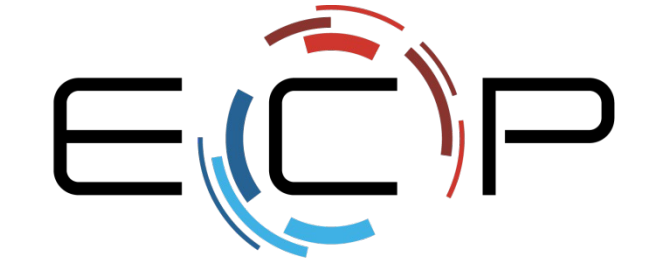
Taking HACC into the Exascale Era:
New Code Capabilities, and Challenges

Esteban M. Rangel, Computational Scientist
(CPS) ANL

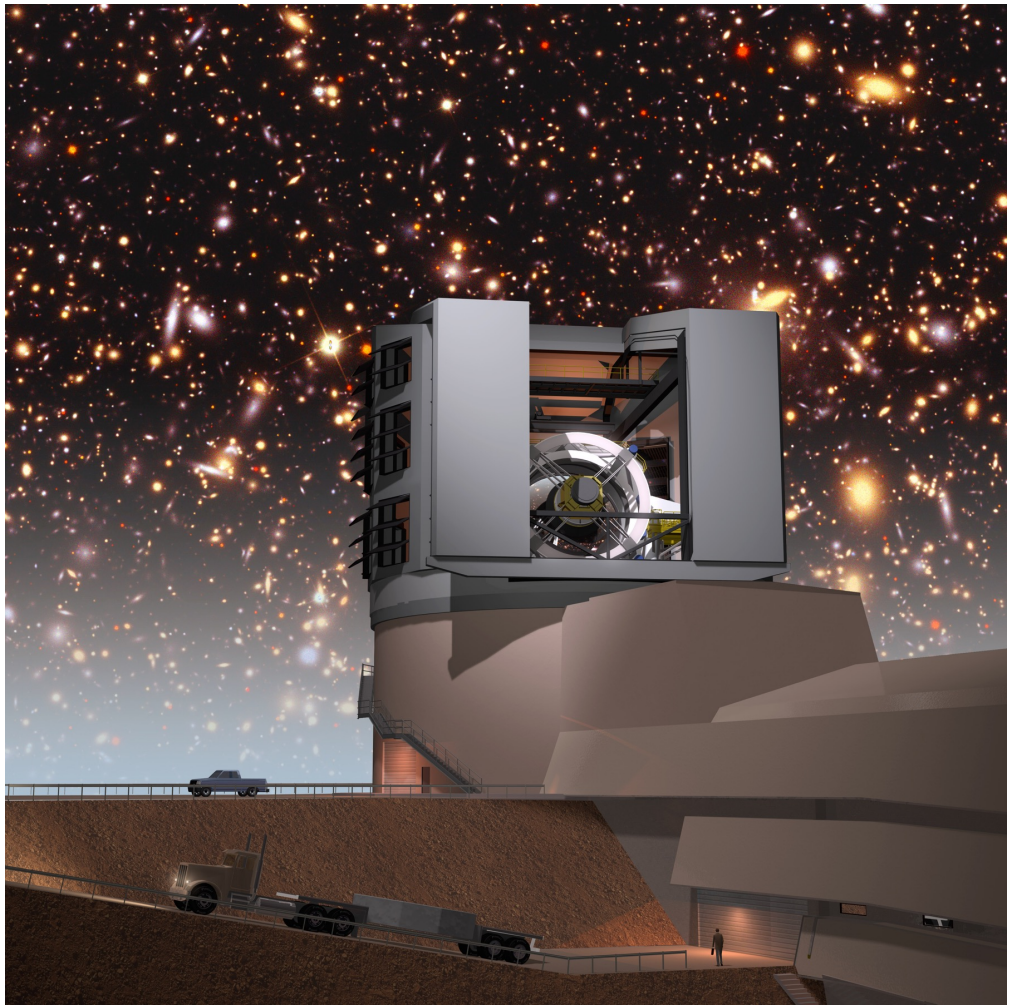
Best Practices for HPC Software Developers (Webinars)



NNSA National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science

Cosmological N-Body Simulations

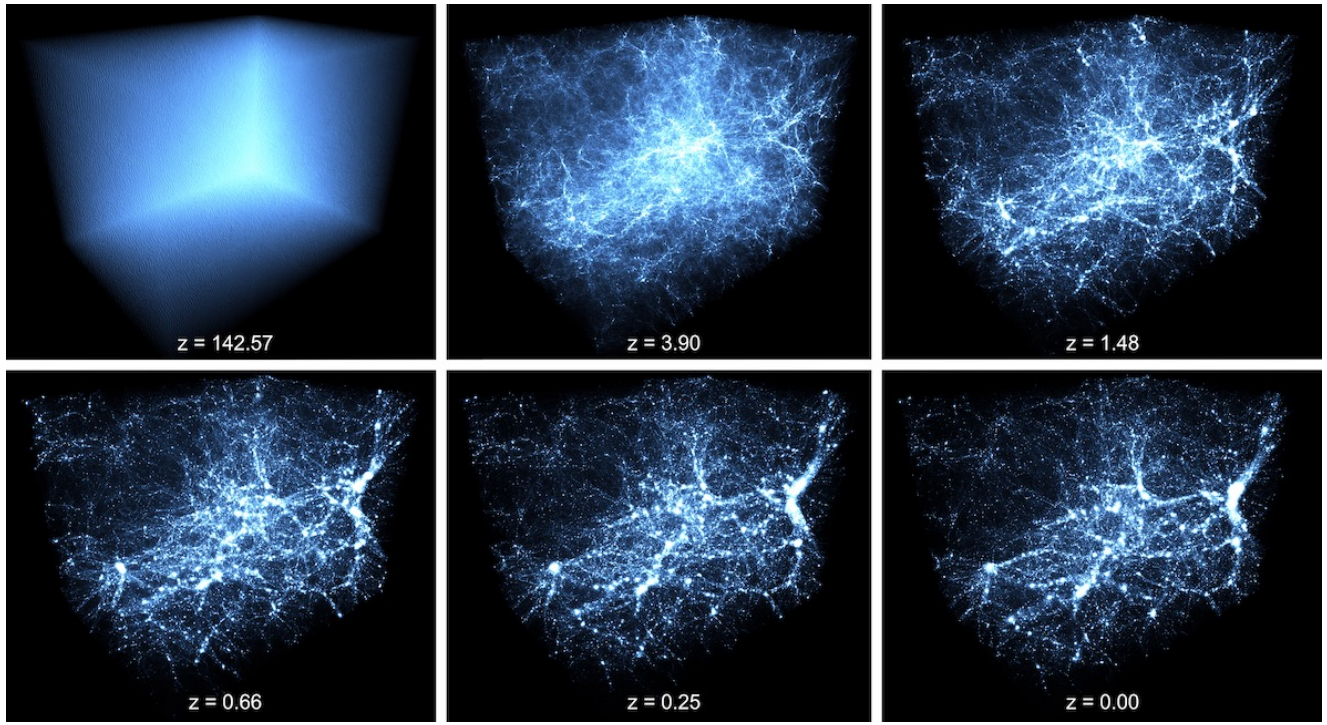
- Why do we run HACC (Hardware/Hybrid Accelerated Cosmology Code)?
- Theoretical research
  - Understand how Large-Scale Structures (LSS) form and evolve over cosmic time
  - Look for signatures of new/interesting physics
- Comparison with observations
  - Grand astronomical surveys
    - Rubin-LSST >$0.5B (NSF + DOE + …)
  - Create mock Universes for survey design
  - Provide theoretical models of summary statistics for data analysis (eg. emulators)
  - Understand data covariance for parameter estimation
  - Single observed Universe means forward-modeling



