National Aeronautics and Space Administration

NASA Advanced Supercomputing (NAS) Division

Dr. Piyush Mehrotra

Division Chief

piyush.mehrotra@nasa.gov

NASA Ames Research Center, Moffett Field, Calif., USA

January 6, 2017

Advanced Computing @ NAS

Supercomputing @ NAS

NASA's Premier Supercomputer Center

Charter: to support all supercomputing requirements of NASA Mission Directorates

Over 500 science & engineering projects with more than 1,350 users

Pleiades: *7.25 PF peak – 11K+ multi-generational nodes; 245K+ cores; #17 on TOP500 (#7 in US): #11 on HPCG*

Supercomputing @ NAS

NASA's Premier Supercomputer Center

Charter: to support supercomputing requirements of all NASA Mission Directorates

Over 500 science & engineering projects with more than 1,350 users

Pleiades: *7.25 P*
generational nodes;
on TOP500 (#7 in U

Electra: *4.78 PF peak – 2304 Broadwell+Skylake nodes; container-based #33 on TOP500*

Supercomputing @ NAS

NASA's Premier Supercomputer Center

Charter: to support supercomputing requirements of all NASA Mission Directorates

Over 500 science & engineering projects with more than 1,350 users

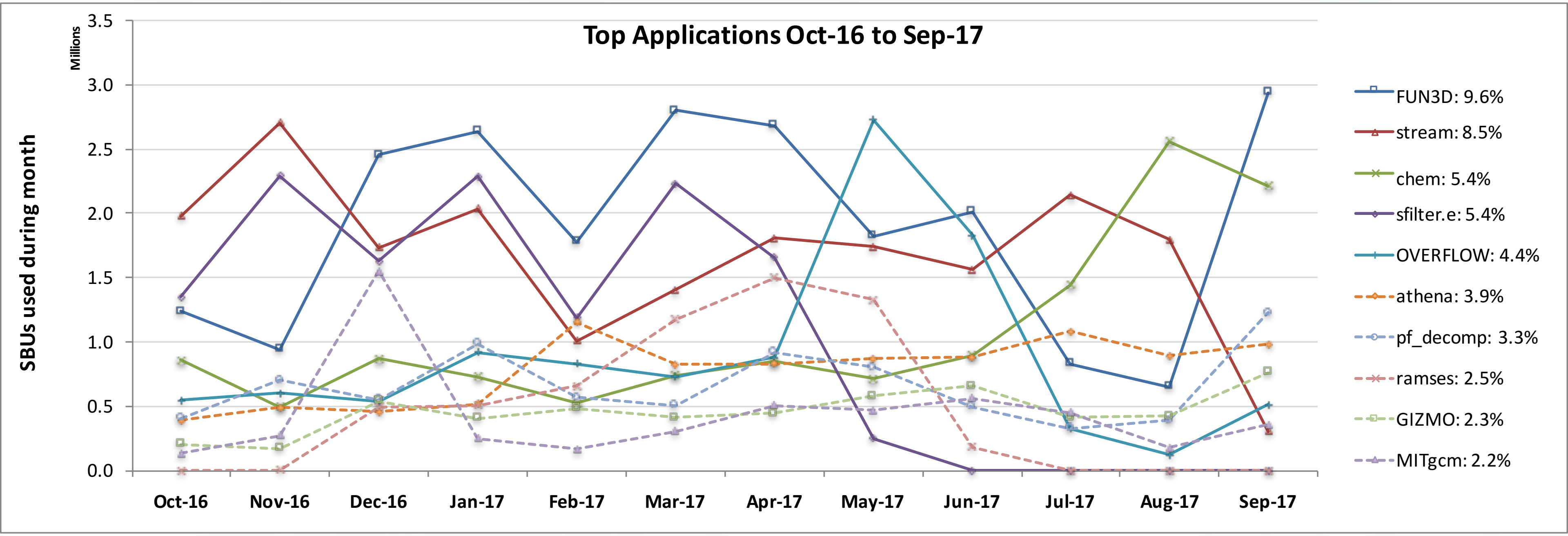
Pleiades: *7.25*
multi-generationa
TOP500 (#

