 2008...

In Uintah dynamic task graph execution needed for more than 100K cores

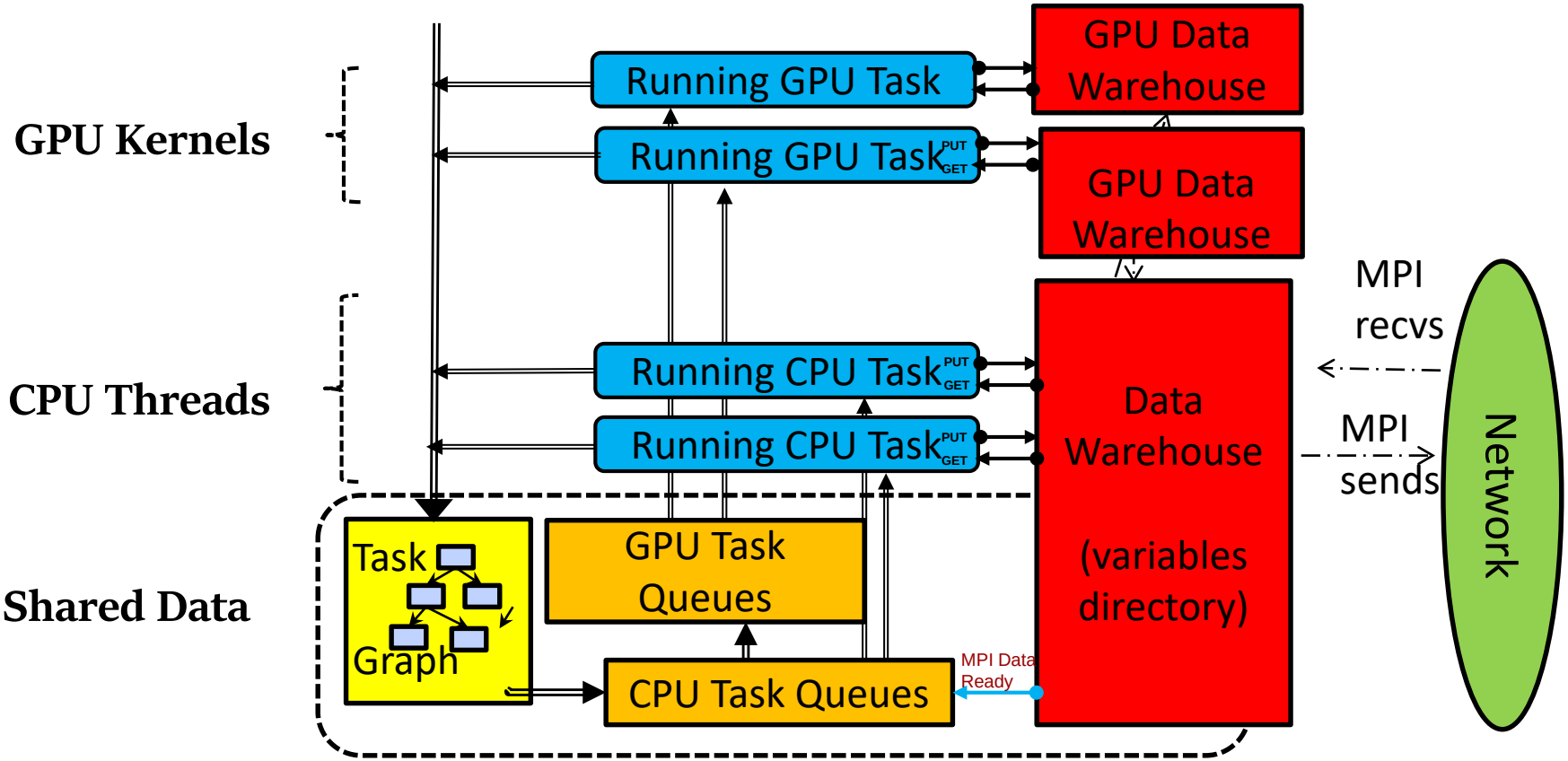
e.g. three compute nodes 12 mesh patches



Execute tasks when possible communicating
as needed. Do useful work instead of waiting
Execute tasks out of order if possible

