

# Portable Scalability and Performance at Exascale using Asynchronous Tasking

**Martin Berzins**

John Holmen (ORNL, ex SCI), Allen Sanderson (SCI), Marta Garcia (ANL),  
INTEL (Mike D'Mello), ANL, Kokkos, Damodar Sahasrabudhe (ex SCI now Cerebras)

Nitish Shingde (SCI), Timothy Blattner (NIST), Walid Keyrouz (NIST),  
Alexandre Bardakoff (NIST – now RIKEN), ANL, NSF ,TACC

Alan Humphrey deserves a special mention for his work on Uintah,  
particularly the thermal radiation model used here.

Thanks also go to the 50+ people involved with Uintah since 1998

## Overview

- (i) Task-based parallelism and Uintah
- (ii) Large Scale Combustion and thermal radiation ray-tracing
- (iii) Achieving high performance with task-based parallelism and Hedgehog
- (iv) Futures and Conclusions

Uintah asynchronous tasking gives **portable performance across Exascale machines** even for low compute intensity applications

Extension of NIST Hedgehog asynchronous task code has a **high percentage of GPU peak** competitive with best GPU codes DPLASMA SLATE for DGEMM

Task-based approaches are also promising for future heterogeneous machines

# Why exascale?



1.012 Exaflops 63774 GPUs with 3.27TB/s HBM to execution per GPU



1.35 Exaflops 37608 GPUs with 3.2TB/s HBM to execution per GPU

## Observations

Frontier is faster on flops but **Aurora has 1.7x the memory bandwidth of Frontier**

It takes about 10 years for the compute power of the top machine in the TOP 500 to be that of the 100<sup>th</sup> Top 500 machine

Developing for the leading edge provides **a decade long software advantage**

# (i) Task-Based Parallelism

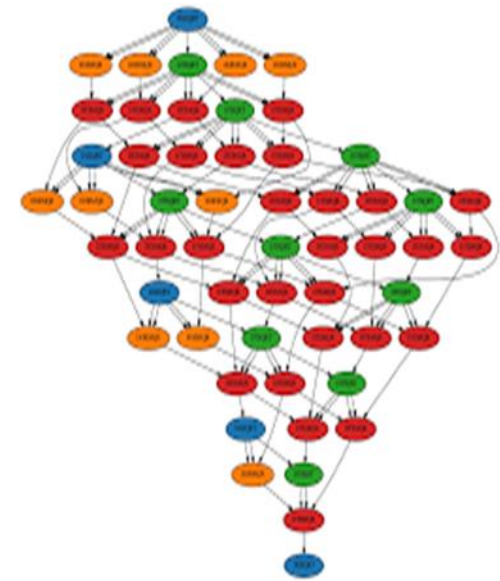
## What didn't ECP do?

Links to AI and Quantum

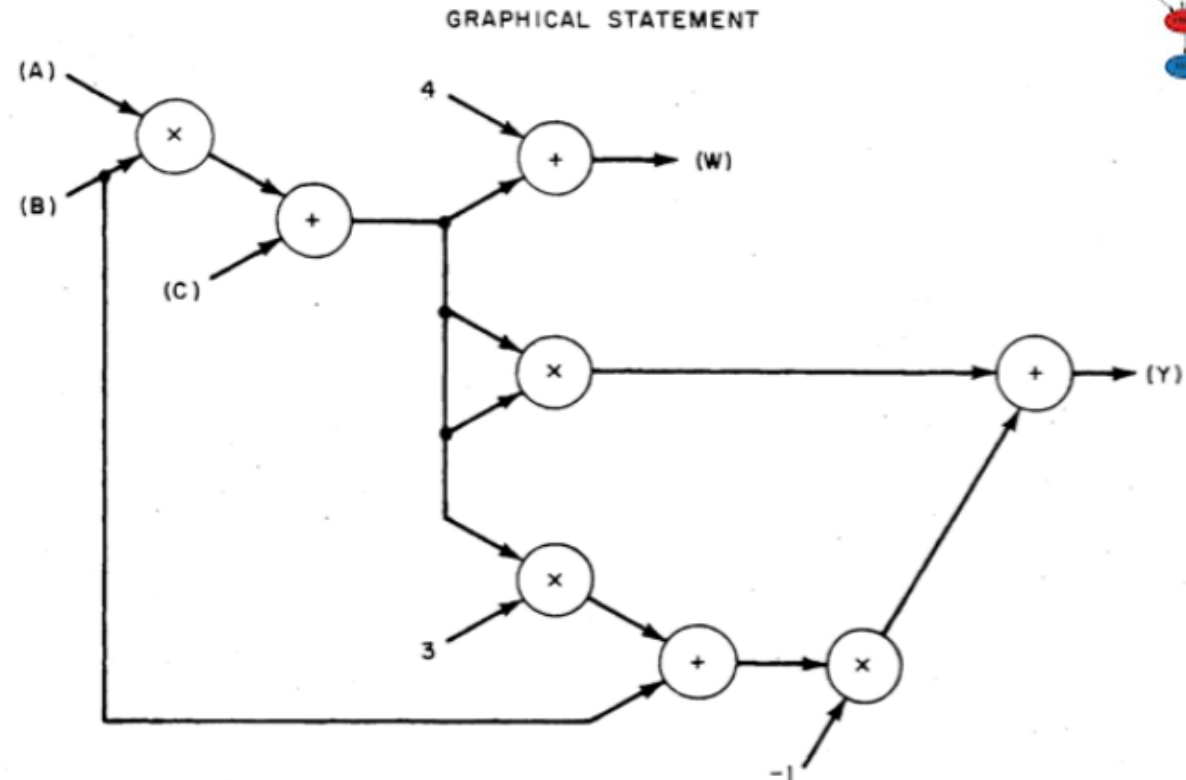
“Limited Progress in task-based and data flow architecture.”

Doug Kothe Salishan 2025  
(personal non-DOE view)

## Original Dataflow Sutherland 1966



Written code  
 $Z = A \times B + C$   
 $Y = Z \times Z - (3Z + B)$



# STANDARD MPI-BASED MESSAGE PASSING

MPI Domain  
Decomposed Program  
Data Structures **A,B**  
Partitioned mesh+halo