Electra: *4.78 PF pea*
Broadwell+Skylak
container-based #33

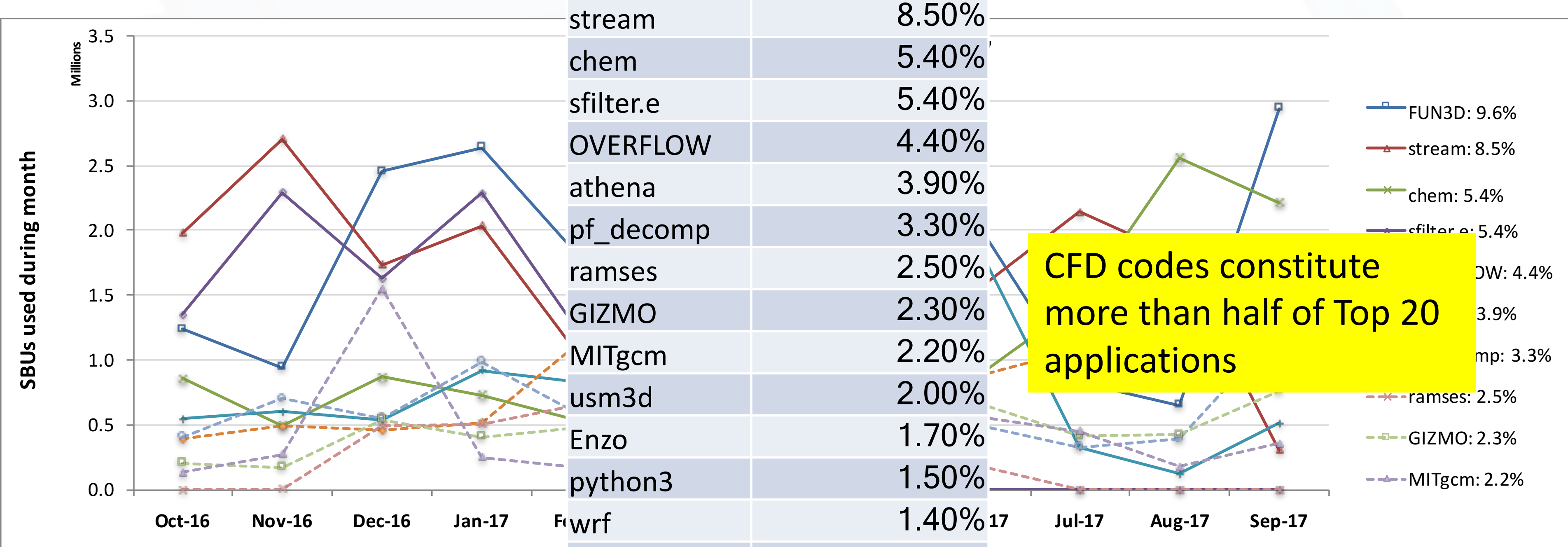
Modular Supercomputing Facility:
Artist's rendering of future facility

Global file system – Lustre and NSF-based > 40 PBs

Application Usage @ NAS FY17



Application Usage @ NAS FY17



Top 20 Applications FY17	
FUN3D	9.60%
stream	8.50%
chem	5.40%
sfilter.e	5.40%
OVERFLOW	4.40%
athena	3.90%
pf_decomp	3.30%
ramses	2.50%
GIZMO	2.30%
MITgcm	2.20%
usm3d	2.00%
Enzo	1.70%
python3	1.50%
wrf	1.40%
wrles	1.30%
a.out	1.30%
lava.mpi	1.20%
arts	1.20%
BATSRUS.exe	1.20%
vasp_std	1.20%
	61.50%