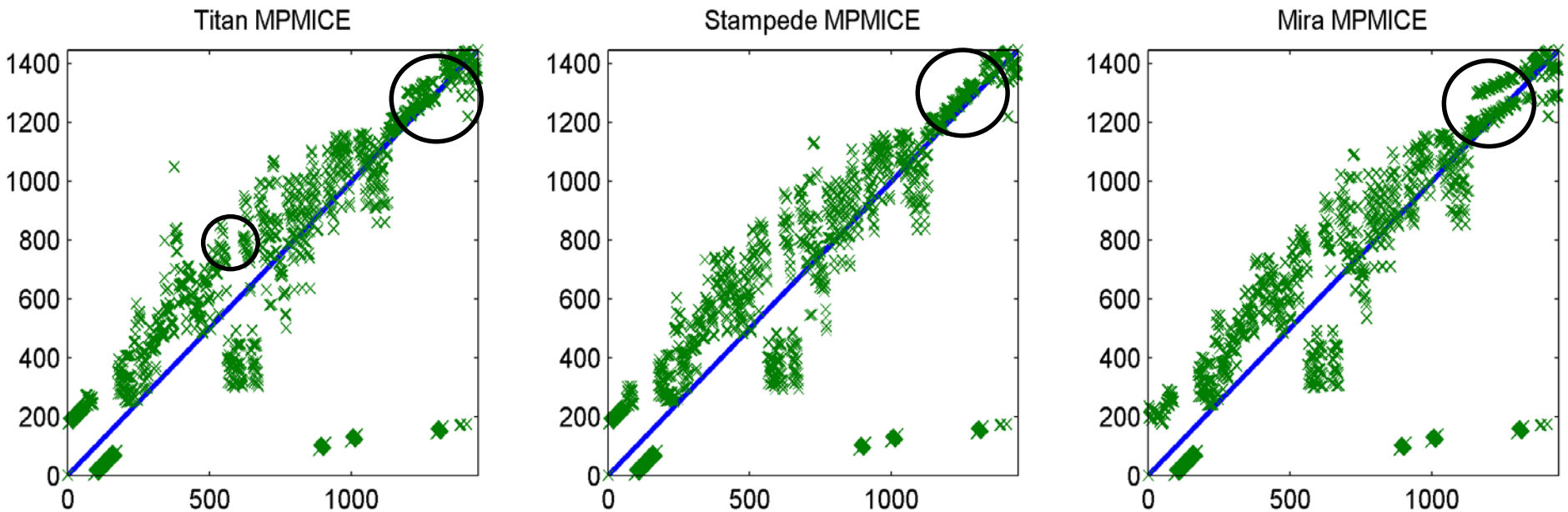
Uintah: Node Architecture

Unified Heterogeneous Scheduler [Meng, Luitjens 2012], more recent versions for KNL GPUs etc



CPU cores and GPUs pull work from task queues

Scalability is at least partially achieved by not executing tasks in order e.g.



Straight line represents given order of tasks . 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

Spanish Fork Accident 8/10/05 NSF PetaApps

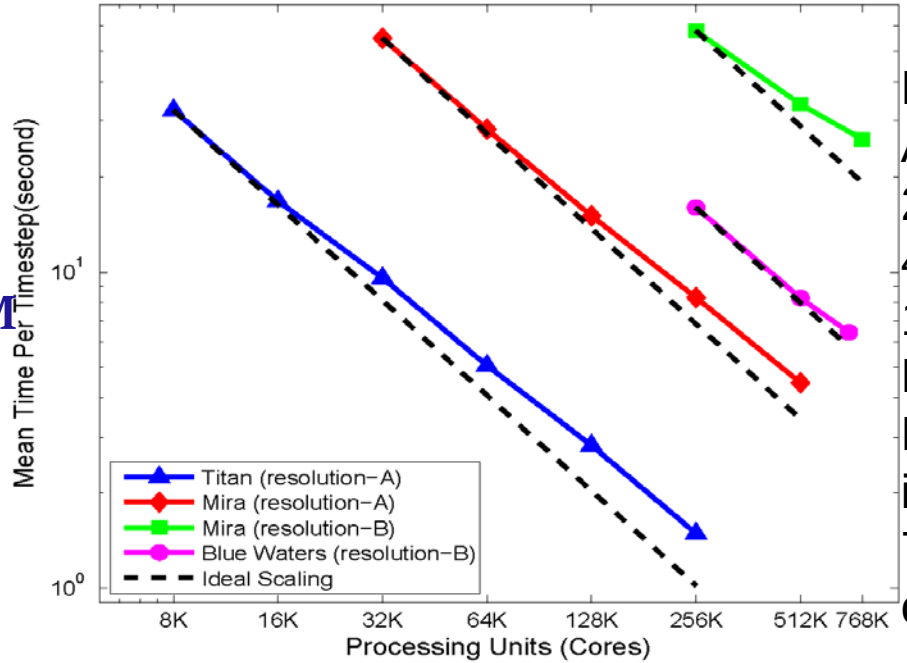
Speeding truck with 8000 explosive boosters each with 2.5-5.5 lbs of explosive overturned and caught fire

Experimental evidence for a transition from deflagration to detonation?

Deflagration wave moves at ~400m/s not all explosive consumed. Detonation wave moves 8500m/s all explosive consumed.

Solution required about 100M cpu hours. Job started on Titan moved to Mira and finished on Stampede.

