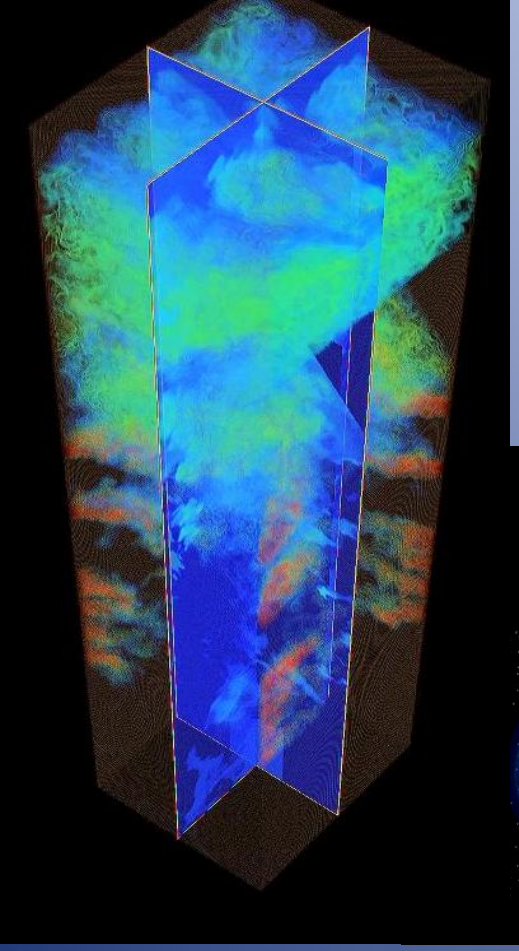
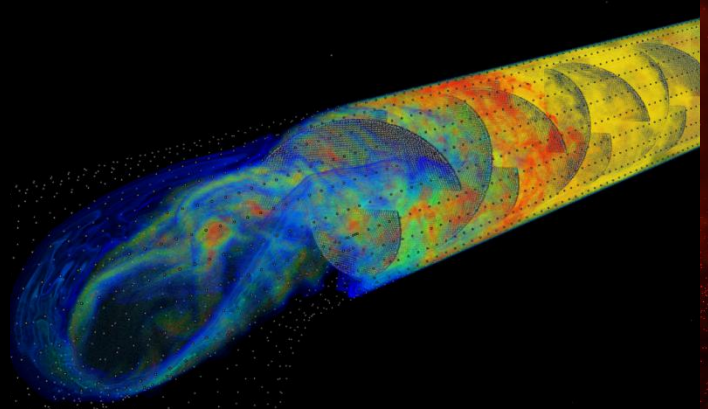
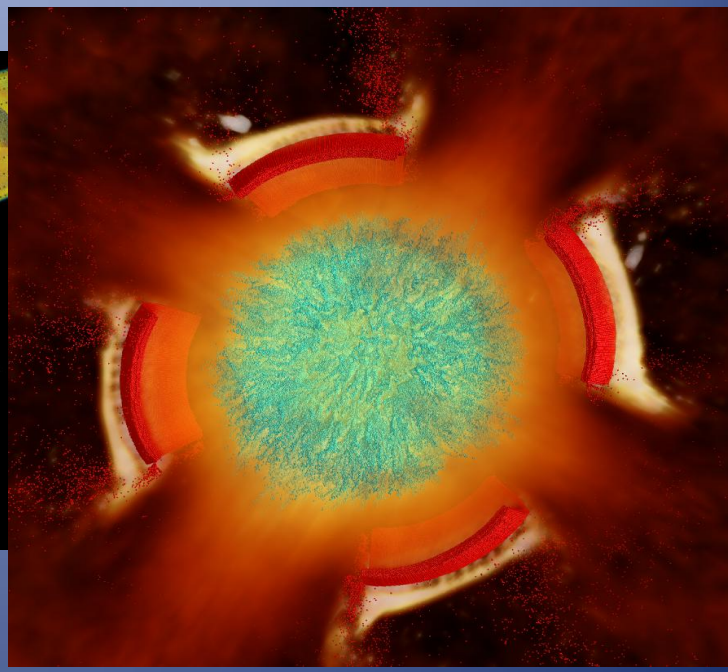


A Unified Approach to Heterogeneous Architectures Using the Uintah Framework





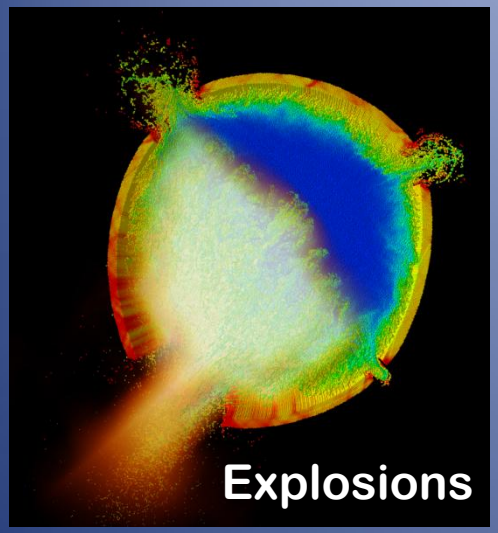
Qingyu Meng, Alan Humphrey
Martin Berzins

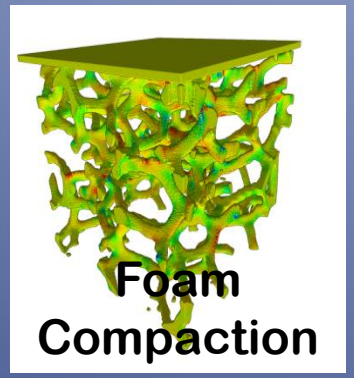
Thanks to: TACC Team for early access to Stampede
John Schmidt and J. Davison de St. Germain, SCI Institute
Justin Luitjens and Steve Parker, Nvidia

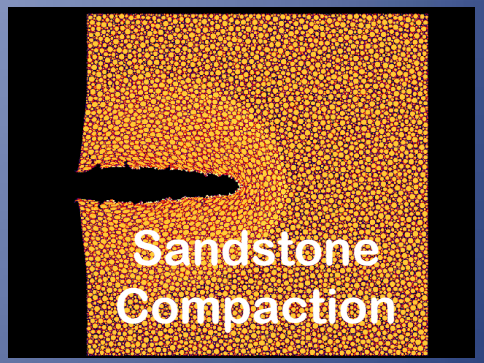
DOE for funding the CSAFE project (97-10), DOE NETL, DOE NNSA
NSF for funding via SDCI and PetaApps

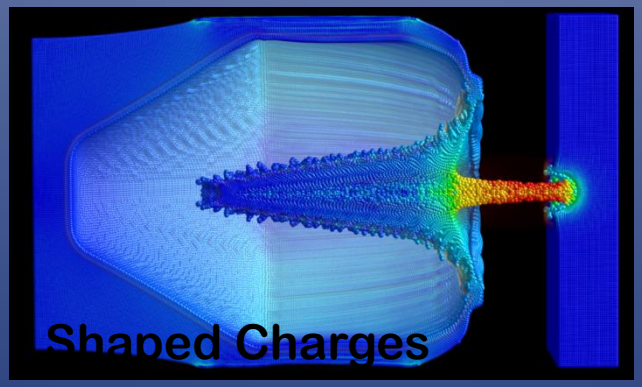
Uintah Overview

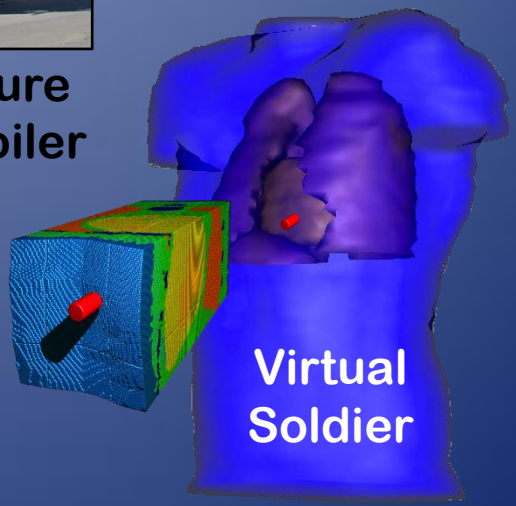
- Parallel, adaptive multi-physics framework
- Fluid-structure interaction problems
- Patch-based AMR using:
  - particles and mesh-based fluid-solve





