Exceptional service in the national interest

Sandia National Laboratories

SPARTA Heterogeneous Full Trinity Runs: Successes and Challenges

Stan Moore and Michael Gallis

IXPUG Annual Fall Conference 2018

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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-NA-0003525. SAND2018-10786 C

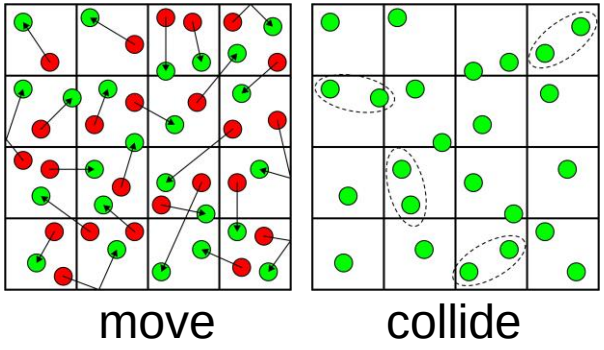
Outline

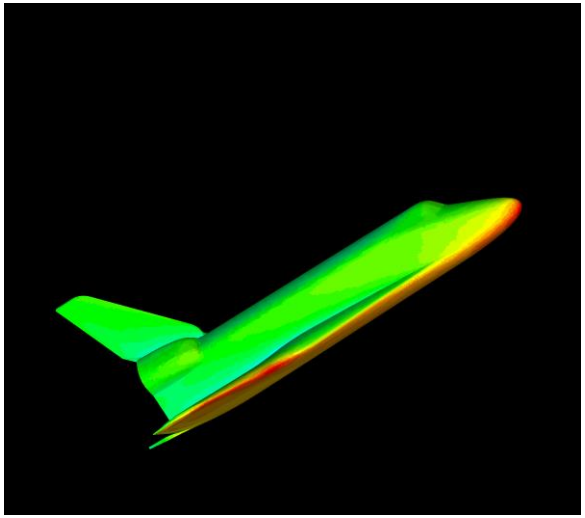
- SPARTA DSMC code
- Trinity supercomputer
- Heterogeneous full Trinity runs
- Challenges
- Lessons learned, towards resilience
- Path forward

Direct Simulation Monte Carlo (DSMC)

Sandia National Laboratories

- Fluids are typically modeled by assuming continuum flow and solving Navier-Stokes equations (i.e. CFD)
- Non-equilibrium, non-continuum conditions cannot be simulated with traditional CFD or reproduced experimentally
- DSMC typically used to simulate low density (rarefied) fluid flows such as hypersonic reentry at high altitudes
- Computational particles move, reflect off boundaries, and collide like real molecules
- Computational and algorithmic advances are now bringing hydrodynamic flows within reach:
  - Instabilities
  - Turbulence

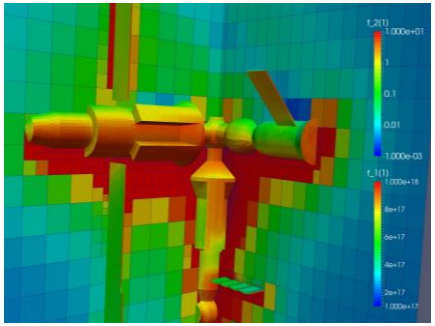




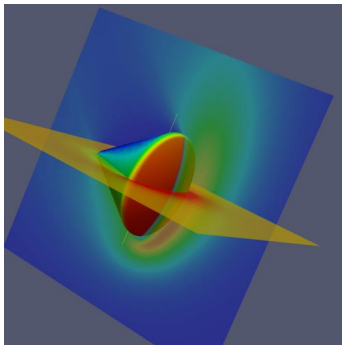
SPARTA

(**S**tochastic **PA**rallel **R**arefied-gas **T**ime-accurate **A**nalyzer)

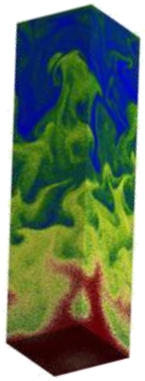
- Direct Simulation Monte Carlo (DSMC) code
- Core developers are Steve Plimpton and Michael Gallis (Sandia National Labs)
- Open-source, http://sparta.sandia.gov