MPI communication

$$\text{TASK D} = \text{AB} + \text{C}$$

MPI communication

$$\text{TASK F} = \text{A/B} + \text{E}$$

MPI Communication

$$\text{TASK A} = \text{A} + \text{D} + \text{F} + \text{G}$$

MPI

MPI READ C

MPI SEND D

MPI READ E

MPI SEND F

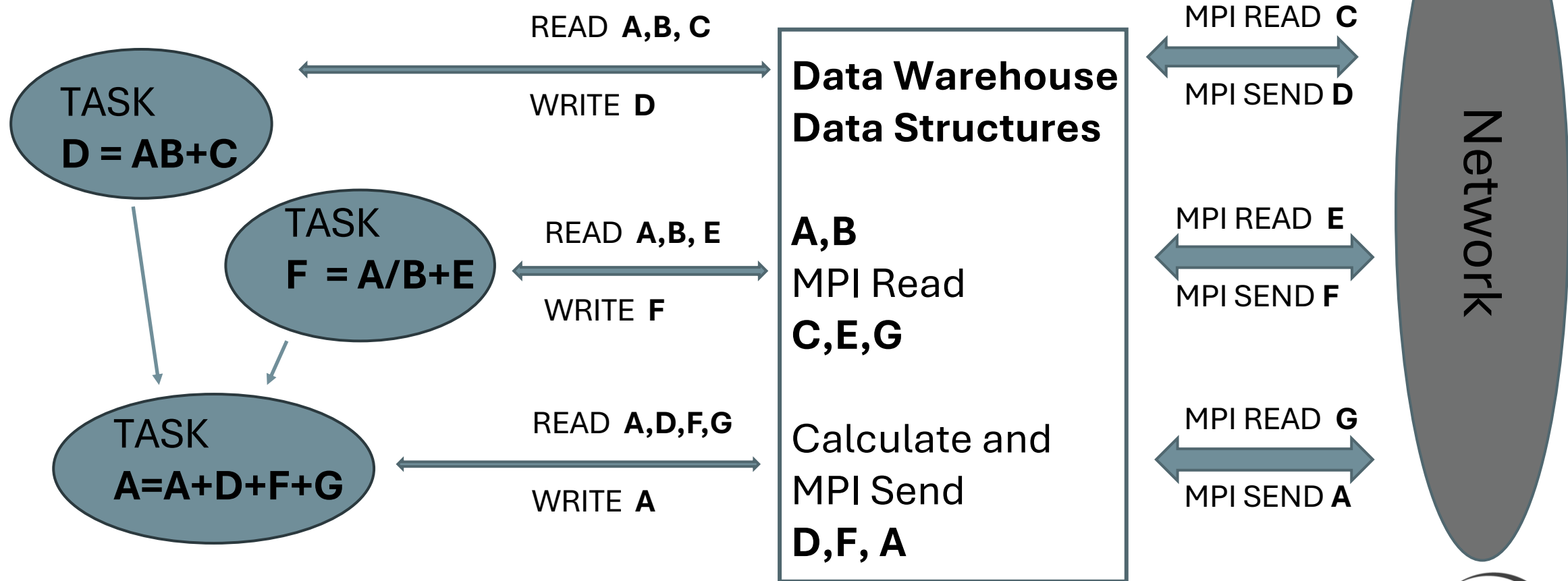
MPI READ G

MPI SEND A

Network

# TASK BASED EXECUTION

First two tasks can execute asynchronously in any order and in parallel depending on when inputs arrive. Limits task waiting.



# Asynchronous Many Task Runtime System Examples

Legion – Stanford

Charm++ - Illinois

Uintah - University of Utah

STAPL – Texas A & M

LFRIC - UK met Office

PaRSEC - Tennessee/ORNL/INRIA

StarPU - Barcelona/INRIA

FleCSI - LANL

HPX - Indiana / Louisiana

Hedgehog - NIST

## Key features of all system:

Adaptive execution of tasks

Ability to hide communications costs including delays

Ability to address heterogeneity

Task specification may not change as code ported however some of runtime system may

# Asynchronous Many Task Runtime Systems

So why isn't everybody doing this?

## Pros:

Asynchronous adaptive execution helps automatic scalability by reducing waiting

The more complex the application the better this works

## Cons:

Many apps can be made/forced to scale with just MPI and enough effort

A runtime is needed to manage asynchronous execution.

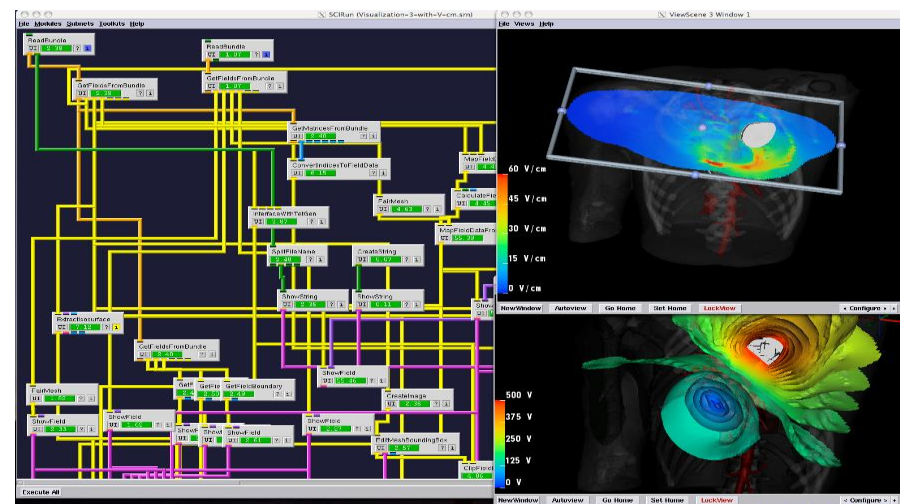
There has to be a clean and careful separation between data and execution

It is hard to transform an existing legacy C++ MPI code to work in this way

# Uintah development timeline

Task based approach by [Steve Parker](#)† originated in SCIRun

Simple programming model- separation of physics and computer science.



- 1998-2007 CSAFE ASCI - **static execution** of task graphs, complex multi-physics . † 1968 - 2024

[Steve goes to NVIDIA](#) ☹️ [MB stepped in.](#)

- 2008-2010 CSAFE Full physics, AMR for fluid-structure [Luitjens, Guilkey, Harman](#)

- 2010-2015 **Adaptive asynchronous. out-of-order task execution on explosions:**

[Luitjens, Meng, Humphrey,](#)

- 2014-2019 PSAAP2 Center Turbulent Combustion –

**full scalability on Titan Mira, Blue Waters – GPU runtime scalability on Nvidia GPUs**

[Humphrey, Sunderland, Peterson, Holmen, Sahasrabudhe](#)

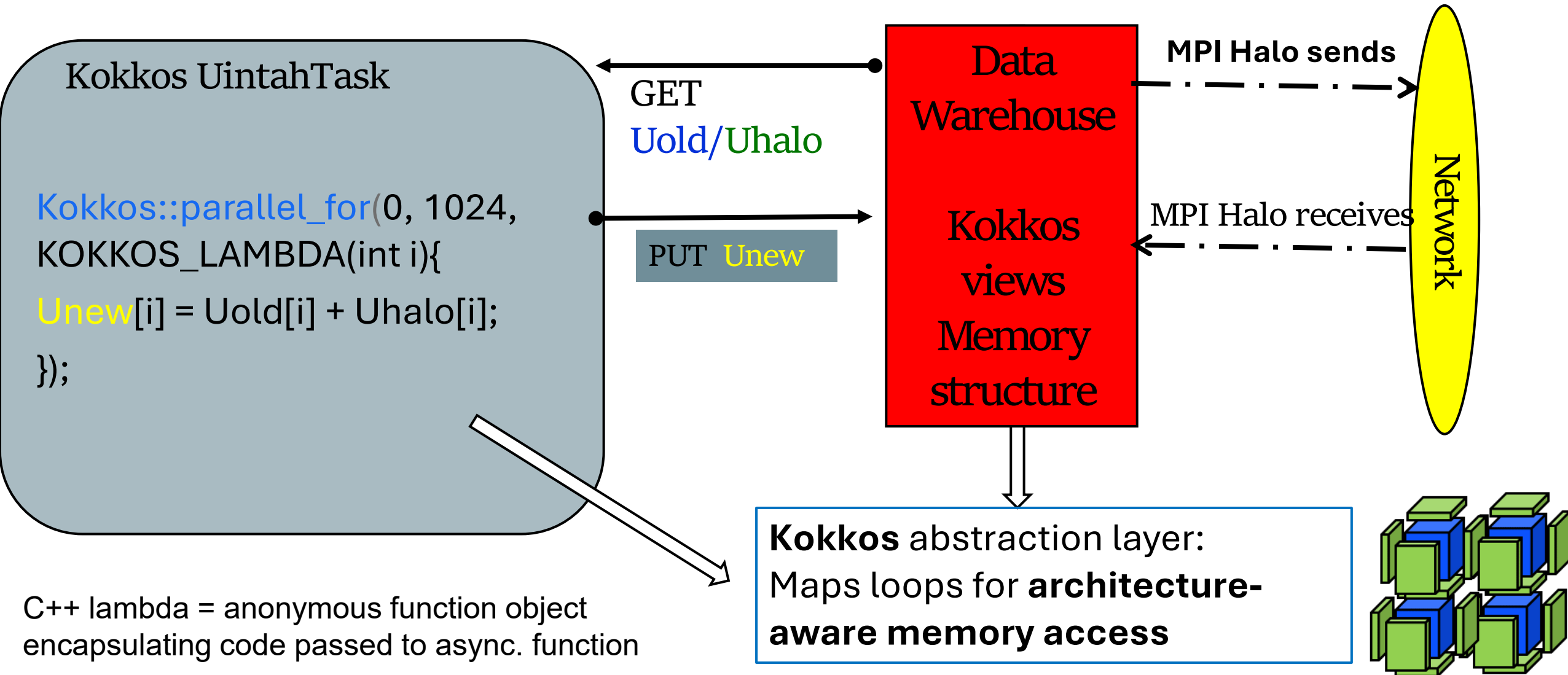
- 2020 -2025 **Fully portable scalability at exascale** [Holmen, Sahasrabudhe, Sanderson](#)