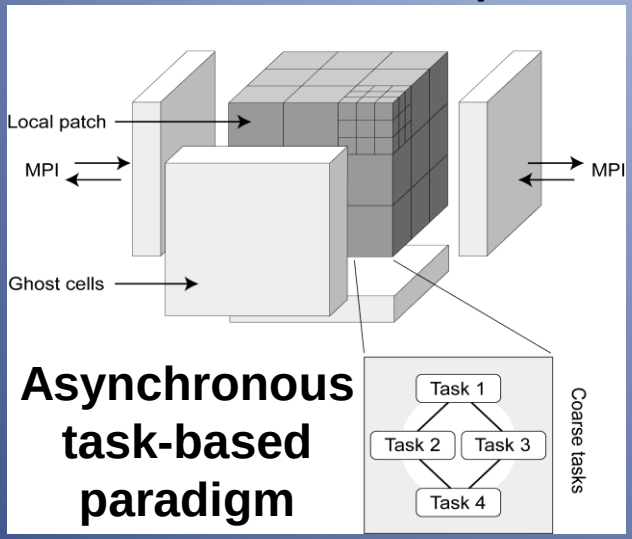


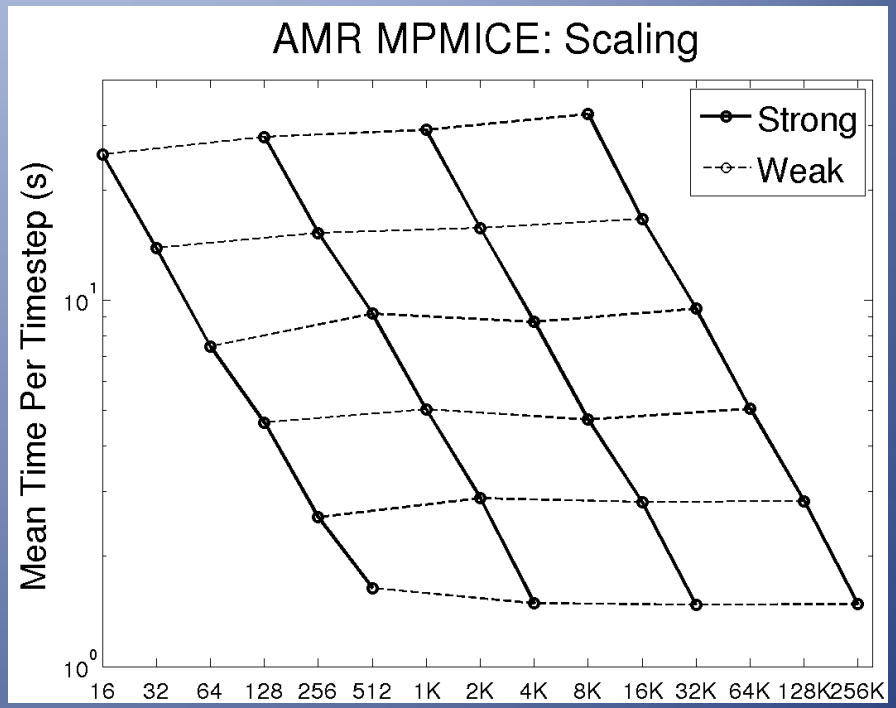




Uintah - Scalability

Patch-based domain decomposition



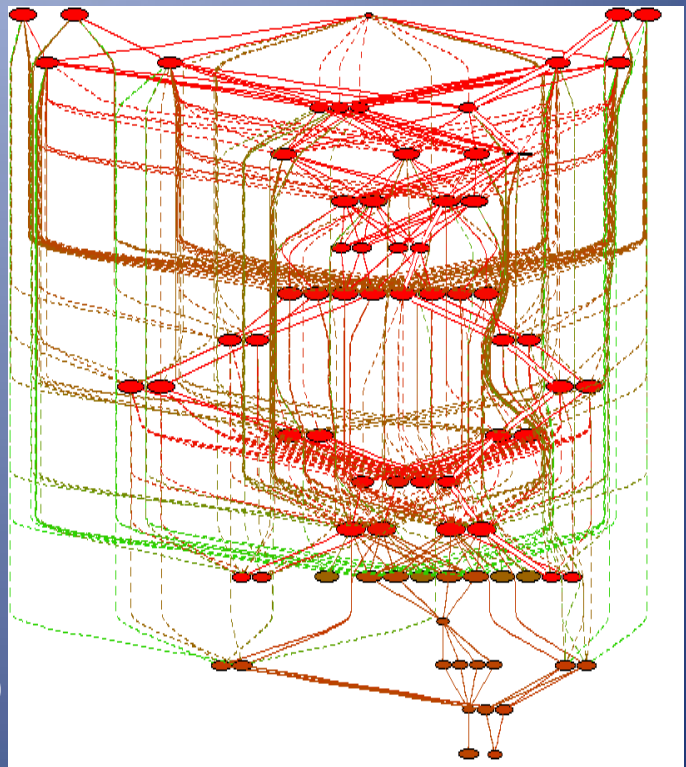


- 256K cores – Jaguar XK6
- 95% weak scaling efficiency & 60% strong scaling efficiency
- Multi-threaded MPI – shared memory model on-node[1]
- Scalable, efficient, lock-free data structures [2]

1. Q. Meng, M. Berzins, and J. Schmidt. "Using Hybrid Parallelism to Improve Memory Use in the Uintah Framework". *In Proc. of the 2011 TeraGrid Conference (TG11),* Salt Lake City, Utah, 2011.
2. Q. Meng and M. Berzins. Scalable Large-scale Fluid-structure Interaction Solvers in the Uintah Framework via Hybrid Task-based Parallelism Algorithms. *Concurrency and Computation: Practice and Experience 2012, Submitted*

Uintah Task-Based Approach

- Task Graph
  - Directed Acyclic Graph
- Asynchronous, out of order execution of tasks
  - Multi-stage work queue design
- **Task** – basic unit of work
  - Key idea
  - C++ method with computation
- Allows Uintah to be generalized to support co-processors and accelerators
- No sweeping code changes