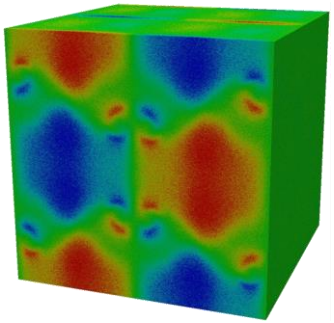
Spacecraft



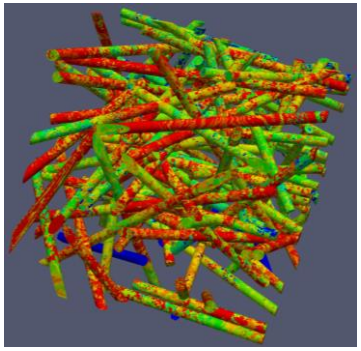
Re-entry



Instabilities



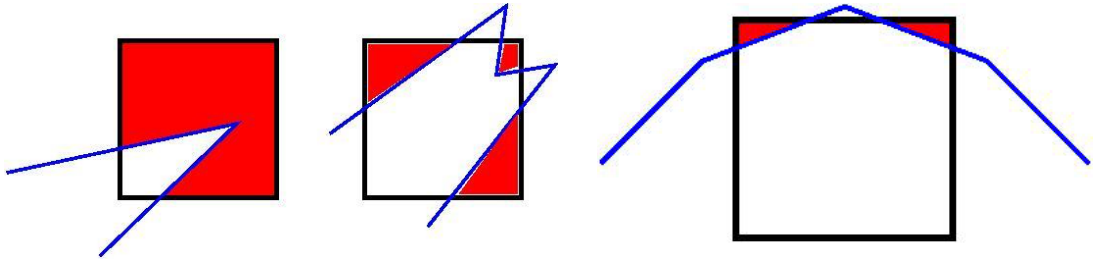
Turbulence

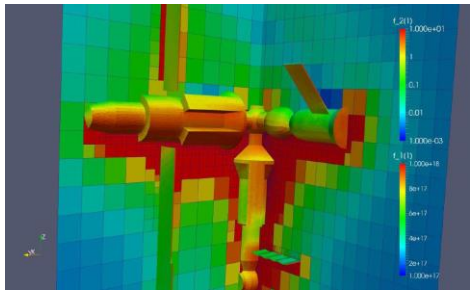


Porous Media

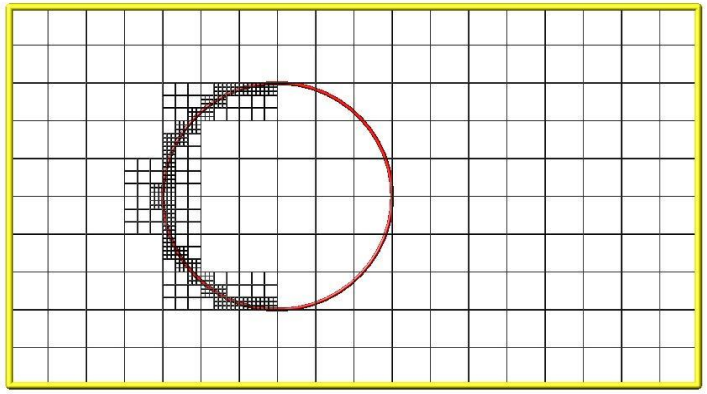
SPARTA Features

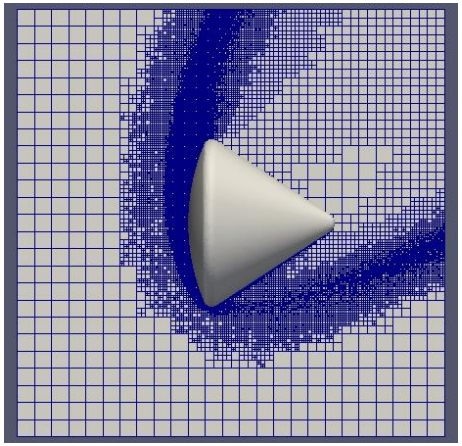
- Structured grids with complex surfaces via cut and split cells





- Hierarchal grids with adaptive mesh refinement

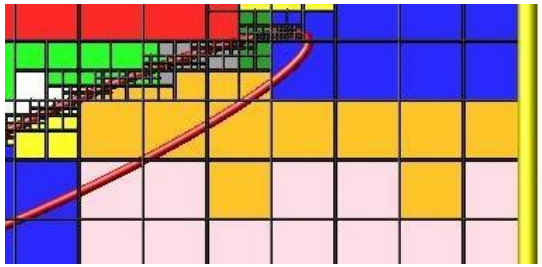


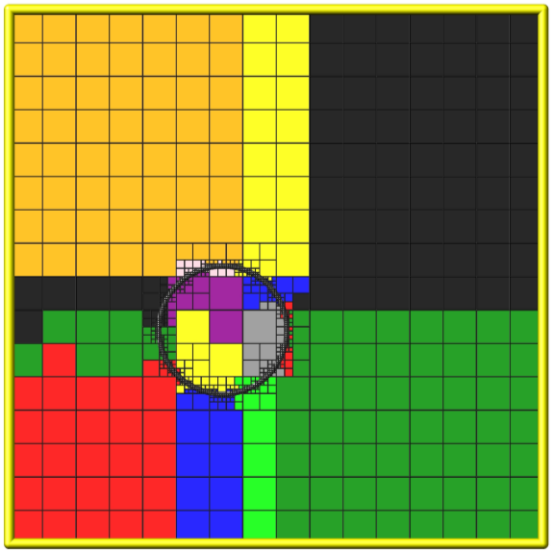


- MPI parallelism using highly scalable domain decomposition

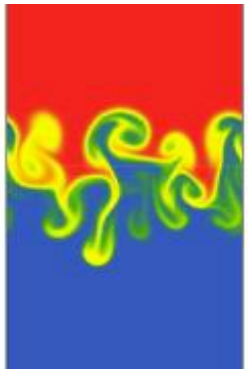
SPARTA Features (cont.)

- Load balancing (static and dynamic)





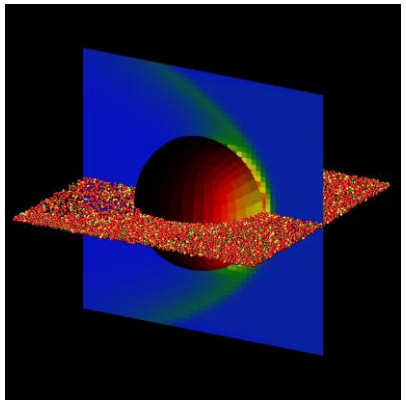
- Gas-phase collisions and chemistry

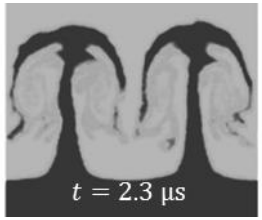


- Surface collisions and chemistry
- Grid cell weighting of particles
- 3D, 2D, and 2D axisymmetric geometries

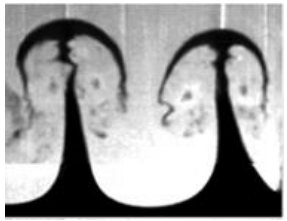
SPARTA Features (cont.)

- Diagnostics
  - global boundary statistics
  - per grid cell statistics
  - per surface element statistics
  - time-averaging of global, grid, surface statistics
- In-Situ Visualization