# Kokkos Performance Portability Library

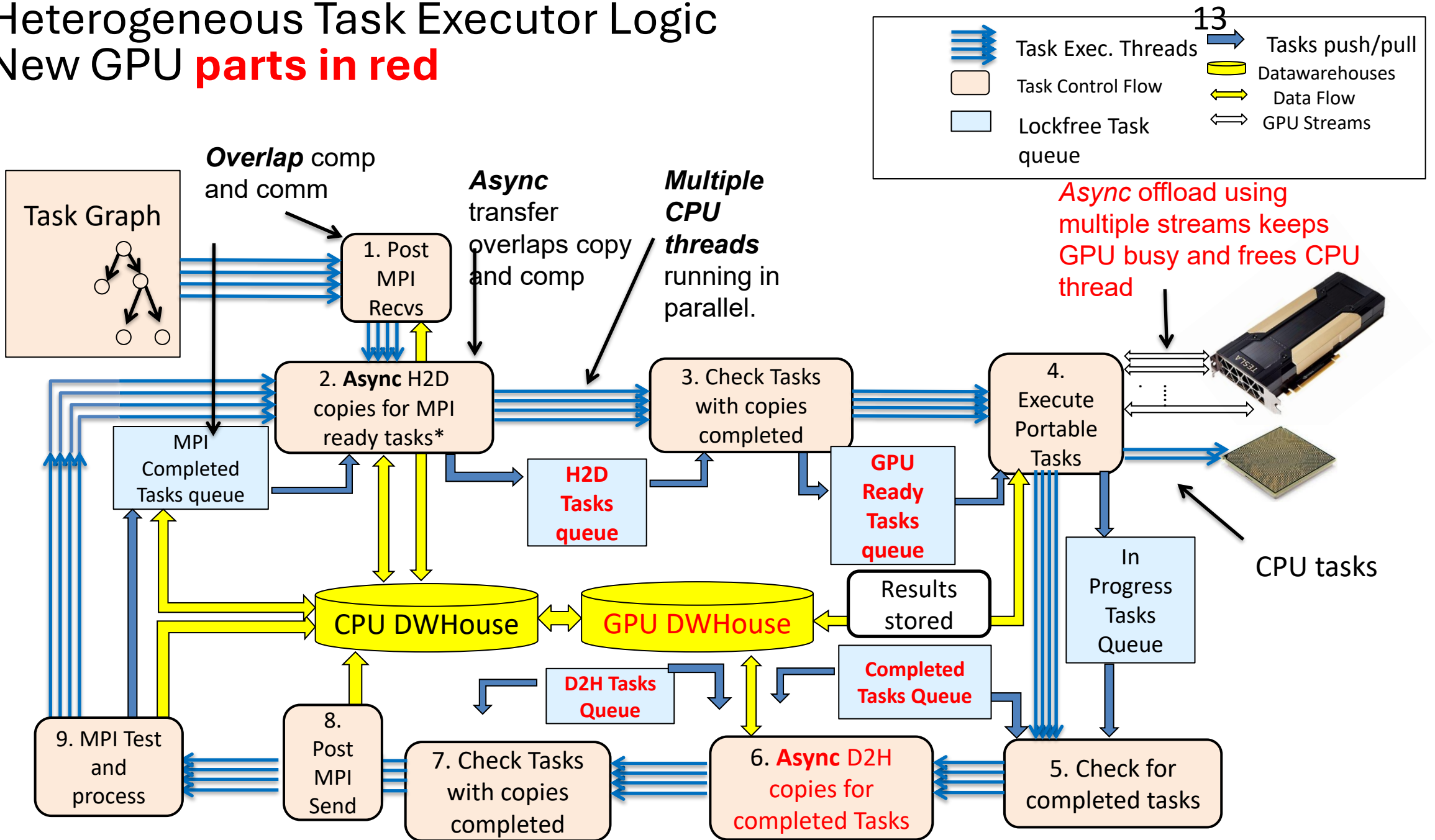
- DOE C++ library for portable, thread-scalable code optimized for CPU, GPU,.....2016 -
- Back-ends to e.g. **CUDA, OpenMP, HIP, and SYCL**
- Provides abstractions to control:
  - how/where kernels are executed,
  - How data is mapped to memory (**Kokkos Views**)
- **Developers still have to write performant code**
- High-level calls mapped to low-level Kokkos calls (legacy interface too [Holmen])
- Calls compiled for target Kokkos back-ends via **C++ template metaprogramming**
- **Widely used, e.g. even in climate codes from China.**
- **Linked to C++ standard 202X**

# Example: Uintah Model for Stencil Timestep



# Heterogeneous Task Executor Logic

## New GPU parts in red



\* If needed, depending on the dependencies / ghost cells

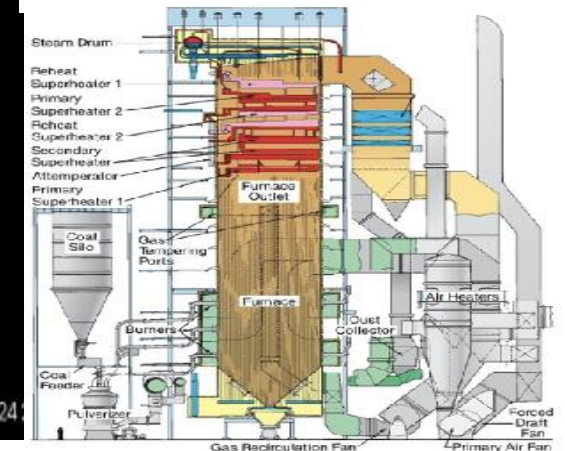
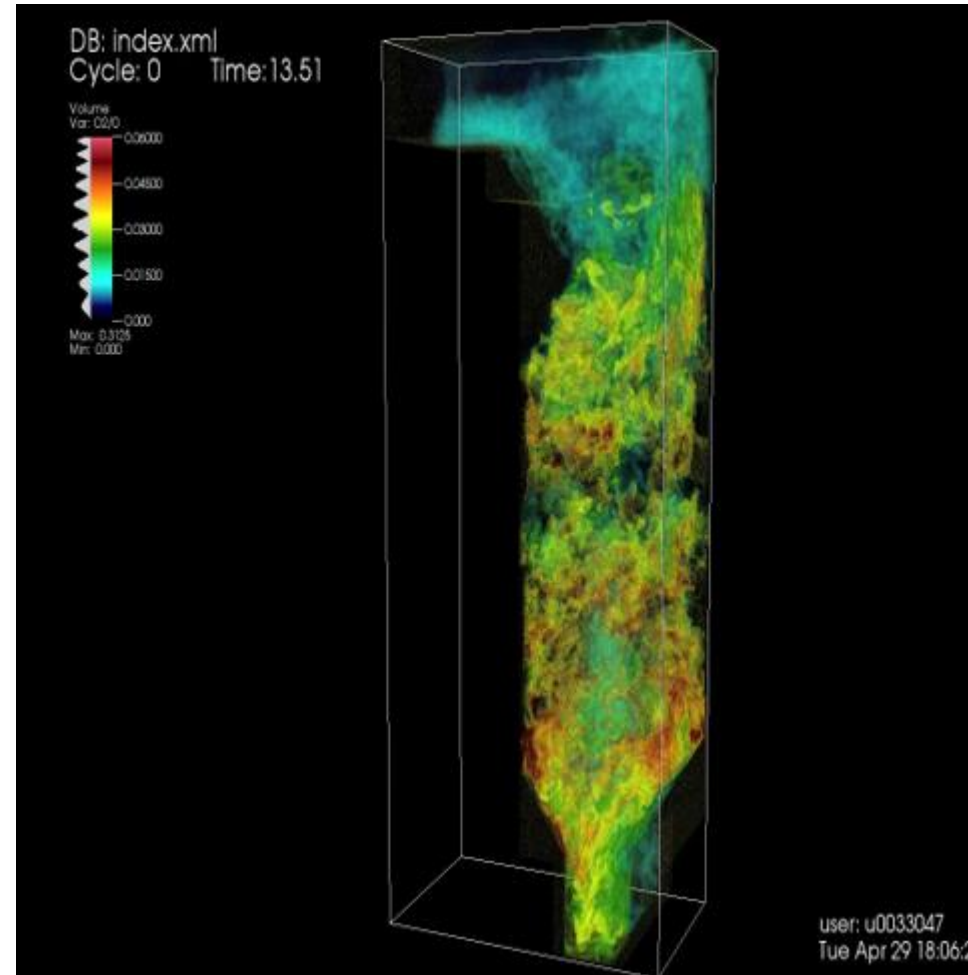
## (ii) Thermal Radiation and Combustion Modeling