4 patch, single level ICE task graph

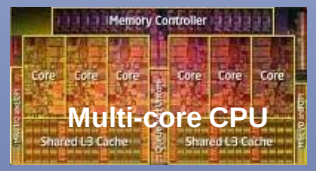
Emergence of Heterogeneous Systems

TACC Stampede
1000s of Xeon Phi Co-processors

Xeon Phi

+

GPU



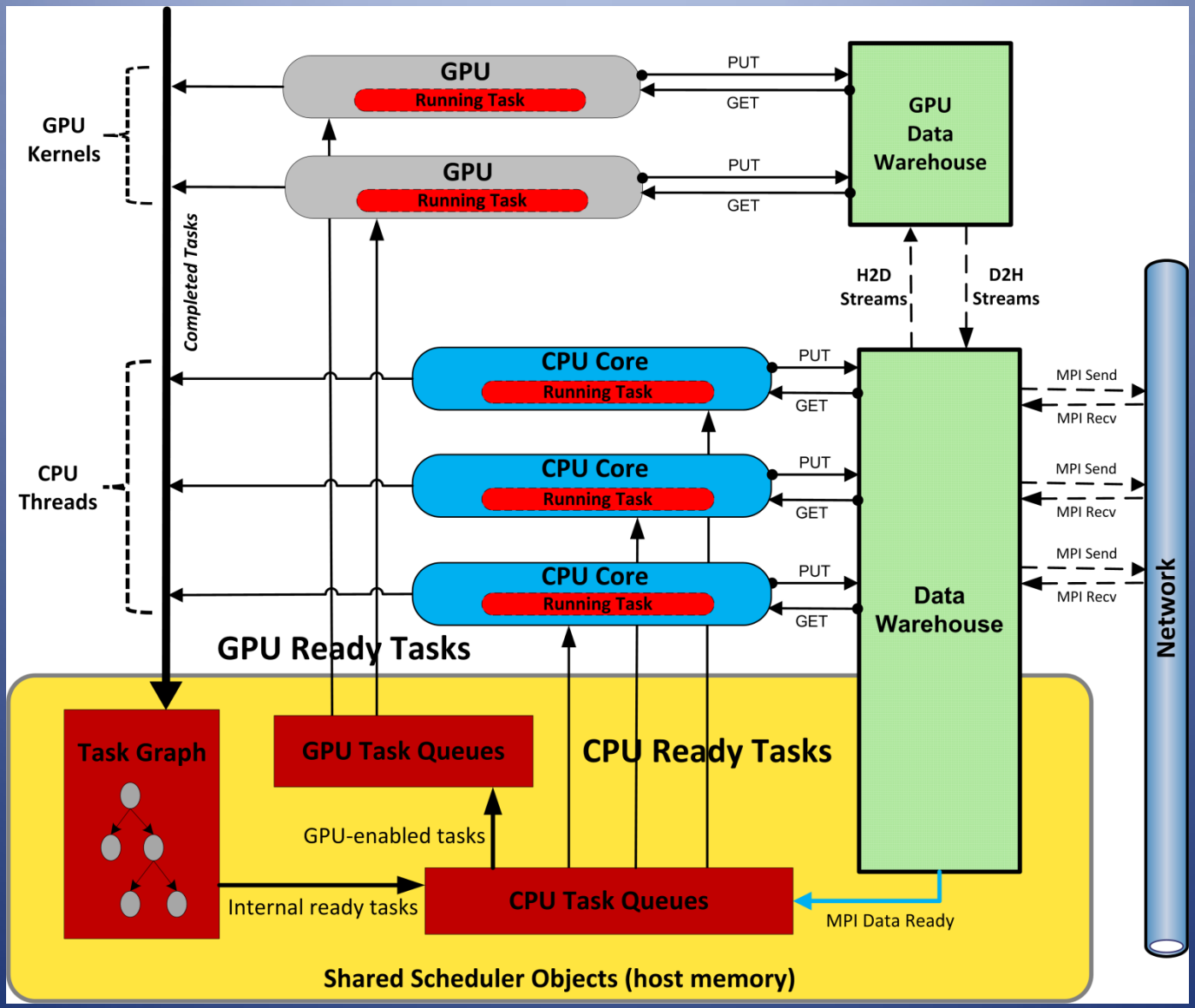
DOE Titan
1000s of Nvidia Kepler GPUs

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

Natural progression:

Accelerator & Co-processor Tasks

Unified Heterogeneous Scheduler & Runtime



GPU support on Keeneland and Titan

The Emergence of the Intel Xeon Phi

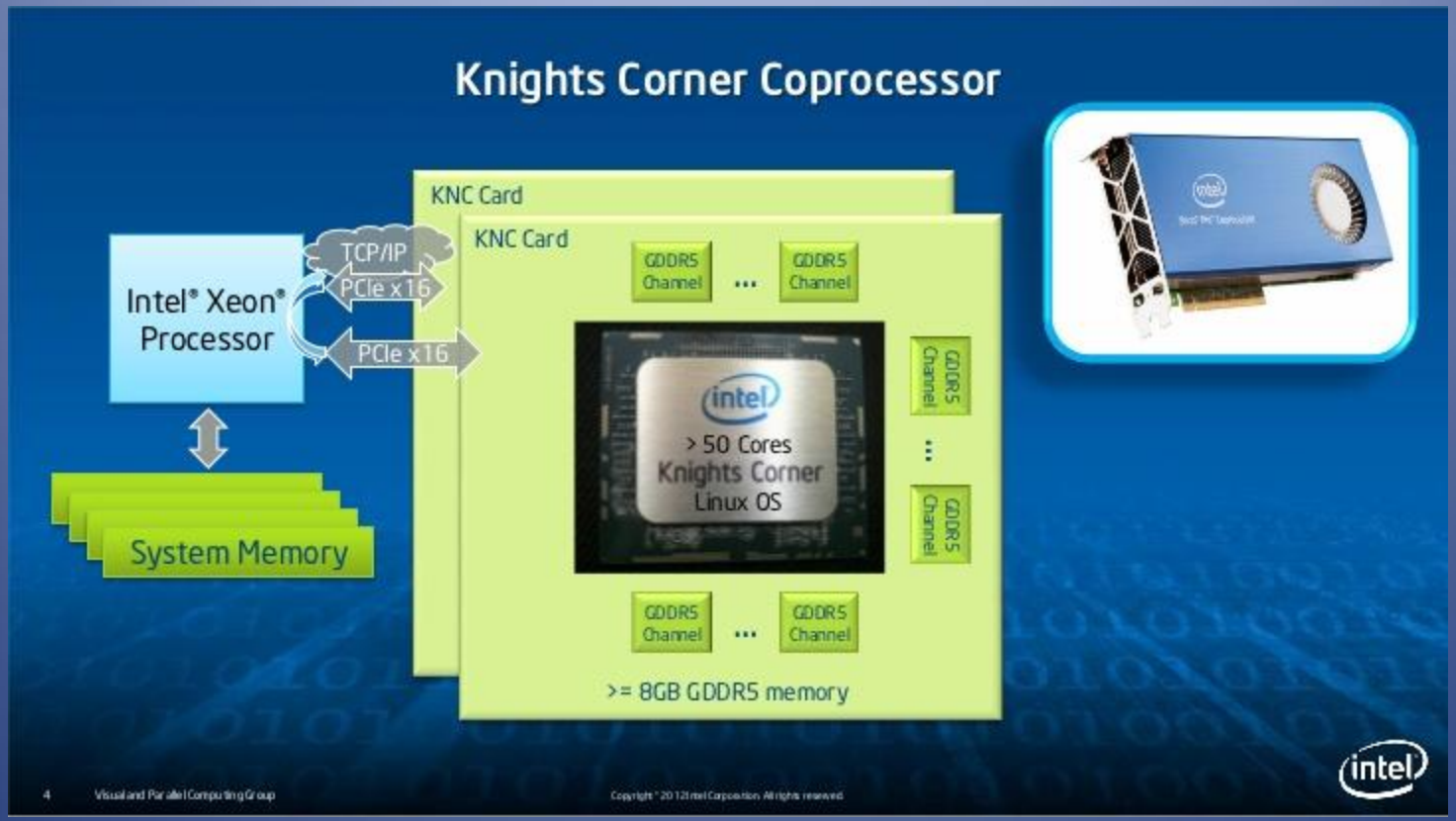
Intel Xeon Phi – What is it?

- Co-processor
  - PCI Express card
  - Light weight Linux OS (busy box)
- Dense, simplified processor
  - Many power-hungry operations removed
  - Wider vector unit
  - Wider hardware thread count
- Many Integrated Core architecture, aka MIC
  - Knights Corner (code name)
  - Intel Xeon Phi Co-processor (product name)

Intel Xeon Phi – What is it?