Rubin LSST: https://www.lsst.org/

HACC N-Body: Matter Distribution

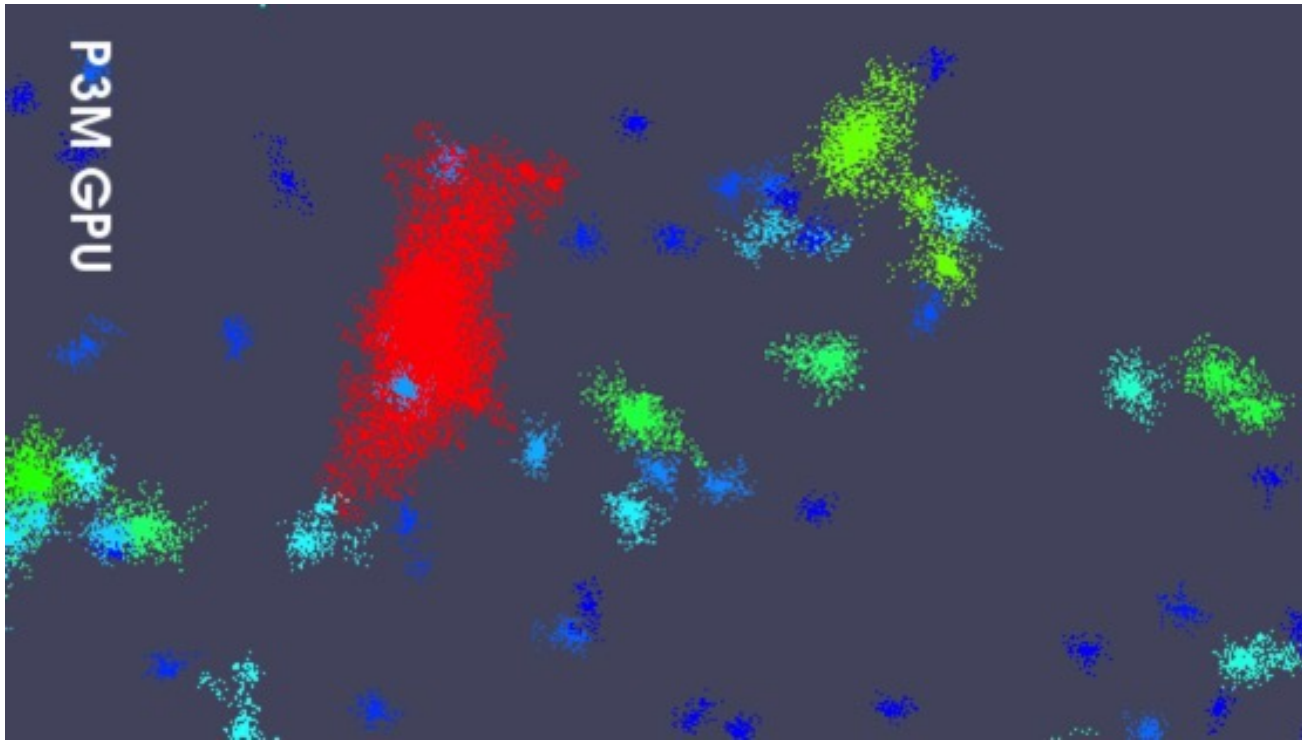
- Gravity
  - As the Universe expands, structure condenses from very smooth initial conditions
  - Dark matter is dominant mass component and is modeled as effectively collision-less.



Evolution of matter distribution over cosmic time for a sub-volume of a HACC simulation.
FarPoint: https://arxiv.org/abs/2109.01956

HACC Analysis: Halos

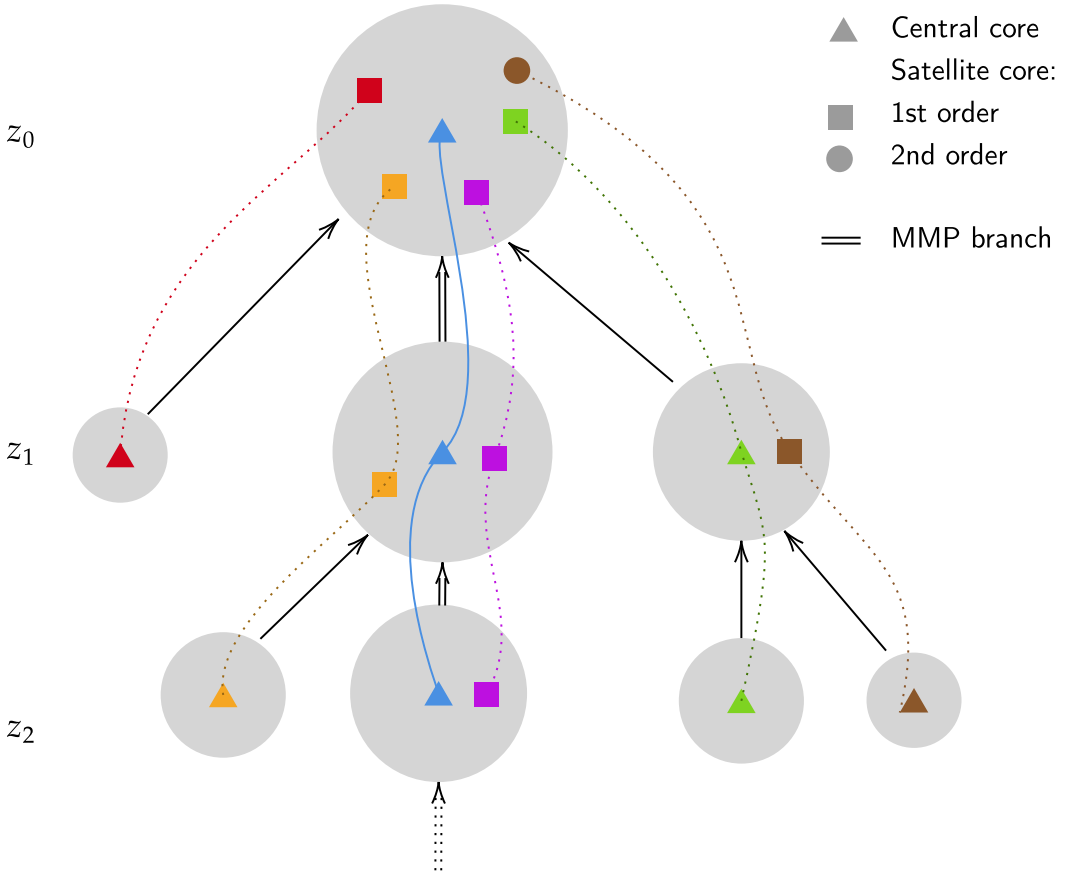
- Dark matter collects into “halos”
- Halos provide deep gravitational potential wells where baryonic matter can collect and eventually cool and condense to form stars and galaxies
- Roughly half of the mass in the Universe ends up in halos by our current epoch
- Halos are identified in simulations by looking for coherent structures with densities >100x of the background density



Particles in a small volume of a HACC simulation colored by halo membership.
HACC: https://arxiv.org/abs/1410.2805

HACC Analysis: Halo Merger Trees

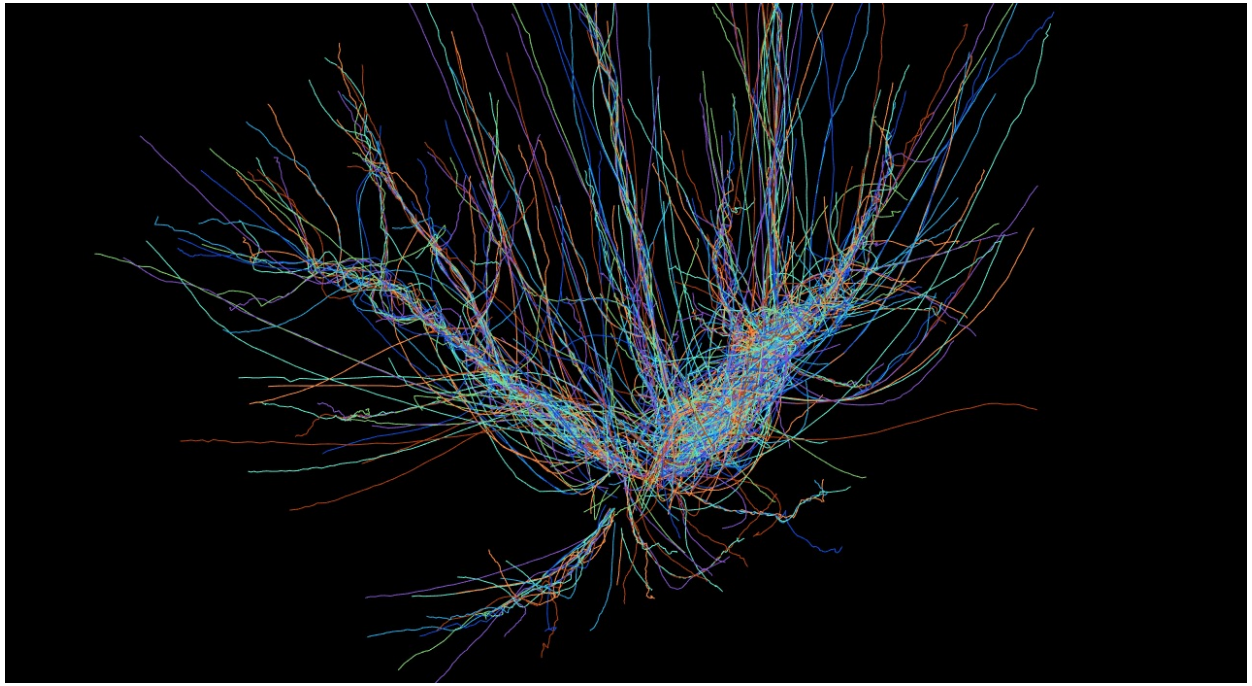
- Halos interact with each other as the Universe evolves, colliding and merging
- The interaction history of halos is important because interactions between galaxies within halos can trigger epochs of star formation, and the total history of star formation in a galaxy determines its luminosity/color
- The interaction histories of halos is summarized in a data structure called a merger tree



