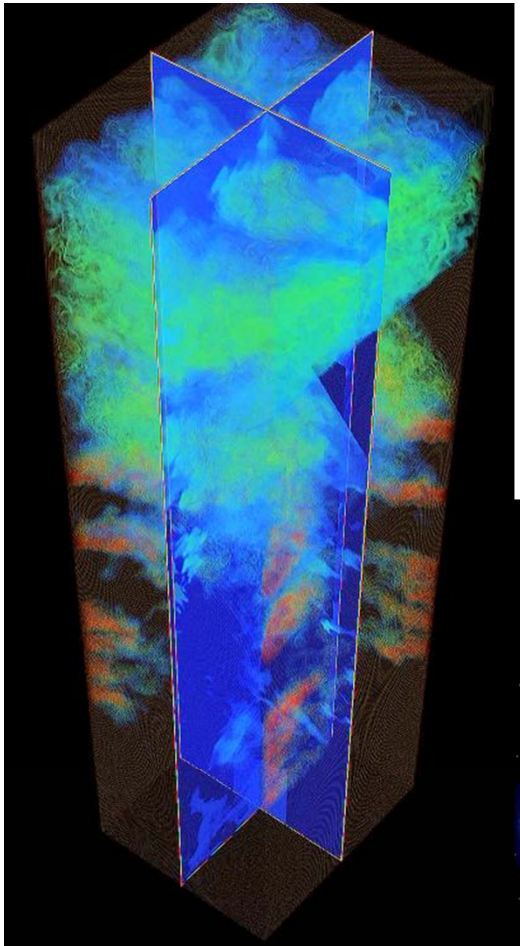
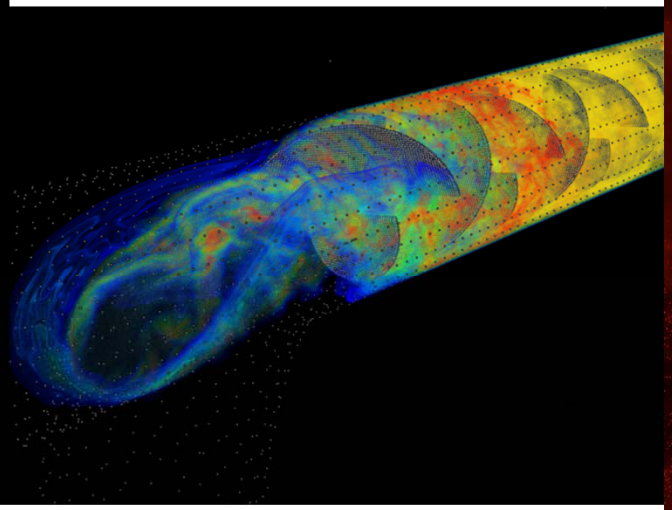
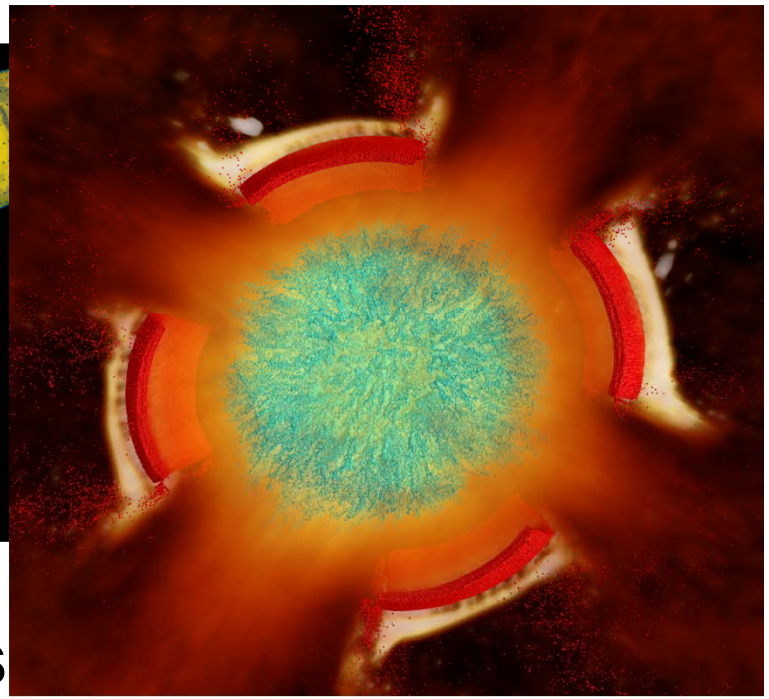
Preliminary Experiences with the Uintah Framework on Intel Xeon Phi and Stampede







Qingyu Meng, Alan Humphrey, John Schmidt, Martin Berzins

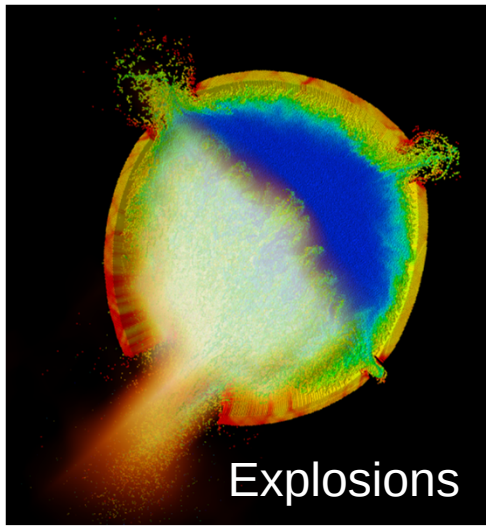
Thanks to: TACC Team for early access to Stampede
J. Davison de St. Germain, Justin Luitjens and Steve Parker

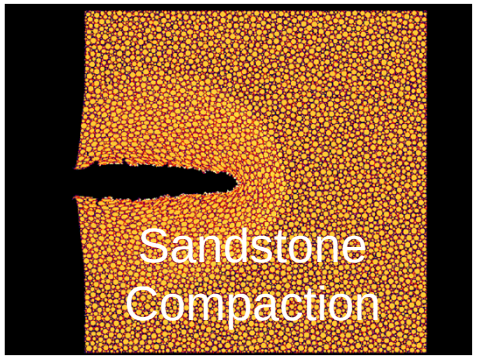
DOE for funding the CSAFE project (97-10), DOE NETL, DOE NNSA
NSF for funding via SDCI and PetaApps

