TRINITY

Trinity: Opportunities and Challenges of a Heterogeneous System

Scott Hemmert, Mike E. Davis, Michael A. Gallis, Nathan Hjelm, John Levesque, James Lujan, Stan G. Moore, David Morton, Hai Ah Nam, Alex Parga, Paul Peltz Jr., Galen Shipman, Alfred Torrez

*SAND*2018-5319 C

Los Alamos NATIONAL LABORATORY EST. 1943

Sandia National Laboratories

National Nuclear Security Administration

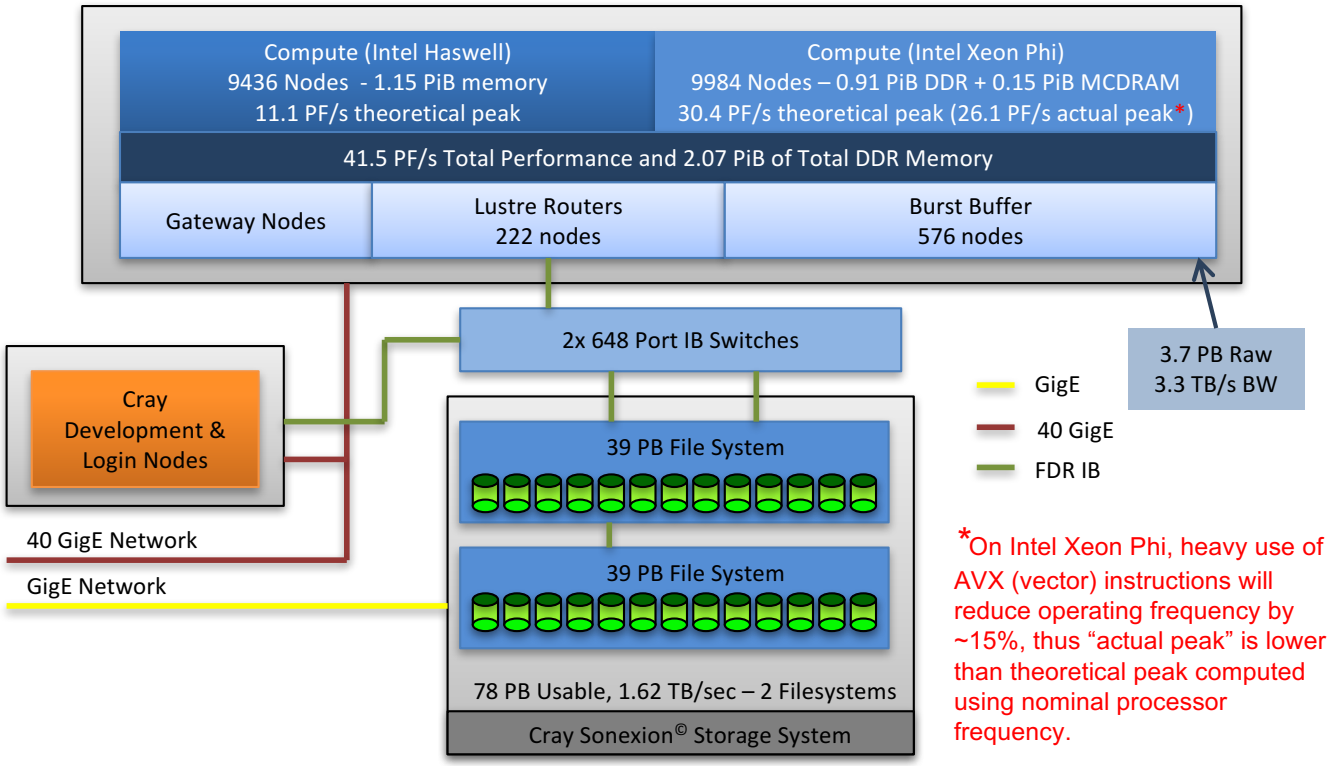
Alliance for Computing at Extreme Scale

Acknowledgements

- The authors would like to thank the EAP code team for their contributions to the Trinity Center of Excellence. The authors would also like to thank the many dedicated staff from LANL, SNL, LLNL, NERSC, Cray, Intel, and SchedMD who spent many long hours making Trinity a success.

- This work was carried out under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory supported by Contract No. DE-AC52-06NA25396 and Sandia National Laboratories.
- Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525

High Level Trinity Configuration



*On Intel Xeon Phi, heavy use of AVX (vector) instructions will reduce operating frequency by ~15%, thus "actual peak" is lower than theoretical peak computed using nominal processor frequency.