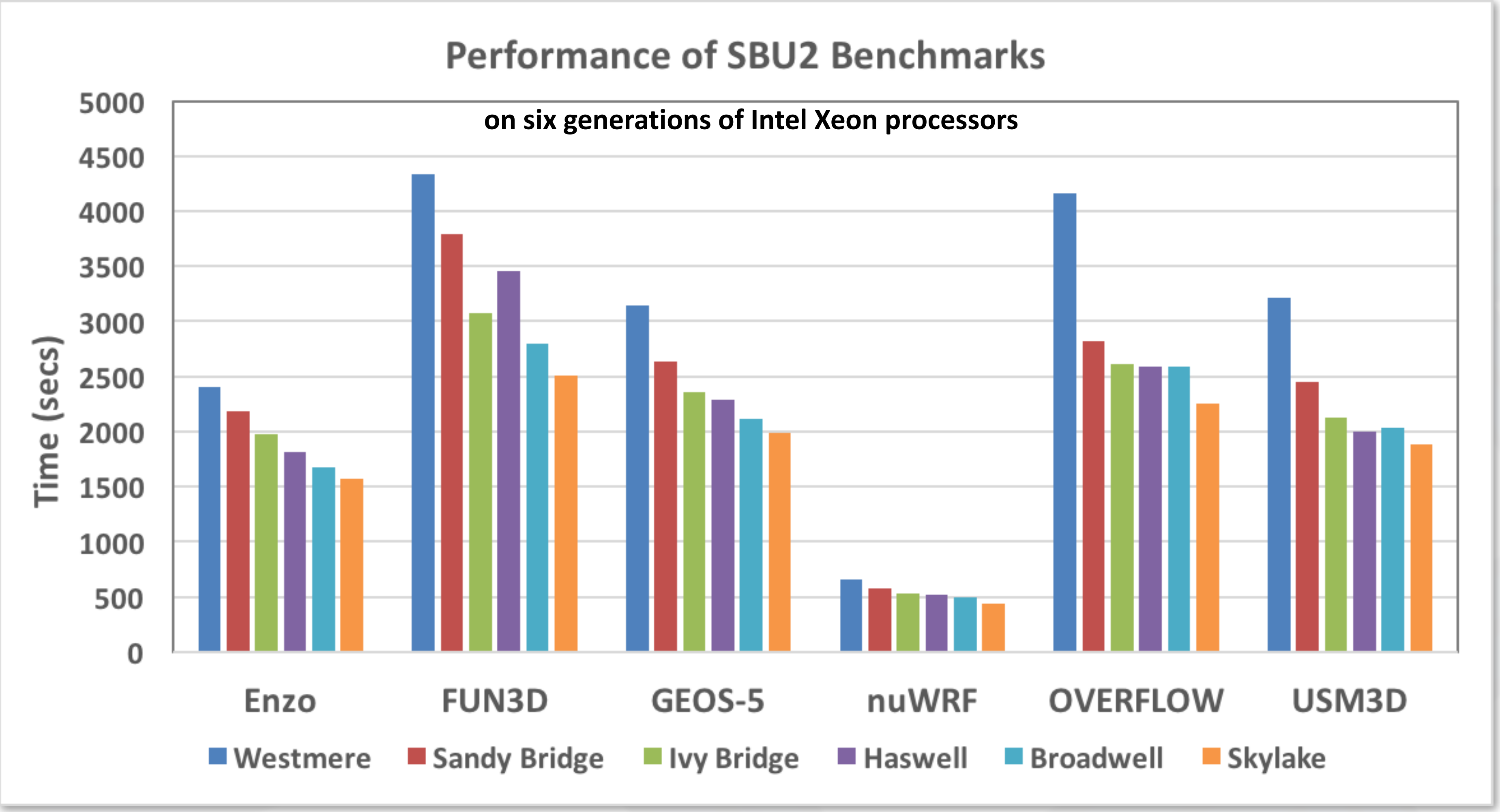
CFD codes constitute more than half of Top 20 applications