# Uintah Arches Applications [Smith(s), Spinti, Thornock +....]

- GE Power 1000MWe “Twin Fireball” boiler
- Supply power for 1M people
- 1mm grid resolution =  $9 \times 10^{12}$  cells
- Combustion pdes standard CFD loops
- Multigrid linear solve in low-mach model
- Coupled particles and algebraic equations
- **Thermal Radiation is the main heat transfer mechanism**

PSAAP problems GE-Alstom boiler 2014-19

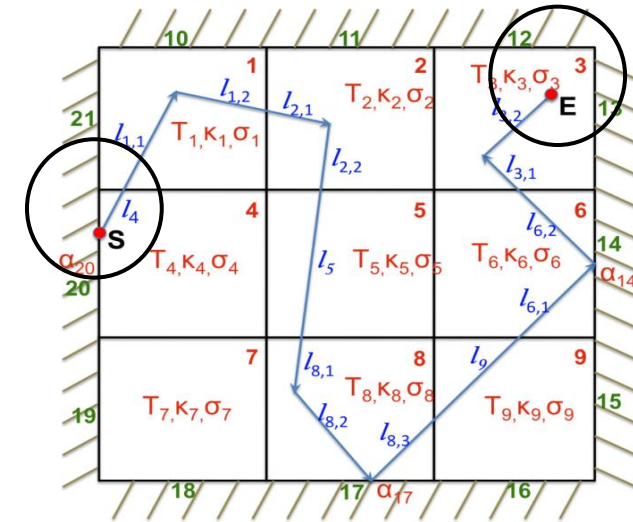
Digital Twin of wood-fired Atikokan Boiler 2020-25



- Heat flux requires integration of incoming intensity about a solid angle with reverse Monte Carlo ray tracing (RMCRT)  $\longrightarrow$
- Radiation-energy coupling incorporated by radiative source term
- Energy equation Temperature field,  $\mathbf{T}$  used to compute **net radiative source term**

$$\int_{4\pi} I_{in} d\Omega \Rightarrow \sum_{ray=1}^N I_{ray} \frac{4\pi}{N}$$

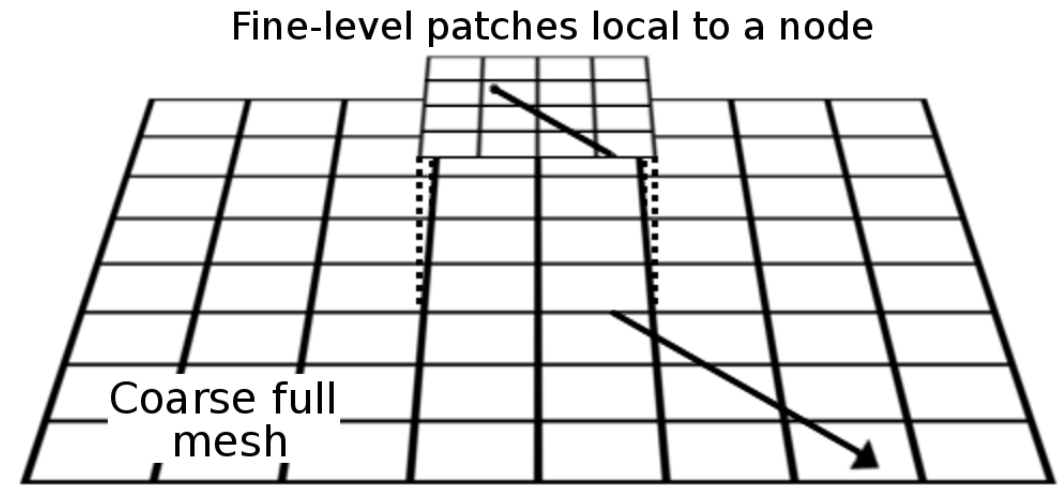
- Rays may traverse multiple patches and compute nodes
- **A Single strong scaling with 2x nodes may double the number of nodes traversed by a ray and hence doubles the overall length and time of communications – destroys scalability**
- Contrast this with stencil calculations in which communications volume doubles but not time taken as all communication is local



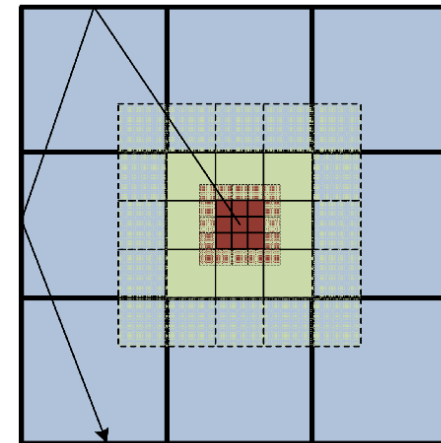
Communications intensive as only 0.7 flops per word

# Multi-Scale Adaptive Mesh RMCRT

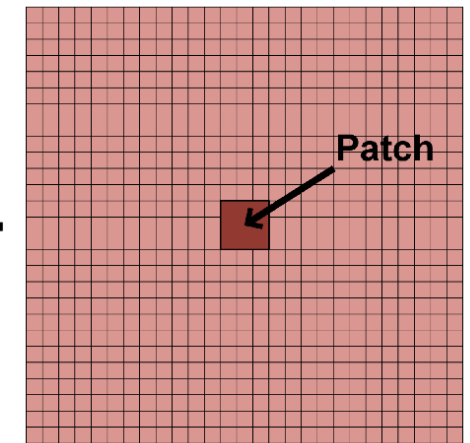
- Each nodes fine mesh is augmented with a **FULL coarse global grid**
- As rays travel away from patch, the stride taken between cells is larger
- Ray Tracing is now local
- **Transmit new information relating to heat fluxes adsorption and scattering coeffs after the raytracing calculation has finished via MPI all-to-all.**
- This reduces computational cost, memory usage and MPI message volume. Makes scalability provably possible



RMCRT Using 3-level Mesh Coarsening Scheme



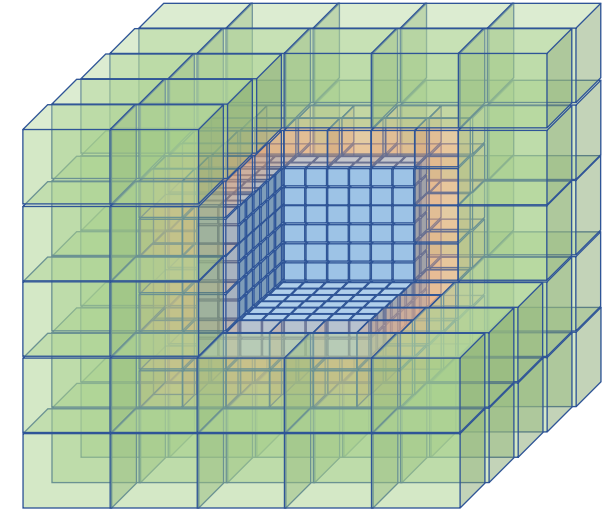
CFD Level Fine Resolution



**RMCRT - 2D diagram of three-level mesh refinement scheme, illustrating how a ray from a fine-level patch (right) might be traced across a coarsened domain (left).**

# Ray Tracing Summary

- (i) Raytracing on a uniform mesh **provably** does not strong scale
- (ii) Raytrace to local halos as in PDE calculation
- (iii) Keep a global coarse mesh on each GPU
- (iv) Calculate coarse mesh heat fluxes on coarse mesh and transmit in coarse mesh MPI all-to all
- (v) **Provable** strong and weak scaling if the global coarse mesh on each GPU is coarse enough



**Full physics using multi-level GPU-RMCRT scales on DOE ORNL Titan using Cuda custom coding gave better than perfect strong scaling**

Cores/GPUs	16K/1K	32K/2K	64K/4K	128K/8K	256K/16K
Time (sec)	821	407	202	99	55
Perfect scaling	821	410	205	102	51