Trinity Architecture Overview

Metric	Trinity		
Node Architecture	Complete Trinity System	Intel Haswell Processor Partition	Intel Xeon Phi Processor Partition
Memory Capacity	2.07 PiB	1.15 PiB	0.91 PiB
Memory BW	>6PB/sec	>1 PB/s	>1PB/s + >4PB/s
Theoretical Peak FLOPS	41.5 PF/s	11.1 PF/s	30.4 PF/s (26.1 PF/s)
Number of Nodes	19,420	9,436	9,984
Number of Cores	980,864	301,952	678,912
Number of Cabs (incl. I/O & BB)	110		
PFS Capacity (usable)	78 PB		
PFS Bandwidth (sustained)	1.45 TB/s		
BB Capacity (usable)	3.7 PB		
BB Bandwidth (sustained)	3.3 TB/s		

Compute Node specifications

	Haswell	Knights Landing
Memory Capacity (DDR)	2x64=128 GiB	96 GiB
Memory Bandwidth (DDR)	136.5 GB/s	115.2 GB/s
# of sockets per node	2	1
# of cores	2x16=32	64 to 72
Core frequency (GHz)	2.3	TBD
# of memory channels	2x4=8	6
Memory Technology	2133 MHz DDR4	2400 MHz DDR4
Threads per core	2	4
Vector units & width (per core)	2x256 AVX2	2x512 AVX-512F
On-chip MCDRAM		16 GB @ > 500 GB/s

Trinity Merge

Trinity Merge (June 2017)

KNL Partition:

Acceptance Network

Accepted Dec 2016

Haswell Partition:

Classified Network

Haswell Cabinets Accepted Dec 2015
KNL Cabinets Accepted Dec 2016

System Merge (Planning)

- Two Partitions
  - Haswell (In Classified Production Environment)
  - KNL (Open Acceptance Network)
  - Separate Aries networks
- Merger Preparation
  - ACES Systems Integration/Production and Cray Team
    - Meetings to discuss every detail of the merger
    - Risk mitigation and fallback plans
      - Run as two separate platforms
      - Revert to Moab

System Merge (Changes)

- System Changes During the Merge
  - Convert scheduler from Moab to Slurm
  - Enable cray hugepages as the default
  - Increasing the number of DataWarp and LNET Routers
  - Reconfiguring repurposed nodes in the KNL partition
  - Almost all changes were staged in version control before the merger to reduce downtime

System Merge (Scheduler)

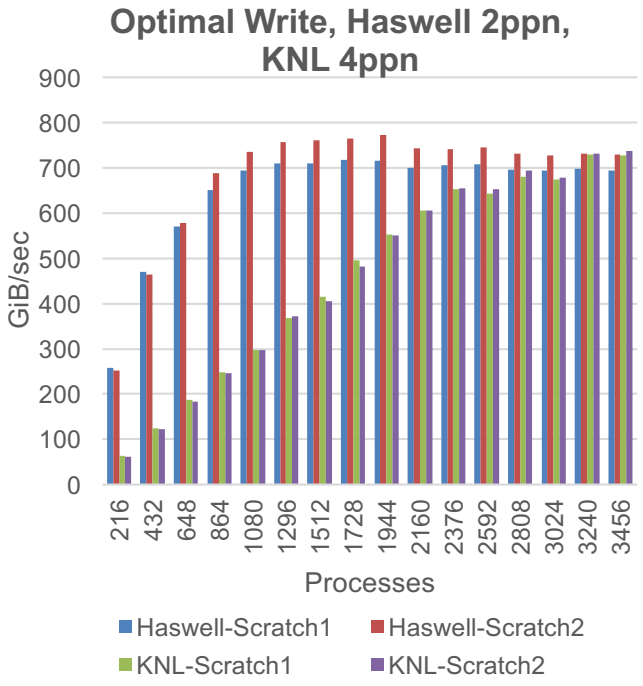
- Slurm Testing
  - Trinity Phase 2 was configured to reboot between schedulers for testing within an 8 hour schedule maintenance window
    - Allowed for scale testing of slurm
    - Test, modify, retest of configuration changes
    - Saved approximately a week of downtime
- Slurm Challenges
  - Loss of transparent process placement that ALPS provided
    - Required testing and documentation to find equivalent functionality
  - PMI_TIMEOUT issues due to loss of transparent ALPS functionality
    - Mitigated by sbcast and increasing the default PMI_TIMEOUT