- Leverage x86 architecture (CPU with many cores)
  - simpler x86 cores, allow more compute throughput
- Leverage existing x86 programming models
- Dedicate much of the silicon to FP ops
- Cache coherent
- Increase floating-point throughput
- Strip expensive features
  - out-of-order execution
  - branch prediction
- Wide SIMD registers for more throughput
- Fast (GDDR5) memory on card

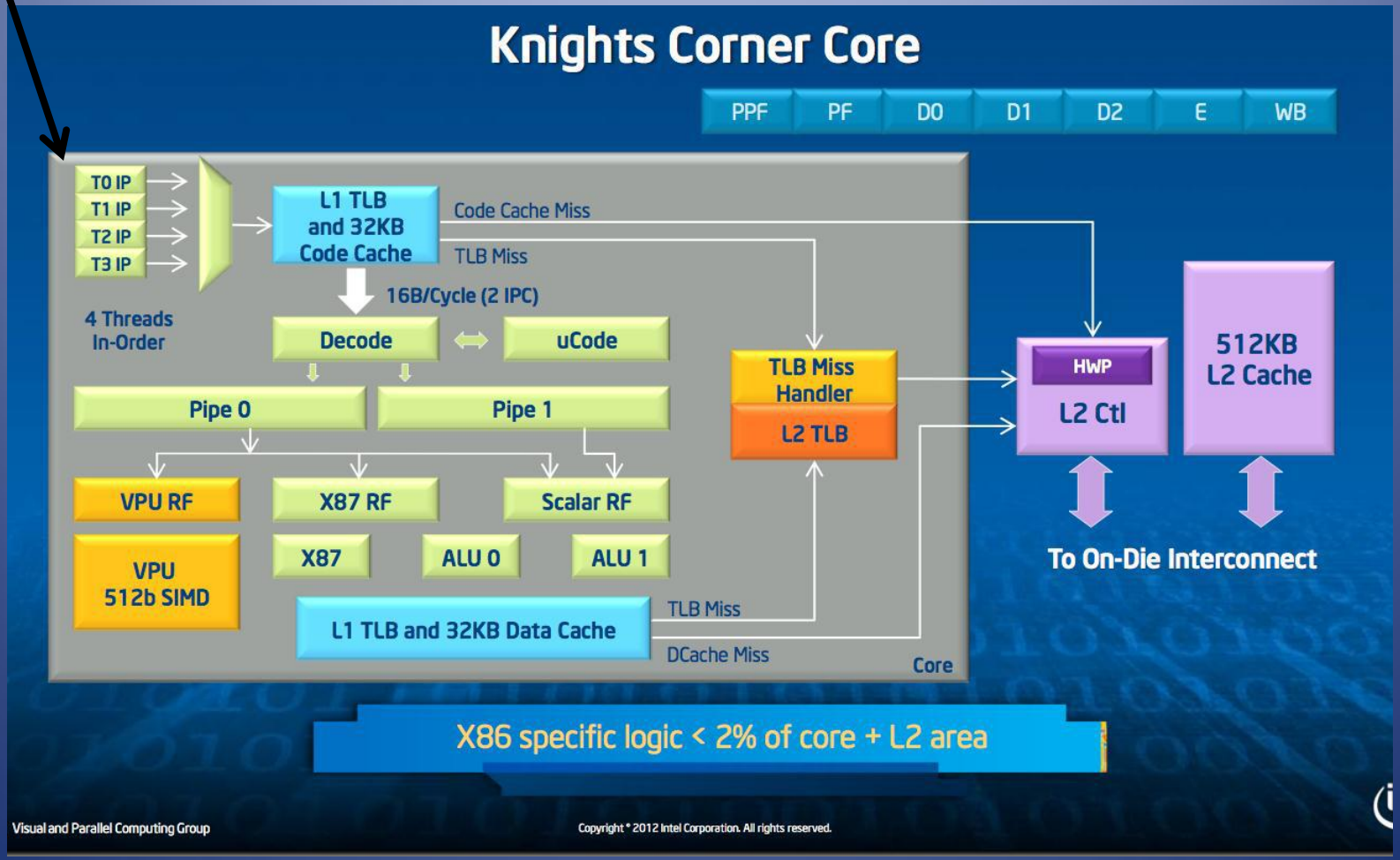
Intel Xeon Phi



George Chrysos, Intel, Hot Chips 24 (2012):
http://www.slideshare.net/IntelXeon/under-the-armor-of-knights-corner-intel-mic-architecture-at-hotchips-2012

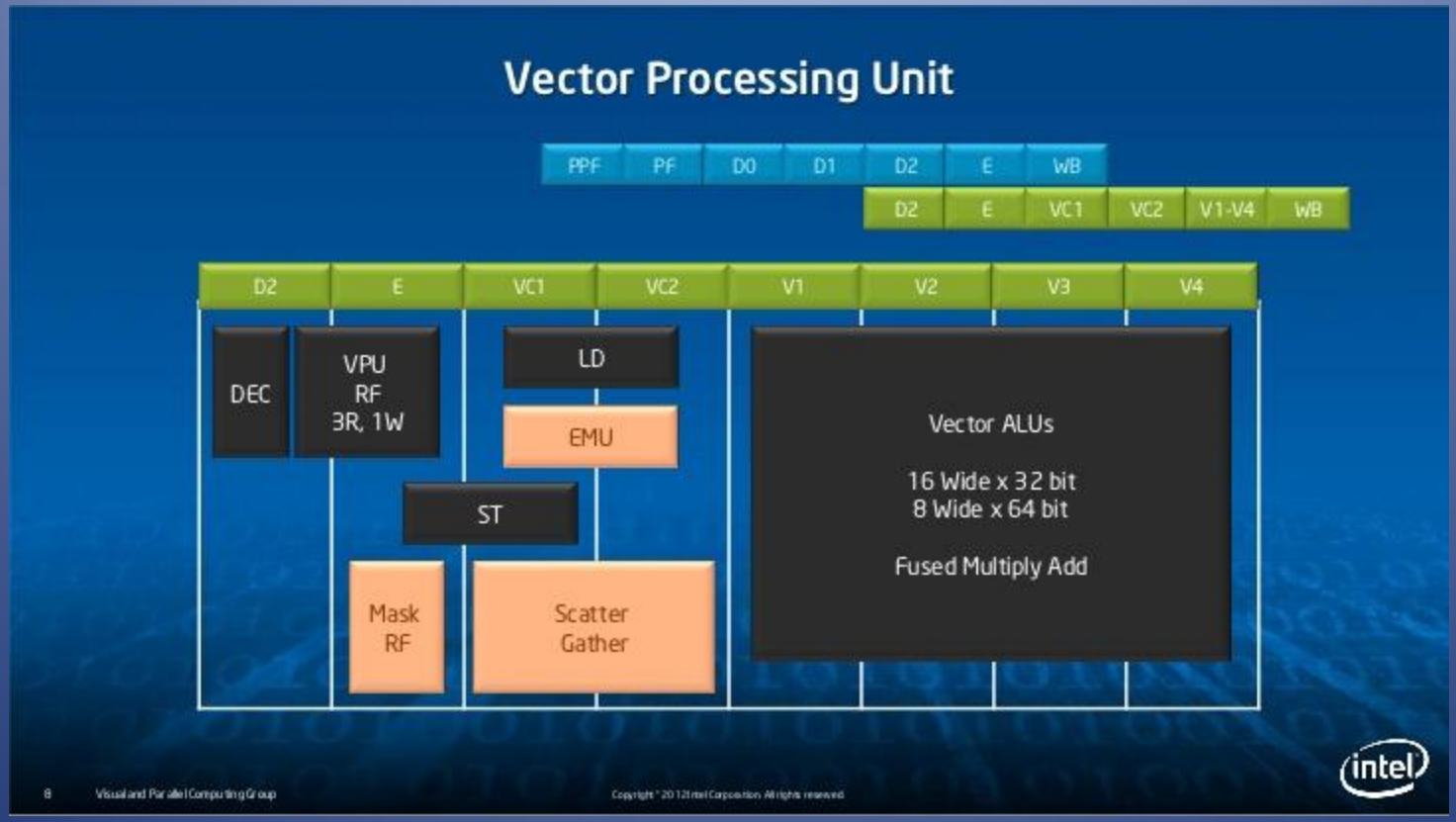
Intel Xeon Phi

4 Hardware threads/core



George Chrysos, Intel, Hot Chips 24 (2012):
http://www.slideshare.net/IntelXeon/under-the-armor-of-knights-corner-intel-mic-architecture-at-hotchips-2012