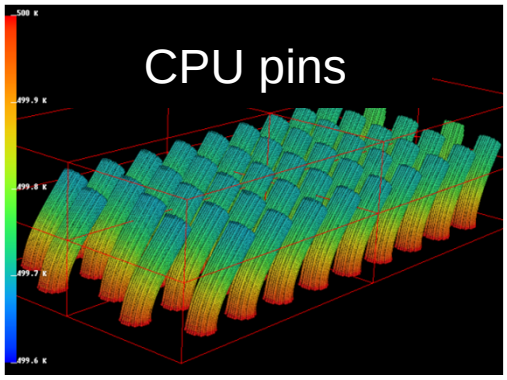
SCI INSTITUTE

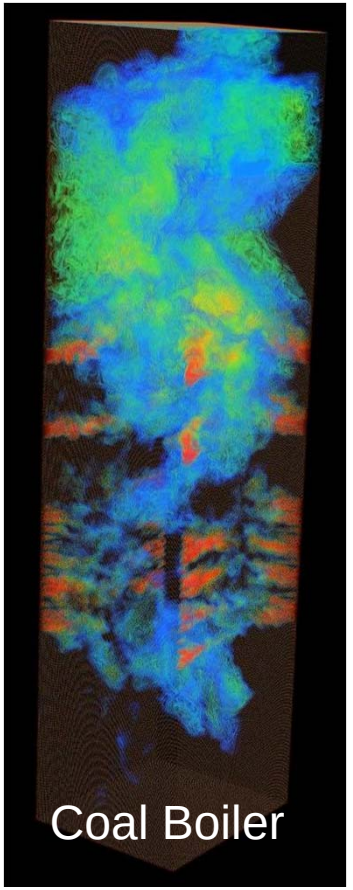
THE UNIVERSITY OF UTAH

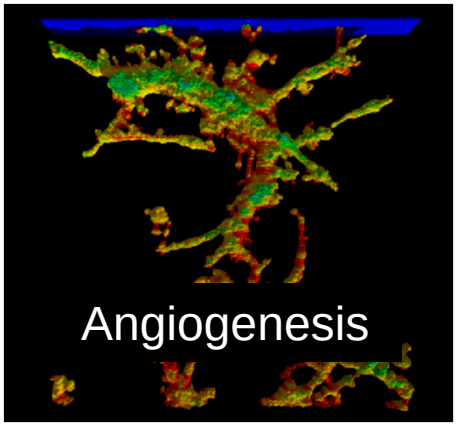
Current and Past Uintah Applications

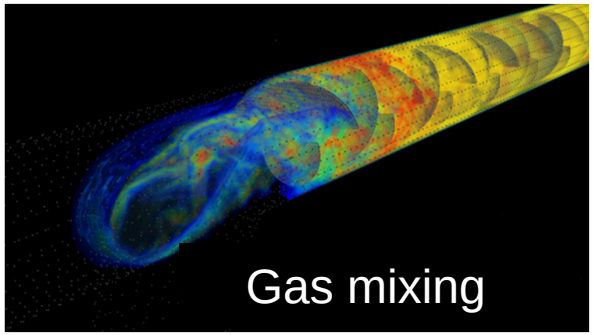




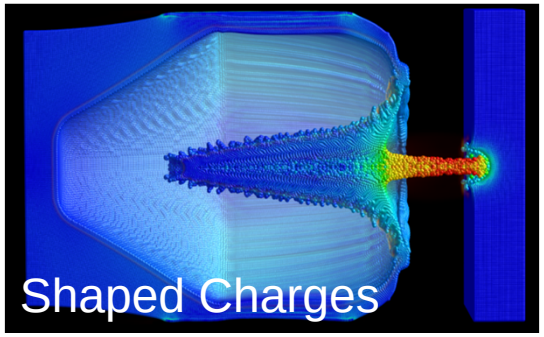


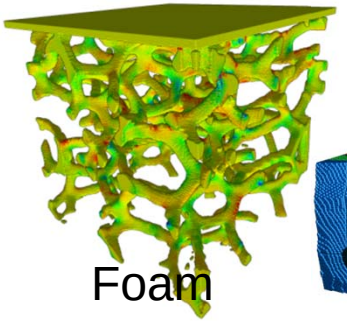




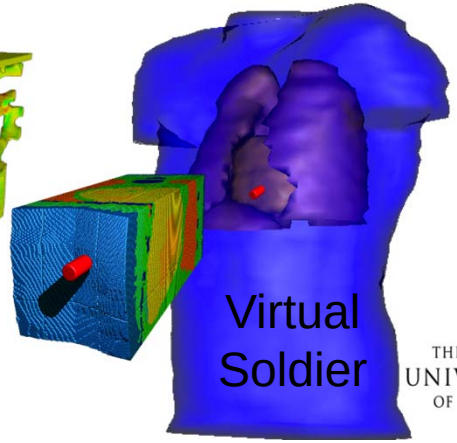


Fires





Foam
Compaction



THE UNIVERSITY OF UTAH

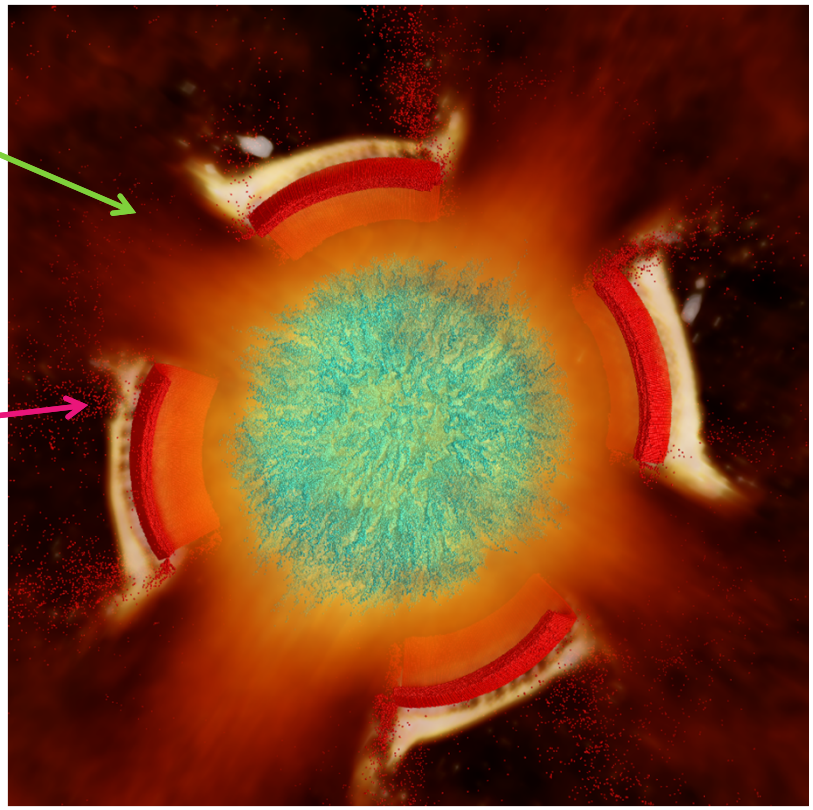
Example: MPM-ICE Algorithm

Metal containers embedded in large hydrocarbon fires.

- ICE is a cell-centered finite volume method for Navier Stokes equations
- ICE now handles fast and slow flows (2009)

- MPM is a novel method that uses particles and nodes
- Cartesian grid used as a common frame of reference

- MPM (solids) and ICE (fluids) exchange data several times per timestep, not just boundary condition exchange.

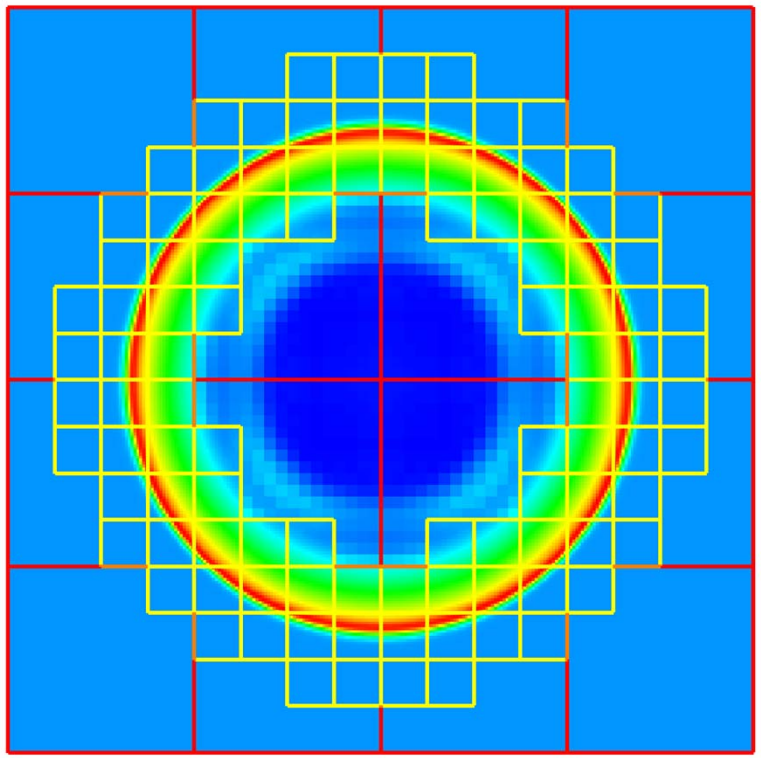


Container with PBX explosive

THE UNIVERSITY OF UTAH

Uintah Parallelism (Data)

Uintah uses both data and task parallelism



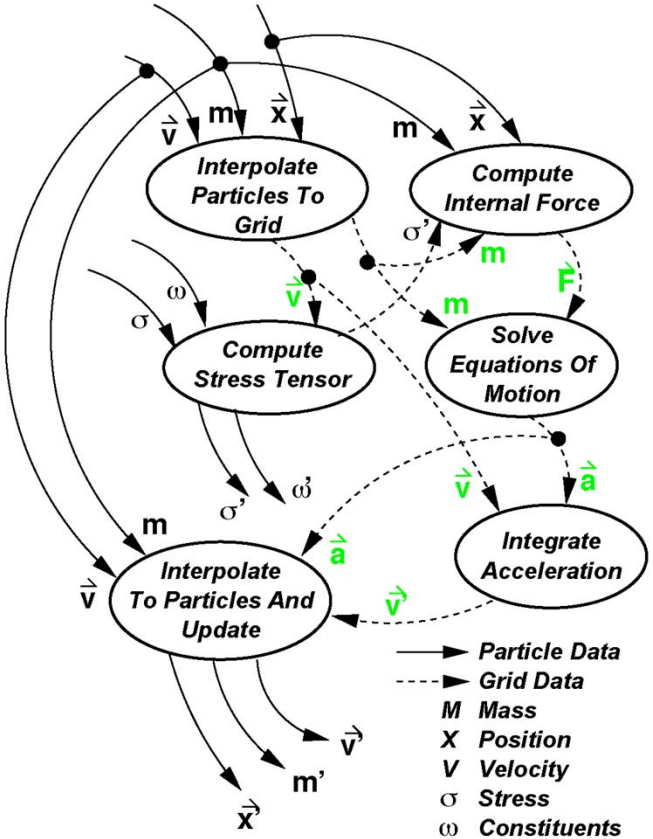
Grid and Patches
(Physical Domain)

- Structured Grid(Flows) + Particles System(Solids)
- Patch-based Domain Decomposition for Parallel Processing
- Adaptive Mesh Refinement
- Dynamic Load Balancing
  - Profiling + Forecasting Model
  - Parallel Space Filling Curves
  - Data Migration

THE UNIVERSITY OF UTAH

Uintah Parallelism (Task)

Uintah uses both data and task parallelism

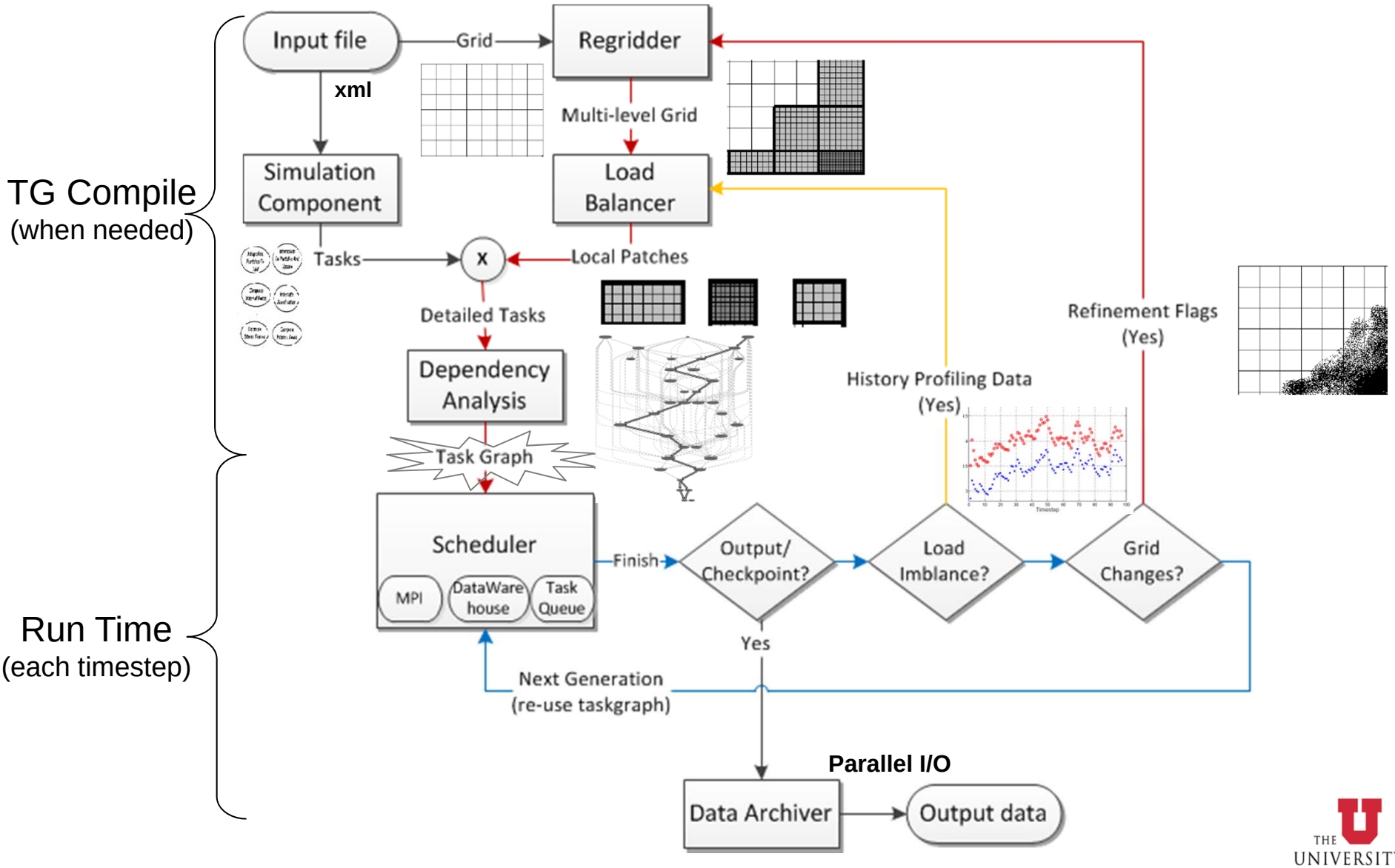


Tasks
(Simulation Methods)