Logical merger tree.
SMACC: https://arxiv.org/abs/2012.09262

HACC Analysis: Halo Core Tracking

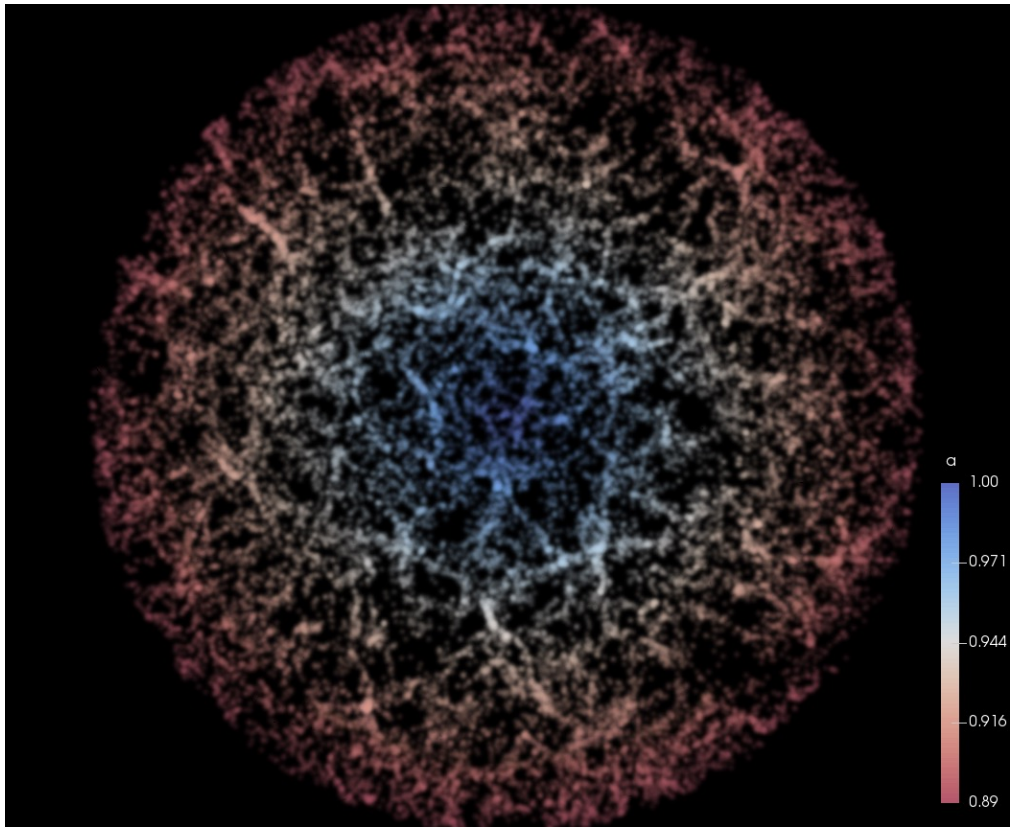
- Very inner part of halo is a tightly-bound core of particles that is not easily disrupted during halo-halo interactions
- Track sub-structure within halos by continuing to track cores even after halos merge
- Core positions are likely good proxies for galaxy locations



Physical trajectories of cores that merge into 1 halo.
OuterRim: https://arxiv.org/abs/1904.11970

HACC Analysis: Lightcones

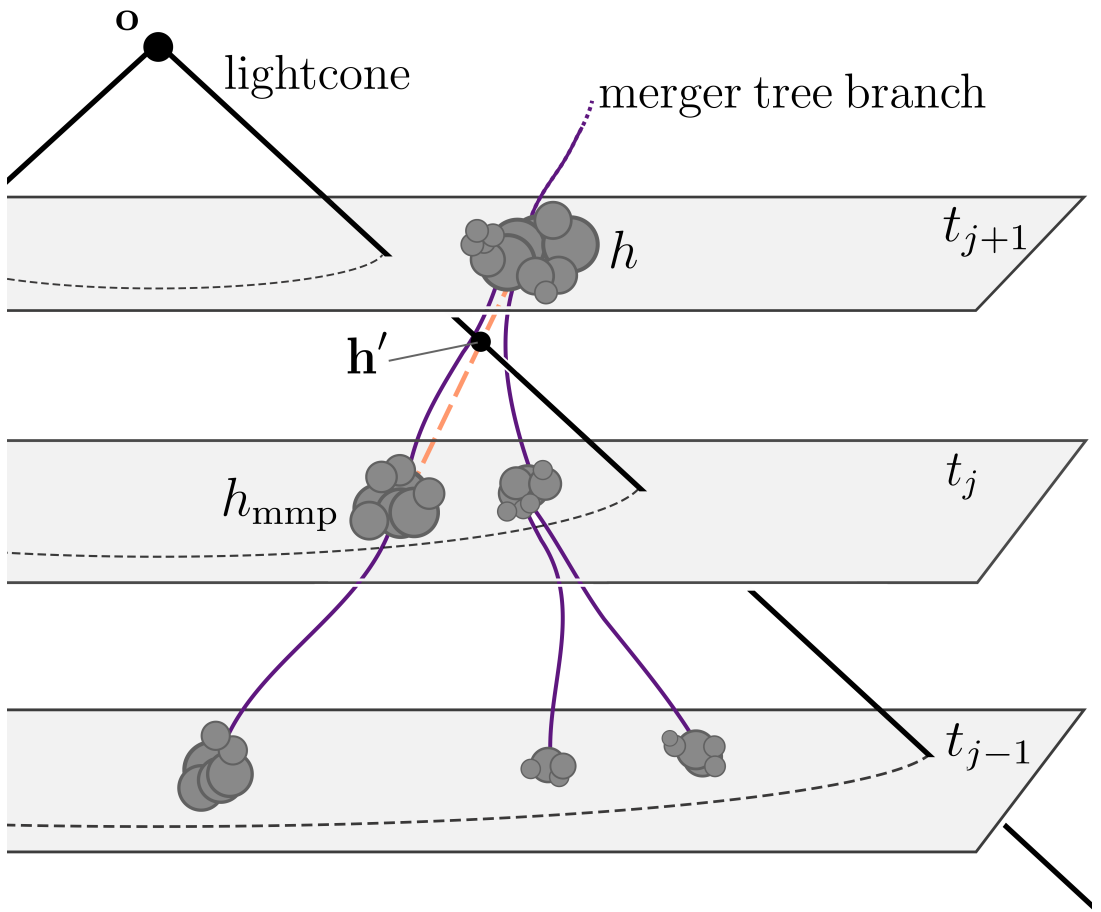
- N-body simulations operate in a comoving gauge, observations are not in same gauge
- Finite speed of light, we observe objects as they were when the light that we are now collecting left the object
- Objects that are farther away have a longer lookback-time
- HACC runs in a fixed-sized box (in comoving/expanding units) with periodic boundary conditions, but we can create a lightcone around an observer by saving the correct spherical shell from each time step



Particle lightcone from a HACC simulation with observer at center; color indicates distance/lookback-time.
OuterRim: https://arxiv.org/abs/1904.11970

HACC Analysis: Halo Lightcones

- Construct halo merger trees to the end of the simulation in the entire simulation volume
- Can go back and figure out where a merger tree intersects an observer's lightcone in order to display information from the merger tree in the right place at the right time



Interpolating merger trees onto an observer's lightcone.
CosmoDC2: https://arxiv.org/abs/1907.06530

Pre-Exascale (Early Petascale Era)