Intel Xeon Phi



George Chrysos, Intel, Hot Chips 24 (2012):
http://www.slideshare.net/IntelXeon/under-the-armor-of-knights-corner-intel-mic-architecture-at-hotchips-2012

Programming for the Intel Xeon Phi and TACC Stampede System

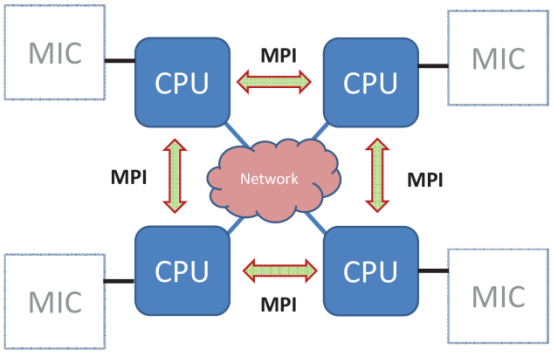
Programming Advantages

- Intel's MIC is based on **x86 technology**
  - x86 cores w/ caches and cache coherency
  - SIMD instruction set
- Programming for MIC is similar to programming for CPUs
  - Familiar languages: C/C++ and Fortran
  - Familiar parallel programming models: OpenMP & MPI
  - MPI on host and on the coprocessor
  - **Any code can run on MIC, not just kernels**
- Optimizing for MIC is similar to optimizing for CPUs
  - **"Optimize once, run anywhere"**
  - Early MIC porting efforts for codes "in the field" are frequently doubling performance on Sandy Bridge.

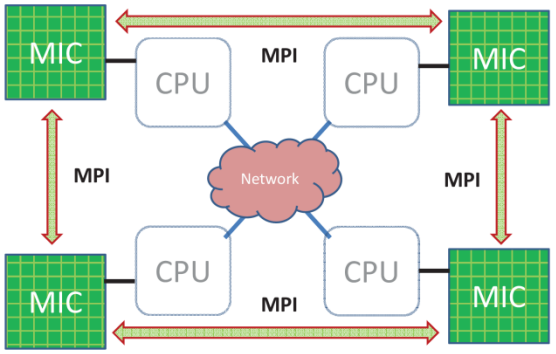
Xeon Phi Programming Models

- Traditional Cluster
  - Pure MPI and MPI+**X**
    - **X: OpenMP, TBB, Cilk+, OpenCL, …**
- Native Phi
  - Use Phi and run OpenMP or MPI programs directly
- MPI tasks on Host and Phi
  - Treat the Phi (mostly) like another host
    - Pure MPI and MPI+X
- MPI on Host, Offload to Xeon Phi
  - Targeted offload through OpenMP extensions
  - Automatically offload some library routines with MKL

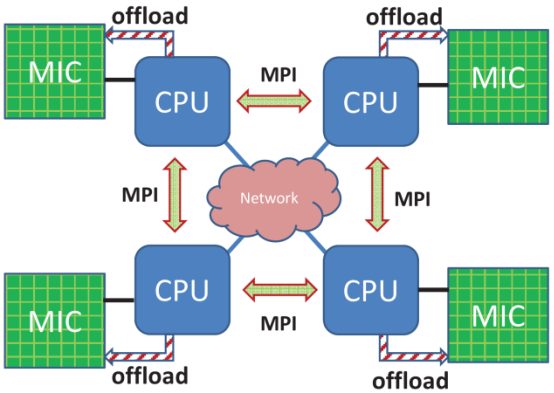
Xeon Phi Execution Models



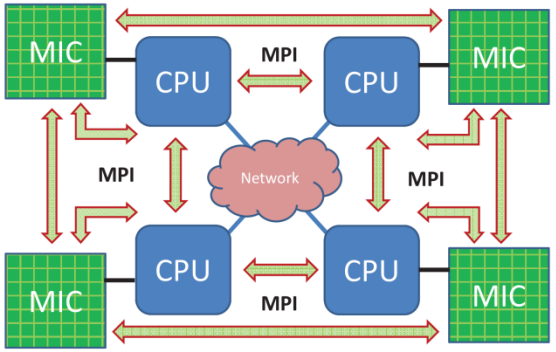
(1) Host-only Model



(2) MIC Native Model



(3) Offload Model



(4) Symmetric Model

Xeon Phi Execution Models

- **Host Only** – Ignore Xeon Phi cards
- **Native**
  - Compile with –mmic, MPI/OpenMP/Pthreads
  - Single node only, can't run multi-node jobs
  - Need one MPI rank on a host CPU
- **Offload**
  - MPI inter-node, OpenMP on-node
  - Synchronous – no overlapping
  - Asynchronous – overlapping with signal/wait
- **Symmetric**
  - MPI internode, OpenMP / Pthreads on-node
  - Use both host and co-processor simultaneously

NSF Stampede System
Texas Advanced Computing Center

- $27.5M acquisition
- 10 petaflops (PF) peak performance
- 2+ PF Linux cluster
  - 6400 Dell DCS C8220X nodes
  - 2.7GHz Intel Xeon E5 (Sandy Bridge)
    - 102,400 total cores
  - 56Gb/s FDR Mellanox InfiniBand
  - 7+ PF Intel Xeon Phi Coprocessor (1.0GHz cores)
  - 500,000+ total cores
    - TACC has a special release: Intel Xeon Phi SE10P
  - 14+ PB disk, 150GB/s
  - 16 1TB shared memory nodes
  - 128 NVIDIA Tesla K20 GPUs

TACC

XSEDE
Extreme Science and Engineering Discovery Environment

Stampede Processor Specs

Arch. Features	Xeon E5 (Sandy Bridge)	Xeon Phi SE10P
Frequency	2.7GHz+turbo	1.0GHz +turbo
Cores	8	61
HW Threads/core	2	4
Vector Size	256 bits, 4 doubles, 8 singles	512 bits, 8 doubles, 16 singles
Inst. Pipeline	Out of Order	In order
Registers	16	32
Caches	L1:32KB, L2:256KB, L3:20MB	L132KB, L2:512KB
Memory	2 GB/core	128 MB/core
Sustained Memory BW	75 GB/s	170 GB/s
Sustain Peak FLOPS	1 thread/core	2 threads/core
Instruction Set	x86+ AVX	x86+ new vector instructions

Stampede Compute Node

Comm layer &
Virtual IP Service For MIC

CPUs and MIC
appear as separate HOSTS
("symmetric"computing)