Much media coverage



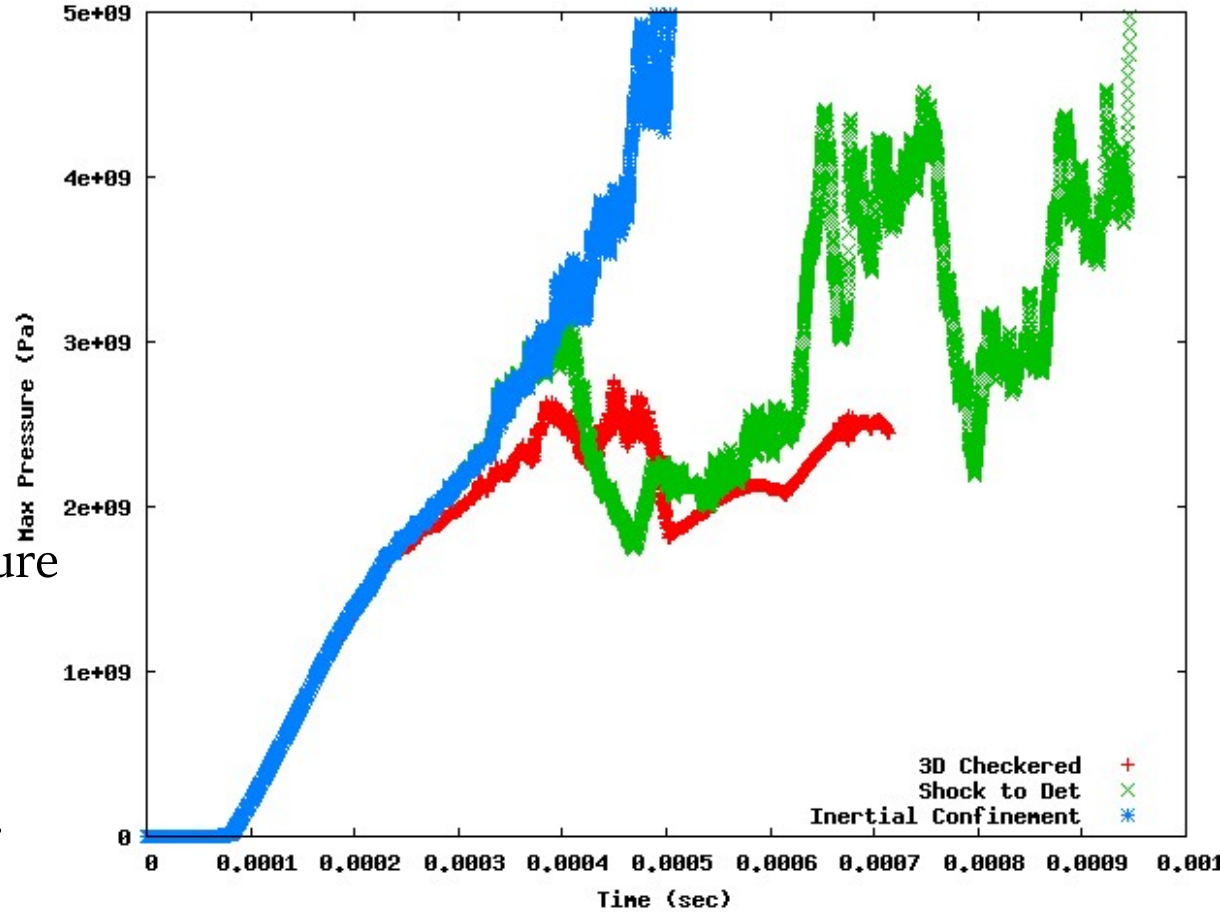
Particle Mesh AMR
29Bn Particles
4Bn mesh Cells
1.2M mesh Patches
Fluid-structure interaction
700-880K cores

Mechanisms of DDT

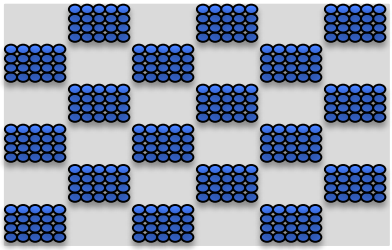
Inertial confinement –

Cylinders closely packed and compact together to form a high density barrier – stops product gases expanding increasing the localized pressure detonation point (5.3 GPa).

Shock to detonation
when cylinders further apart. Cylinders burn for long time (elevated pressure around 1GPa) before being impacted by pressure or deflagrating particles impacting the cylinder.



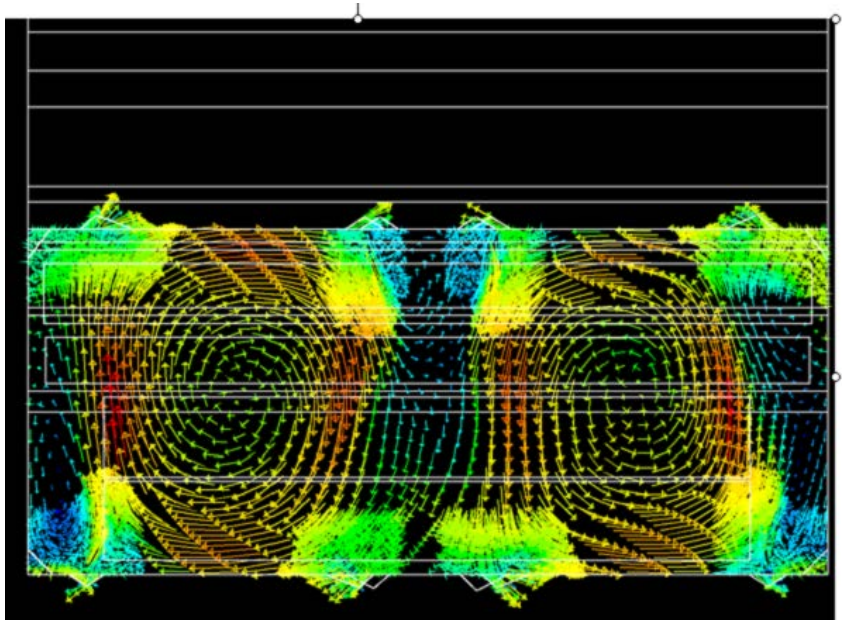
Alternate packing density leads to lower pressures and explosive being burned without detonation