History

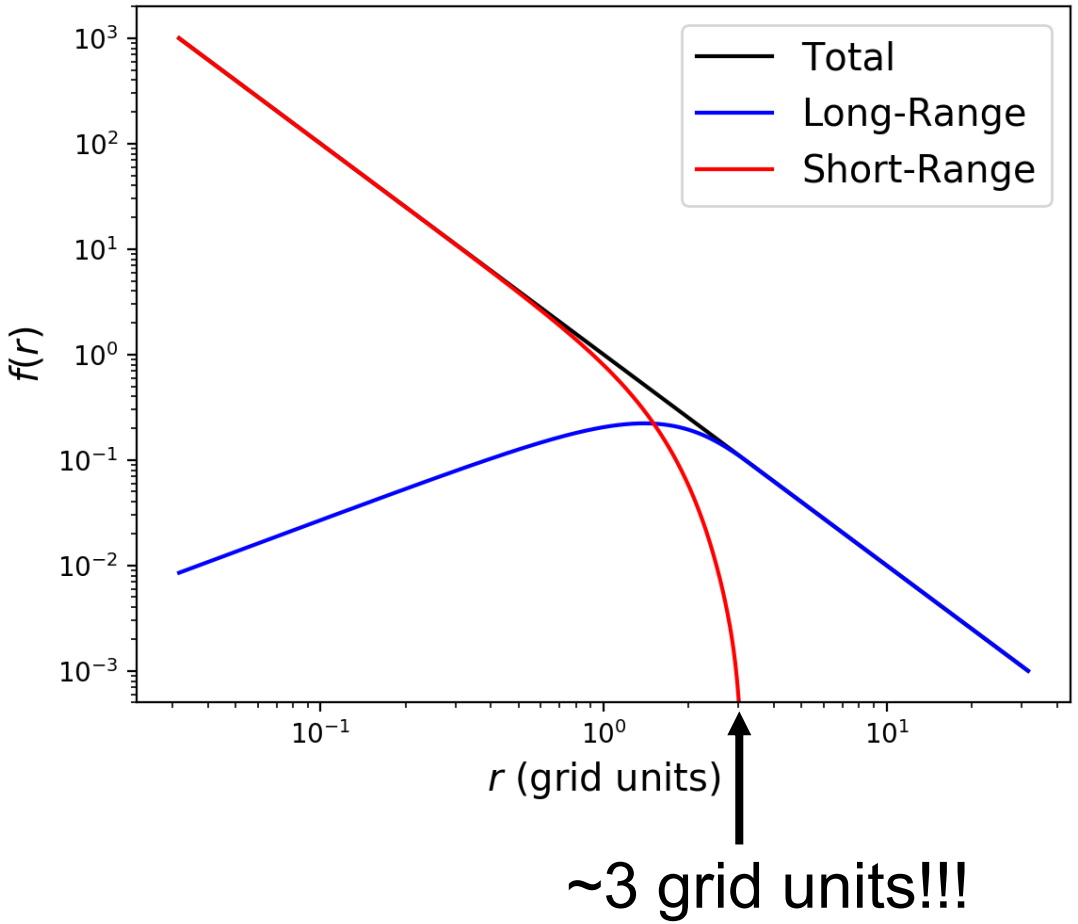
- Predecessor code was originally developed as a gravity-only cosmological N-body structure formation code written for Los Alamos National Laboratory's IBM Roadrunner supercomputer, which featured IBM Cell Broadband Engine accelerators.

Design

- Force-splitting
  - Long-range component of gravity was calculated with particle-mesh methods and a distributed-memory Fourier transform-based Poisson solver implemented in MPI.
  - Short-range component of gravity calculated using direct particle-particle comparisons and implemented in C with intrinsics to take advantage of the FLOPS available on the Cell accelerators.

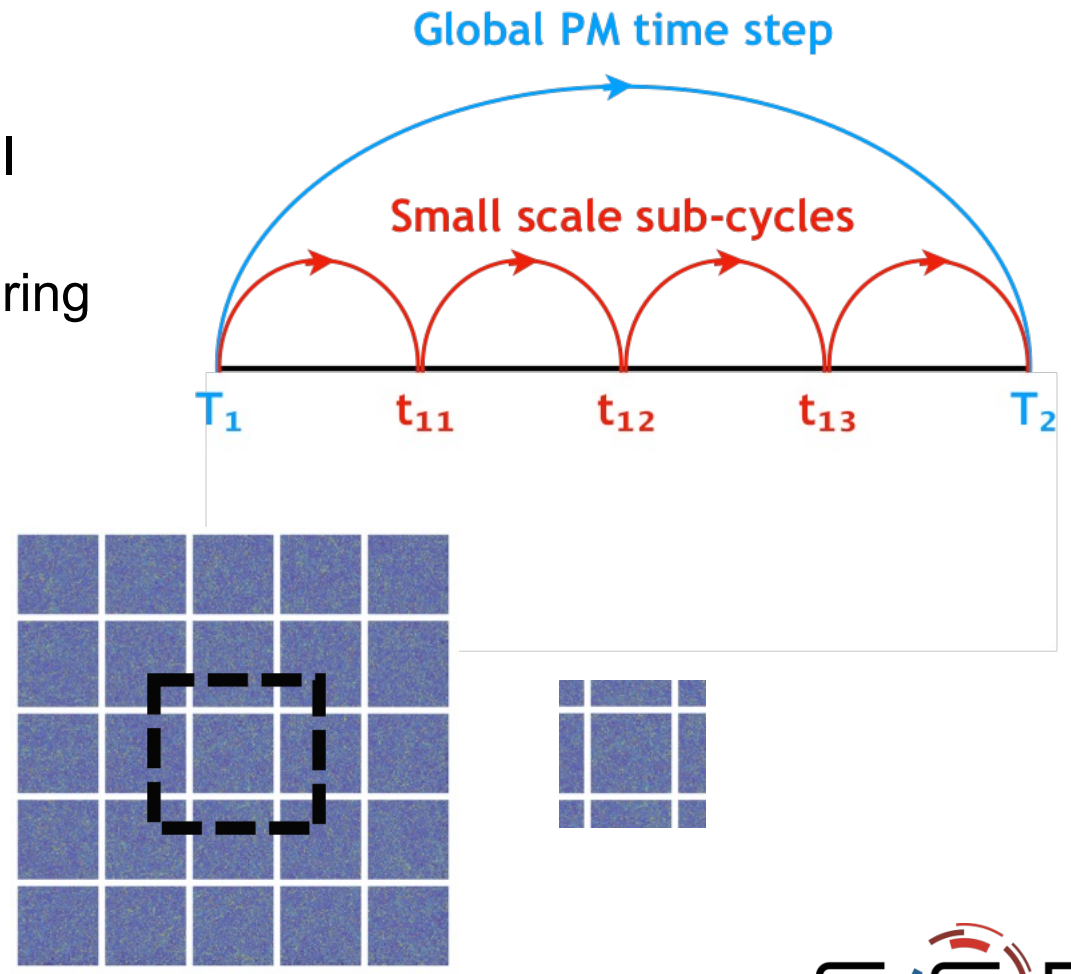
Gravity Force Splitting

- Hardware/Hybrid Accelerated Cosmology Code (HACC)
  - Gravity is infinite and unshielded
  - 1 kpc force resolution in 1 Gpc box, 10^6 dynamic range
- Operator splitting
  - Kick: forces used to update particle velocities; positions fixed
    - Long-range: Particle-Mesh, deposit onto grid, FFT-based Poisson solver, ~10^4 resolution from ~10k^3 grids, requires double precision
    - Short-range: Particle-Particle interactions, FLOPS intense, maximize architecture, ~10^2 resolution, single precision sufficient
  - Stream: velocities used to update positions; velocities fixed
  - Symplectic integration
- HACC Spectral Force Handover Technology™
  - Use low-order Cloud-in-Cell (CIC) deposit
  - Spectral shaping reduces noise and emulates smoother deposit
  - Extremely compact, ~3 grid units, limit particle comparisons



HACC Design

- Overloading (between MPI ranks)
  - Each MPI rank caches a thin shell of particles from immediate neighbor MPI ranks
  - No particles need to be exchanged during sub-cycles for short-range force calculation
  - Refresh particle cache periodically between outer time steps



Preparing for Exascale (Frontier and Aurora)

Challenges

- More increased compute capabilities than memory (e.g., Summit->Frontier)
  - ~8x more FLOPS (peak)
  - ~3x more memory
- Exascale systems will have multiple programming models and frameworks
  - CUDA, HIP, SYCL, OpenMP, Kokkos, …
- CPU analysis routines (on the host) are becoming a larger fraction of the overall execution time.

CRK-HACC

- Adds baryonic physics in addition to gravity
  - New Conservative Reproducing Kernel (CRK) formulation of Smoothed Particle Hydrodynamics (SPH)
  - Resolves some discrepancies with grid-based hydrodynamic schemes
  - non-radiative hydrodynamics
  - sub-grid models for radiative cooling, star formation, and feedback from supernovae and Active Galactic Nuclei (AGN)

ECP Software and Technology (ST) Projects

Collaborators

- ArborX
  - Fast GPU-accelerated geometric search library
- VeloC/SZ
  - Low overhead checkpointing
  - Lossy data compression where the error can be bound and controlled
- ALPINE/ZFP
  - Visualization

HACC

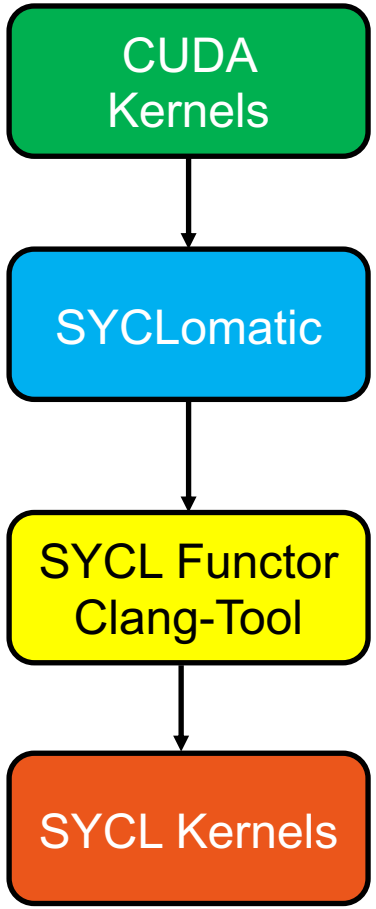
- ArborX
  - FOF halo-finding
  - AGN center-finding (hydro sub-grid)
  - SOD halo-finding
- VeloC/SZ
  - Checkpoint/restart
  - Compressed analysis outputs