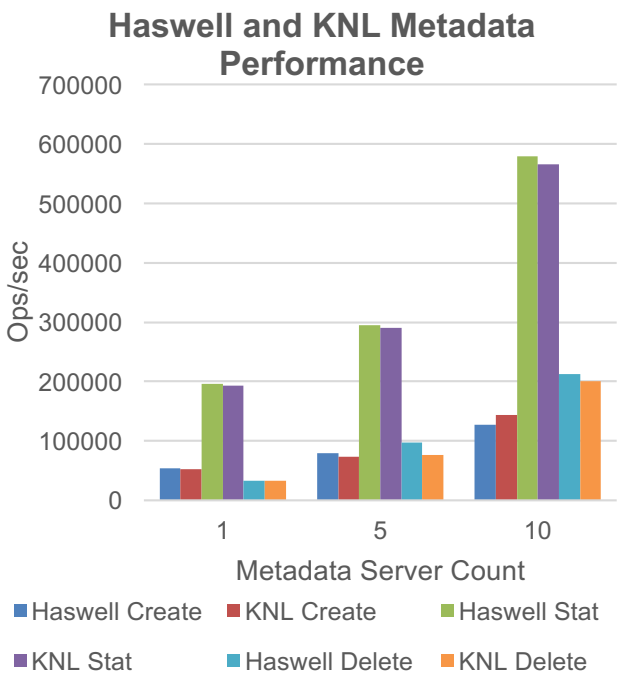
System Merge (Scale Issues)

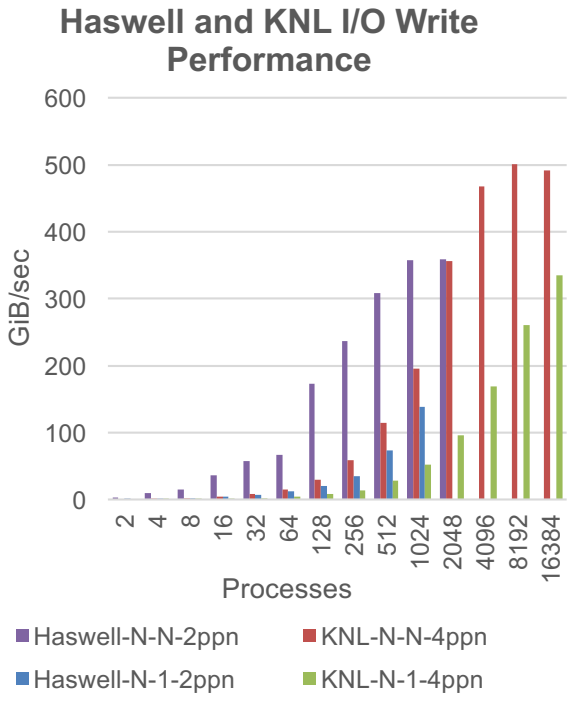
- Cray, the linux kernel, and Slurm issues
  - Hugepages
    - Enabled by default for users during the merger
    - Cray's hugepage memory usage was not accounted for by the kernel
      - Caused out of memory issues that killed nodes
      - OOM messages were not being passed back to the user
      - Further complicated by ACES use of vm.overcommit to limit memory usage which triggered this failure
- Scale issues with rebooting KNLs for mode changes
  - Over 2K nodes caused large perturbances in Slurm's ability to schedule jobs
  - Disruptive to the entire network
  - ACES decided to disable user definable KNL modes for all but 100 nodes