- Uintah Task – serial code w/o MPI/thread on a generic patch
  - Require Variable(s) with ghost cells
  - Compute Variable(s)
  - Call-back Function
- Separation of user code and parallelism
- Framework automatically generate MPI messages
- Easy to switch/combine multiple algorithms

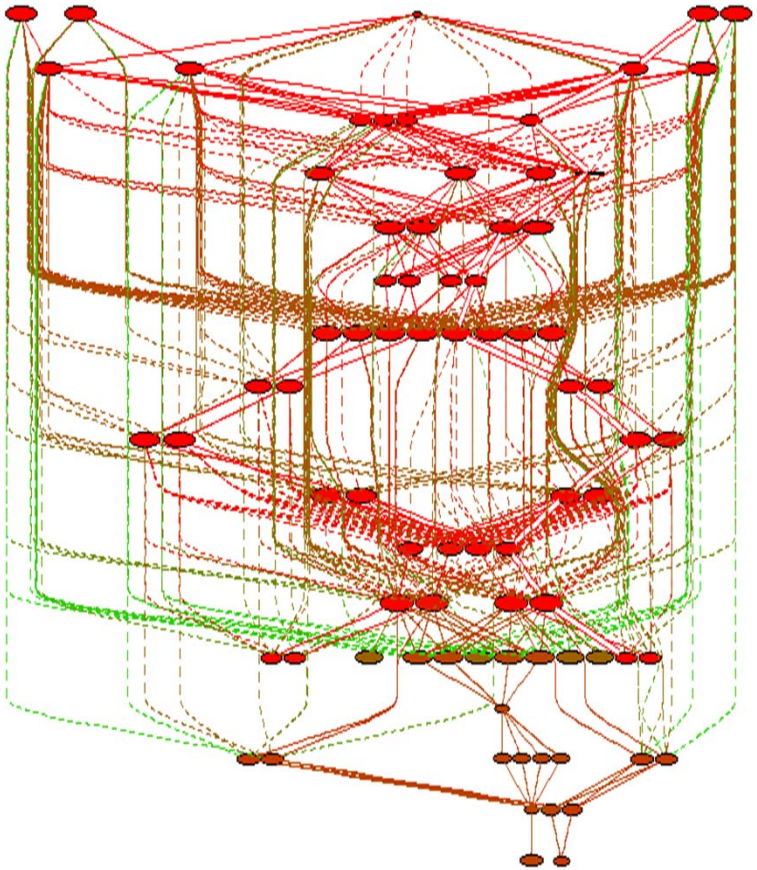
THE UNIVERSITY OF UTAH

How Uintah Works



THE UNIVERSITY OF UTAH

Uintah Task Graph



4 patches single level ICE task graph

- Intermediate representation for Uintah runtime system
- **Distributed:** only creates detailed tasks on local and neighboring patches
- Compiled by the framework
- Internal dependencies

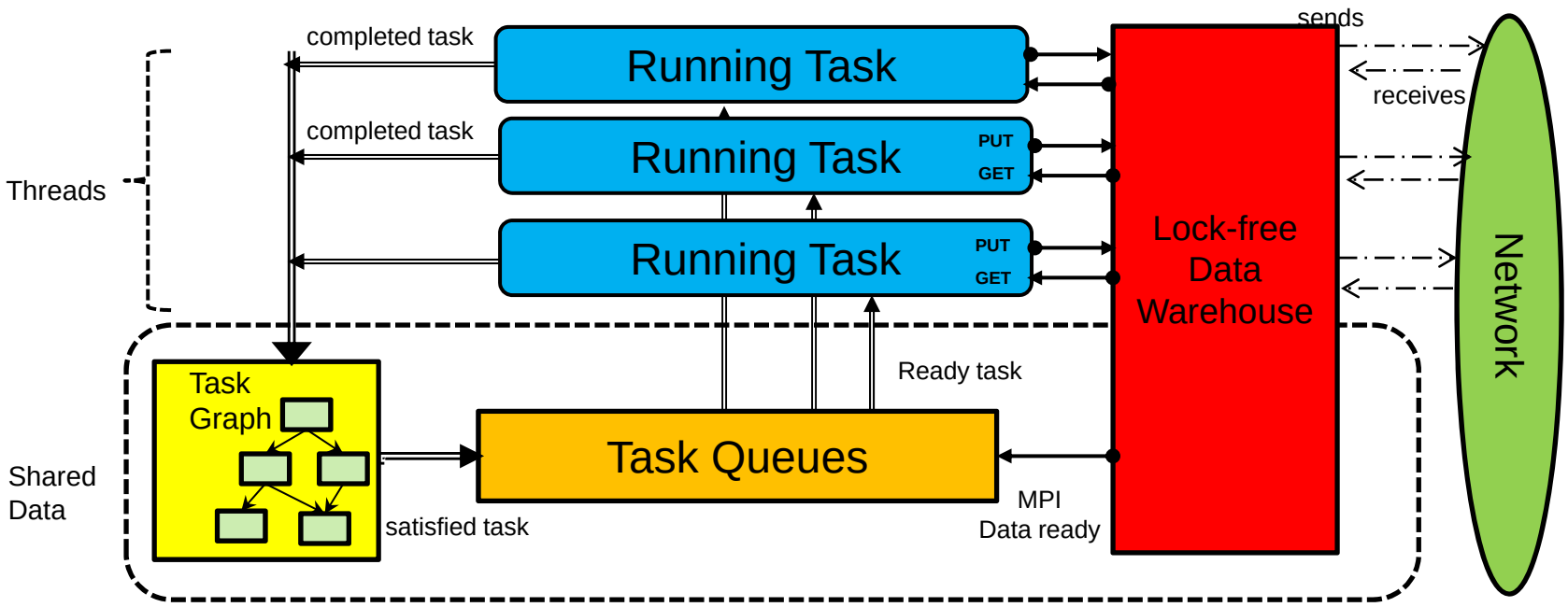
  ->Edges
- External dependencies

  ->MPI message tags

Uintah Runtime System: How Uintah Runs Tasks

- **Memory Manager**: Uintah Data Warehouse (DW)
  - Variable dictionary (hashed map from: Variable Name, Patch ID, Material ID keys to memory)
  - Provide interfaces for tasks to
    - Allocate variables
    - Put variables into DW
    - Get variables from DW
  - **Automatic Scrubbing** (garbage collection)
  - **Checkpointing & Restart** (data archiver)
- **Task Manager**: Uintah schedulers
  - Decides when and where to run tasks
  - Decides when to process MPI

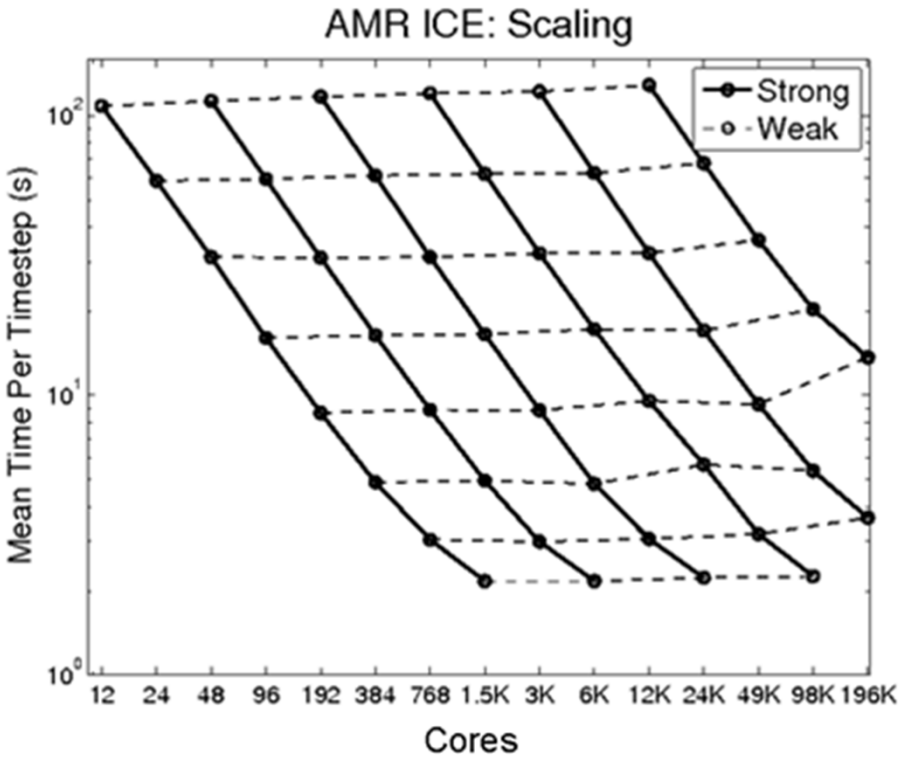
Hybrid Thread/MPI Scheduler (De-centralized)