PCIe X16

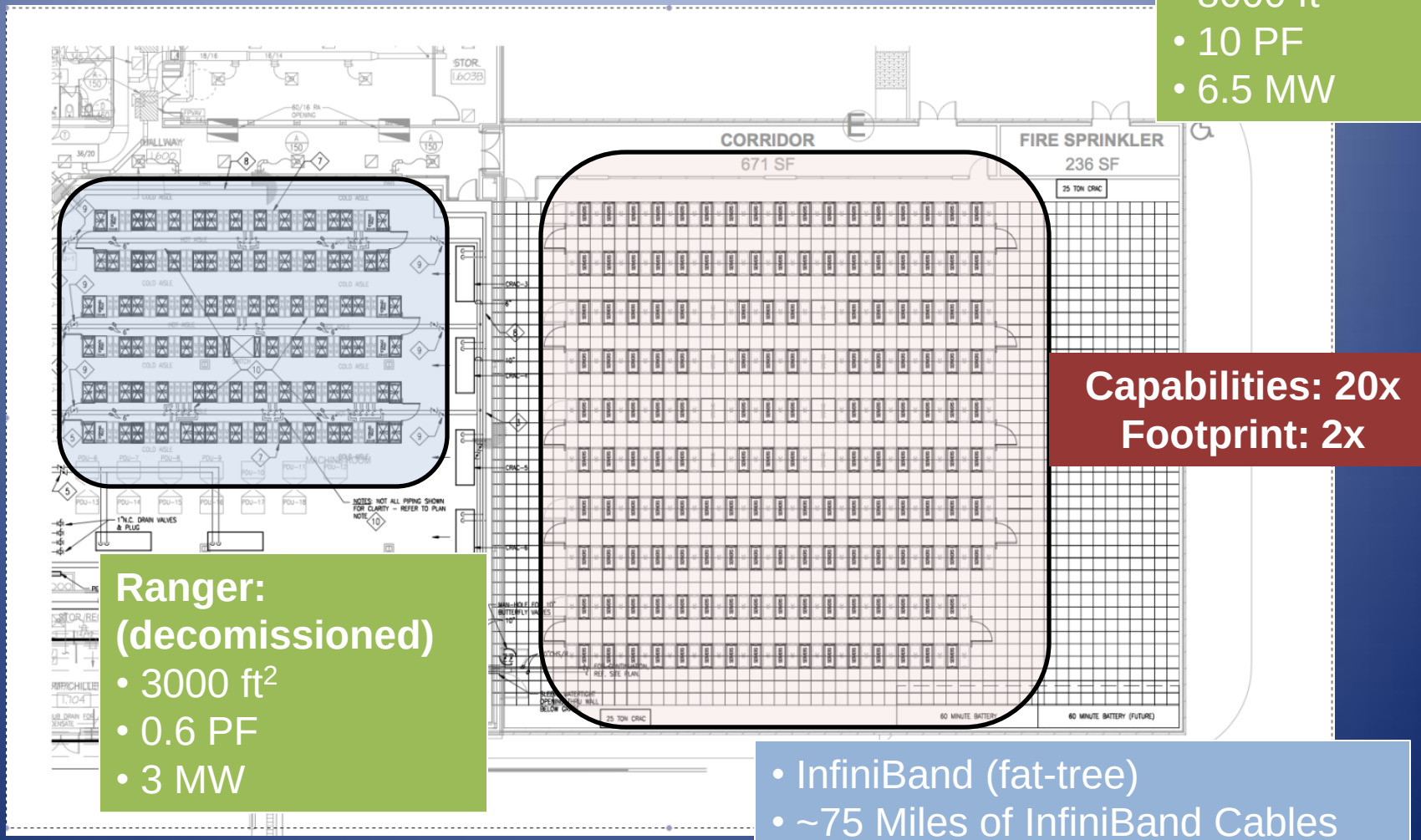
- Two Xeon E5 8-core CPUs
- 16 cores
- 32G RAM

- Xeon Phi Coprocessor
- 61 lightweight cores 8G RAM
- Each core has 4 hardware threads
- Runs micro Linux OS (BusyBox)

TACC

TACC Facilities Footprint (Stampede & Ranger)

Stampede:
- 8000 ft^2
- 10 PF
- 6.5 MW



Capabilities: 20x
Footprint: 2x

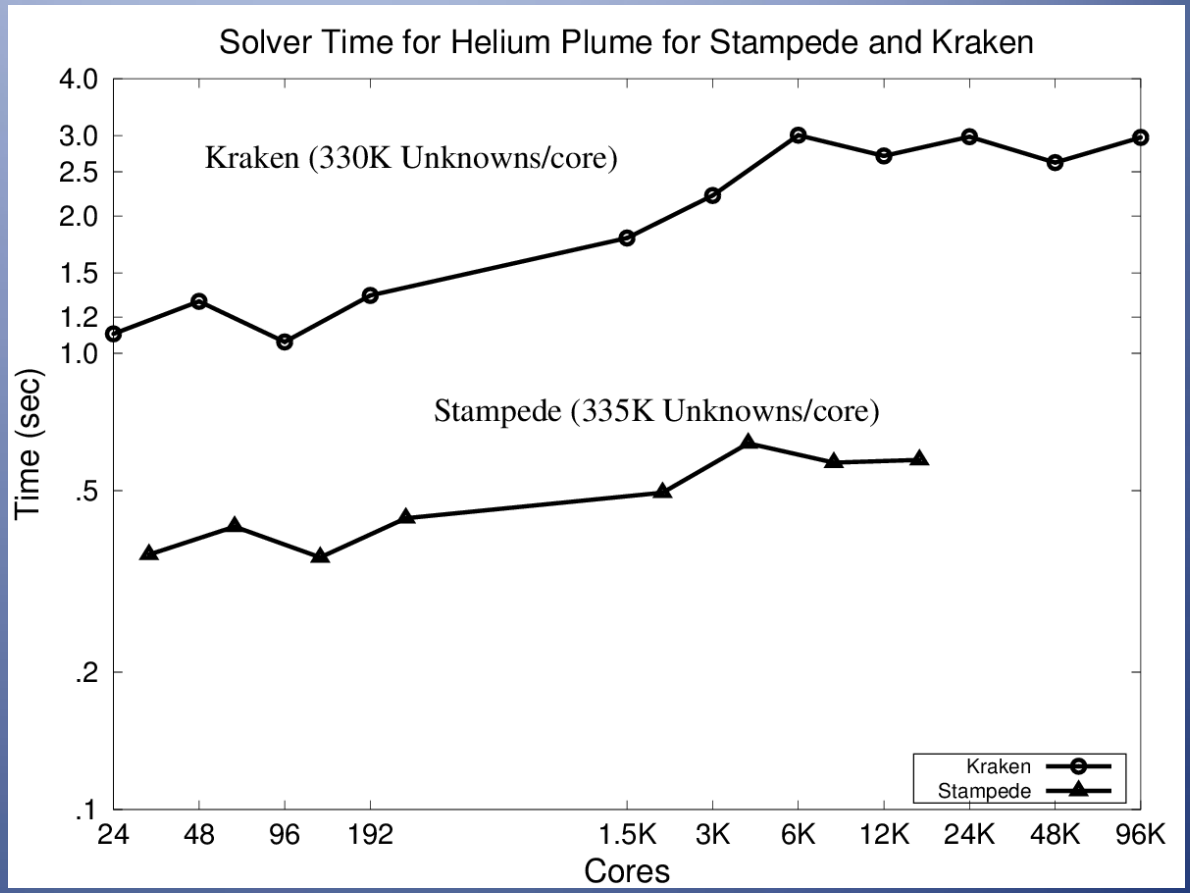
Ranger: (decomissioned)
- 3000 ft^2
- 0.6 PF
- 3 MW

- InfiniBand (fat-tree)
- ~75 Miles of InfiniBand Cables

Stampede footprint figure courtesy, Bill Barth, TACC

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

Host-only Model

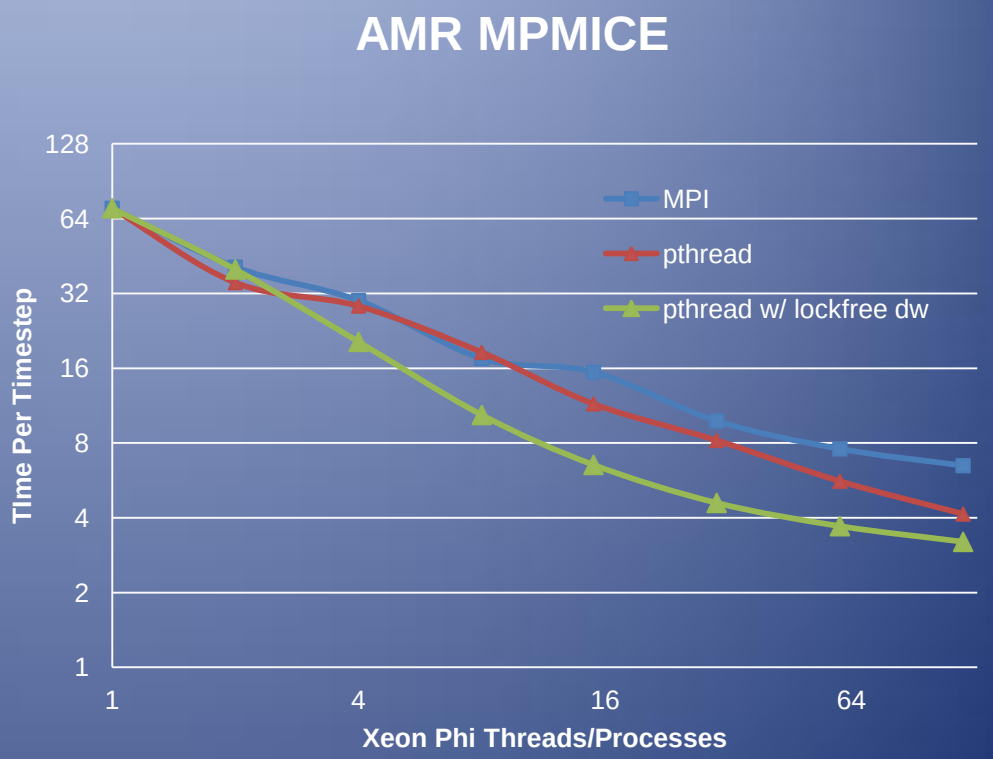


- Intel MPI issues beyond 2048 cores (seg faults)
- MVAPICH2 required for larger core counts