I/O Post-Merge

Parallel Filesystem (Lustre) Performance

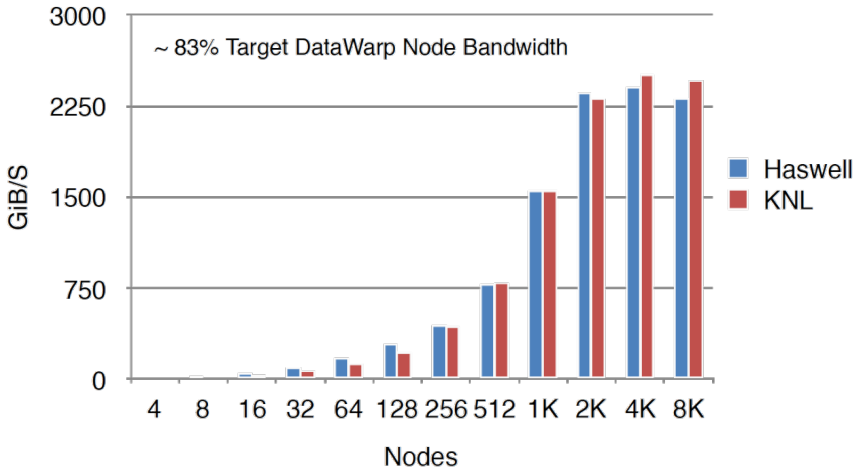




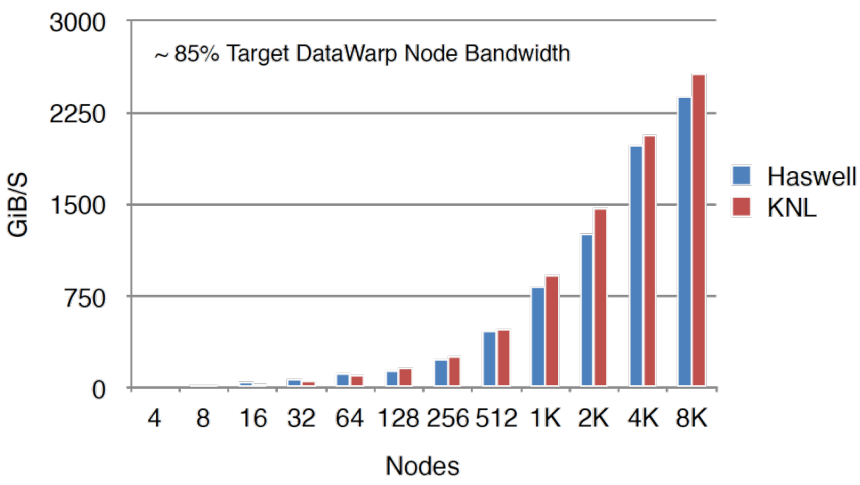


DataWarp N-1 Performance (Using IOR)

Write, N-1
4 PPN, 128 GiB/node, 1MiB xfer



Read, N-1
4 PPN, 8 GiB/node, 1 MiB xfer



DataWarp/Slurm Ongoing Issues

- DataWarp functionality is tightly coupled to the Work Load Manager (WLM),
  - Changed from Moab to Slurm during the merge.
  - Most functionality has been re-established after the merge to stage-in and stage-out data
  - Support for several complex workflows involving chained jobs are still unavailable under Slurm and are actively being developed.

- DataWarp is great for performance, but we need it to function as originally designed with ALPS
  - Need better support for ensuring stage out of files prior to execution of a dependent job
  - Need a more flexible method to stage in data generated from a “parent job” prior to invocation of the associated “child job”
  - Current workarounds - either don’t stage in files or use a persistent burst buffer allocation

Hybrid HPCG

Intel HPCG 2.4 on Merged Trinity

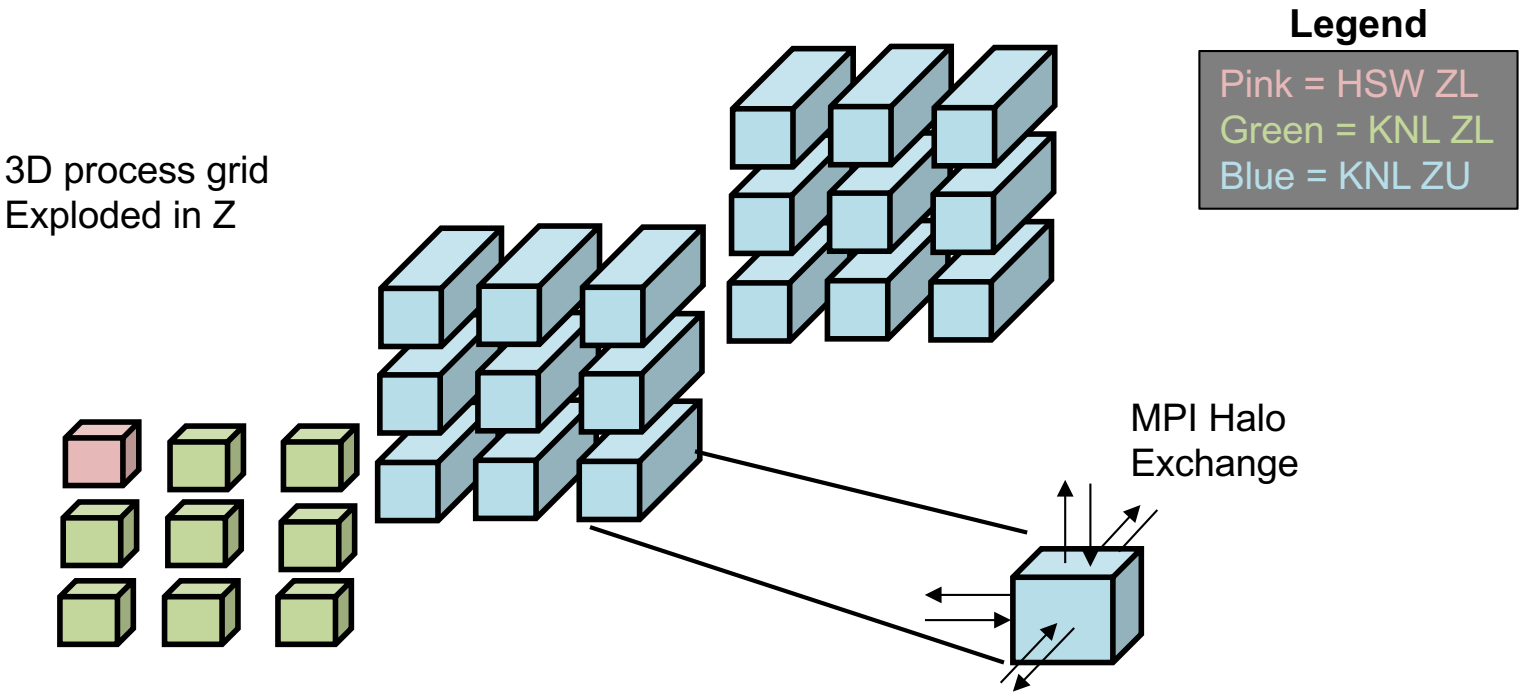
- 2-region feature designed and implemented in reference version 2 by Mike Heroux in September 2016
  - Allows for running on both Haswell and KNL nodes of Trinity
- 2-region feature integrated into Intel-optimized version by Alexander Kalinkin in Sept 2016
  - “Engineering” release, with licensing restrictions
- Advice on KNL thread/core assignment provided by Alexander Kalinkin