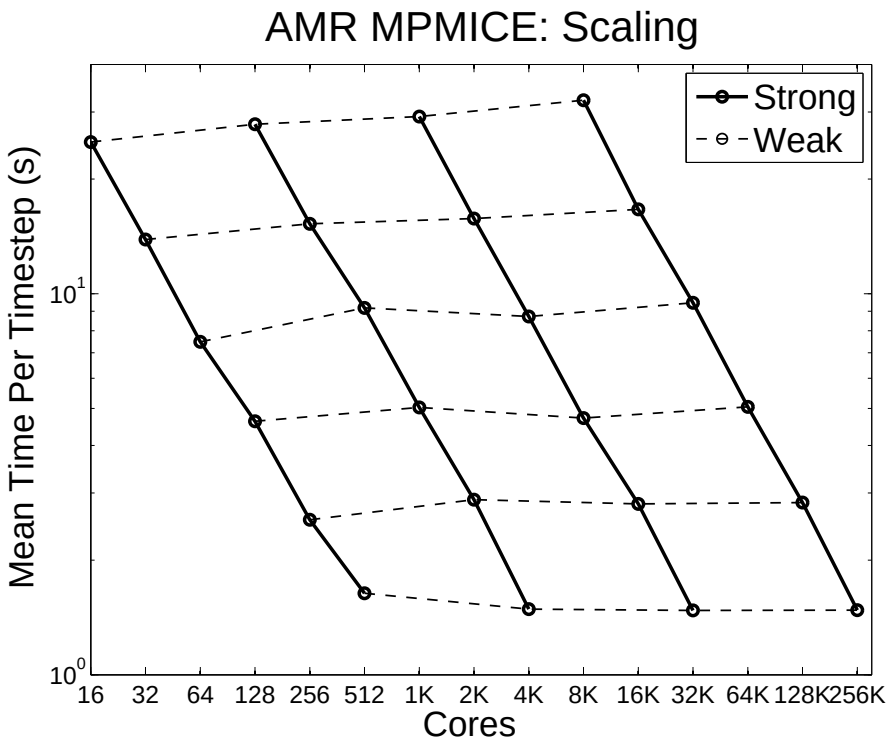


- **No control thread**: All threads directly pull tasks from task queues, thread-safety required for all data structure
- **Fully Overlapping**: All threads process MPI sends/receives/collective and execute tasks, *mpi_thread_multiple*, multiple communicators required
- Use **lock-free** data structure to avoid locking overhead

THE UNIVERSITY OF UTAH

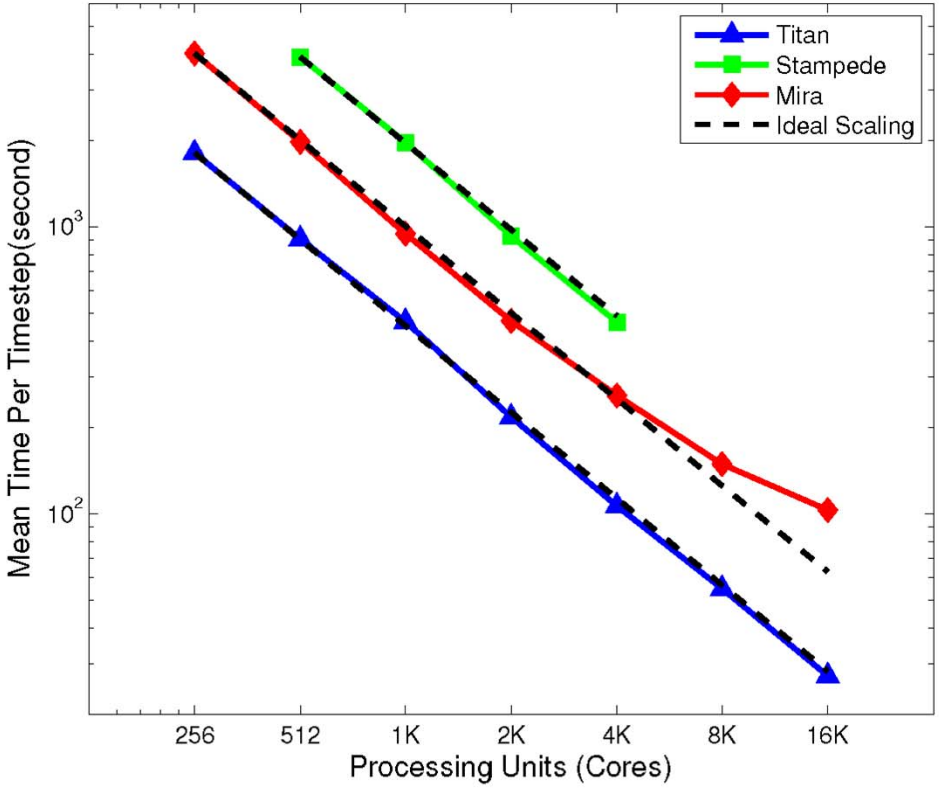
Uintah Scalability



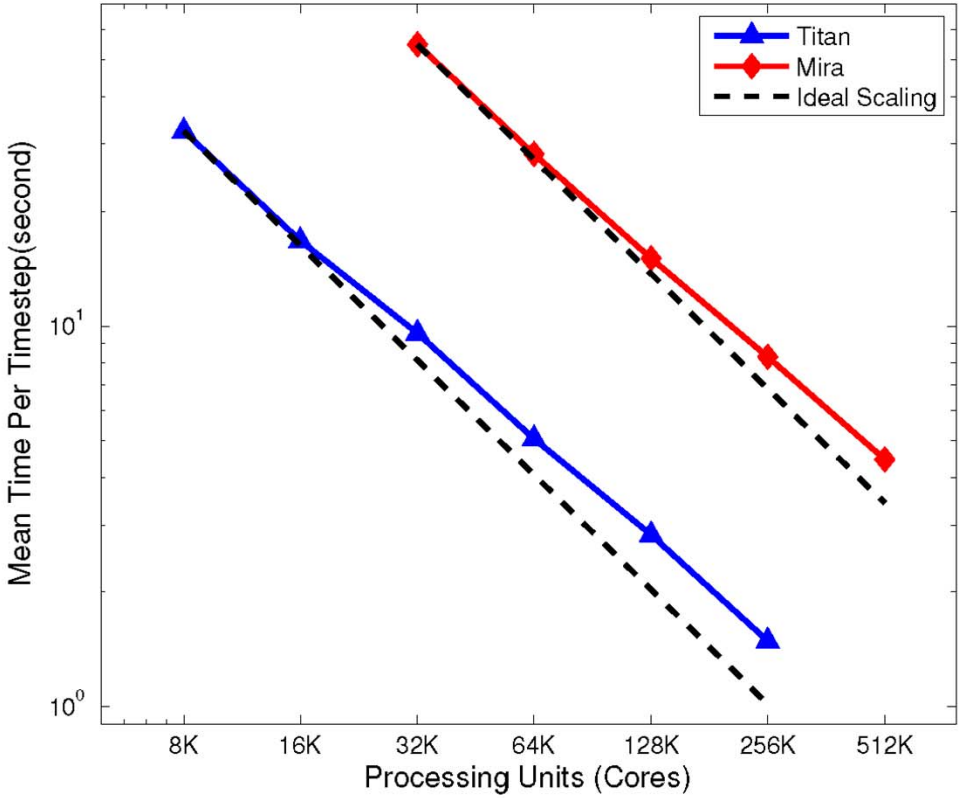


- Ability to run large simulations (MPI-only runs out-of-memory)
- Achieve much better CPU Scalability
- 95% weak scaling efficiency on 256K cores on Titan

Uintah Scaling on Titan, Mira, Stampede



2-Level RMCRT



AMR MPMICE

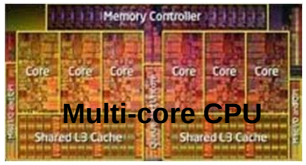
Emergence of Heterogeneous Systems

TACC Stampede
1000s of Xeon Phi Coprocessors

Xeon Phi

GPU

+

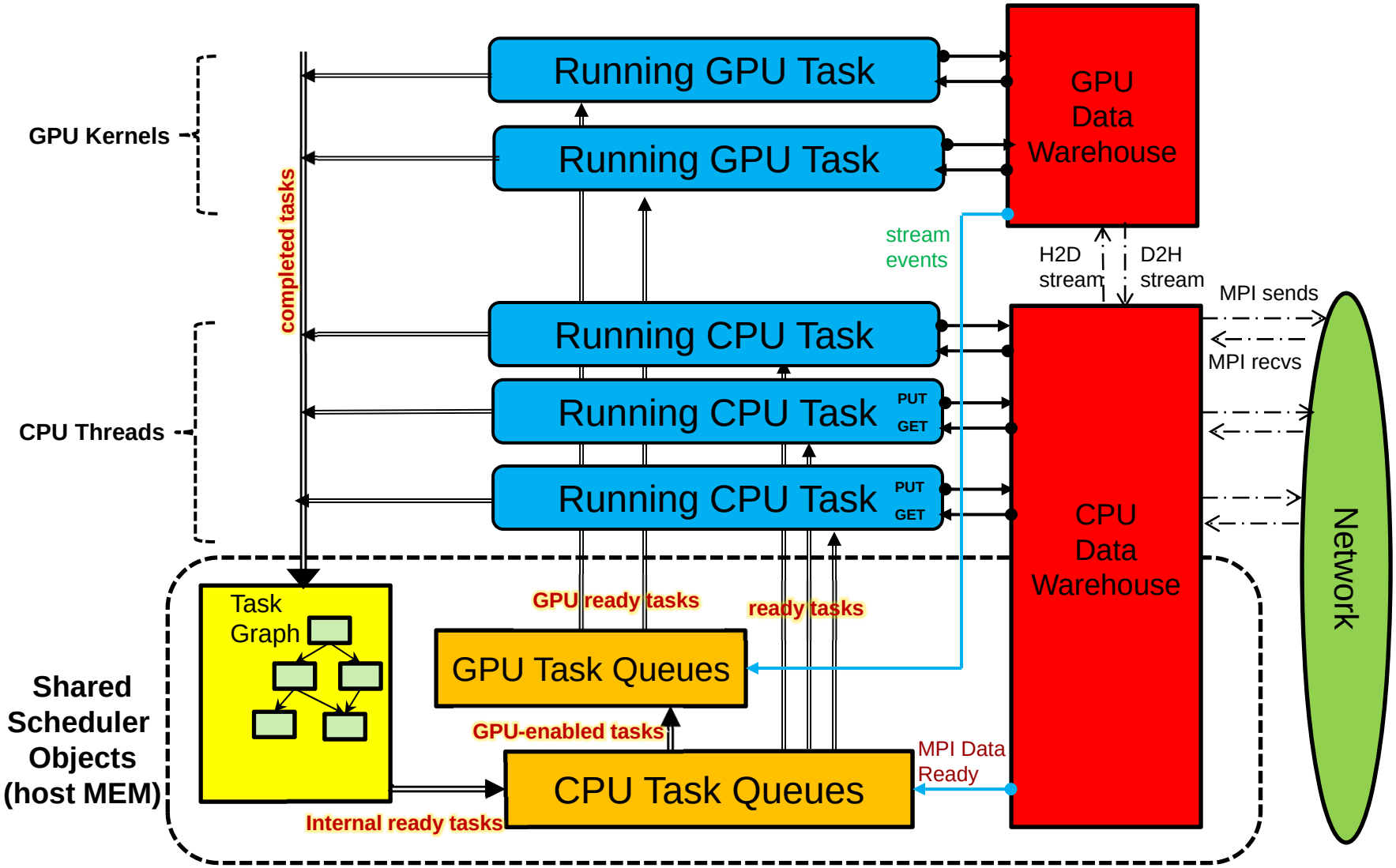


DOE Titan
1000s of GPUs

- **Motivation** - Accelerate Uintah Components
- Utilize all on-node computational resources
- Uintah's asynchronous task-based approach well suited for Co-processors and Accelerator designs