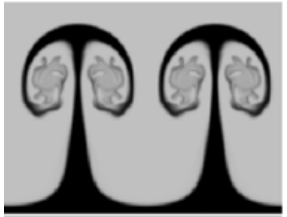




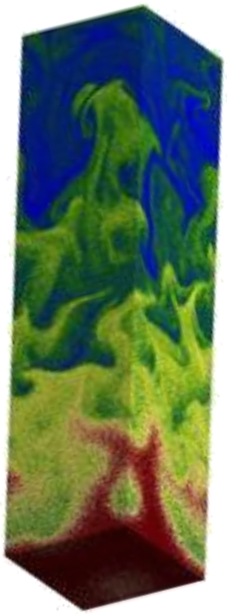
DSMC



Experiment



Navier-Stokes



- And more

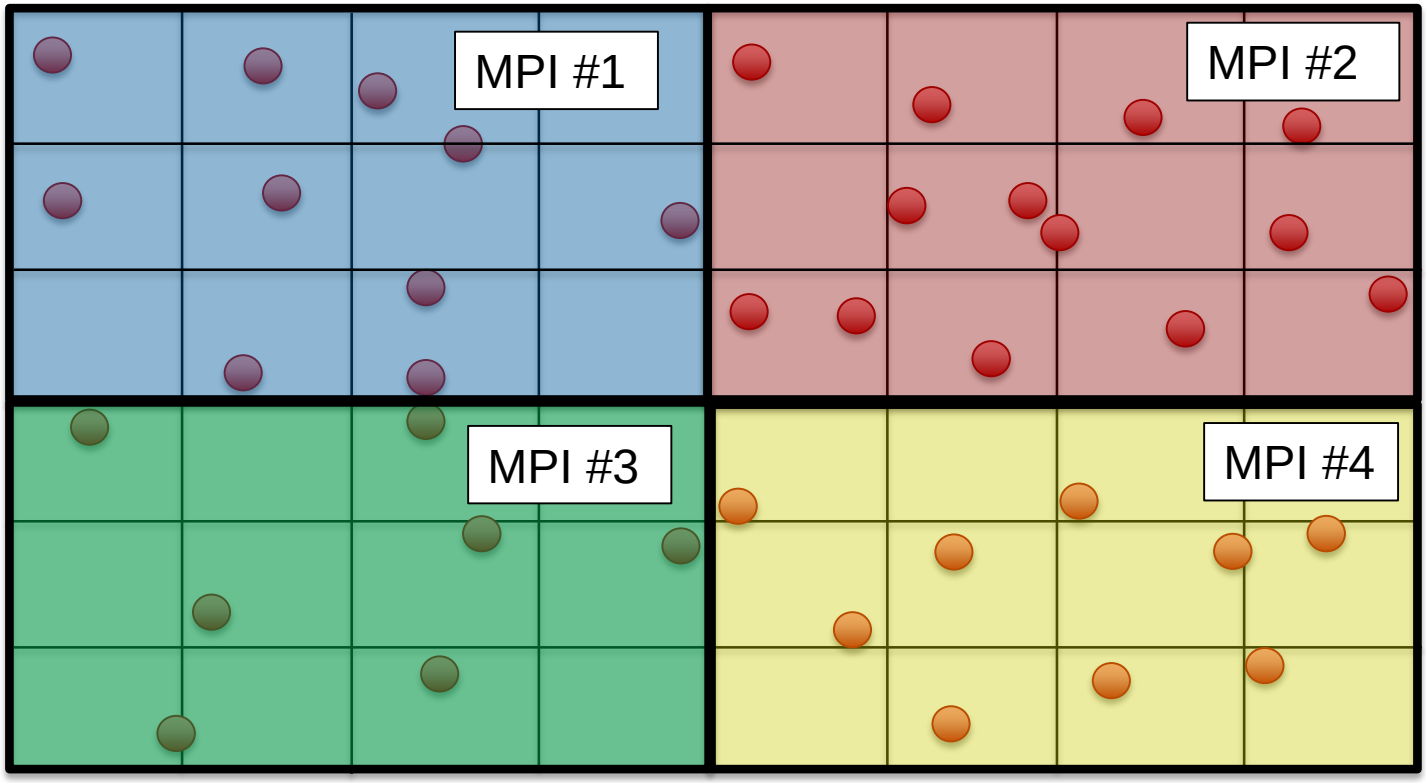
Kokkos Library

Sandia National Laboratories

- Modern HPC hardware is complicated and ever changing
- The Kokkos library is an abstraction layer between the programmer and these platforms: write C++ code once using Kokkos abstractions
- Kokkos:
  1. Parallel dispatch—threaded kernels are launched and mapped onto backend languages such as CUDA, OpenMP, or Pthreads
  2. Kokkos views—multidimensional arrays with polymorphic memory layouts that can be optimized for a specific hardware (such as C-style layout right vs Fortran-style layout left)
- Used on top of existing MPI parallelization (MPI + X)
- Goal is performance portability
- Open-source, can be downloaded from https://github.com/kokkos/kokkos
- Kokkos version of SPARTA is implemented as an optional add-on package

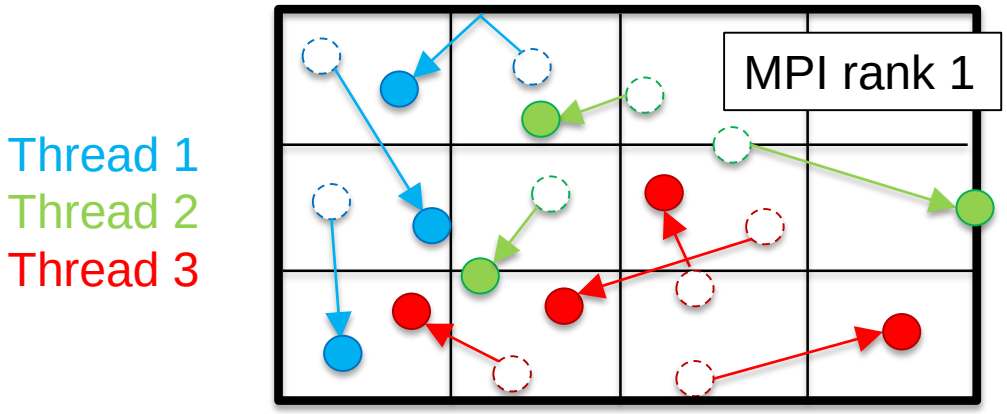
MPI Parallelization Approach

- Domain decomposition: each processor owns a portion of the simulation domain and particles therein



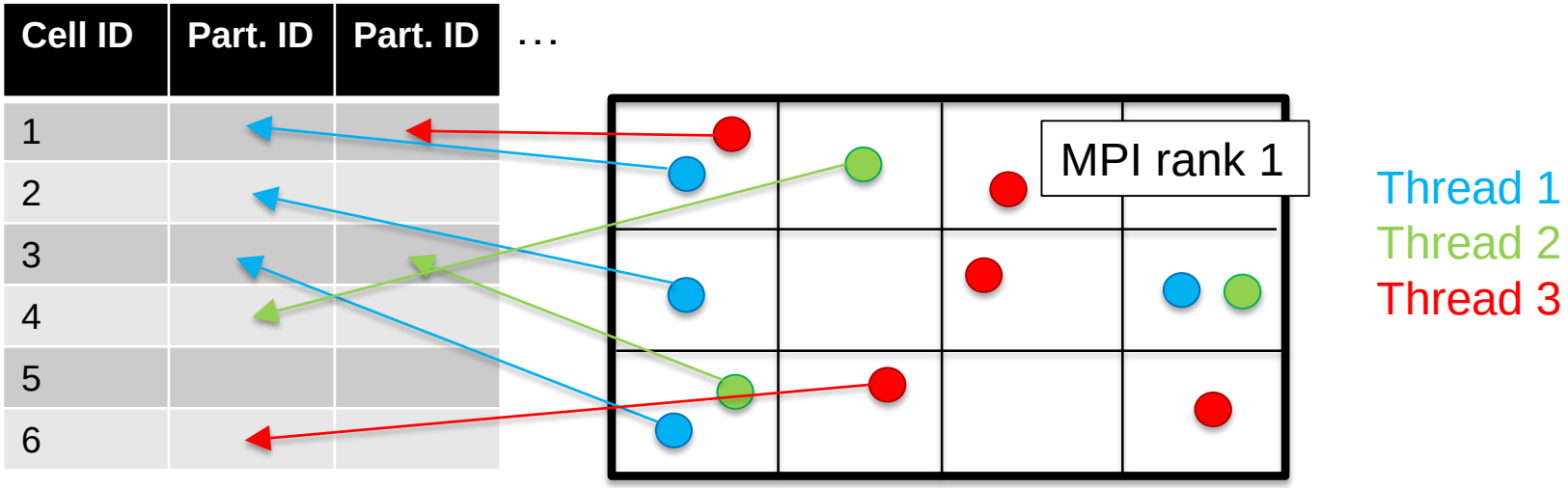
Kokkos Threaded Move

- One thread pushes particles for a timestep or micro-iteration
- Intermediate grid crossings are found
- Statistical accumulators (i.e. number of moves, number of surface collisions, etc.) use either a parallel reduction or an atomic reduction on a global variable