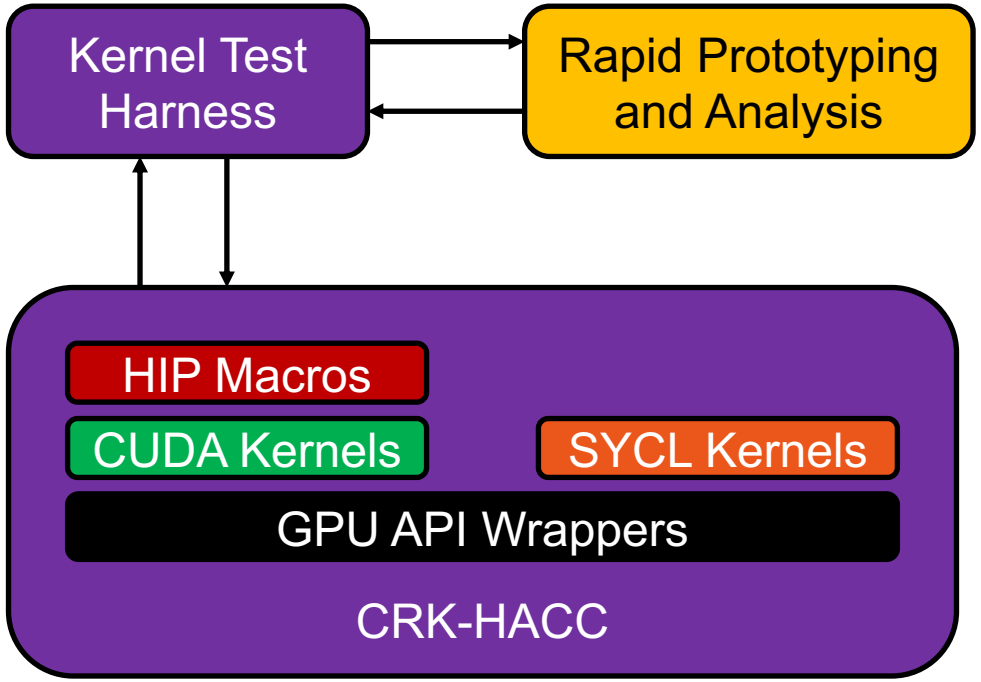
Preparing for Aurora

- Primary *and ongoing* development of CRK-HACC uses CUDA, with HIP support (for Frontier) through macro transformations of API calls.
- SYCL was chosen by the HACC team for running on Aurora.
- HACC would support multiple build implementations, as it has historically, to exploit low-level programming model features to achieve the best possible performance on target systems.

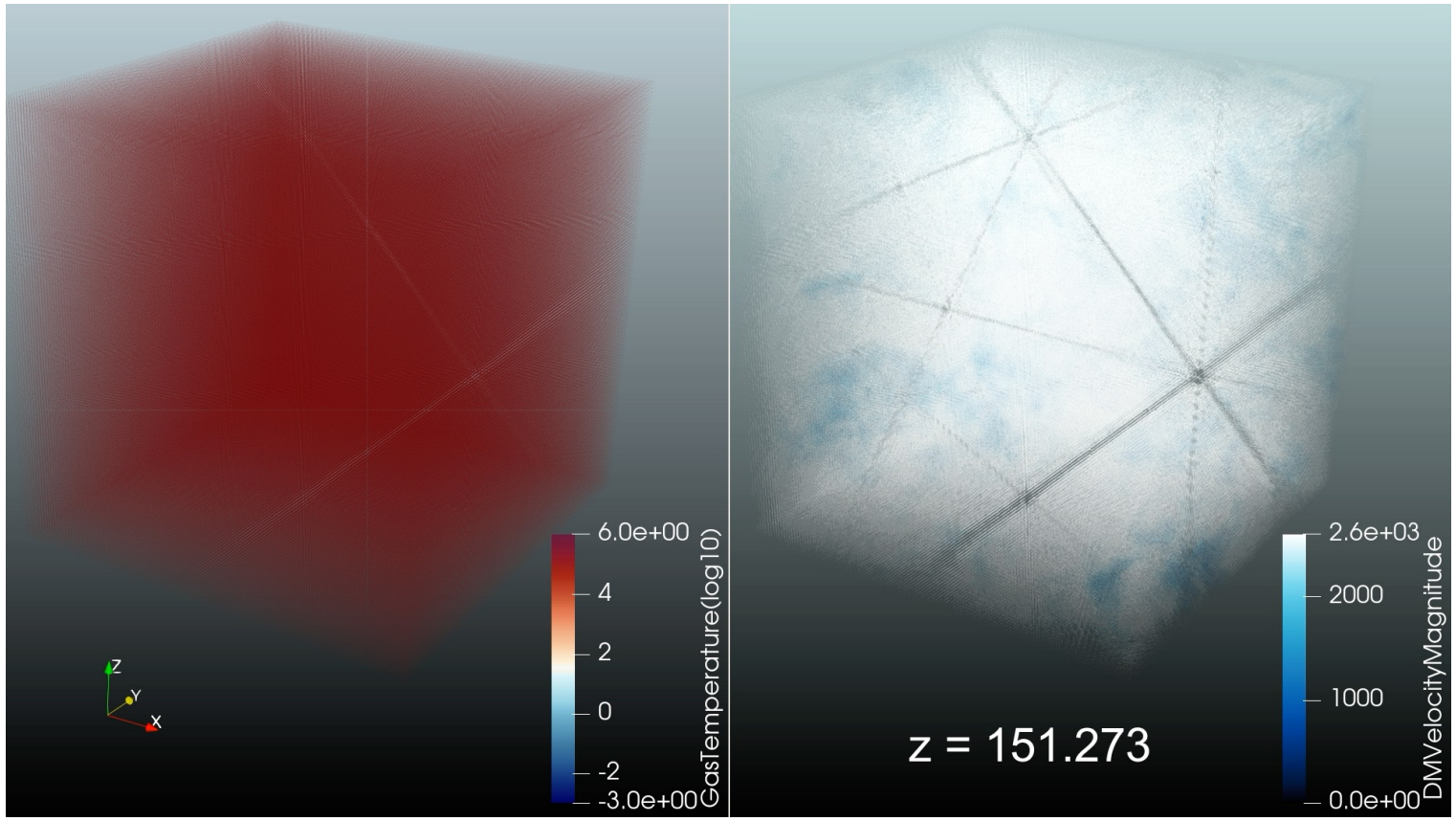
CUDA to SYCL Migration

Semi-automated Migration Pipeline





Adiabatic Hydro Simulation on Sunspot (Aurora TDS)



Credit: Silvio Rizzi, Argonne LCF

Performance, Portability, and Productivity Study, using SYCL

A Performance-Portable SYCL Implementation of CRK-HACC for Exascale

Esteban M. Rangel
erangel@anl.gov
Argonne National Laboratory
USA

S. John Pennycook
john.pennycook@intel.com
Intel Corporation
USA

Adrian Pope
apope@anl.gov
Argonne National Laboratory
USA

Nicholas Frontiere
nfrontiere@anl.gov
Argonne National Laboratory
USA

Zhiqiang Ma
zhiqiang.ma@intel.com
Intel Corporation
USA

Varsha Madananth
varsha.madananth@intel.com
Intel Corporation
USA

Paper to appear in the P3HPC Workshop as part of SC23

Experimental Setup

Hardware Configuration for Systems

System	CPU	GPU	FP32 Peak per GPU
Aurora	2 x Intel® Xeon® CPU Max 9470C, 52 cores	6 x Intel® Data Center GPU Max 1550	45.9 TFLOPS
Polaris	1 x AMD EPYC 7543P, 32 cores	4x NVIDIA A100-SXM4-40GB	19.5 TFLOPS
Frontier	1 x AMD EPYC 7A53, 64 cores	4 x AMD Instinct MI250X	53 TFLOPS

Problem Size

2x 512^3 particles

5 timesteps (4 fixed sub-cycles)