# Moving to Summit, Frontier and Aurora 2017 - 2025

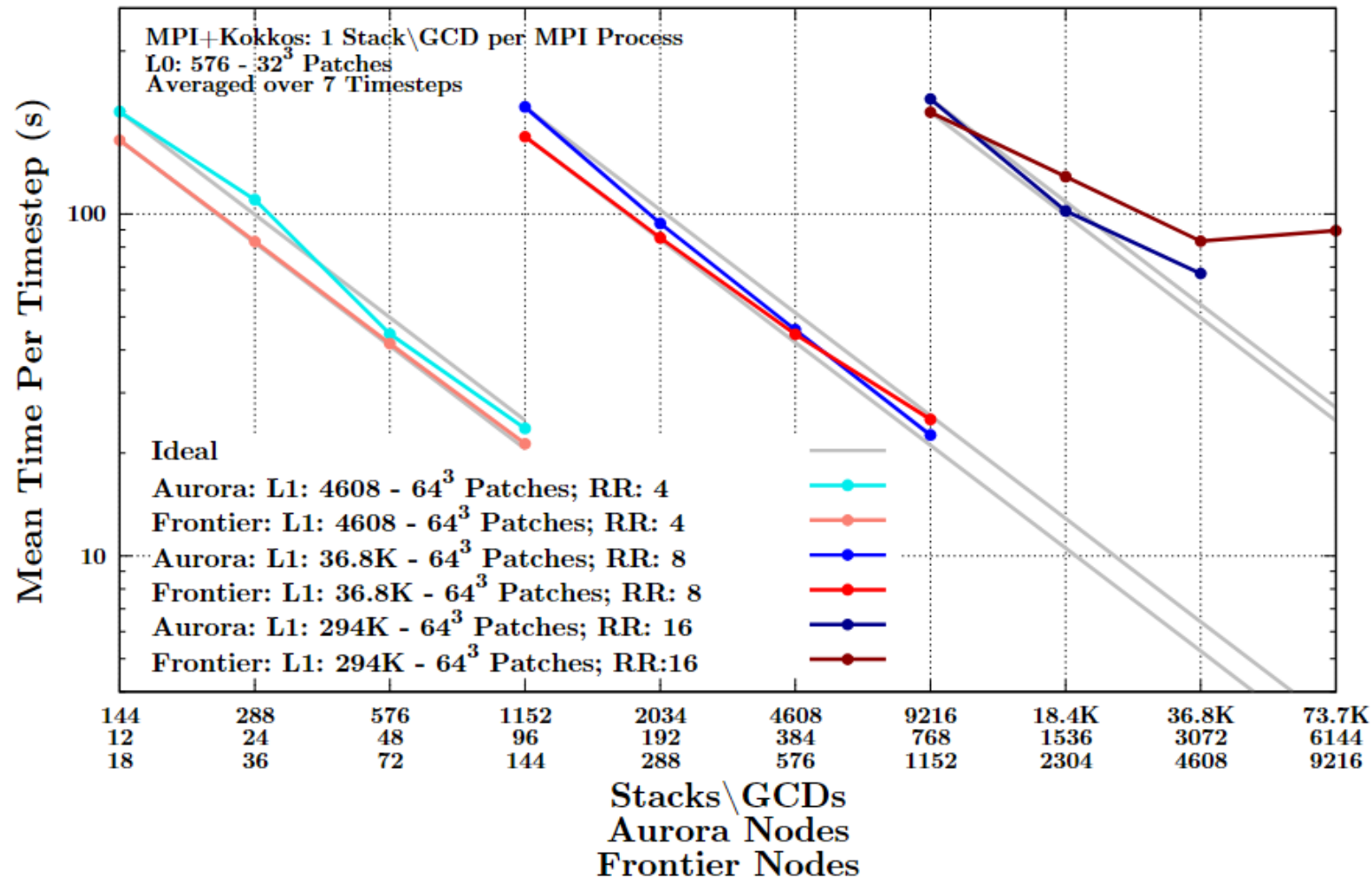
- (i) ORNL Summit port by John+Damodar after porting to LLNL Lassen . Developed combustion benchmark coupling cfd loops, linear solve (hypre) and ray-tracing radiation.
- (ii) Marta, John, Allen and Intel worked very hard to make the Aurora port happen in a very dynamic environment.
- (iii) Lots of support from Kokkos team and Christian Trott.
- (iv) Allen rewrote Uintah runtime to use Kokkos.
- (v) High level of systems support from Argonne and Intel.
- (iv) In 2024 everything started to work well and many runs at almost full scale on Aurora happened routinely- one of the first applications groups to do this

# Aurora\* & Frontier Strong-Scaling Studies Full Benchmark

- CFD benchmark loops +hypre MG solve+radiation
- Problem redesigned with aggressive mesh coarsening when weak-scaled
- Improved weak-scaling
- Excellent to OK strong-scaling but scalability barriers in Arches
- Encouraging results on first attempts to run at scale with largely naïve port of Arches

\* This work was done on a pre-production supercomputer with early versions of the Aurora software development kit

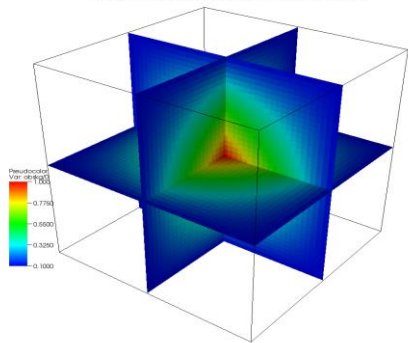
Modified Burns and Christon Benchmark - Strong Scaling  
Arches - Hypre - 2-Level RMCRT



# Thermal Radiation 3D Ray Tracing Aurora

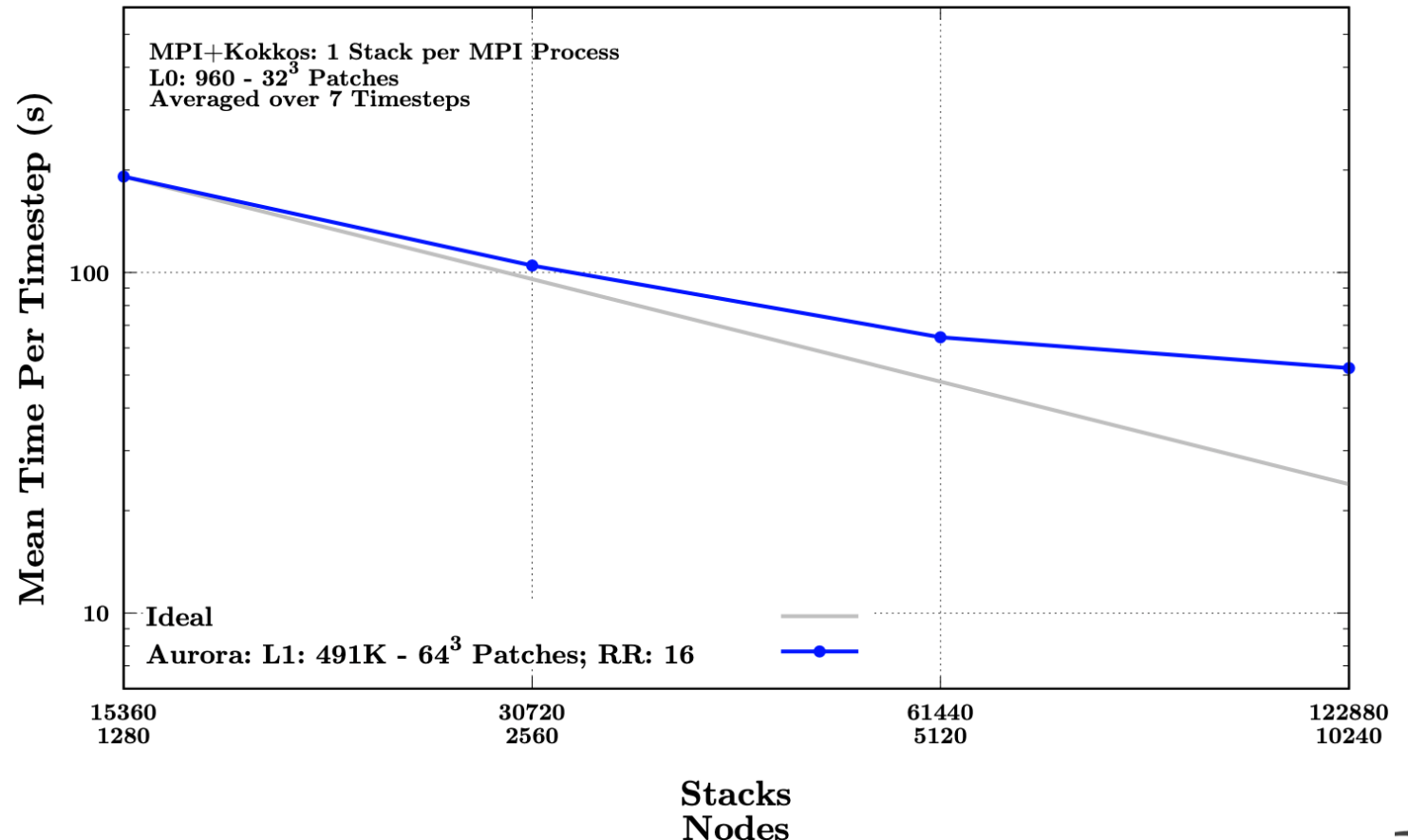
Global unpredictable all-to-all communications, low compute intensity. Strong scaling: more nodes with fewer cells – more communications