Kokkos Threaded Sort

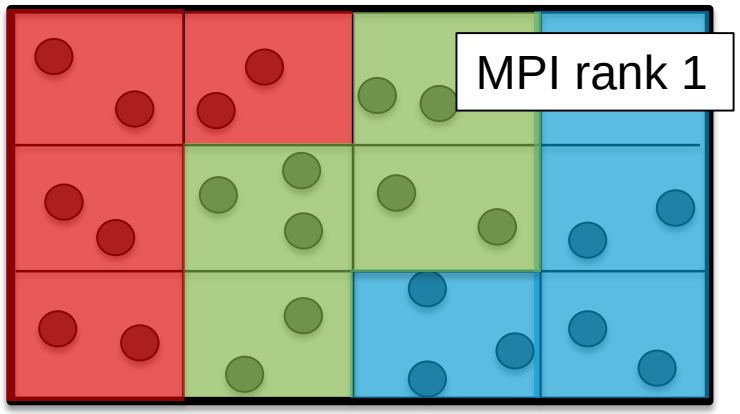
- Threads loop over particles to sort by grid cell
- 2D array of grid cells vs particle IDs is created, along with 1D array of counts of particles in each cell
- Requires thread atomics to avoid write conflicts
- If 2D array is too small, increase second dimension, realloc, and try again
- Periodically reordering particle list by cell id can improve performance



Kokkos Threaded Collide

- Each thread processes all the collisions in a grid cell
- Nearest neighbor algorithm also supported

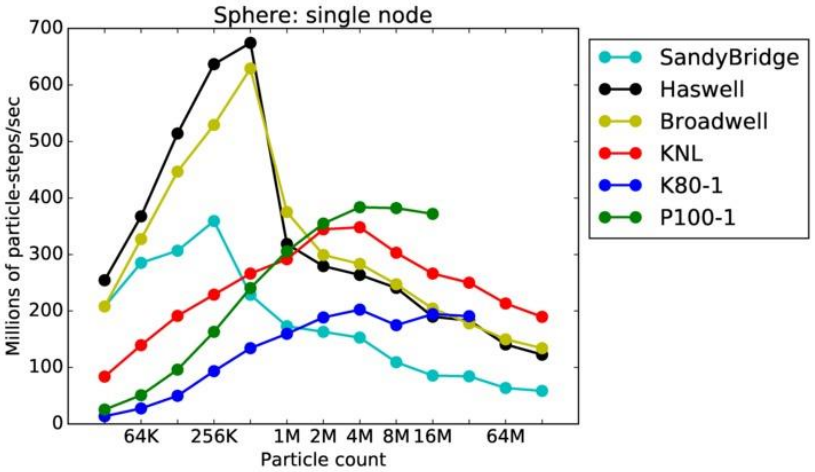
Thread 1
Thread 2
Thread 3

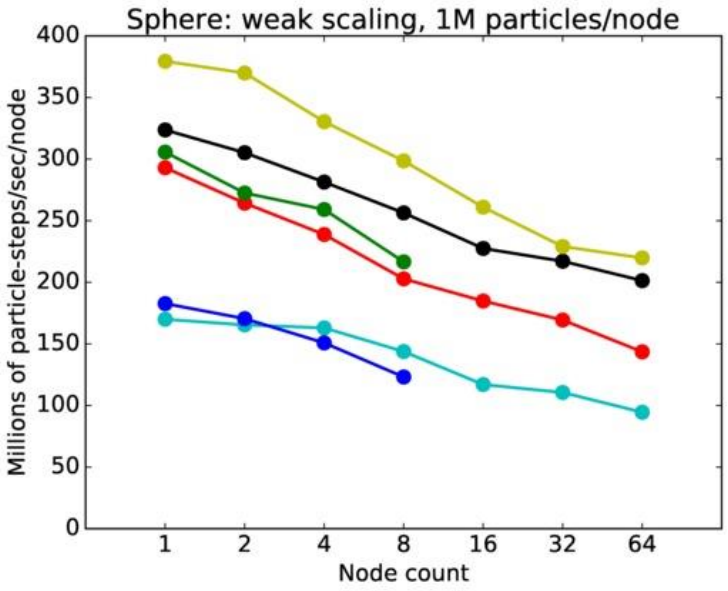


Performance Portability: Sphere Benchmark

Sandia National Laboratories

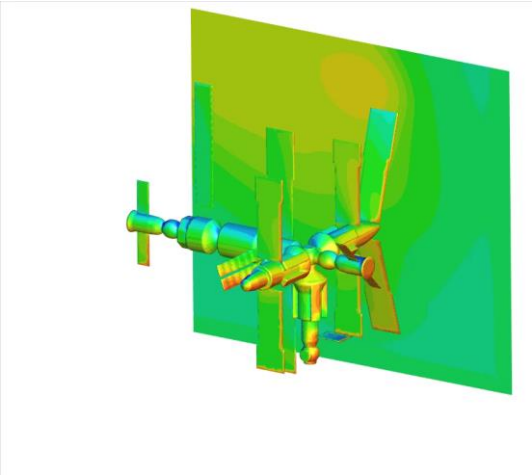
- Benchmark: particles flowing around a sphere
- 1 CPU node or 1 GPU
- Best performance using either Kokkos or MPI-only
- Large cache effect for small problem sizes