8 MPI ranks

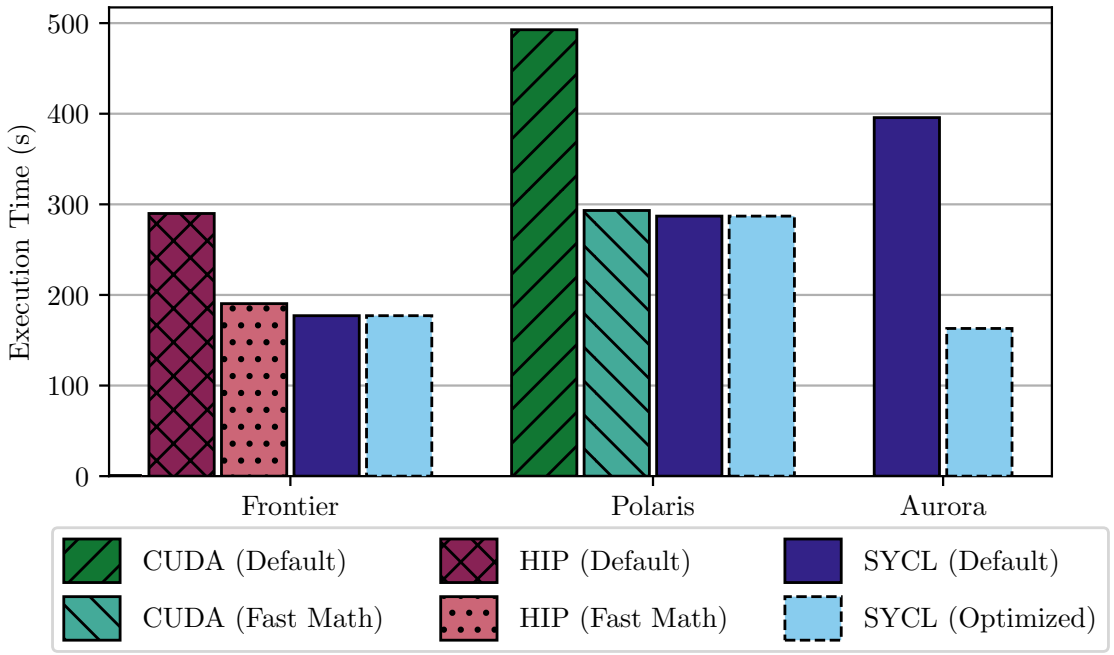
Aurora: 1 rank/tile

Polaris: 2 ranks/GPU
note: measured ~11% lower efficiency

Frontier: 1 rank/GCD

Initial Results

Aggregate of all GPU Kernels



- Fast Math optimizations were not enabled by default on all compilers.
- Frontier HIP code uses a wavefront size of 64, and the SYCL code uses a sub-group size of 64
- Polaris CUDA code uses a warp size of 32, and the SYCL code uses a sub-group size of 32
- Aurora SYCL code uses a sub-group size of 32 and/or 16

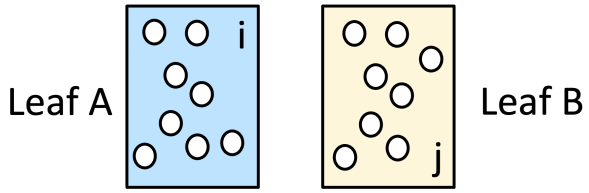
Performance results are based on testing dates and configurations used and may not reflect all publicly available updates.

Optimizations to GPU (solver) Kernels

Hotspot Kernels

1. *Geometry*: measures the volumes of gas particles
2. *Corrections*: computes the reproducing kernel coefficients of the higher order smoothed particle hydrodynamics (SPH) solver
3. *Extras*: evaluates the density and state gradients
4. *Acceleration*: calculates the momentum derivative
5. *Energy*: solves the derivative of the internal energy.

The SIMD lane data layout of the "half-warp" algorithm. Lanes [0-15] load and update particles from leaf A, while lanes [16-31] operate on particles from leaf B.

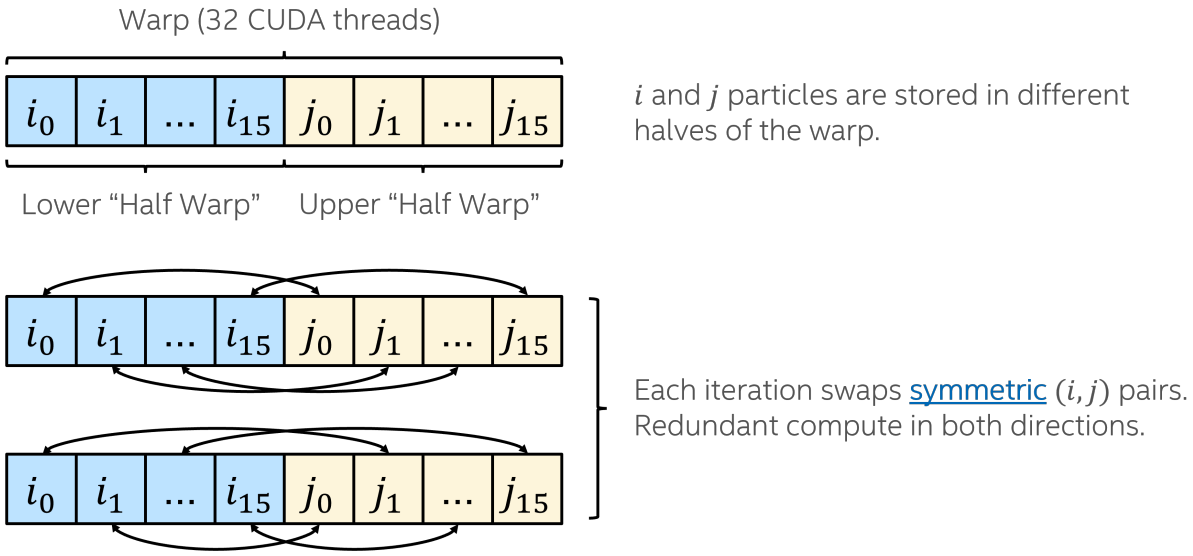


Particles are organized in a tree with "fat leaves" containing an SYCL work-group/CUDA block-sized number of particles.

Particle SIMD layout

Lane ID	0	1	…	15	16	17	…	31
Leaf	A	A	A	A	B	B	B	B

Optimizing Cross-lane Communication



- The communication pattern of the "half-warp" algorithm for interacting particles from leaves A and B within the same warp.
- This represents one of the total ($|LeafA| \times |LeafB|/warp\_size$) instances required.
- The pair-wise symmetry is critically important for the correctness of the algorithm
- XOR-based shuffle pattern implemented as the `__shfl` intrinsic for CUDA `sycl::select_from_group` in SYCL

Optimizing Cross-lane Communication

Intel® Data Center GPU Max 1550 assembly snippets for `sycl::select-from-group`

Elements are gathered from the registers specified in **a0** and written into **r2** using *indirect register access*

```
...
shl (16|M0)   r24.0<1>:uw   r82.0<2;1,0>:uw   0x2:uw
add (16|M0)   a0.0<1>:uw    r24.0<1;1,0>:uw   0x640:uw
mov (16|M0)   r2.0<1>:ud    r[a0.0]<1,0>:ud
...
```

alternative instruction sequence employing *register regioning* is more performant but not always achievable by the compiler

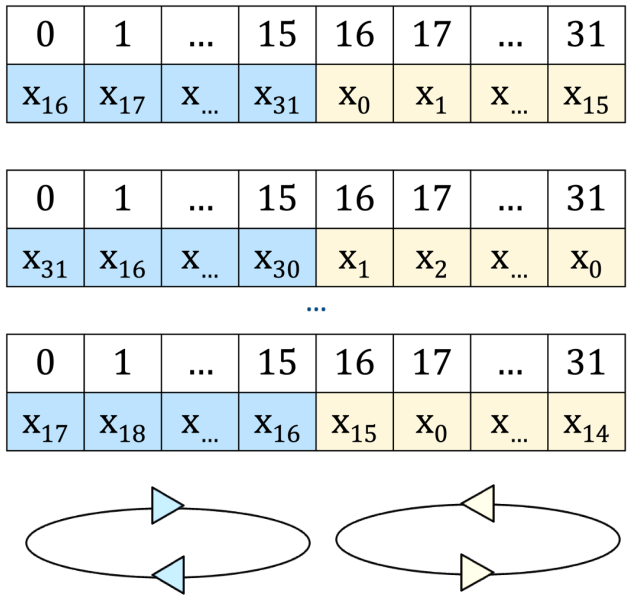
```
...
add (16|M0)   r24.0<1>:f   r68.0<1;1,0>:f   -r14.0<0;1,0>:f
add (16|M0)   r26.0<1>:f   r68.0<1;1,0>:f   -r14.1<0;1,0>:f
add (16|M0)   r30.0<1>:f   r68.0<1;1,0>:f   -r14.2<0;1,0>:f
...
```

Cross-lane Communication Strategies explored

- Shared Local Memory
  - Uses sycl::local_accessor to reserve a small amount of work-group local memory per sub-group to communicate instead of via registers.
- Broadcasts
  - Restructure loops so that sufficient information is known about the communication pattern at compile-time to generate more efficient assembly.
- Optimized Instruction Sequences (Intel)
  - Explicitly code the assembly instructions for each communication step needed.

Optimizing Cross-lane Communication (Intel)

Specialized butterfly-shuffle communication pattern, which provides the same pair-wise symmetry property of the XOR-based pattern



Register view of the specialized butterfly-shuffle

r0	X_0	X_1	X_2	X_3	X_4	...	X_{14}	X_{15}
r1	X_0	X_1	X_2	X_3	X_4	...	X_{14}	X_{15}
r2	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	...	X_{30}	X_{31}
r3	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	...	X_{30}	X_{31}

Example of performing the i=1 instance of the butterfly-shuffle.