HPCG Run Parameters on Merged Trinity

- Run on 9372 Haswell and 9906 KNL nodes
  - 9372 * 2 ranks/node = 18744 ranks * 16 threads/rank
  - 267 KNL * 2 ranks/node = 534 ranks * 34 threads/rank
  - 9639 * 4 ranks/node = 38556 ranks * 34 threads/rank
  - 57834 ranks total
  - Process grid (nx,ny,nz) = (27,42,51)
  - pz=13, so 27x42x17 processes in region 1, 27x42x34 in 2
  - Local grid 160x160x112 in region 1, 160x160x152 in 2
  - QuickPath option used (single iteration through timed region)

HPCG 2-Region Split

(figure dervied from: http://icl.cs.utk.edu/graphics/posters/files/SC14-HPCG.pdf)



SLURM Support for HPCG 2-Region Split

- What SLURM can do (--multi-prog)
  - Separate executables for separate nodes
  - Order ranks by role (HSW_ZL, KNL_ZL, KNL_ZU)
  - Separate OMP_NUM_THREADS for separate nodes
  - Separate –nz arguments for separate regions
- What SLURM cannot do
  - Separate –ntasks-per-node for separate nodes
  - Separate –cpus-per-task for separate nodes
  - Separate –cpu_map for separate nodes
- Trinity support code developed to handle this
- Dual-partition sbatch option (-p any) provided by Paul Peltz

HPCG Trinity Support Code

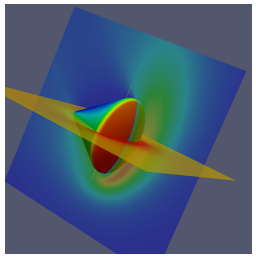
- Multiprog_driver
  - Launched by srun
  - Detects node type (/proc/cpuinfo)
  - Launches xhpcg_avx2 or xhpcg_knl, as appropriate
  - For xhpcg_knl, uses numactl –membind=1

- Post-MPI_Init setup
  - Determine role of rank (HSW_ZL, KNL_ZL, KNL_ZU)
  - Reorder ranks (HSW_ZL first, then KNL_ZL, KNL_ZU last)
  - Pin ranks and threads to cores
  - “Bench” superfluous ranks on lower-region nodes
  - Adjust NZ parameter to match ZL or ZU, as appropriate

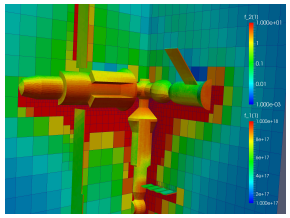
Hybrid SPARTA

SPARTA

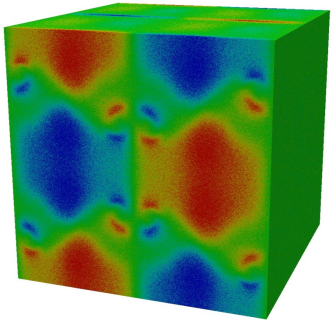
- Direct-Simulation Monte-Carlo (DSMC) code
- DSMC accurately models high-altitude hypersonic re-entry flow, resolves length scales at the particle level
- Non-equilibrium, non-continuum conditions cannot be simulated with traditional CFD or reproduced experimentally
- Can be used for:



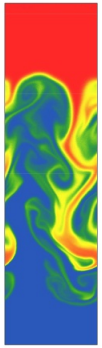
Re-entry



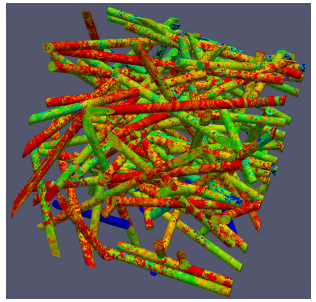
Spacecraft



Turbulence



Instabilities



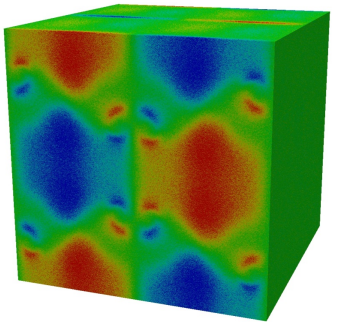
Porous Media

Summary

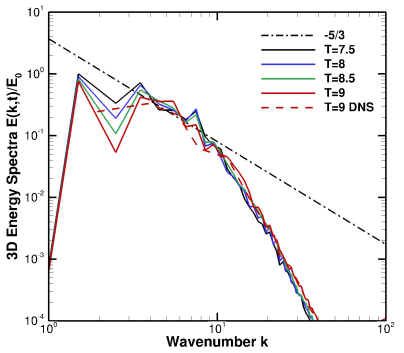
- A heterogeneous run on full Trinity was performed using SPARTA, March 9-12
- Over 19,000 nodes (9200+ Haswell and 9900+ KNL) and 1.2 million MPI processes were used
- The run was successful, with SPARTA running for several hours with good performance
- Several challenges encountered, and some unresolved issues remain

Problem Studied: Taylor--Green Flow

- DSMC applied to simulate nearly incompressible turbulent energy evolution for Reynolds numbers 450-500
- In the incompressible limit, the DSMC simulations agree with corresponding Navier-Stokes Direct Numerical Simulation (DNS) results
- Offers the possibility of gaining new insights into turbulence by directly linking molecular relaxation processes to macroscopic transport processes
- Studied by Michael Gallis (Sandia) using Sequoia. For this DAT, used same parameters as Sequoia to verify correctness and compare performance
- 3D grid with 8 billion grid cells (2000 x 2000 x 2000), ~45 particles/cell = 360 billion particles. Using 1.2 million MPI ranks = ~300,000 particles/MPI rank



Turbulence



Heterogeneous Job Launch

- Haswell = 32 physical cores x 2 hardware threads, AVX2 vector extensions
  - Configuration: 64 MPI ranks, 1 thread / rank (using hyperthreads for MPI ranks)
- KNL = 68 physical cores x 4 hardware threads, AVX-512 vector extensions
  - Configuration: 64 MPI ranks, 4 threads per rank (using hyperthreads for Open MP)
- Currently can't launch more than one srun command. Must use same sbatch and srun options for each node type. I.e. must use the same --cpu_bind" and "-c" values. Must either use same number of MPI on each node, or use a wrapper to "bench" (i.e. send to MPI finalize) extra MPI ranks and create a sub communicator that excludes benched ranks
  - Build one executable for Haswell: use Kokkos Serial backend (no OpenMP), AVX2
  - Build one executable for KNL: use Kokkos OpenMP backend, AVX-512
- Use a driver program written by Mike Davis (Cray) to stitch the two executables together: the driver program determines the node type and then launches either the HSW or KNL executable, based on node type

Challenge 1: Hardware Failures

- 6 hardware failures during the full Trinity run: 3 SIGBUS errors and 3 node failures
  - Node failures, such as a kernel panic, cause an entire node to go down until it is rebooted by admins
  - SIGBUS errors do not cause the node to go down, but will kill the srun command
    - Workaround for SIGBUS: sbcast to /tmp and statically link executable

- Addressing Hardware Failures:
  - Put srun command in a loop. Write out checkpoint files frequently, and automatically restart using the latest checkpoint file. Request extra nodes in case of node failure.
    - Worked perfectly for SIGBUS errors, job automatically restarted
    - Didn't work for node failures, may work in future Slurm version
  - Submit multiple jobs to queue instead of just 1 robust job. Leave out a few nodes in case of node failure.
    - Worked for node failures (but job has to wait through queue if not in DAT)
  - Use DataWarp burst buffers to reduce file I/O time (each checkpoint file was 20+ TB)