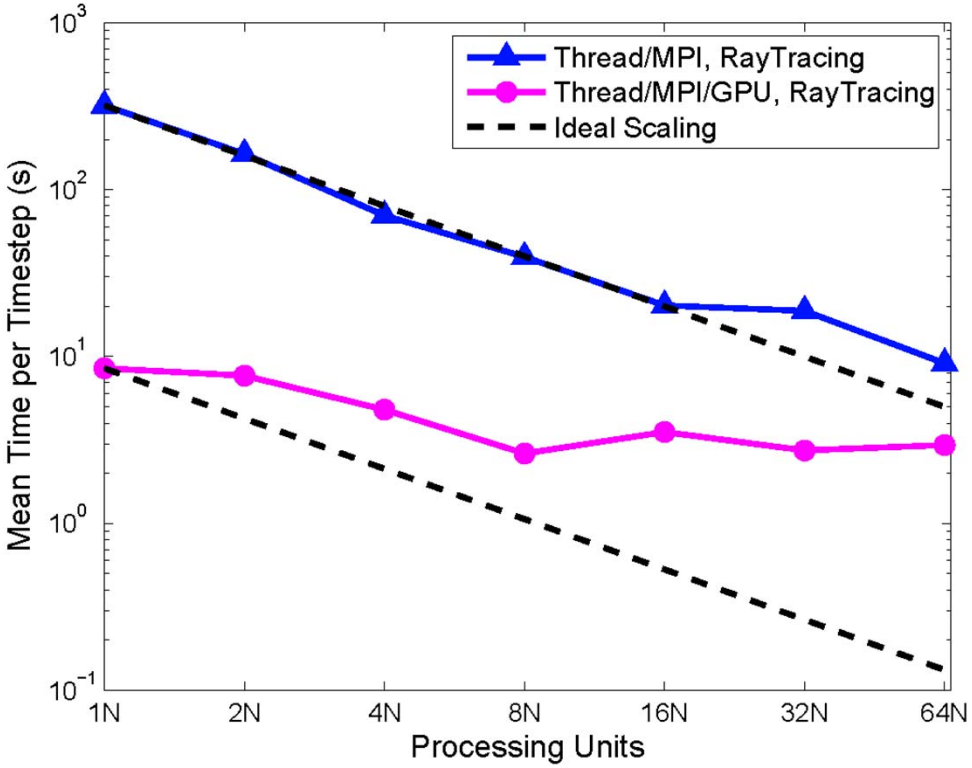
THE UNIVERSITY OF UTAH

Unified Heterogeneous Scheduler (GPU offload)



THE UNIVERSITY OF UTAH

Performance and Scaling Comparisons

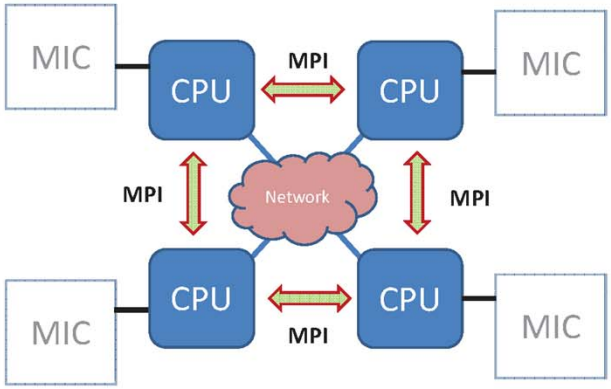


- Uintah strong scaling results when using:
  - Multi-threaded MPI
  - Multi-threaded MPI w/ GPU
- GPU-enabled RMCRT
  - **All-to-all** communication
  - 100 rays per cell 128^3 cells
  - Need port Multi-level CPU tasks to GPU

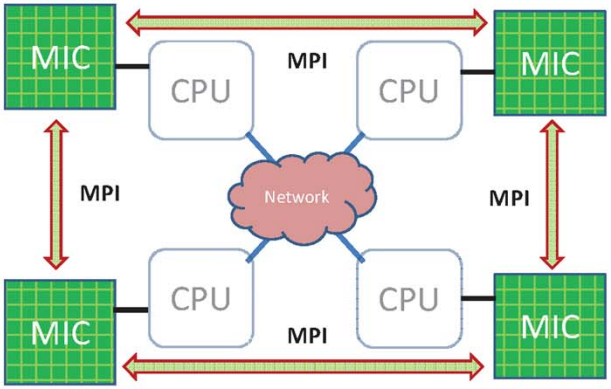
Cray XK-7, DOE Titan
Thread/MPI: N=16 AMD Opteron CPU cores
Thread/MPI+GPU: N=16 AMD Opteron CPU cores + 1 NVIDIA K20 GPU