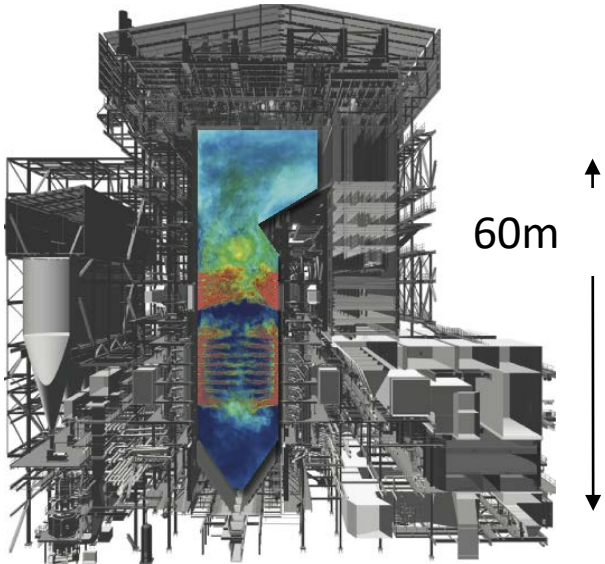


This example drove Uintah Scaling to 800K cores

NNSA PSAAP2 Existing Simulations of GE Clean coal Boilers

- Large scale turbulent combustion needs mm scale grids 10^14 mesh cells 10^15 variables (1000x more than now)
- Structured, high order finite-volume discretization
- Mass, momentum, energy conservation
- LES closure, tabulated chemistry
- PDF mixing models
- DQMOM (many small linear solves)
- Uncertainty quantification

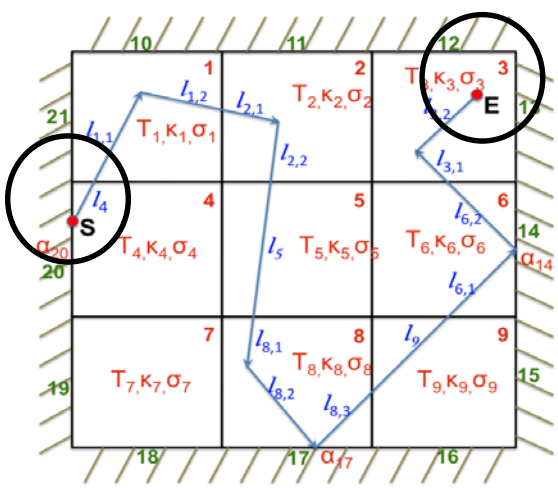


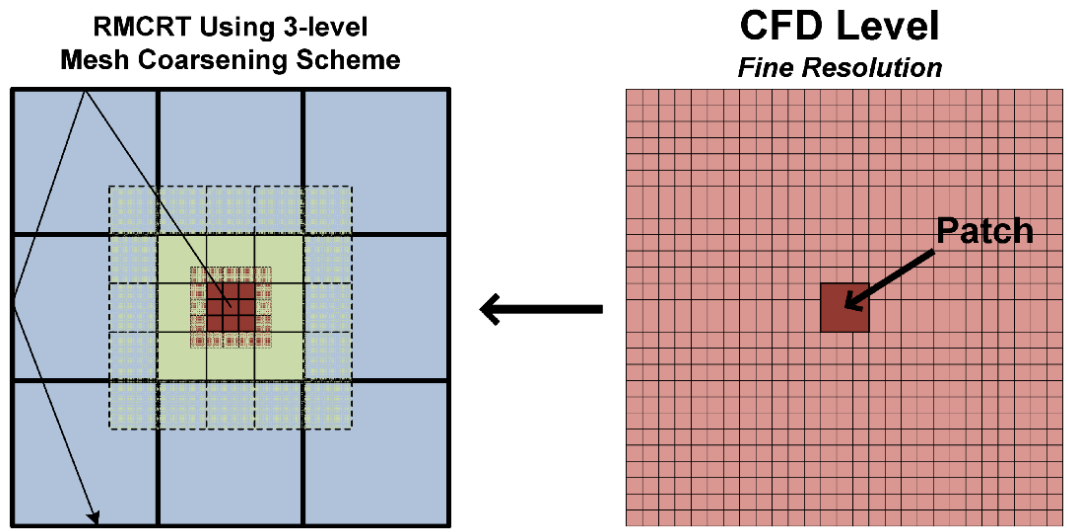


- Low Mach number approx. (pressure Poisson solve up to 10^12 variables. 1M patches 10 B variables
- Radiation via Discrete Ordinates – many hypre solves Mira (cpus) or ray tracing Titan (gpus).
- FAST I/O needed PIDX