Challenge 2: Slow Nodes

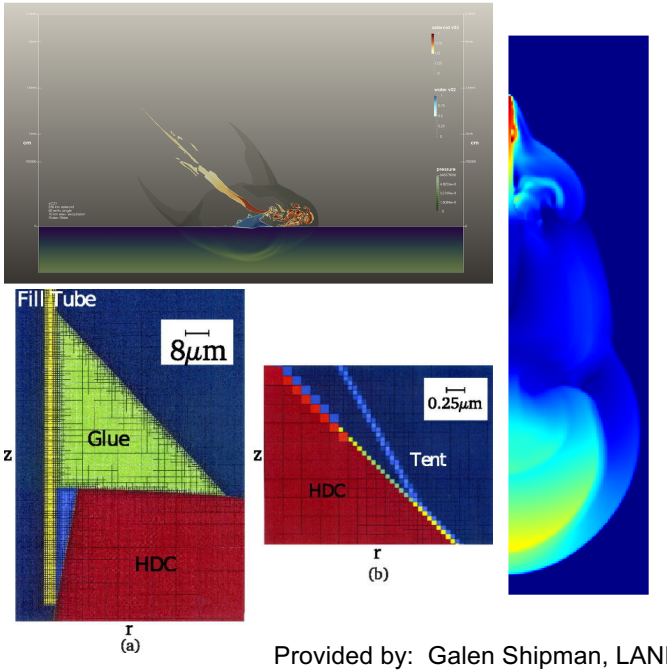
- Initially, performance was much slower than expected (but the code was running correctly and making forward progress)
  - Built-in timers showed most of the time was spent outside of normal computation (particle move, sort, collide) and in the “other” category → suggests severe load imbalance
  - If even only one node is running slow, the performance of the entire simulation will be bad due to MPI barriers or other synchronization points in the program

- Addressing Slow Nodes (simulation ran over 20x faster after mitigations):
  - Add timers to compare performance of slowest KNL MPI rank vs slowest Haswell MPI rank.
    - Slowest Haswell is many times slower
- Find hostname of slow Haswell node using a script and exclude slow node from allocation
- Repeat to find and exclude a second bad Haswell node

Center of Excellence
Example Interaction: xRAGE

xRAGE can model a variety of multi-physics problems

- Asteroid impact simulations
- Shape charge experiments
- Inertial Confinement Fusion simulations



Provided by: Galen Shipman, LANL

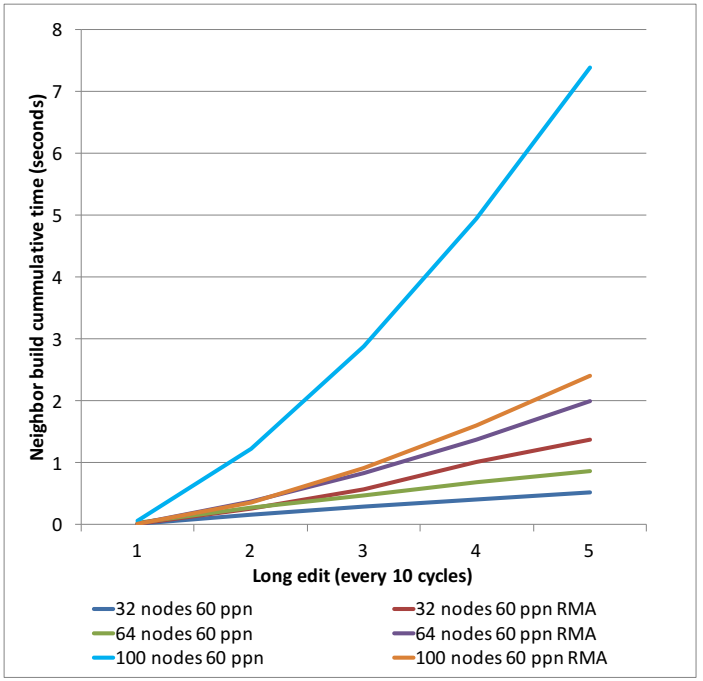
Morrison, David, e al. "Asteroid Generated Tsunami: Summary of NASA/NOAA Workshop." (2017).

Masser, Thomas, et al. *Shaped Charge Comparisons. Pagosa and xRage Code Study*. No. LA-UR-17-27355. Los Alamos National Laboratory (LANL), 2017.