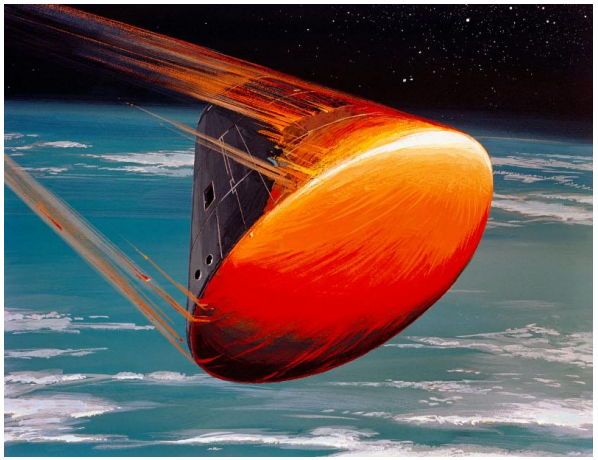


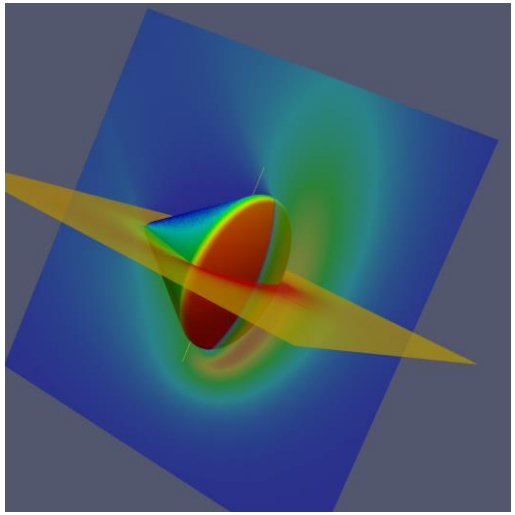
Typical Problems

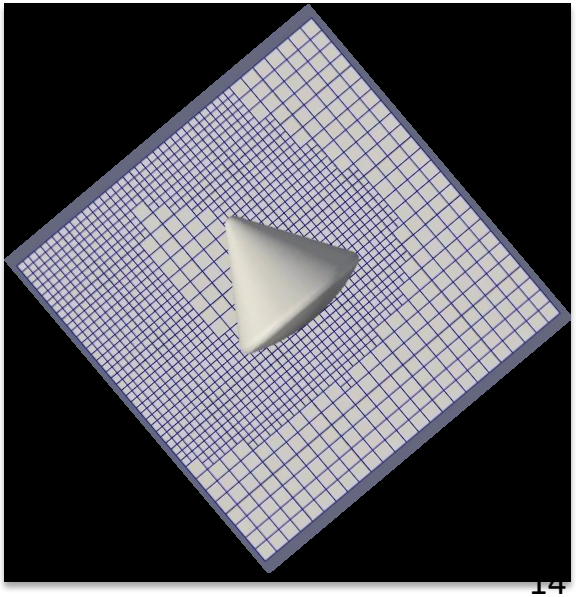
- Fluid flow around the Mir space station
- The plane shows the temperature profile, with the surface colored according to the heat flux



- 3D simulation of Apollo capsule re-entry
- Grid refined to 5 levels

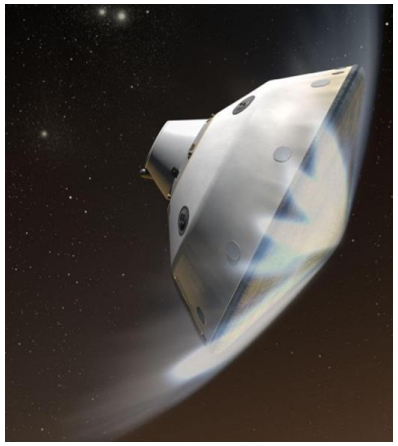


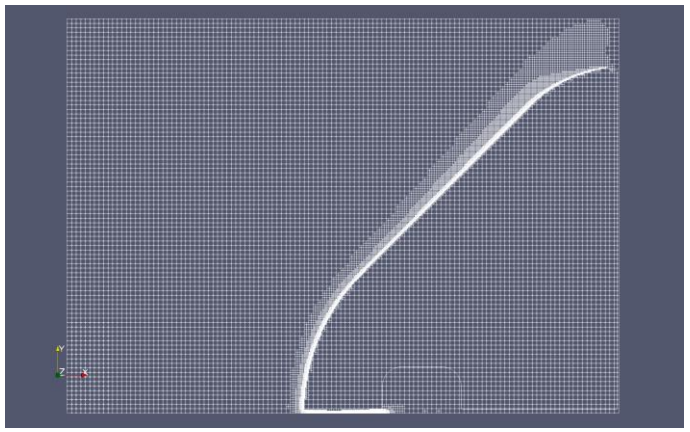


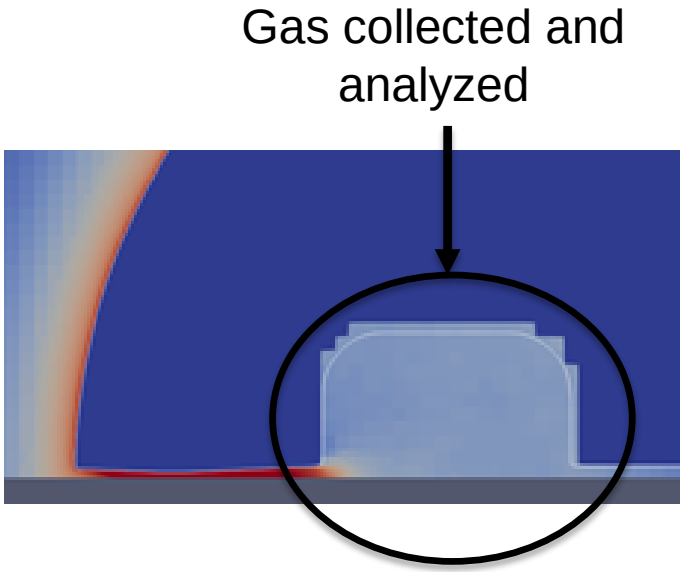


Cupid's Arrow Test Case

- Mission concept: smallsat that could sample Venus' upper atmosphere
- Axisymmetric simulation
- Grid refined to 13 levels to capture mean free path! (highest density inside nozzle)



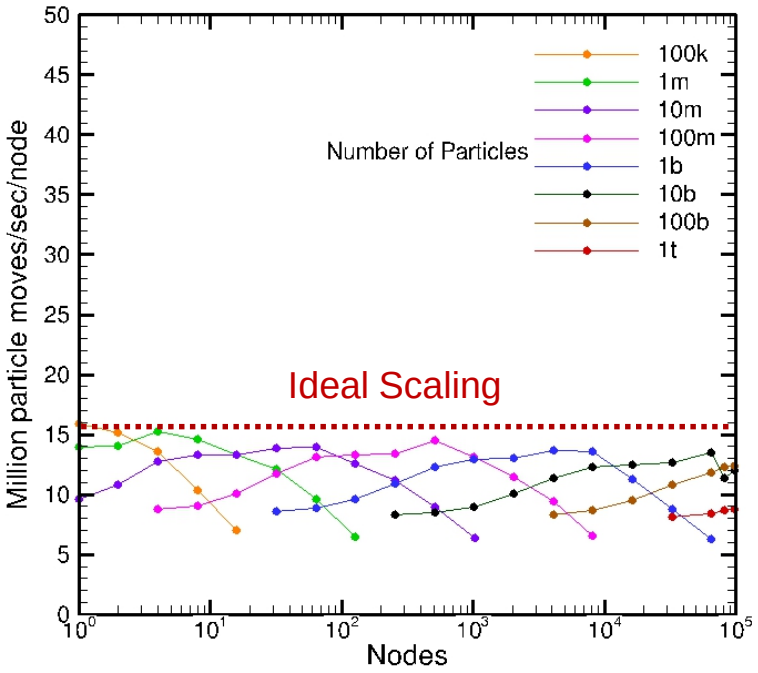




[1] https://www.jpl.nasa.gov/news/news.php?feature=6791
[2] https://trs.jpl.nasa.gov/bitstream/handle/2014/45775/15-4602_A1b.pdf?sequence=1

Large Scale Simulations

- DSMC algorithm scales well and is a good candidate for exascale computing
- SPARTA runs on 1.57 million BG/Q cores (Sequoia), largest simulation = **1 trillion grid cells**



EXASCALE COMPUTING PROJECT

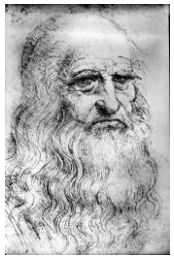
Why DSMC?