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

Uintah on Xeon Phi Native Model

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

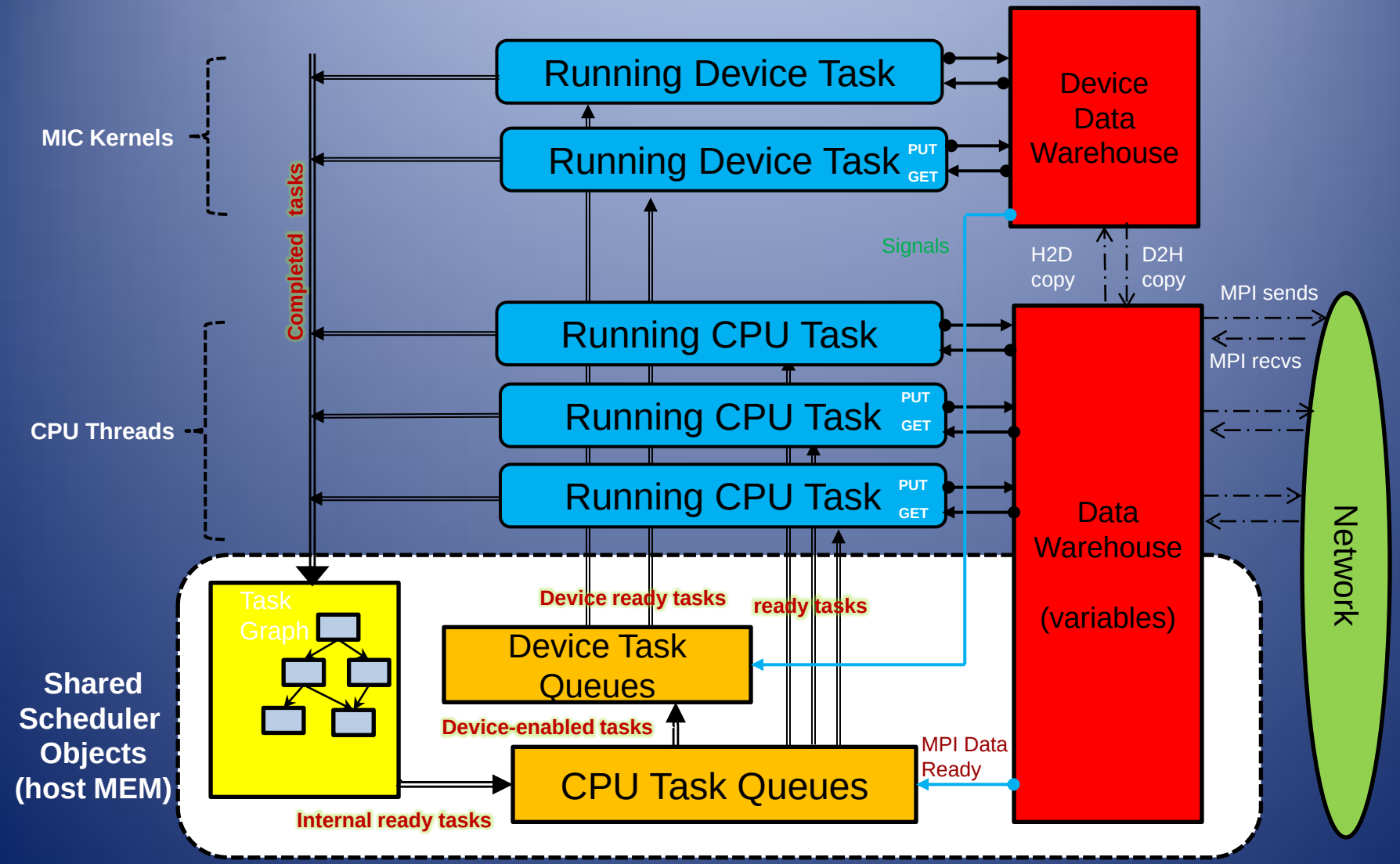


2 MPI processes per Phi core , up to 120 processes

2 threads per Phi core, up to 120 threads

Lock-free multi-threaded MPI

Unified Heterogeneous Scheduler & Runtime Offload Model



Uintah on Xeon Phi Offload Model

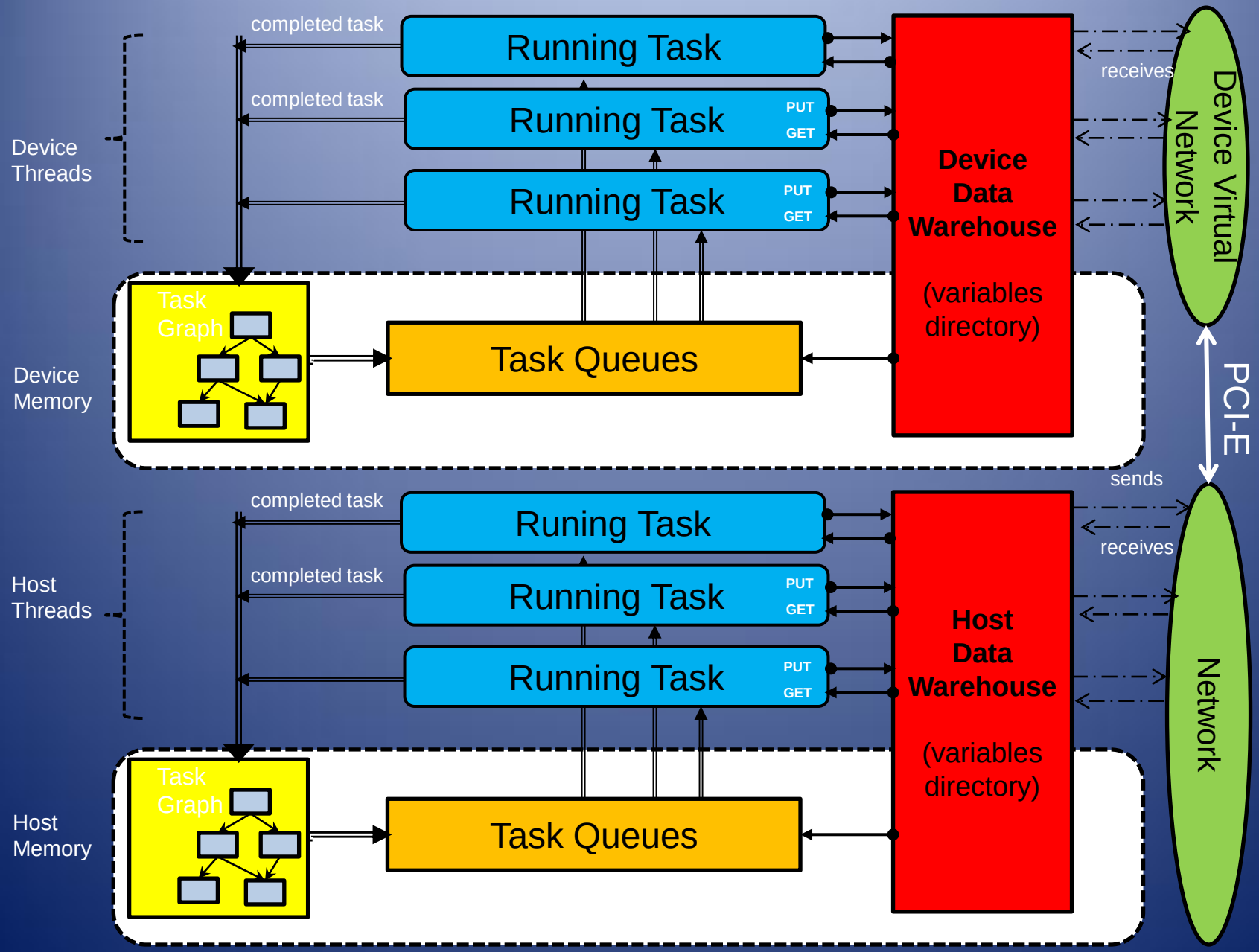
- Use compiler directives (#pragma)
  - Offload target: #pragma offload target(mic:0)
  - OpenMP: #pragma omp parallel
- Find copy in/out variables
- Functions called in MIC must be defined with __attribute__((target(mic)))
- Hard for Uintah to use offload mode
  - Rewrite highly templated C++ methods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