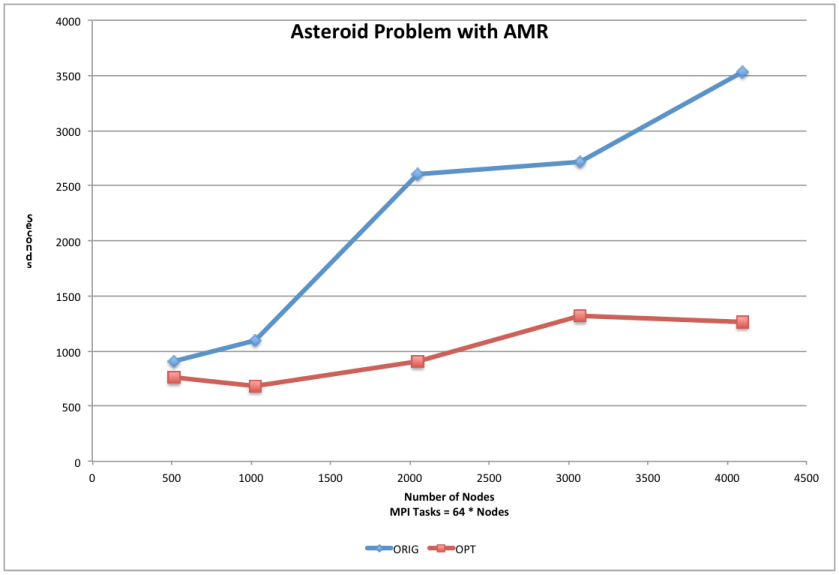
Haines, Brian M., et al. "The effects of convergence ratio on the implosion behavior of DT layered inertial confinement fusion capsules." *Physics of Plasmas* 24.7 (2017): 072709.

Recent work in optimization of communication

- xRAGE makes extensive use of alltoall and allreduce communication
- Alltoall used to build communication pairs during mesh reconfiguration (your neighbors can change every cycle)
- Identified this scalability bottleneck with the COE using CrayPAT
- Implemented an RMA based alltoall solution that takes advantage of sparsity
- Incorporated a asynchronous barrier optimization allowing overlap of synchronization and use of the RMA window
- Cray delivered an optimized Cray MPICH for our RMA use-case → DEC PE



Significant Scalability Improvements in AMR via RMA Alltoall and Comm Caching

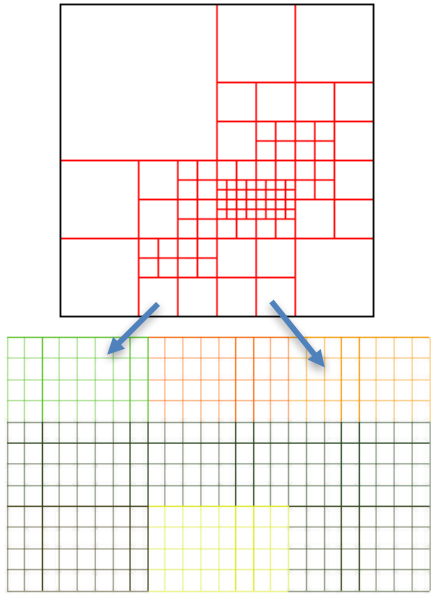


- The results are outstanding!
- ~3x performance improvement at scale
  - Total application runtime!

- RMA based Alltoall
- Caching and reuse of AMR communication pattern

Improvement in building AMR to structured mesh map

- xRAGE can generate mappings between the AMR mesh and a structured mesh
- At each cycle the AMR mesh changes and the mapping must be rebuilt
- In some problems this was taking up to 60% of runtime (1200 Nodes on KNL)
- Reduced this time to 8% of runtime
- Identifying the performance issue was challenging
- Working with the COE we identified a load imbalance in the calculation of the topology intersections using CrayPAT, a handful of ranks were taking a degenerate code path
- LANL staff then implemented a in-place heap sort of the mesh topology and a binary search significantly reducing the load imbalance
- Improved memory usage reporting (hugepages not included in RSS)



Trinity is delivering valuable production computing time to NNSA

- The complete heterogeneous Trinity supercomputer, with both Haswell and KNL nodes, is delivering valuable production computing time to the NNSA to solve significant problems and provide new opportunities for scientific investigation.
- The journey of deploying Trinity and the challenges presented by the scale and complexity of the system are becoming the norm as each generation of supercomputer pushes the limits.
  - Beyond the initial deployment, each software upgrade to Trinity requires dedicated efforts to ensure performance expectations persist through its lifetime.
  - It is only through close collaborations with our vendor partners that we are able to stand up these systems that are essential to pushing the limits of our scientific knowledge.


CRAY
Alliance for Computing at Extreme Scale
TRINITY
NNSA
Los Alamos
Sandia National Laboratories

Questions?