THE UNIVERSITY OF UTAH

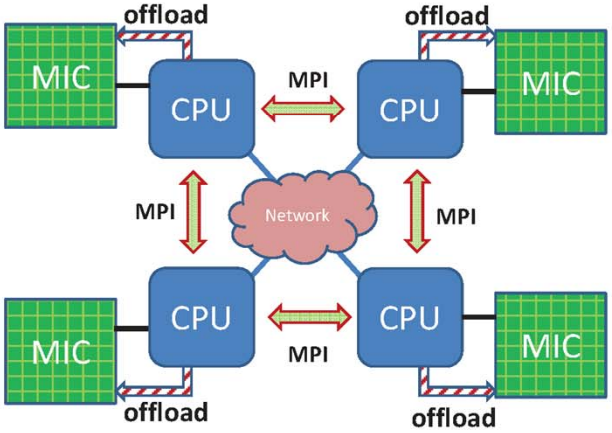
Xeon Phi Execution Models



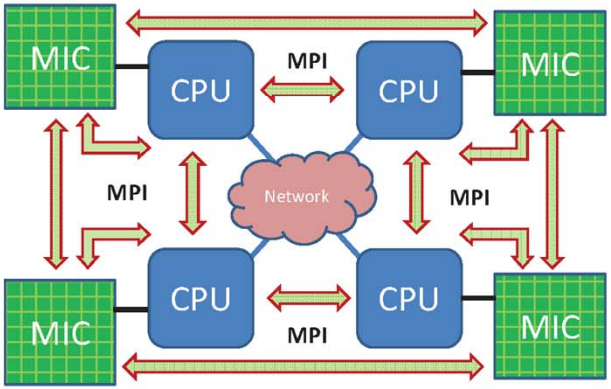
(1) Host-only Model



(2) MIC Native Model



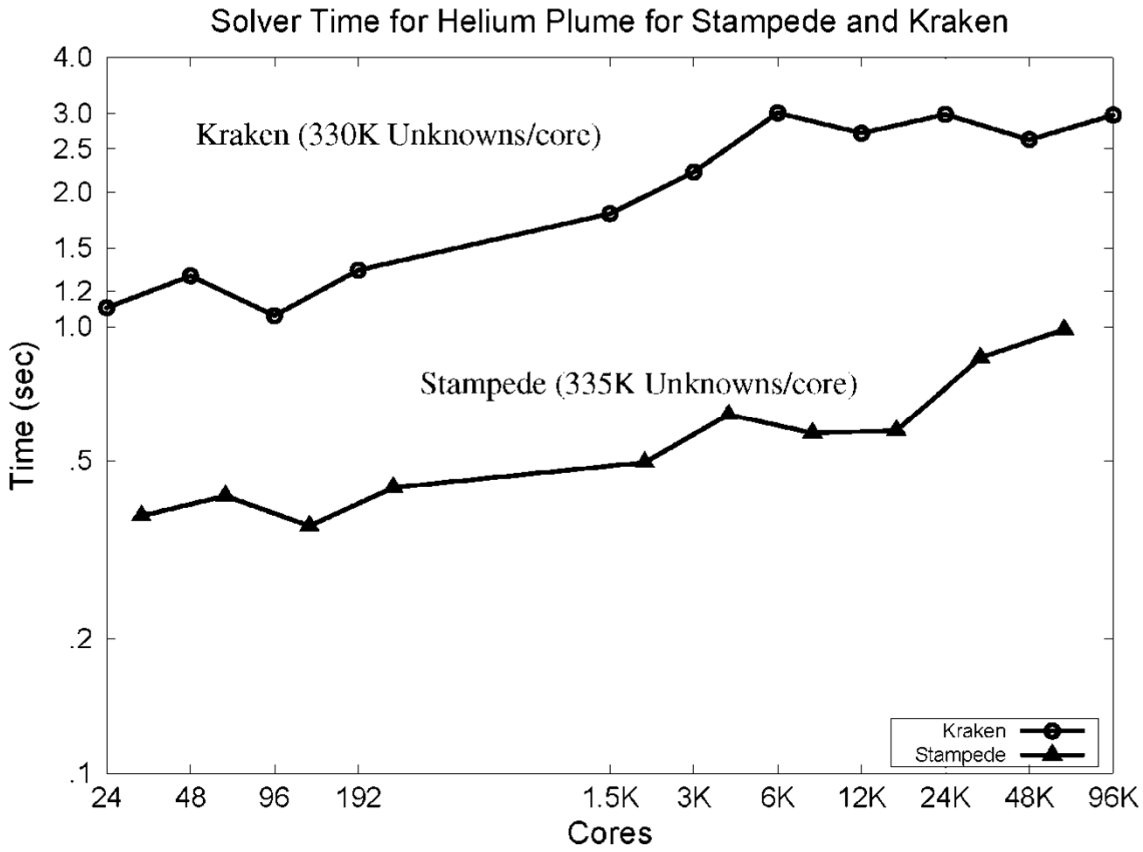
(3) Offload Model



(4) Symmetric Model

THE UNIVERSITY OF UTAH

Uintah on Stampede: Host-only Model



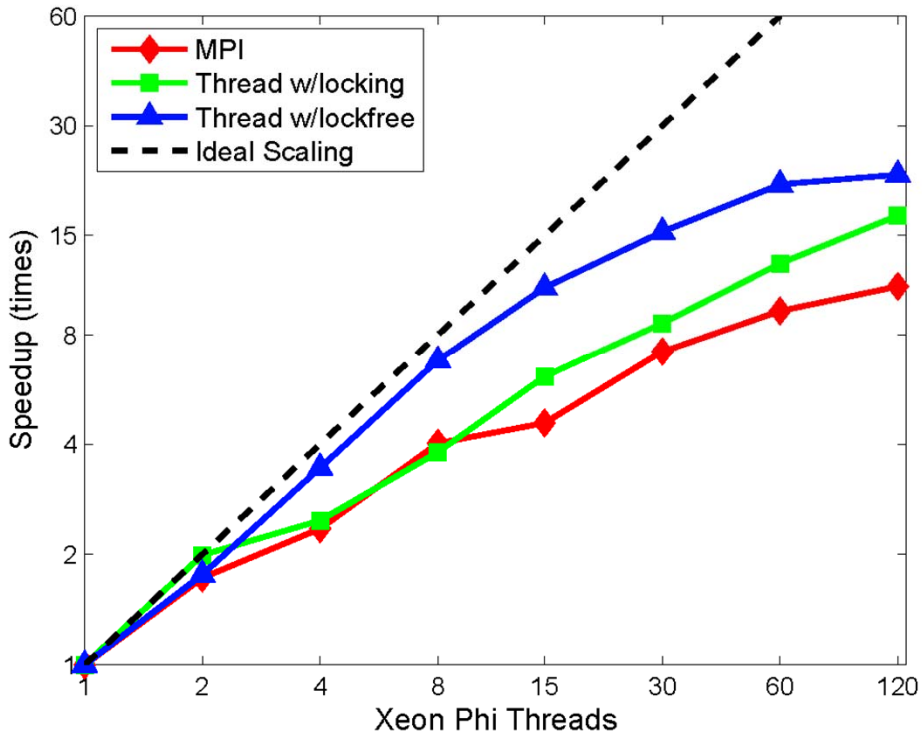
- Intel MPI issues beyond 2048 cores (seg faults)
- MVAPICH2 required for larger core counts

- Using Hypre with a conjugate gradient solver
- Preconditioned with geometric multi-grid
- Red Black Gauss Seidel relaxation - each patch

THE UNIVERSITY OF UTAH

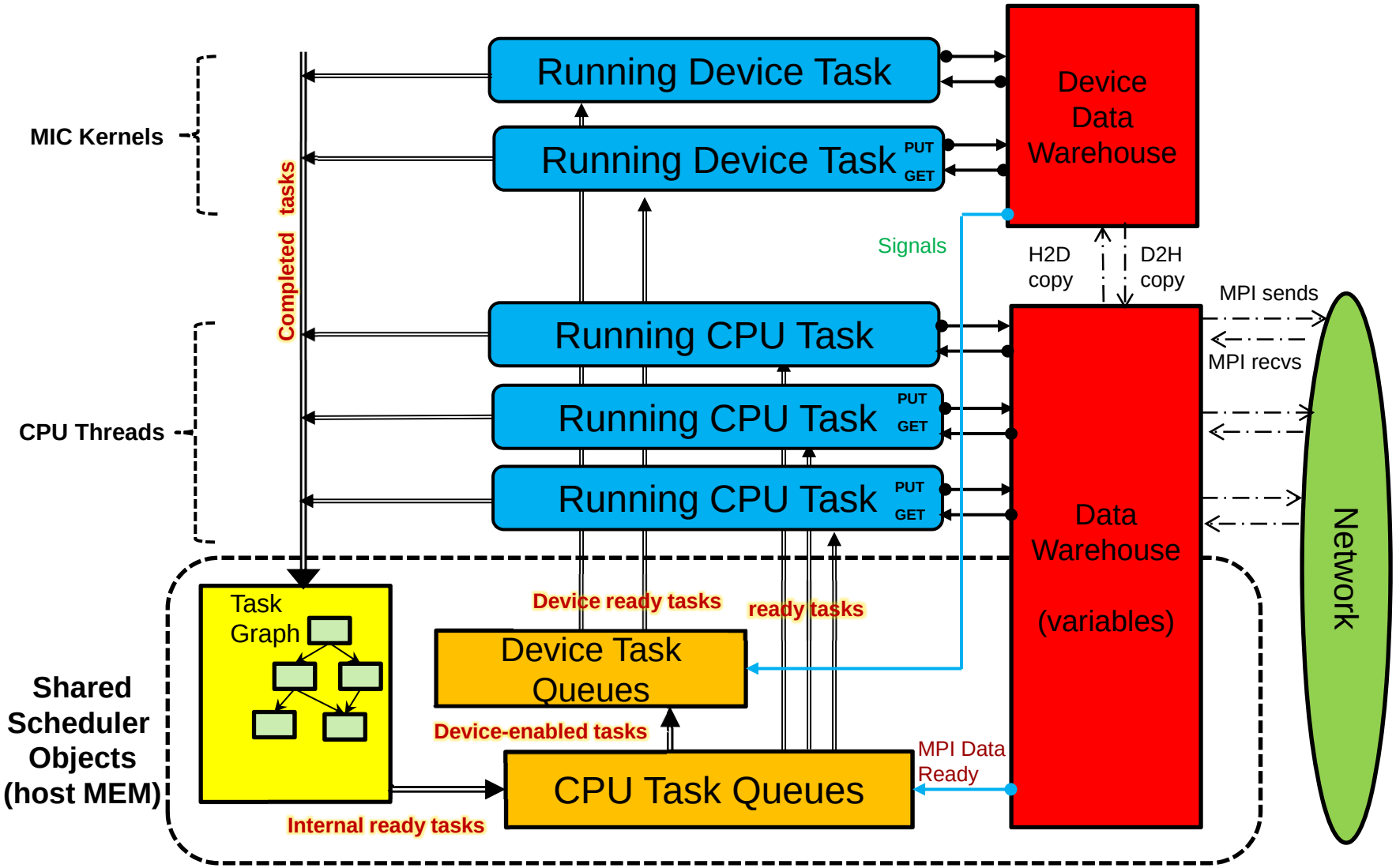
Uintah on Stampede: Native Model

- Compile with –mmic
  - cross compiling
- Need to build all Uintah required 3p libraries
  - libxml2
  - zlib
- Run Uintah natively on Xeon Phi within 1 day
- Single Xeon Phi Card



THE UNIVERSITY OF UTAH

Unified Heterogeneous Scheduler & Runtime Offload Model

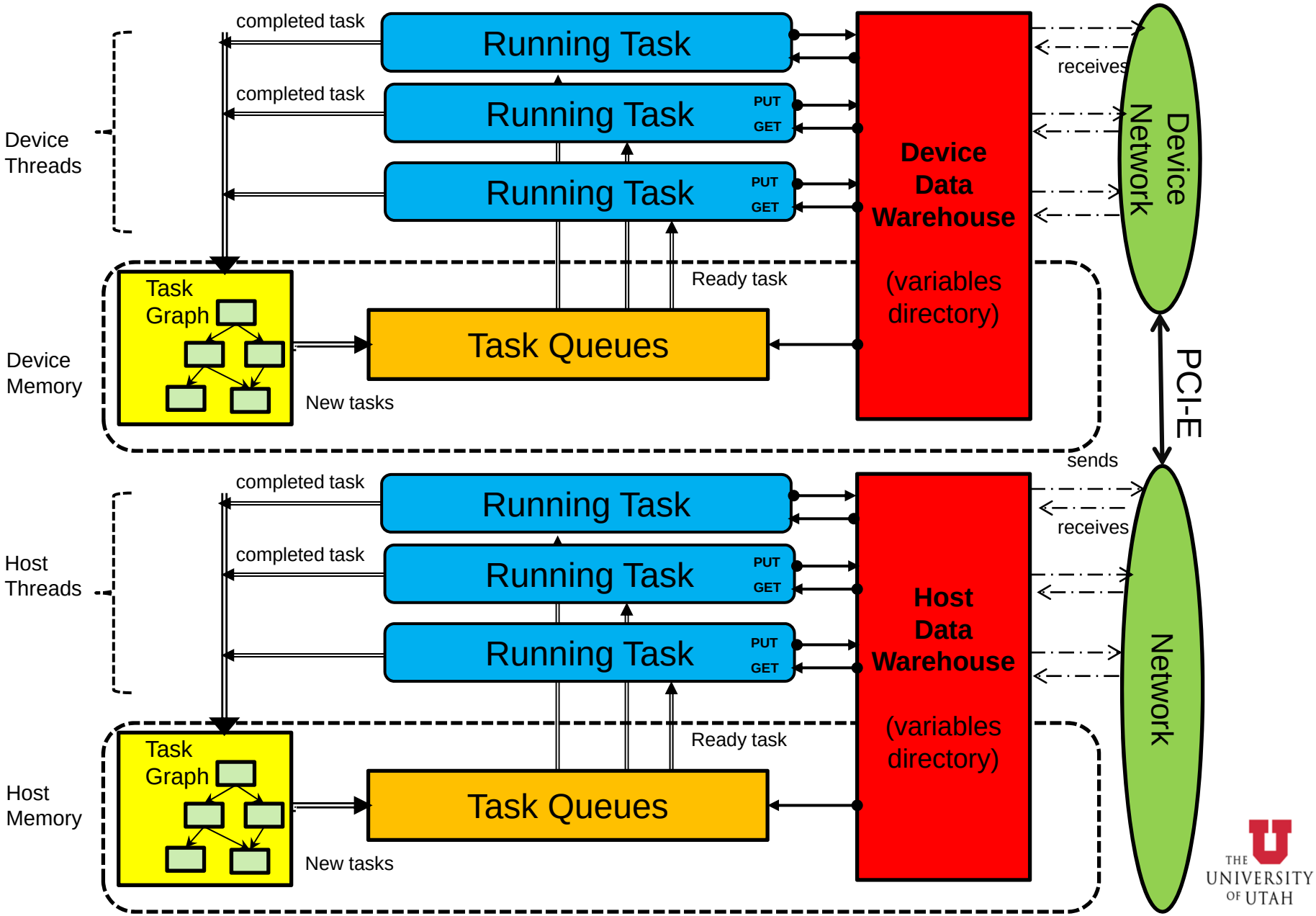


THE UNIVERSITY OF UTAH

Uintah on Stampede: Offload Model

- Use compiler directives (#pragma)
  - Offload target: #pragma offload target(mic:0)
  - OpenMP: #pragma omp parallel
- Find copy in/out variables from task graph
- Functions called in MIC must be defined with __attribute__((target(mic)))
- Hard for Uintah to use offload mode
  - Rewrite highly templated C++ methods with simple C/C++ so they can be called on the Xeon Phi
  - Less effort than GPU port, but still significant work for complex code such as Uintah