Radiation Overview

- Including Radiation means that every one of 10^10 cells may be connected to every other cell
- Model radiation using Monte Carlo ray tracing (RMCRT)
- Replicate AMR versions of the mesh on each node
- Ray trace in parallel
- Radiative properties and radiative fluxes calculated on each node and their **AMR** values transmitted to minimize communication.

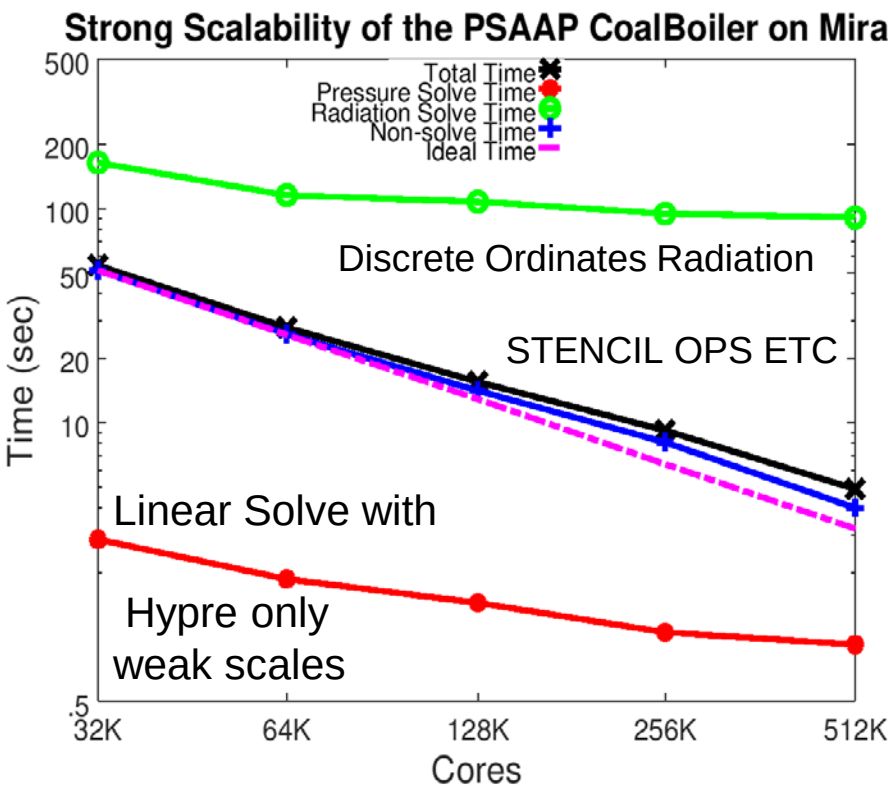


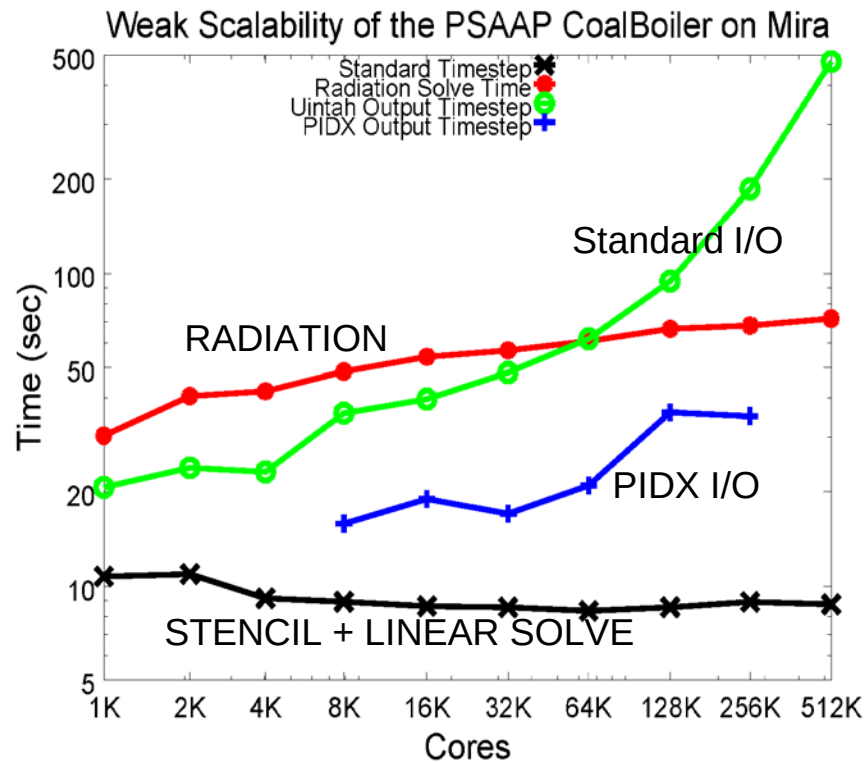


Full physics multi-level GPU-RMCRT scales on Titan

Cores/GPUs	16k/1k	32k/2k	64k/4k	128k/8k	256k/16k
Time (sec)	821.13	407.31	202.69	99.39	55.06