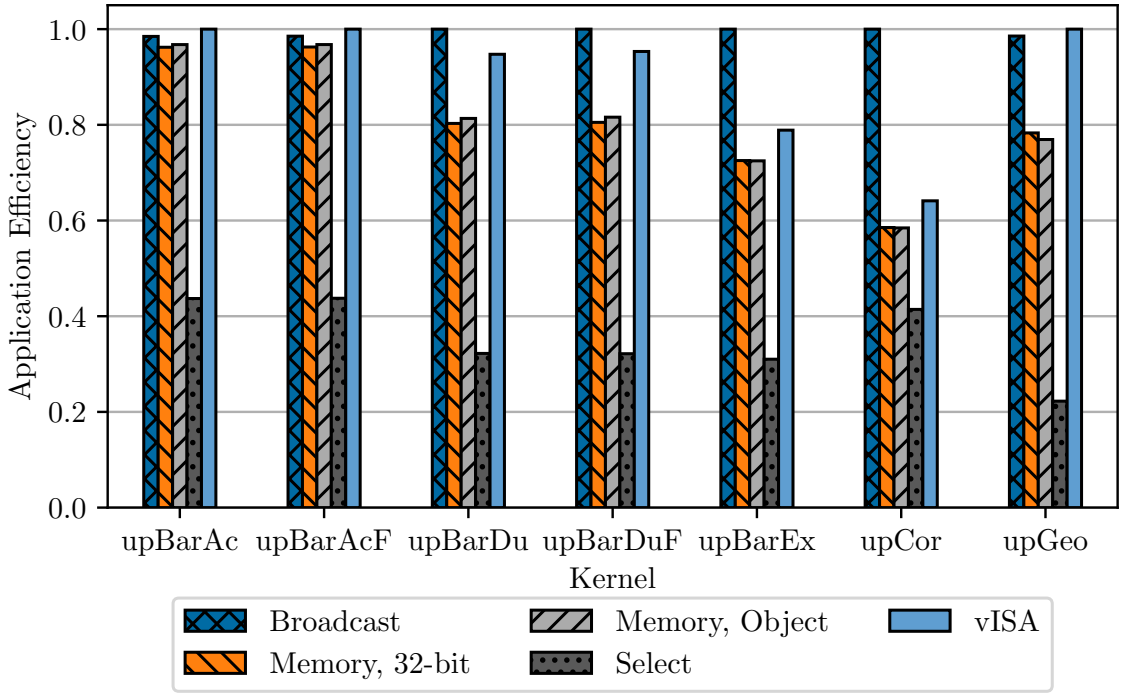
The “wrap-around” feature of the register file is exploited.

Efficiently performed with 4 **mov** instructions

Optimization Results (Hotspot Kernels)

Aurora

Intel® Data Center GPU Max 1550

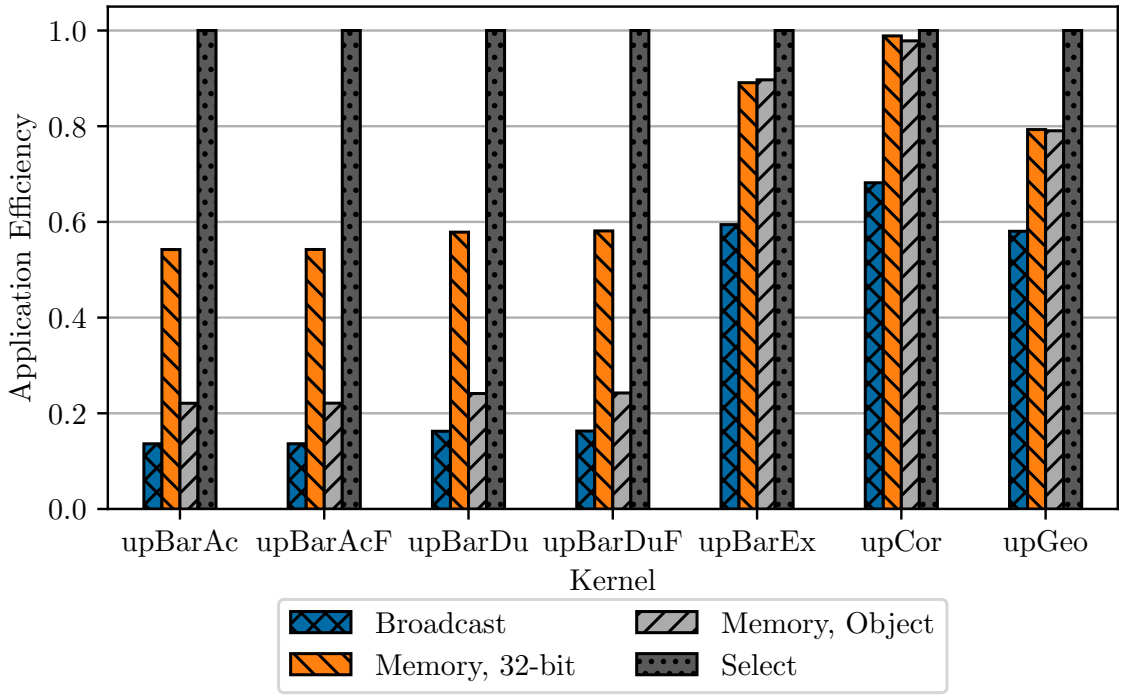


- Broadcast uses a sub-group size of 16, all other variants use a sub-group size of 32
- Restructuring the loops to use broadcasts also allows us to generate fewer atomic instructions, more noticeable in the Extras and Corrections kernels

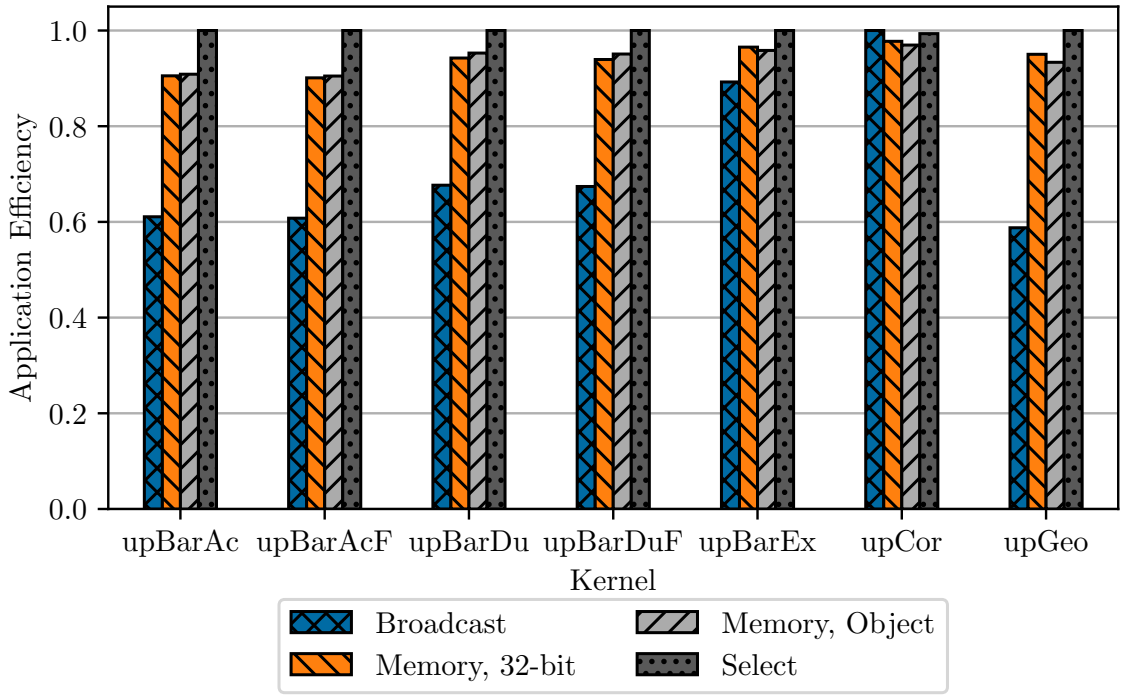
Performance results are based on testing dates and configurations used and may not reflect all publicly available updates.