Sandia National Laboratories

- Exascale computing, despite its 50x leap over current petascale computing capabilities, does not guarantee that using larger and more refined CFD simulations will produce a corresponding leap in the accuracy of simulations
- Traditional CFD is only as accurate as the underlying equations being solved
- May be shortcomings in those equations, and including additional physics found at the molecular level could improve the accuracy of CFD simulations
- Examine difficult problems on a smaller, more fundamental scale using DSMC

Turbulence: The Greatest Unsolved Problem

Sandia National Laboratories



Leonardo da Vinci

Importance and intractability of understanding turbulence was recognized early on!

Richard Feynman

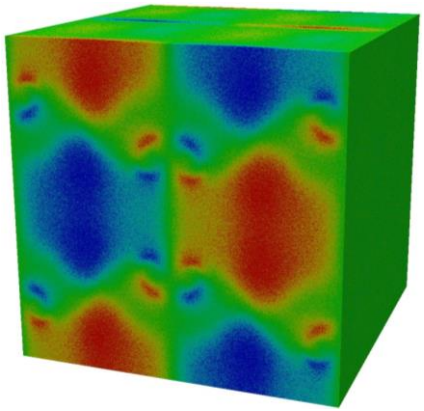
"Turbulence is the most important unsolved problem of classical physics." (Nobel Prize, quantum electrodynamics)

"When I meet God, I'm going to ask him two questions: why relativity? And why turbulence? I really believe he'll have an answer for the first."

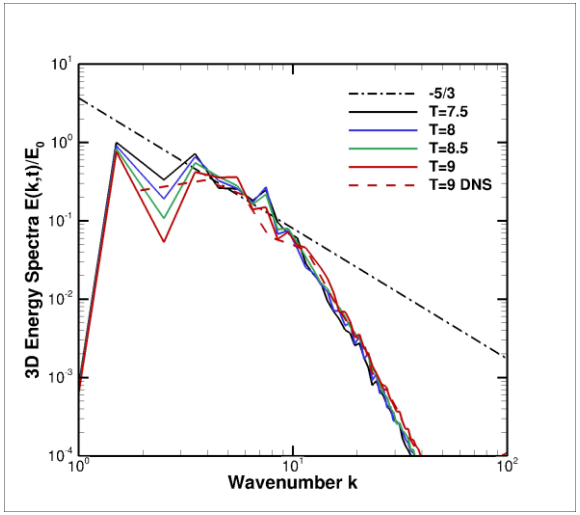
--attributed to physicist Werner Heisenberg (though probably apocryphal)

Problem Run: 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
- Need really big simulations to simulate turbulence at the molecular level!
- 3D grid with 8 billion grid cells (2000 x 2000 x 2000), ~45 particles/cell = 360 billion particles
- Work done by Michael Gallis (Sandia)



Turbulence



Trinity

Sandia National Laboratories

- Cray XC40 product located at Los Alamos National Lab
- Over 19,000 nodes (9200+ Haswell and 9900+ KNL)
- Haswell = 32 physical cores x 2 hardware threads, AVX2 vector extensions
- KNL = 68 physical cores x 4 hardware threads, AVX-512 vector extensions, MCDRAM high-bandwidth memory

Why Heterogeneous?

Sandia National Laboratories

- Trinity has 9200+ Haswell and 9900+ KNL nodes
- Can use more memory and compute
  - Can run bigger problems
  - Boost throughput of production runs
- System hardening and resilience: large runs expose weaknesses in hardware and software
  - Just takes 1 node failure to kill the job
  - Some issues only manifest in large simulations
- Large-scale computing initiative: use the machine as it was designed

> "I know how to make 4 horses pull a cart -- I don't know how to make 1024 chickens do it." [or 19,000 mice, or 1.2 million ants…]
> -- *Enrico Clementi*

Heterogeneous Job Launch

Sandia National Laboratories

- Mike Davis (Cray) previously ran HPL and HPCG benchmarks heterogeneously across both partitions of Trinity. No science code had every run at full scale though.
- Used a driver program written by Mike Davis to stitch the two executables together
- Build one executable for Haswell: use Kokkos Serial backend (no OpenMP), AVX2
- Build one executable for KNL: use Kokkos OpenMP backend, AVX-512
- Driver program determines the node type and then launches either the HSW or KNL executable, based on node type
- Must use special “any” partition that is a union of KNL and Haswell partitions