Mira BGQ Strong and Weak Scaling Examples
INCITE runs with full GE Boiler





Porting Uintah to future Exascale architectures

Two main steps

(i) Modify the scheduler to run tasks in a way that takes advantage of the machines in question

Use experience with many different architectures to do this quickly

(ii) Find a way of transforming the loops in a task so that performance portability is possible

Either hand tune loops or use Kokkos DOE Sandia to have portable loops for which it is possible to get performance.

(iii) PDSEC 2018 Uintah paper considers porting to a challenging architecture Sunway TaihuLight . Uintah subset no large apps , open code for runtime.

Kokkos Abstraction for Portability and Node Performance

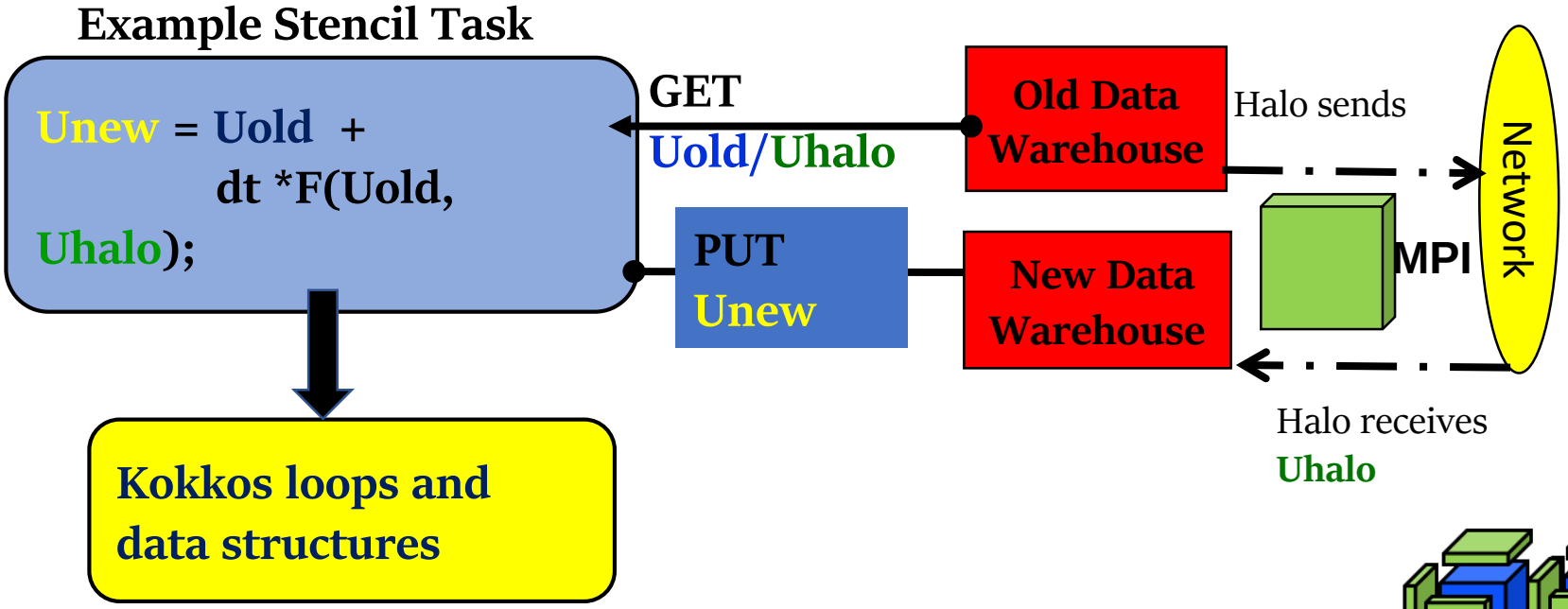
- **Use Kokkos (or Raja) abstraction layer that maps loops onto machine (CPU GPU KNL) efficiently using cache aware memory models and vectorization / Openmp**

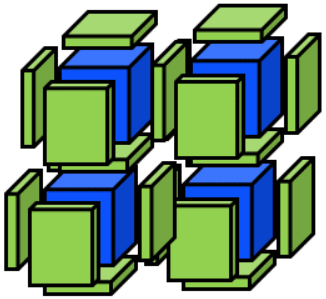
- Uses **C++ template metaprogramming** for compile time data structures and functions and allows vectorization
- **Replace patch grid iterator loops**
- **for** (auto itr = patch.begin(); itr != patch.end(); ++itr) {
  IntVector iv = *itr;
  A[iv] = B[iv] + C[iv];} becomes