Uintah on Xeon Phi Symmetric Model

- Best fits current Uintah model
- Xeon Phi directly calls MPI
- Two MPI processes per node:
- 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

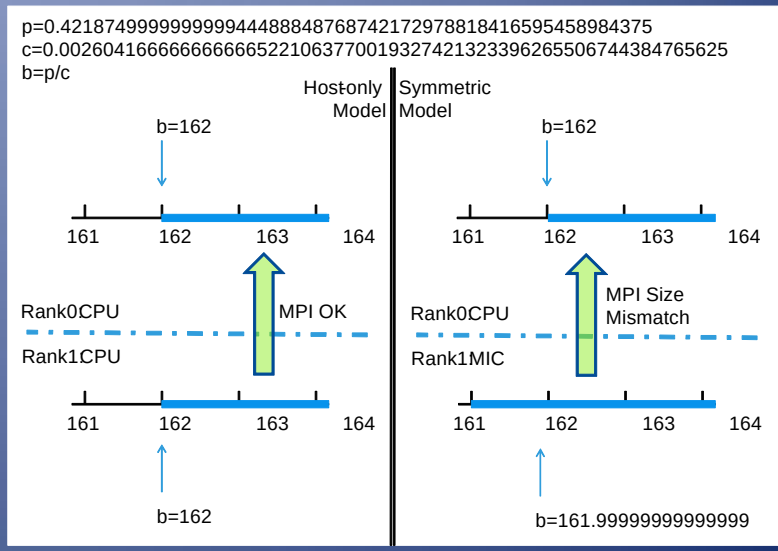
  mpiexec.hydra -n 8 ./sus – nthreads 16 : -n 8./sus.mic –nthreads 120
- No major Uintah code changes

Unified Heterogeneous Scheduler & Runtime (symmetric)



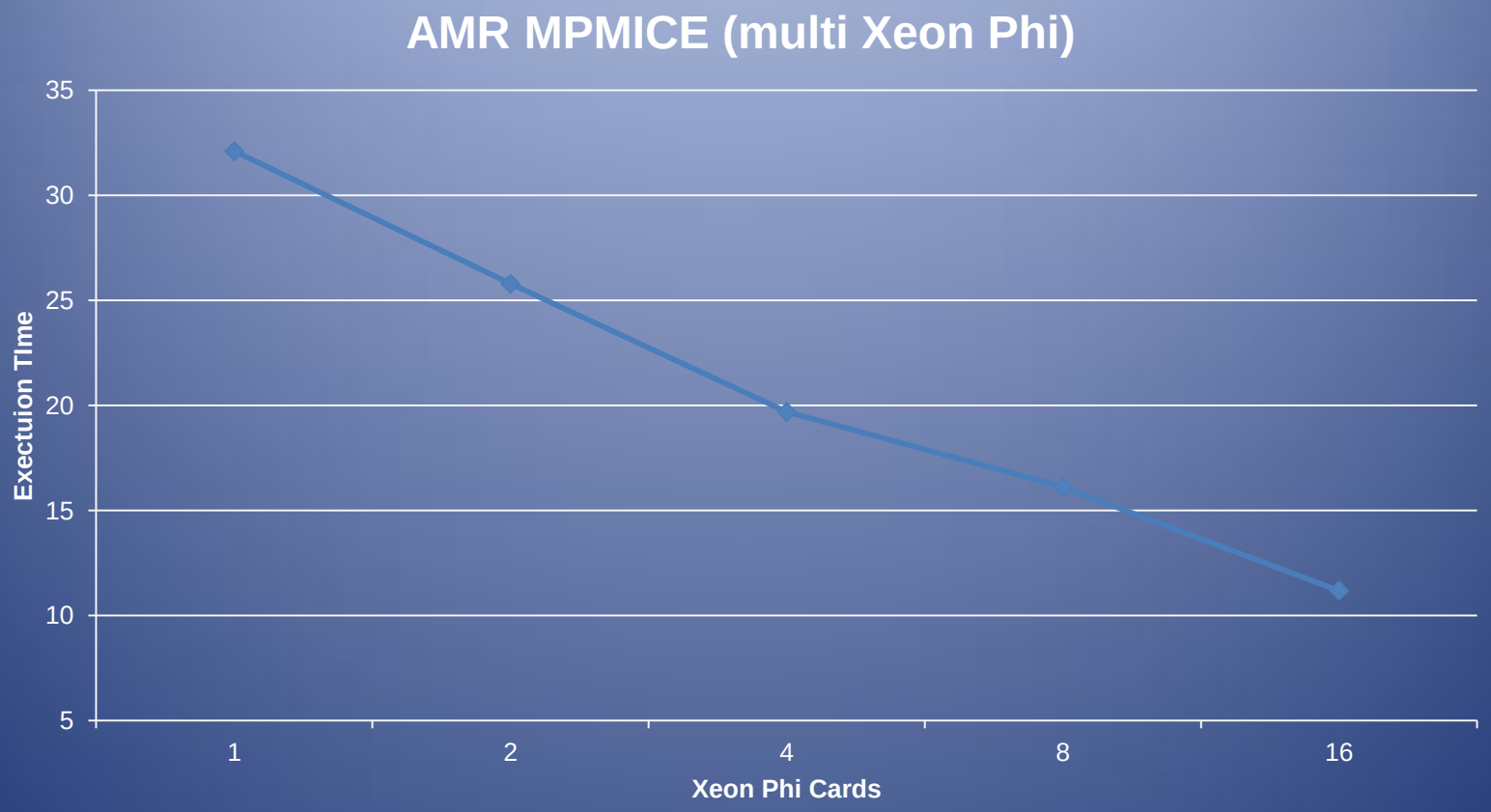
Symmetric Model Challenges

- **Native debugging with *gdb***
  - Built **gdb** for ourselves from source
  - Weren't aware of **idb** / **idbc** (need idb server/client setup)
- **Different floating point accuracy on host and co-processor**



- **MPI message mismatch issue**
  - Any control logic based on result of FP calculation may choose different execution routes on Xeon Phi and host
  - Size of MPI send/rcv buffers determined by FP result
  - Caused Uintah MPI message mismatch (MSG_TRUNC)

Scaling Results on Xeon Phi Symmetric Model



60 threads per MPI process, 2 MPI processes per Xeon Phi card
16 threads per MPI process, 1 MPI processes per host CPU

Scaling results limited by development queue allowances for symmetric model

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)

Questions?

http://www.uintah.utah.edu

SCI INSTITUTE