Unified Heterogeneous Scheduler (MIC symmetric)



THE UNIVERSITY OF UTAH

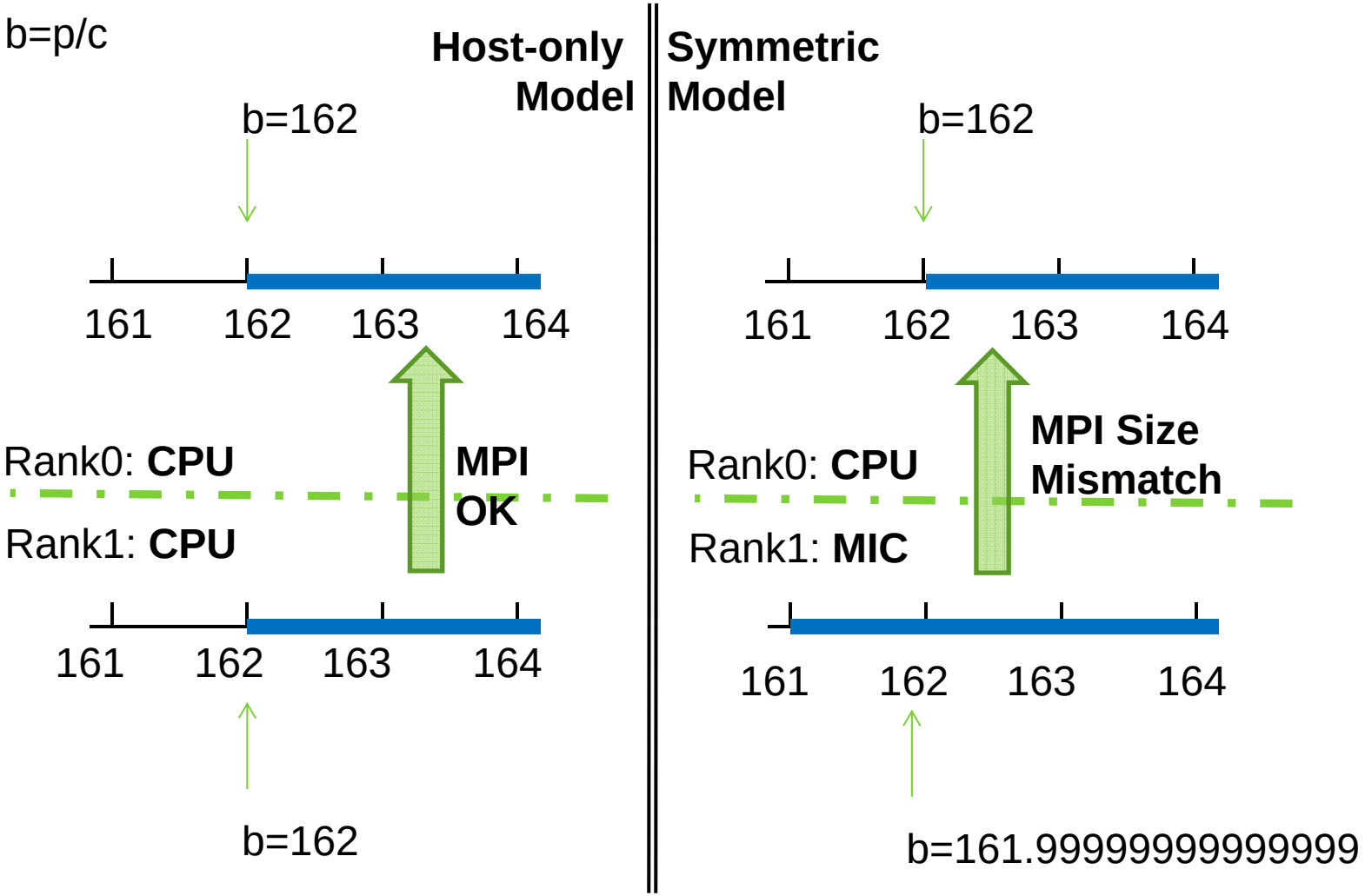
Uintah on Stampede: Symmetric Model

- Xeon Phi directly calls MPI
- Use Pthreads on both host CPU and Xeon Phi:
  - 1 MPI process on host – 16 threads
  - 1 MPI process on MIC – up to 120 threads
- Currently only Intel MPI supported

  mpiexec.hydra -n 8 ./sus – nthreads 16 : -n 8./sus.mic –nthreads 120
- Challenges: Different floating point accuracy on host and co-processor
  - Result Consistency
  - MPI message mismatch issue: Control related FP operations

Example: Symmetric Model FP Issue

p=0.421874999999999944488848768742172978818416595458984375
c=0.00260416666666666652210637700193274213233962655067443847656 25
b=p/c

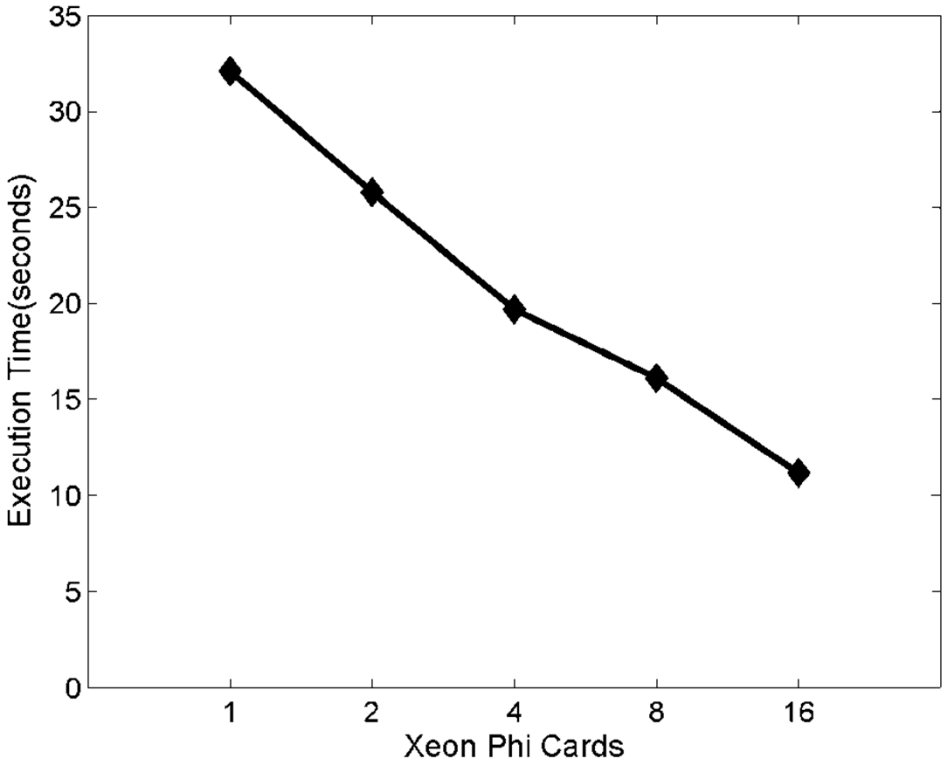


Control related FP operations must use consistent accuracy model

THE UNIVERSITY OF UTAH

Scaling Results on Xeon Phi

- Multi MIC Cards (Symmetric Model)
- Xeon Phi card: 60 threads per MPI process, 2 MPI processes
- host CPU :16 threads per MPI process, 1 MPI processes
- Issue: load imbalance - profiling differently on host and Xeon Phi



Multiple MIC cards

THE UNIVERSITY OF UTAH

Current and Future Work

- Load Balancer
  - **Cannot treat all MPI ranks uniformly**
  - Profile CPU and Xeon Phi separately
  - Need separate forecast model for each
- Address different cache sizes
  - New **regridder** to generate large patches for CPU and small patches for Xeon Phi
- Explicitly use long vector
- Asynchronous offload model
  - _Offload_signaled(mic_no, &c)

THE UNIVERSITY OF UTAH

Questions?