- **parallel_for**(patch.range(), LAMBDA(int i, int j, int k) {
  A(i,j,k) = B(i,j,k) + C(i,j,k)});

- **Uintah is a structured mesh use case for Kokkos.**
- **Speedup of 3x on 1 core , 50x on 16 cores , up to 100x on a gpu**

Modified : Uintah Programming Model for Stencil Timestep



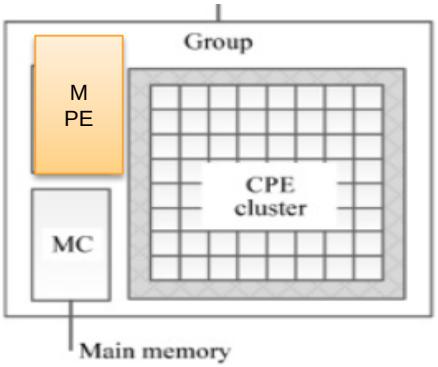


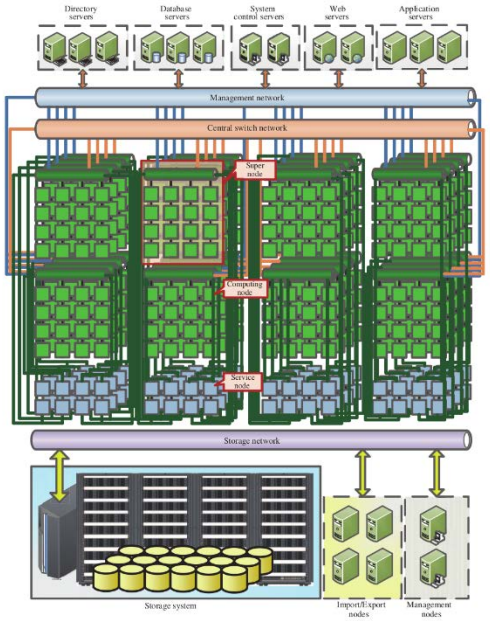
```
Uintah::BlockRange range ( patch->getCellLowIndex(), patch->getCellHighIndex() );
Uintah::parallel_for ( range, [&]( int i,  int j, int k ) {
  char_rate[I,j,k] = 0.0;
  . . .
}
```

Automatically calls non-Kokkos, Kokkos OpenMP or Kokkos cuda, depending on build

Sunway TaihuLight Architecture:

- Each Sunway Compute node contains 4 **core groups (CGs**
- **CG :** 1 Management Processing Element (**MPE)** and **64 CPE** Computing Processing Elements
- **MPE** handles the main control flow / management, communications and computations and shares its memory with.....
- **cpes** are used to perform computations. These can be considered as "coprocessor" used to offload computations. With 256 vector instructions. Cacheless but with shared scratch memory 64K (LDM)
- 10M cores 93PF **vectorization and comms hiding keys to success**.





Source https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/201609/Dongarra-ascac-sunway.pdf

Example PDE moved to Sunway using NSF funding + open source code

Original C++ kernel for 3D Burgers' equation

Fortran kernel vectorized with SIMD intrinsics for d2udz2 calculation

```fortran
VECTOR256 :: v0, v1, v2, v3, v_d2udz2
! d2udz2 = (-2 * u0(i,j,k) +u0(i,j,k-1) &
!           + u0(i,j,k+1)) * (z_dx*z_dx)
v0 = SIMD_CMPLX(-2d0, -2d0, -2d0, -2d0)
call SIMD_LOADU(v1, u0(i,j,k))
call SIMD_LOADU(v2, u0(i,j,k-1))
call SIMD_LOADU(v3, u0(i,j,k+1))
v0 = SIMD_VMAD(v0, v1, v2)
v0 = SIMD_VADDD(v0, v3)
tmp2 = z_dx * z_dx
call SIMD_LOADE(v2, tmp2)
v_d2udz2 = SIMD_VMULD(v0, v2)
```

```
#pragma omp parallel for collapse(2)
    for(k=a; k<b; ++k)
    {
        for(j=c; j<d; ++j)
        {
#pragma simd
            for(i=e; i<f; ++i)
            {
                double px = anc0 + c0 * i, py = anc1 + c1 * j, pz = anc2 + c2 * k;

                double dudx = -1 * (u_data[k][j][i] - u_data[k][j][i - 1]) * initialW(px, currTime) /dx_x;
                double dudy = -1 * (u_data[k][j][i] - u_data[k][j - 1][i]) * initialW(py, currTime) /dx_y;
                double dudz = -1 * (u_data[k][j][i] - u_data[k - 1][j][i]) * initialW(pz, currTime) /dx_z;

                double d2udx2 = (-2 * u_data[k][j][i] + u_data[k][j][i - 1] + u_data[k][j][i + 1]) /(dx_x * dx_x);
                double d2udy2 = (-2 * u_data[k][j][i] + u_data[k][j - 1][i] + u_data[k][j + 1][i]) /(dx_y * dx_y);
                double d2udz2 = (-2 * u_data[k][j][i] + u_data[k - 1][j][i] + u_data[k + 1][j][i]) /(dx_z * dx_z);

                double du = dt * ((dudx + dudy + dudz) + (double)VISCOSITY*(d2udx2 + d2udy2 + d2udz2));

                new_u_data[k][j][i] = u_data[k][j][i] + du;
            }//for(i=ti; i<ti_max; ++i)
        }//for(j=tj; j<tj_max; ++j)
    }//for(k=tk; k<tk_max; ++k)
```