Isothermal unit cube: with cold and black walls, and an interior of gray, nonscattering, absorbing/emitting material

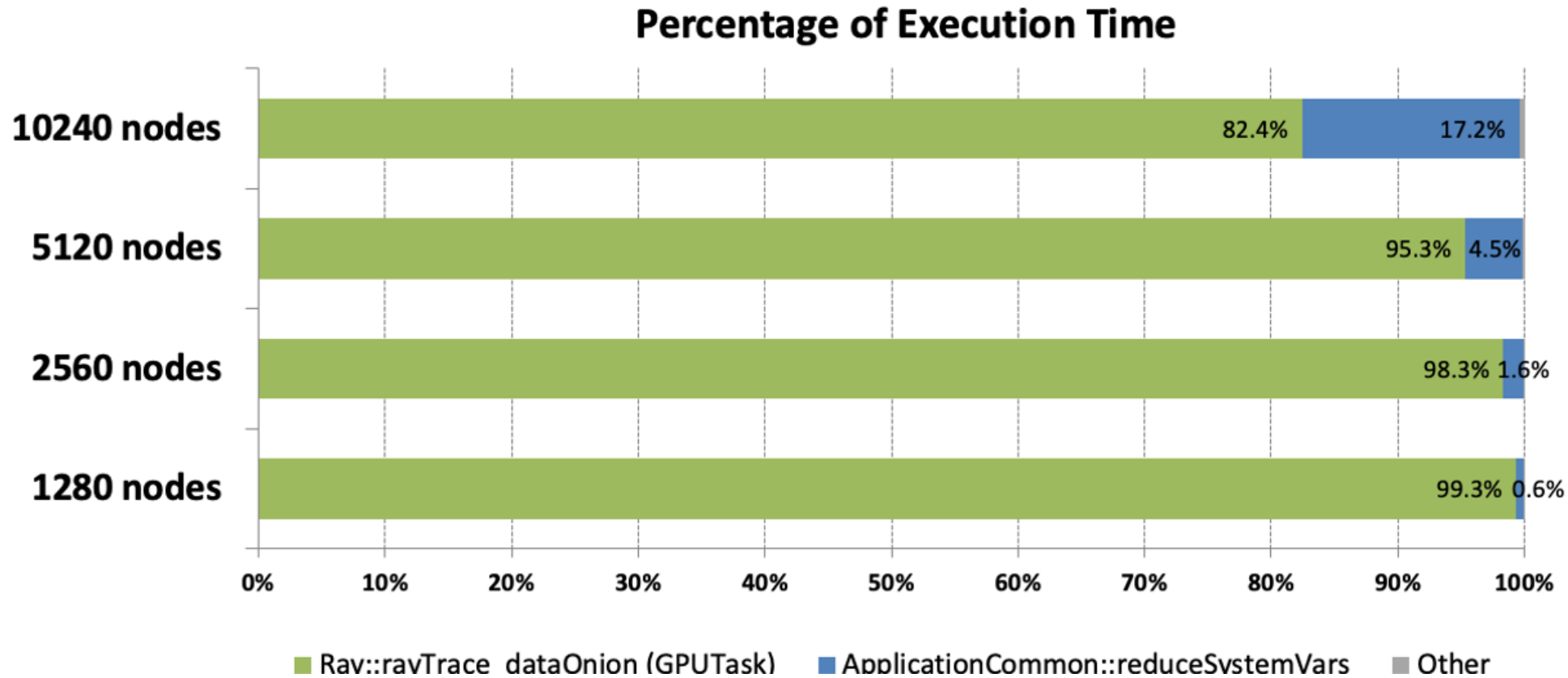


Initial Conditions:  
Uniform temp field varying absorption coefficient

Burns and Christon Benchmark - Strong Scaling  
2-Level RMCRT  
ALCF - Aurora System

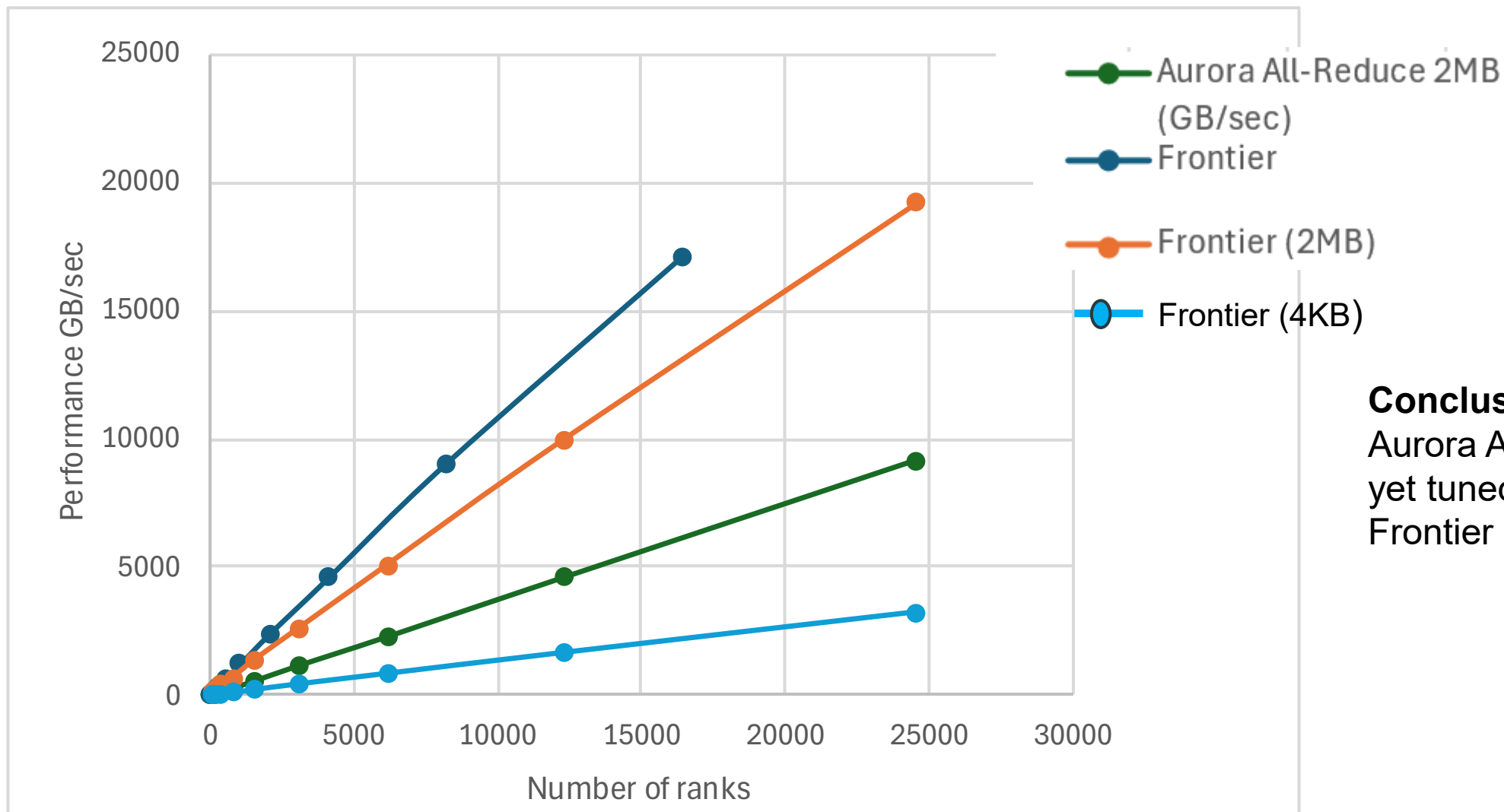


# Aurora Strong Scalability



Doubling the node count causes **all-reduce time** to go up by 2.5x to 4x

# MPI All-Reduce Debug Results Aurora vs Frontier



## Conclusion

Aurora All-Reduce not yet tuned compared to Frontier perhaps

## (iii) High Performance Task-Based Parallelism with NIST Hedgehog Software

1. NIST Parallel library for **coarse grain parallelism** Blattner, Bardakoff
2. Based on a **data-flow** graph model
3. Static task graphs
4. Using **data pipelining** for performance
5. **No scheduler**
  - Relies only on OS to manage threads
  - Dataflow execution based only on the presence of data
6. **Header only** library
7. **Good GPU peak** on single node DGEMM
8. **Matrix Warehouse Task similar to Uintah Runtime**
9. **DGEMM Matrix-Matrix multiplication algorithm same as Dongarra et al. block cyclic**