Optimization Results (Hotspot Kernels)

Polaris
NVIDIA A100-SXM4-40GB



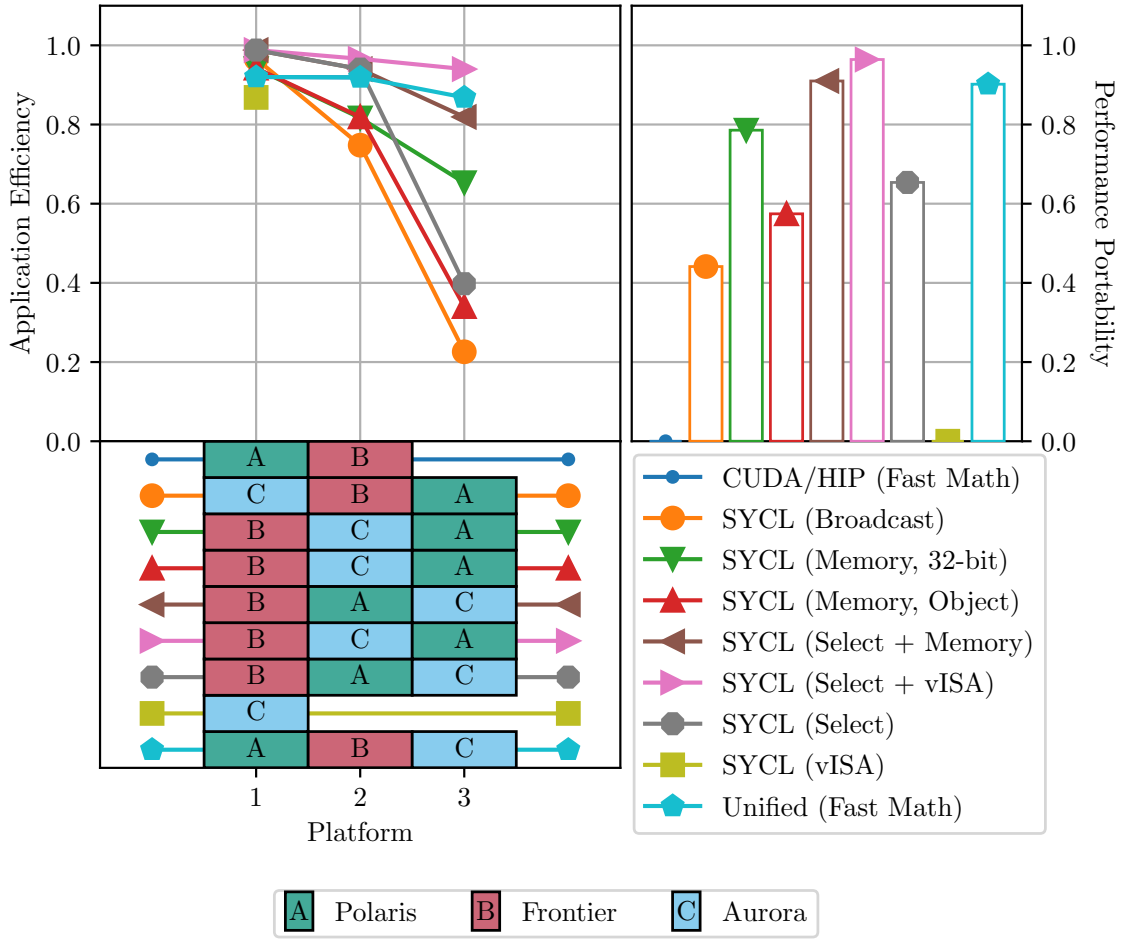
Frontier
AMD Instinct MI250X



Performance results are based on testing dates and configurations used and may not reflect all publicly available updates.

Performance Portability Analysis

Cascade plot of application efficiency and performance portability of CRK-HACC variants.



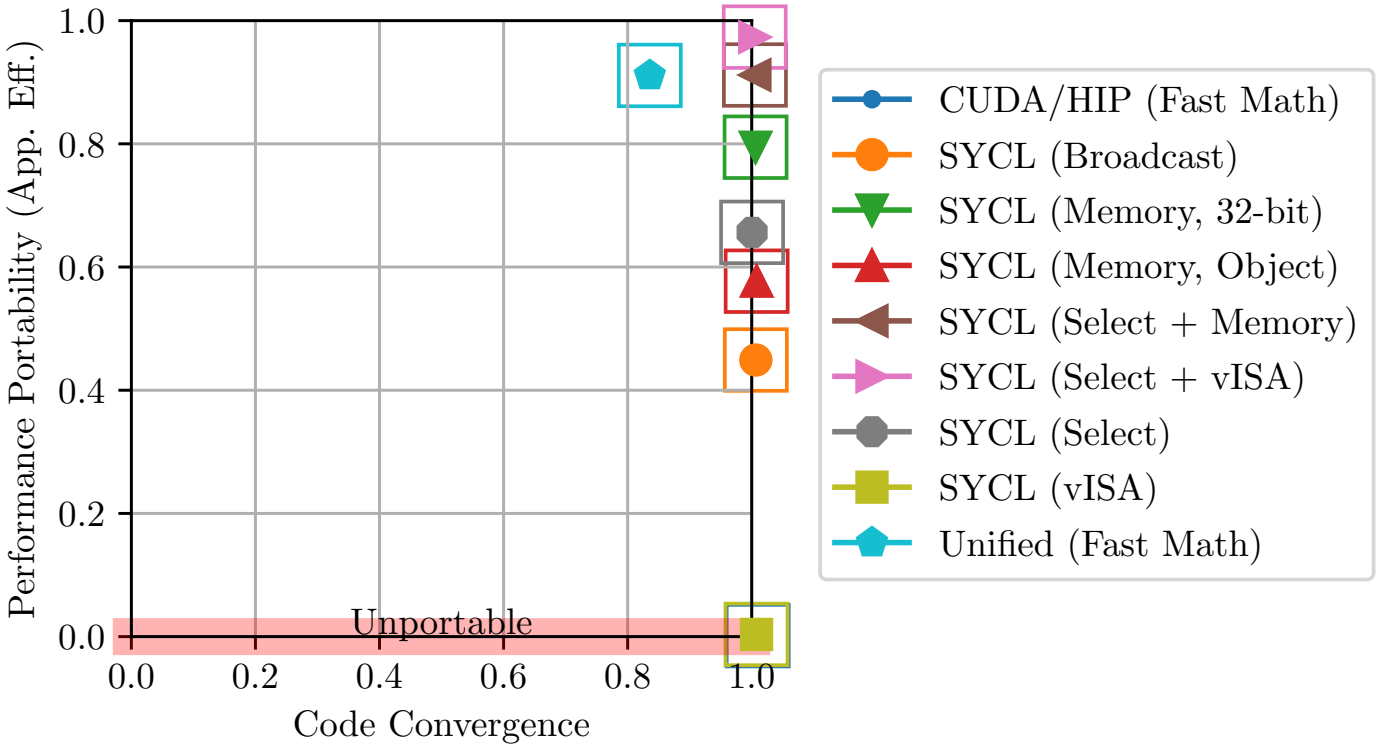
$$\mathrm{PP}(a,p,H) = \begin{cases} \dfrac{|H|}{\sum\limits_{i \in H} \dfrac{1}{e_i(a,p)}} & \text{if } \forall i \in H \\ e_i(a,p) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where a is an application, p is a specific input problem, H is the set of platforms of interest, and $e_i(a, p)$ is the efficiency with which application a solves problem p on platform i

application efficiency is calculated relative to a hypothetical application that is able to use the best version of each kernel on every platform

Productivity Analysis

A navigation chart showing the performance portability and code convergence of CRK-HACC variants



Code Divergence

$$\mathrm{CD}(a, p, H) = \binom{|H|}{2}^{-1} \sum_{\{i,j\} \in H \times H} d_{i,j}(a, p)$$

where $d_{i,j}(a, p)$ represents the distance between the source code required to solve problem p using application a on platforms i and j (from platform set H).

Disclaimers

- Performance varies by use, configuration and other factors. Learn more at https://www.intel.com/performanceindex
- Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See Slide 4 for configuration details. No product or component can be absolutely secure. Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.
- Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others. Khronos is a registered trademark and SYCL and SPIR are trademarks of The Khronos Group Inc.
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that code included in this document is licensed subject to the Zero-Clause BSD open source license (OBSD), http://opensource.org/licenses/0BSD.

Conclusion

- HACC has introduced new physics (hydrodynamics and sub-grid modeling) into the simulation capabilities, made possible with the increased computing power of Exascale supercomputers.
- Described a process to migrate and maintain a CUDA codebase to SYCL
- Identified that “shuffle” operations are not performance-portable from NVIDIA to Intel GPUs
- Developed a straightforward workaround to replace “shuffles” with local memory operations that can be generally useful to other developers.
- Demonstrated the practical potential for writing performance portable applications in SYCL ultimately achieving a performance portability of 0.96 with near-zero code divergence -- and a pure SYCL implementation performance portability of 0.91.

Key Takeaways

- Code (outside solvers), e.g., *in situ* analysis, are becoming bottlenecks and need GPU acceleration.
- The increased complexity of code makes maintaining multiple implementations more burdensome and highlights the need for performance-portable programming models.
- The SYCL version of CRK-HACC is an exciting proof-of-concept for using a single programming model across GPUs from Intel, NVIDIA, and AMD without sacrificing performance.

Acknowledgments

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.
- 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.

Thank you

Contact:
Esteban Rangel, CPS
erangel@anl.gov