Alstom Clean Coal Boiler Simulation: RMCRT offloaded, Flow simulation on CPU

Software Homepage http://www.uintah.utah.edu/

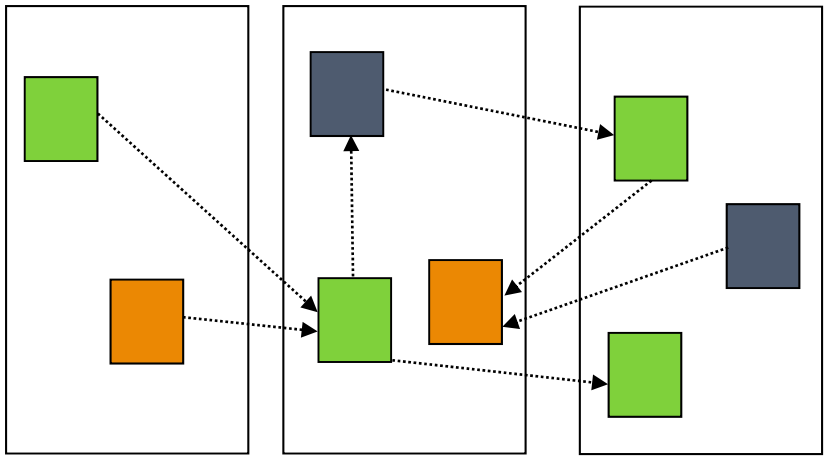
Science Track Talk: Jacqueline Beckvermit, Wednesday, 5:00-5:30pm

The Influence of an Applied Heat Flux on the Violence of Reaction of an Explosive Device

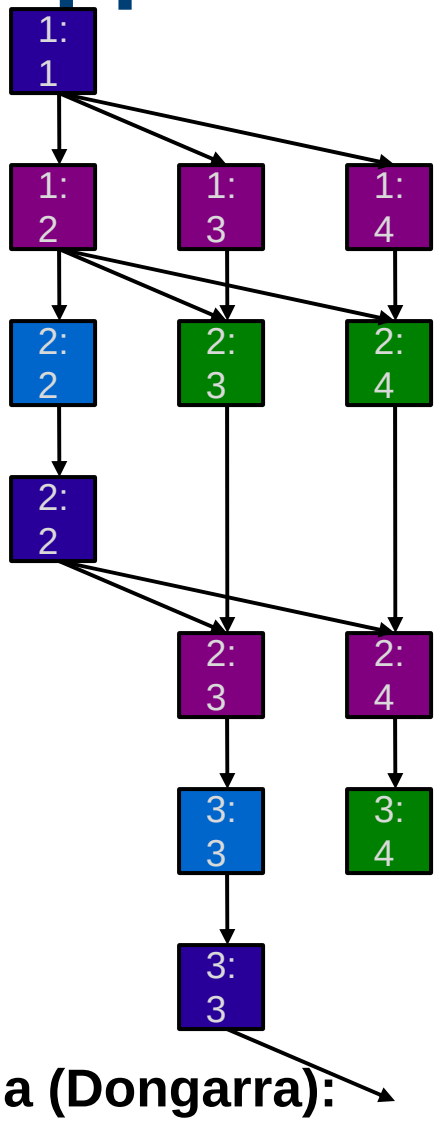
SCI INSTITUTE

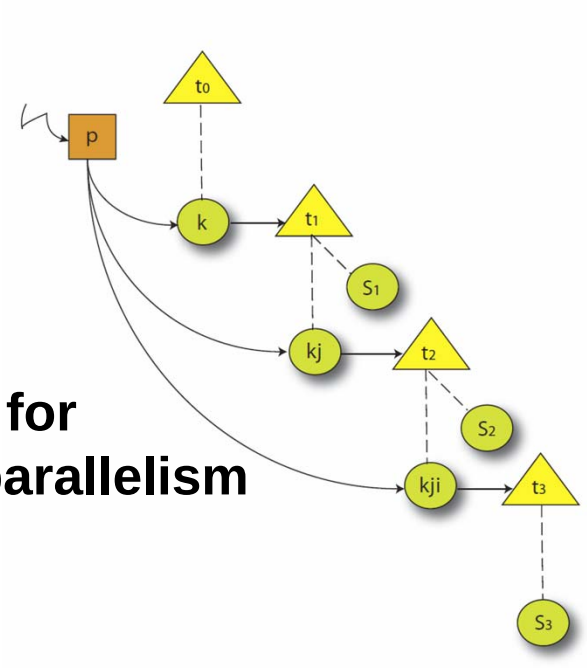
THE UNIVERSITY OF UTAH

Graph Based Applications



Charm++: Object-based Virtualization





Intel CnC:
new language for
graph based parallelism

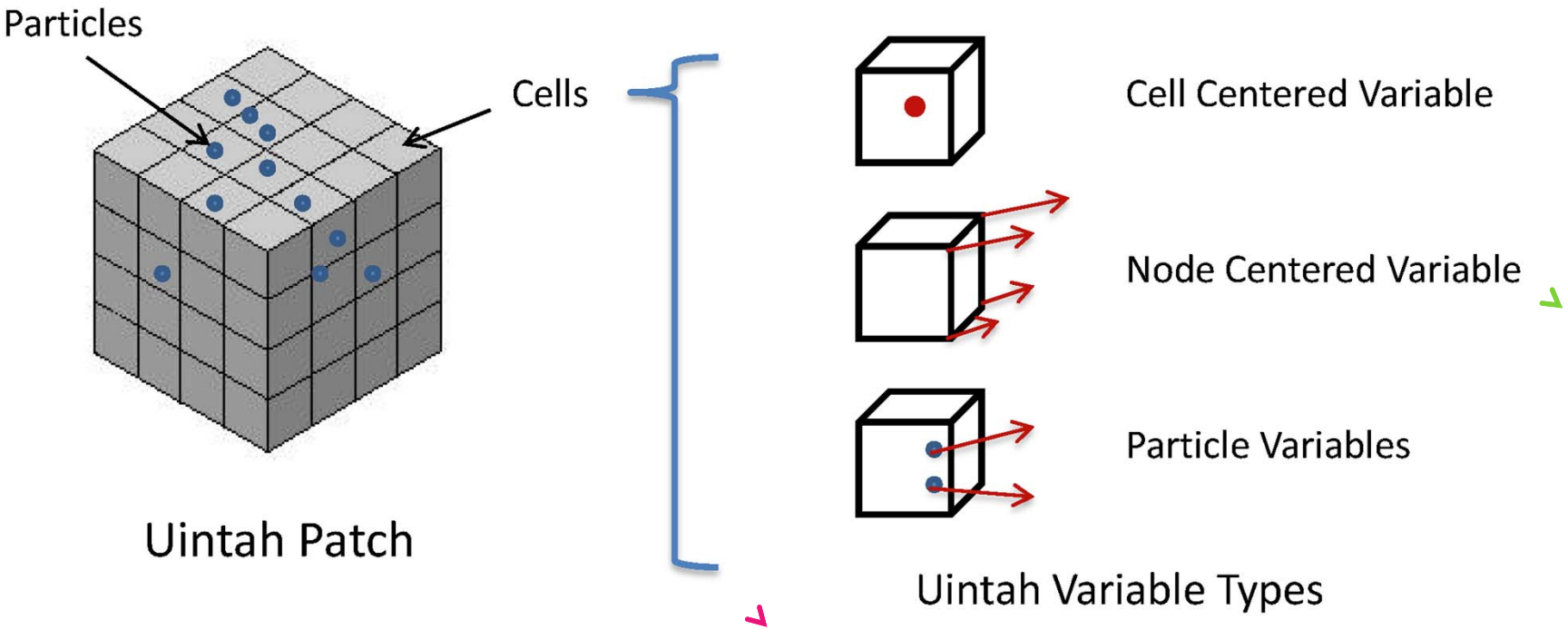
Plasma (Dongarra):
DAG based
Parallel linear algebra software

Software Model for Exascale

- Silver model for Exascale software which must:
  - Directed dynamic graph execution
  - Latency hiding
  - Minimize synchronization and overheads
  - Adaptive resource scheduling
  - Heterogeneous processing
- Graph-based asynchronous-task work queue model

(DARPA software report, 2009)

Uintah Patch and Variables



Structured Grid Variable (for Flows)

- Cell Centered, Node Centered, Face Centered

Unstructured Points (for Solids)

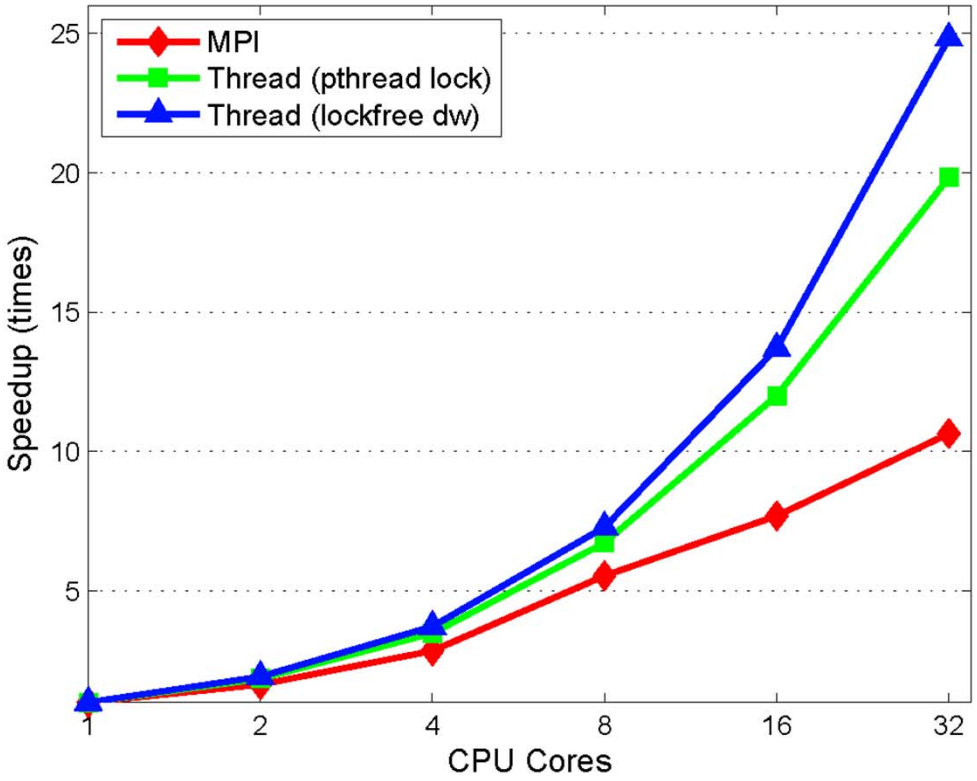
- Particles
- Atoms

Uintah on Stampede

- **Host Only**
- **Native**
  - Compile with –mmic (cross compiling)
  - Third-party libs (zlib, libxml2)
  - Single Xeon Phi Card
- **Offload**
  - Use compiler directives (#pragma)
  - Functions called in MIC must be defined with __attribute__((target(mic)))
  - Need rewrite simple C/C++
- **Symmetric**
  - Best fits current Uintah model
  - Xeon Phi directly calls MPI
  - Different floating point accuracy

THE UNIVERSITY OF UTAH

Performance Comparison



AMR MPMICE with

- MPI-only
- Pthread & lock-based DW
- Pthread & lock-free DW

2.4X speed up (Pthread with lock-free DW Vs MPI-only)

Cray XE6 node, 32 AMD Opteron cores

THE UNIVERSITY OF UTAH