# SGEMM Single Node DGX1 8V100s TFLOPs

M = N = K	DPLASMA	Hedgehog	Peak %
128K	101.5 ± 0.4	98.9 ± 1.76	79%
160K	99.9 ± 0.6	96.8 ± 1.8	77%
192K	-	106.6 ± 0.6	85%

Nitish used MPI and worked with NIST. DPlasma used Summit and INRIA team and used Nvidia specific calls

Preliminary **Fugaku** Results –need to Numa and compiler tune still

# NSF TACC Vista 576 H200s Pflops

Nodes (GPUs)	M = N = K	Hedgehog PetaFlops	Peak %
64	640K	3.469 ± 0.008	81%
128	1000K	6.625 ± 0.022	77%
256	1250K	13.79 ± 0.021	80%
512	1500K	24.57 ± 0.318	72%

- (i) EcoBoost mode single core  
One arithmetic pipeline  
Hedgehog Sgemm 40%  
Sgemm + Open MP 52%
- (i) Normal mode 2arithmetic pipelines  
Sgemm+OpenMP 94%  
Hedgehog Sgemm 61.2% 10/2/25

# (IV) Future Task-Based Approaches: Quantum-HPC Hybrid Platform in Japan

Masaaki Kondo, Kento Sato, Miwako Tsuji, Riken

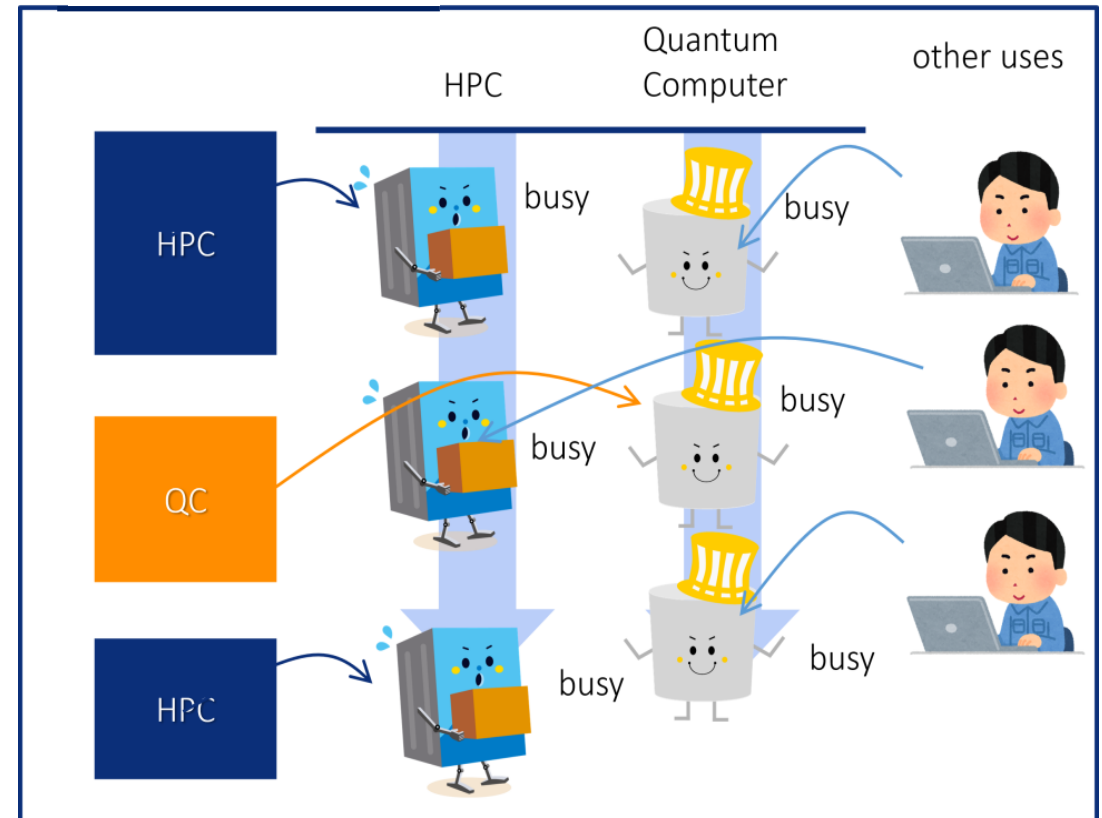
See Satoshi Matsuoka's talk

Riken aims to avoid waiting times in quantum computers and supercomputers

- Uses separate quantum kernels and HPC kernels in their applications
- Defines the order of their execution

Job-schedulers in both computers may arrange jobs to maximize the throughput in their systems.

Job may have to wait the result from another job, but the waiting time does not consume resources



Source <https://inpx.science/workshop/2025-inpx-workshop/>

# Vision for 2<sup>nd</sup> Layer, task-based programming?

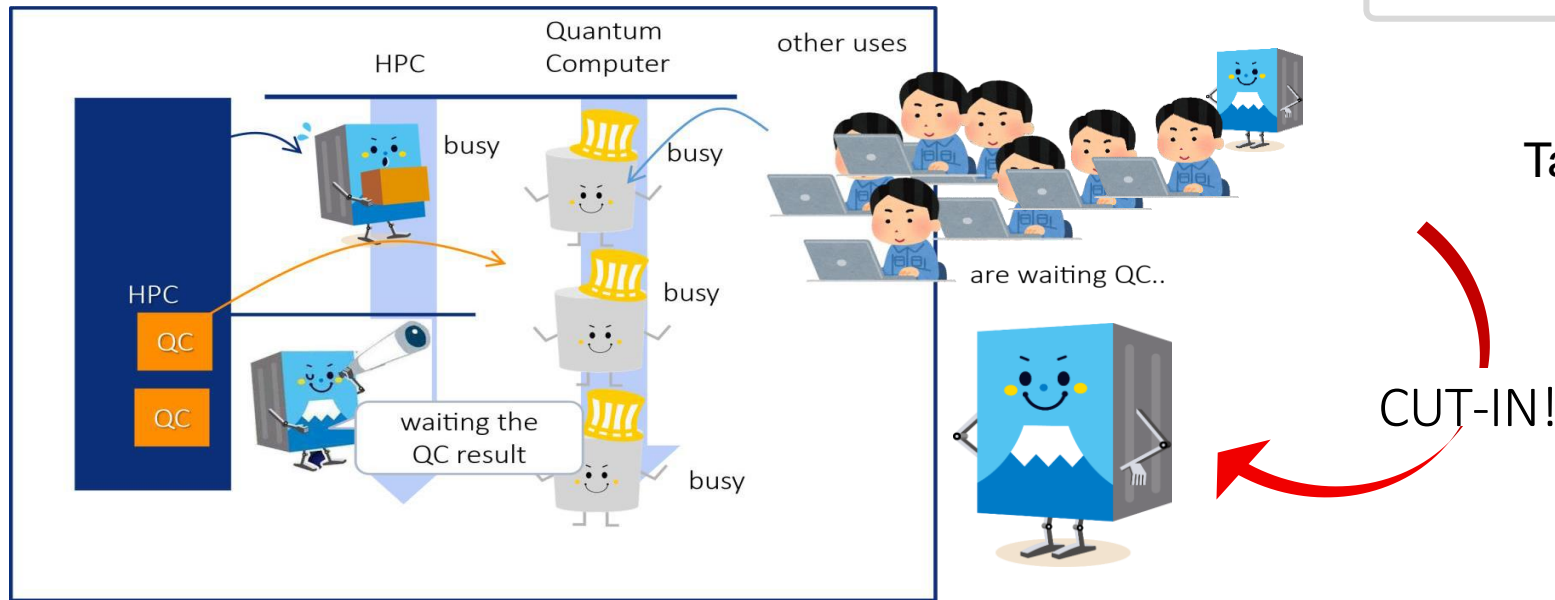
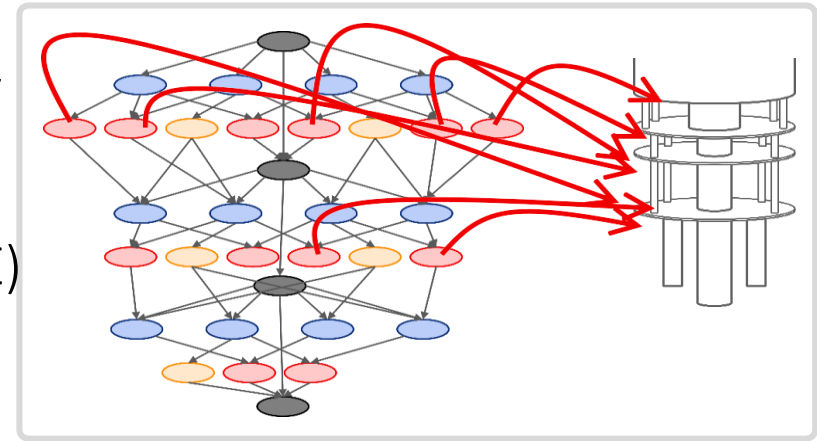
Tightly coupled quantum and HPC kernels in a single job