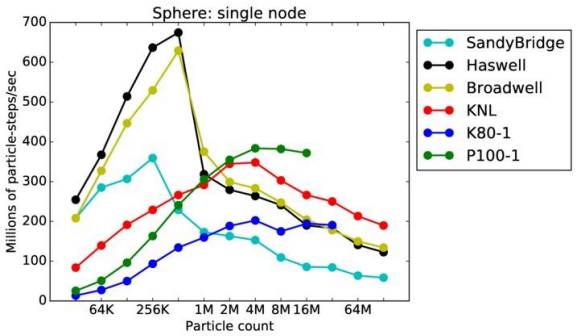
Heterogeneous Job Launch

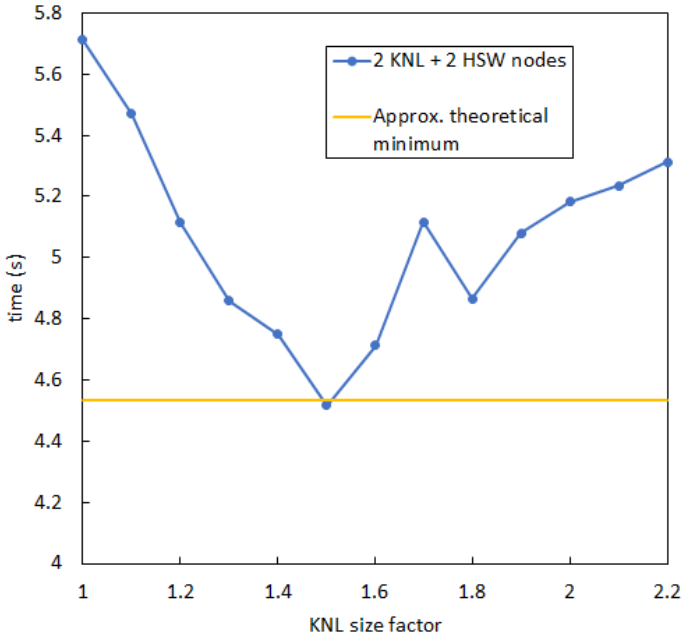
Sandia National Laboratories

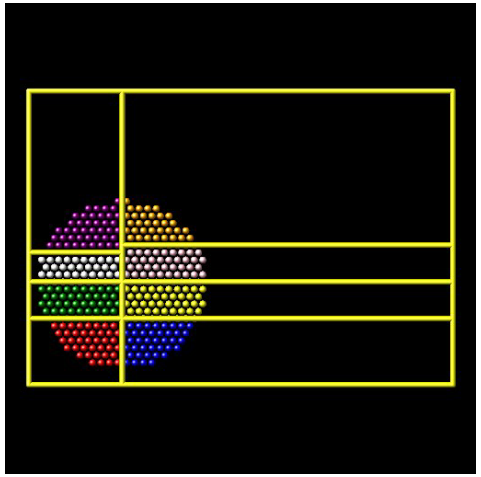
- Currently can't launch more than one srun command. Must use same sbatch and srun options for each node type. I.e. must use the same --cpu_bind" and "-c" values. Must either use same number of MPI on each node, or use a wrapper to "bench" (i.e. send to MPI finalize) extra MPI ranks and create a sub communicator that excludes benched ranks
- Use 64 MPI on each node type, plus 4 OpenMP threads on KNL (uses hyperthreading), KNL in quad, cache mode
- MPI calls must match up exactly between executables or program may hang
- Kokkos gives flexibility by allowing Serial or OpenMP threading

Load Balancing for Heterogeneous Runs

- Static recursive bisectioning (RCB): manually shift work between KNL and Haswell partitions using a constant factor
- Dynamic RCB: timer based, iteratively shifts work between KNL and Haswell partitions over time







Cray DataWarp Burst Buffers

Sandia National Laboratories

- Cray XC40 DataWarp™ (DW) burst buffer utilizes Intel p3608 solid-state drives (SSDs)
- Can substantially increase I/O bandwidth for some applications
- Trinity: 576 burst buffer nodes each with 6.4 TB capacity and a peak sequential read and write bandwidth of ~5.7 GB/s, Total = 3.6 PB and 3.2 TB/s
- SPARTA supports writing N MPI ranks → M files (with M <= N), e.g. 29 TB checkpoint broken into 16,000 files
- Use persistent DW allocation to eliminate stage-in overhead and dependencies
- Stage out checkpoint files from DW to Lustre after every job (later show a better way)

Resilience

Sandia National Laboratories

- Periodically write out checkpoint files to DataWarp (each checkpoint = ~29 TB)
- Toggle between two filenames to save space
- Submit multiple jobs to the queue
- When a job dies for any reason, the next job restarts using the latest checkpoint file
- If a checkpoint file is corrupted (e.g. job dies while writing out last file) switch to backup file
- Fairly robust (but not bulletproof)

Results

Sandia National Laboratories

- 3 full Trinity heterogeneous runs so far
- SPARTA ran on entire Trinity supercomputer using over 19,000 nodes and **1.2 million MPI ranks** (9200+ Haswell CPU nodes and 9900+ KNL Xeon Phi nodes) for several hours with good performance
- Tried to learn from each full heterogeneous run and make workflow more robust to failure modes discovered (but new failure modes also appeared)
- Goal: submit job and walk away

March 2018: First Heterogeneous Trinity Run

Sandia National Laboratories

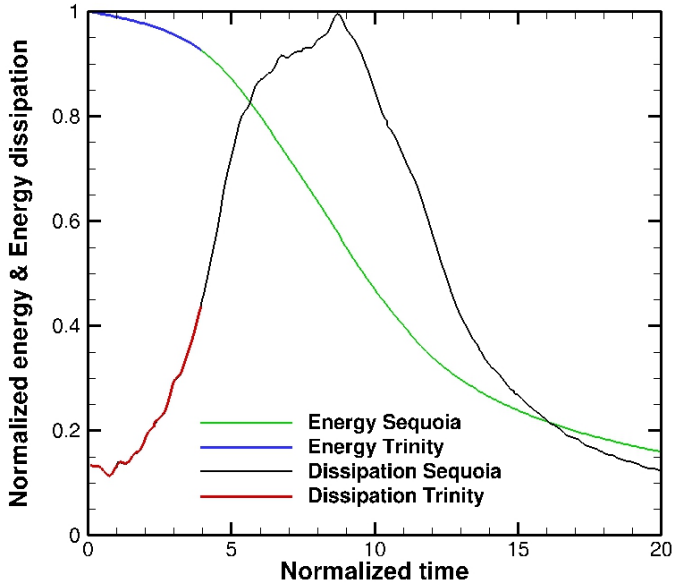
SPARTA ran for several hours producing useful results, but several challenges were encountered:

- SPARTA hung when reading in checkpoint files, fixed during allocation
- SPARTA hung when using dynamic load balancing, so no load balancing used
- At first, SPARTA ran very slow but still made forward progress, found 2 slow Haswell nodes and excluded them from the allocation
- 6 hardware errors that killed the job: 3 SIGBUS errors and 3 node failures
- Subtle checkpoint corruption, had to manually switch to backup file

Performance

First full heterogeneous run:

- SPARTA performance on full Trinity = 8 timesteps/s
- 1/3 Sequoia = 2.3 timesteps/s → full Sequoia = 6.9 timesteps/s (neglecting MPI scaling overhead)
- Without checkpoint file I/O: 1/3 Sequoia = 2.6 timesteps/s → full Sequoia = 7.8 timesteps/s
- Dynamic load balancing not turned on



Energy and energy dissipation for a Re=500, Ma=0.3 Taylor-Green Argon flow.

May 2018: Second Heterogeneous Trinity Run

Sandia National Laboratories

SPARTA ran for several hours producing useful results, dynamic load balancing worked, but several challenges were encountered:

- Started with a healthy allocation of nodes, but after 4 hours into the job, a Haswell node started running 5x slower than normal (during the night) and severely degraded job performance until we discovered the issue in the morning
- 3 node failures that killed the job each time
- 3 bus errors that killed the job each time (could have been prevented, workaround wasn't implemented correctly)
- After DAT, DW had a solid-state drive (SSD) failure and all of the checkpoint data was lost ☹

June 2018: Third Heterogeneous Trinity Run

Sandia National Laboratories

Produced useful results (230,000 timesteps worth of data), but there were also some issues:

1. Slow KNL node was found and excluded at the beginning of the DAT
2. Three node uncorrectable errors that killed the job
3. Two sbcast failures that crashed the job on re-start
4. Multiple DataWarp stage-outs happening at the same time
5. DataWarp filesystem became partially read-only which led to data loss, and corrupted both checkpoint files
6. Had to start simulation over ☹

Challenges

Sandia National Laboratories

1. Bugs in SPARTA code (now fixed)
2. SIGBUS errors (now fixed, at least for SPARTA)
3. Node uncorrectable errors
4. Slow nodes
5. Data loss leading to checkpoint file corruption and other issues when using Cray DataWarp burst buffers

1. SPARTA Issues

Sandia National Laboratories

- Code hung when reading in checkpoint files at full scale. Cause: bit shift too large, needed to use “1L” literal and 64 bit integer. Undefined behavior, didn’t show up on Sequoia at same scale. Fixed during first DAT
- Code hung when using dynamic load balancing. Issue didn’t show up on small testbed for small problems. Kokkos hash table lookup failed. Fixed after first DAT
- Lesson learned: just because a code runs fine on a few nodes doesn’t mean it will run fine on 1.2 million MPI ranks!