Sunway specific changes

Infrastructure and Scheduler: 200 lines of new code

- Updated offloading and polling mechanism using OpenACC

Computational Kernel / Task: 200 lines of new code

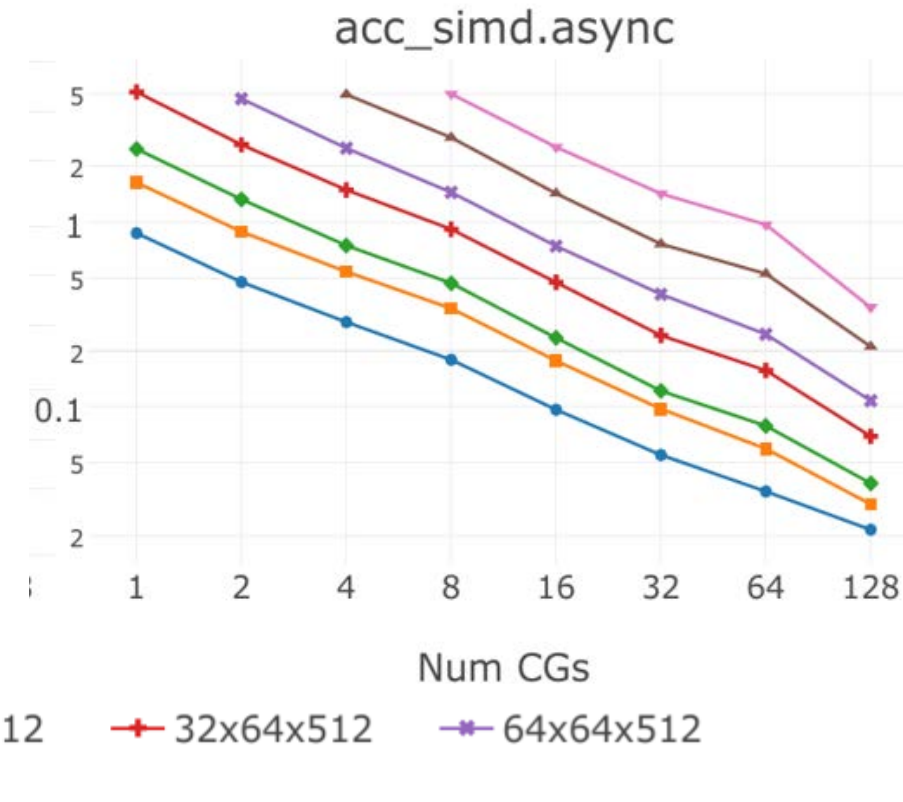
- Porting of Kernel: -main comp. kernel rewritten using Fortran, C, OpenACC and native athread runtime as CPEs do not support C++ low level SIMD instructions
- Need to use athreads low-level SIMD commands to overcome OpenACC slowdowns

Optimizations:

Tiling: The CPE part of scheduler divides tiles among CPEs.

Vectorization: Used native SIMD vector intrinsics for vectorization

Perfect scaling out to 8192 cores on Sunway development queue. IPDPS PDSEC 2018 paper



A Uintah Library for new applications – help DOE NSF and Industry users benefit from Uintah

- **Evolve Uintah**
- Uintah core library + separate application components
- Improve long term maintainability
- Expand the concept of a mesh patch to include unstructured grids and meshless Particle methods,
- Possible new Application directions: Nektar++, QMC, FEM, SPH

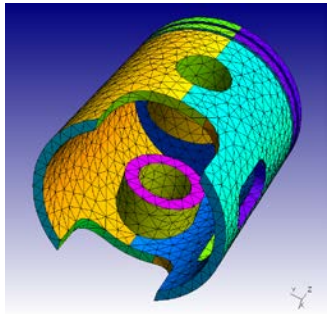
Backwards compatibility – Most Important

Little change to existing components

Portability layers, i.e. Kokkos Existing

Broaden the idea of patches and halos to include:

- Present Structured Grids –Unstructured Grids – NNSA type (combination of multiple different element types)
- Particles for meshless methods – SPH, MD
- Graphs -- potential



NEKTAR++
SPECTRAL/HP ELEMENT FRAMEWORK

Summary

Past and present investments in

I. People
II. good code and algorithm design of
III. a programming model and an
IV. adaptive asynchronous communication-hiding runtime system
V. with a portability layer

Make it possible to:

(i) independently develop complex physics code which is then unchanged
(ii) while scaling complex engineering calculations and
(iii) Using results to drive engineering design
(iv) Provide a viable path to exascale