- Difficult to separate quantum and HPC kernels completely

Aim to avoid waiting time in large scale HPC jobs

Multiple quantum kernel calls from HPC in a short time (ex. VQE)

- Difficult to wait for each of the quantum circuit executions



Tasking Approach to Quantum

Source <https://inpx.science/workshop/2025-inpx-workshop/>

# Conclusions and Publications

Task-based approaches such as Uintah are very successful but require a high software investment. Can we

- (i) Reduce this software cost and make the best use of large post-exascale machines?
- (ii) Extend out to HPC+ QC AL/ML ?

J.K. Holmen, M. García, A. Bagusetty, A. Sanderson and M. Berzins. "**Making Uintah Performance Portable for Department of Energy Exascale Testbeds**". In: Zeinalipour, D., et al. Euro-Par 2023: Parallel Processing Workshops. Euro-Par 2023. Lect. Notes in Comp. Sci., 14352, 115-126. Springer, 2024

J.K. Holmen, M. García, A. Sanderson, A. Bagusetty and M. Berzins, "**Lessons Learned and Scalability Achieved When Porting Uintah to DOE Exascale Systems**". In: Caino-Lores, S., et al. Euro-Par 2024: Parallel Processing Workshops. Euro-Par 2023. Lect. Notes in Comp. Sci, 15385, 231-242. Springer, 2025.

M. García, J.K. Holmen, and M. Berzins, "**Scaling Uintah on the Aurora Exascale System up to 122,880 Intel Ponte Vecchio Xe Stacks**". In Practice and Experience in Advanced Research Computing (PEARC 25), July 20-24, 2025, Columbus, OH, USA. ACM, New York, NY, USA

Nitish Shingde, Timothy Blattner, Alexandre Bardakoff, Walid Keyrouz, and Martin Berzins. 2024. **An Illustration of Extending Hedgehog to Multi-Node GPU Architectures Using GEMM**. SN Comput. Sci. 5, 5 (May 2024).

# ACKNOWLEDGMENTS



## Marta García

Computational Scientist at the Computational Science Division (CPS) at Argonne National Laboratory

This research was supported by and used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

This work is associated with an ALCF Aurora Early Science Program project.

Some of this work was done on a pre-production supercomputer with early versions of the Aurora software development kit.



## John K. Holmen

HPC Engineer at the National Center for Computational Sciences (NCCS) at the Oak Ridge National Laboratory

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

## Walid Keyrouz and Tim Blattner

NIST provided access to ANL and TACC Machines and the connection to Riken.

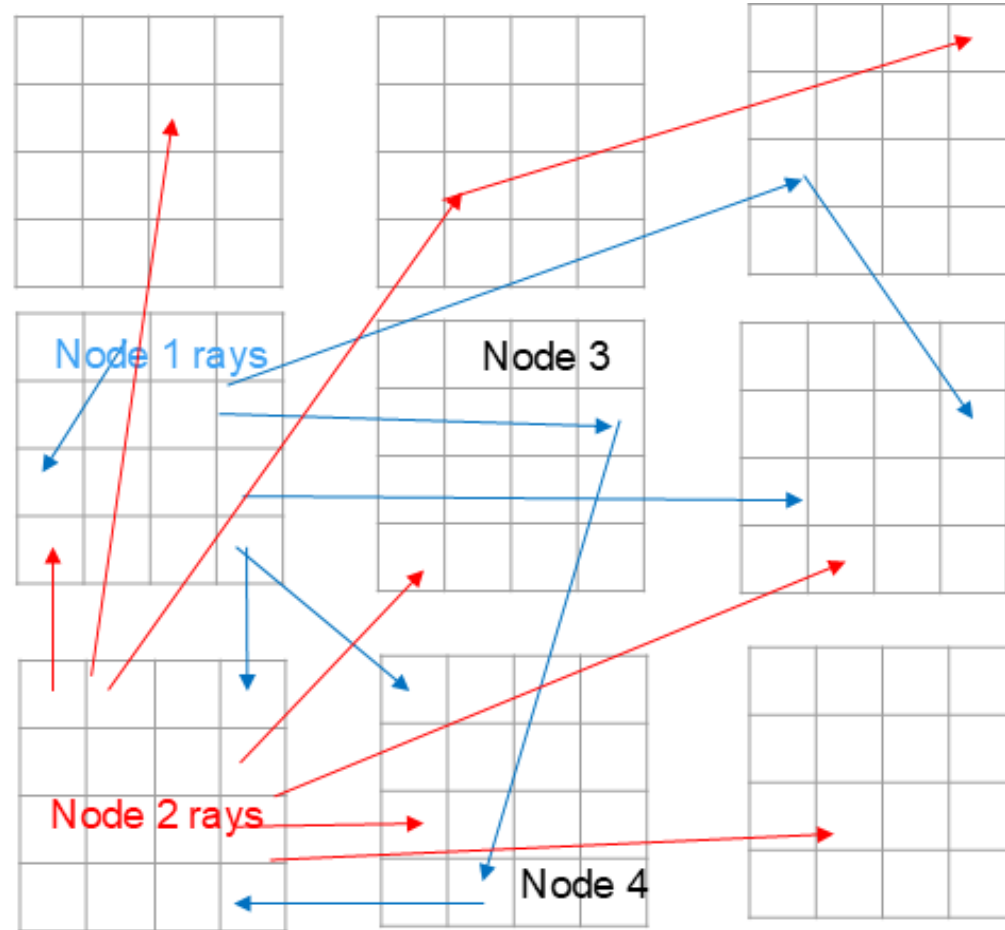
Thanks to all who helped at NIST, TACC, ANL and Riken

Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Rays trace across multiple mesh patches in a way that is unpredictable

Has a potentially chaotic communications pattern.

As we strong scale the communications increase unpredictably when there are fewer mesh patches per compute node



Ha Ha  
**NO SCALABILITY**  
As rays traverse multiple mesh patches



### Scalability breakdown as rays traverse multiple patches

# Debug Results

		Frontier		Aurora	
	Calls	Mean	Total	Mean	Total
Kokkos run tasks	1584	135	2e+5	209	3e+5
Kokkos for:raytrace	46080	4.1	1.9e+5	3.43	1.6e+5
Ze event host synch	170352			0.93	1.6e+5
MPI_allreduce	3168	4.0	1.3e+4	45.8	1.4e+5

Ze event host synch synchronizes host and gpu on Aurora

MPI\_allreduce implementation is perhaps the problem?

Attempts to solve this are ongoing.