2. SIGBUS Errors

Sandia National Laboratories

- Job killed due to “SIGBUS error” or “bus error” after a few hours
- Kills the srun command but does not cause the node to go down
- Workaround: statically link executable and place in /tmp directory on compute nodes
- Cause: bug in Lustre, paged out text part of executable and then tried to read from it
- Permanent Solution: SU28 patch, now fixed on Trinity, at least for SPARTA ☺

3. Node Uncorrectable Errors

- Job killed due to “node failure”
- Kills the srun command AND causes the node to go into “down” state
- Cause: hardware failure such as CATERR (BANK 255), ROB timeout (Bank 0 Bus), Bank 2 Bus/L2, BANK 4 FEC, etc.
- Mostly on KNL nodes
- SPARTA has a much higher node failure rate than average—SPARTA algorithms seems to stress KNL hardware (recent 24 hour, 5000 KNL node jobs had 80% failure rate)

Image from: https://www.top500.org/news/intel-dumps-knights-hill-future-of-xeon-phi-product-line-uncertain/

Workaround for Node Uncorrectable Errors

Sandia National Laboratories

Good:

- Submit multiple jobs to queue
- When a job dies due to node failure, the next job restarts using the latest checkpoint file
- But lose allocation and must wait through queue again → overhead

Better:

- Request a few (i.e. 3) extra nodes more than are used in srun command
- Put srun command in a loop, use the “#SBATCH --no-kill” option: when a node drops out, restart using latest checkpoint file and one of the backup nodes
- Avoids allocation loss and overhead of requeuing
- But fails (hangs) with current Slurm version on Trinity, supposed to be fixed in Slurm 17.11.6 or later with the "slurmsmwd" daemon running
- Could also strip down SPARTA code to find which kernel is stressing out the KNL hardware, but would require expending huge amount of computing resources and time

4. Slow Nodes

- Nodes running 4x to 20x slower than normal
- Almost always on Haswell nodes
- Cause: memory DIMM encountering a high rate of correctable errors (i.e. bursting DIMM)
- If even only one node is running slow, the entire simulation will run slow due to MPI barriers or other synchronization points in the program
- During second full run, started with all healthy nodes, but a few hours into the run, a node became slow

Workaround for Slow Nodes

Sandia National Laboratories

- At the beginning of each run, check for slow nodes using simple standalone MPI program: each MPI rank calculates PI constant using a series expansion
- Program touches a lot of memory, which is key for finding nodes with memory DIMM issues
- Finds hostname of slower-than-average nodes
- Slow nodes are excluded from srun command before job runs
- SPARTA also periodically checks for slow nodes in load balance routine, kills job if slow node is found, next job excludes slow node

5. DataWarp issues

Sandia National Laboratories

- Sometimes writes to DW fail, leading to data loss and checkpoint file corruption
  - Sometimes returns errno and sometimes silent
  - Sometimes massive data loss and sometimes very subtle
  - Sometimes hardware cause identified, sometimes no apparent cause
- During third full run, some DW nodes became read-only, leading to catastrophic checkpoint file corruption
- Trying to read in corrupted files causes segfault or SPARTA error. Very difficult to detect subtle corruption outside of SPARTA (file size and file count in first full run looked fine)
- Count bytes SPARTA “thought” it wrote out vs file size on DW; if these don’t match data loss occurred and files are corrupt

Other DataWarp issues

Sandia National Laboratories

- Slow stage-in and stage-out to/from Lustre when using thousands of files, stage-in can time out and job won't ever run
- Using #DW directives and staging out at end of job + two jobs failing in a short time = multiple stage-outs of the same data to the same Lustre directory

DW Painful Lessons Learned

Sandia National Laboratories

- Solid-state drives (SSDs) used for DW can and do fail. Don't just leave data on DW, stage out from persistent DW to Lustre at least every 24 hours!
- For high profile runs, stage out to Lustre every few hours and don't overwrite all the valid checkpoint files with corrupted ones! (otherwise can't continue simulation)
- Good to stage out to unique Lustre directories, at least on a per-job basis
- Sometimes DW goes into bad state and data loss occurs, which corrupts checkpoint files
- When DW fails, need to fall back to Lustre

Current Work

Sandia National Laboratories

- Originally used #DW directives in batch script, can only stage out to end of job
- Now using libdatawarp API instead of #DW directives to allow immediate stage out of checkpoint files from DW to Lustre (thanks to Giovanni Cone from LANL)
- Stage out to unique Lustre directories for each job, avoids total loss of checkpoint files
- Need to fall back to Lustre when data loss is detected with DW
- Waiting to test robust recovery: e.g. request a few extra nodes, put srun in a loop, and restart using the latest checkpoint file

Benefits of Resilience

Sandia National Laboratories

- Running production simulations on Trinity through queue: 5000 KNL node jobs with a walltime limit of 24 hours
- Submitted several jobs to the queue and then walked away (actually flew to Hawaii for vacation for a week and a half)
- 5 jobs ran while I was gone: 1 job ran for 24 hours and timed out, other 4 jobs died due to node hardware uncorrectable errors before 24 hours elapsed
- Despite job failures, still made significant forward progress
- Caveat: did not use DataWarp…still need to harden DW workflow against total checkpoint file corruption

Looking Forward

Sandia National Laboratories

- Slurm is adding advanced features for heterogeneous runs, see https://slurm.schedmd.com/heterogeneous_jobs.html
- Can stitch two separate srun commands together into a single MPI_COMM_WORLD (no special driver program needed)
- Can fine-tune srun commands for each partition, i.e. can use different # of MPI, core specialization, different affinity, etc.
- Don't need a special "any" partition, can use two separate accounts
- Easy to run heterogeneous production runs outside of DATs!
- Doesn't work with Cray MPI yet ☹, hopefully fixed in Slurm 18

Conclusions

Sandia National Laboratories

- Heterogeneous (Haswell + KNL) runs on 19,000+ nodes with up to 1.2 million MPI ranks are possible
- Full system heterogeneous runs stress out the system and are very challenging (have to deal with weaknesses of both partitions)
  - KNL = die due to uncorrectable hardware errors
  - Haswell = slow due to bursting memory DIMM
- Workflow hardening is very beneficial and necessary (wish it was less necessary)
- Running a short sanity check on each node and scanning for slow or bad nodes before production runs is highly recommended (recently found 3 slow Haswell nodes during early-user testing)

Sandia National Laboratories

Questions?

Discussion/Suggestions?

Thanks to Mike Davis (Cray), and Joel Stevenson, Ben Santos, and other Trinity HPC support personnel (LANL and SNL)