SBU Benchmarks

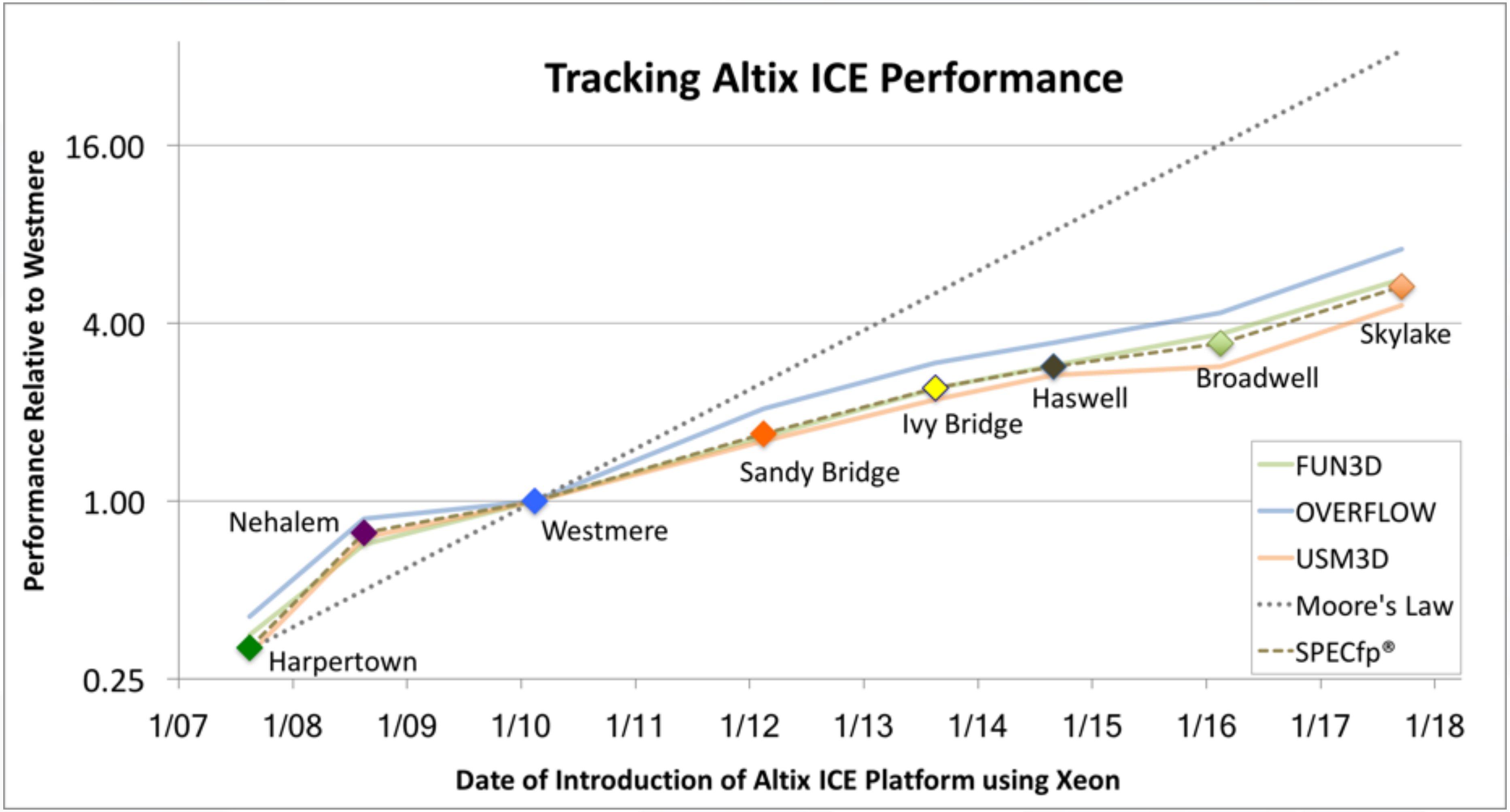
- ***Standard Billing Unit (SBU)*** is a measure of application cost running on minimum allocatable unit (MAU) of a system for a given node type
- Used for usage accounting and tracking across node types
- Also used for benchmarking and performance comparisons
- The first set of SBU benchmarks (SBU1) was released in 2011 with Intel Westmere as baseline
- SBU2 Benchmark Suite under development
  - Utilizes Intel Broadwell as baseline
  - Updated test cases with increased MPI rank counts
  - 30 mins execution on most recent node type in 2016 (Broadwell)
  - Adjusted weight factors for workloads in 2016

Application	Missions	Version	Testcase
FUN3D	ARMD/HEOMD	13.1	1.7B cells, 2016 MPI ranks
OVERFLOW	ARMD/HEOMD	2.2l	753M grid points, 2016 MPI ranks
USM3D	ARMD/HEOMD	2016	623M cells, 2016 MPI ranks
Enzo	ASTRO	2.5	cosmology sim, 196 MPI ranks
GEOS-5	SMD (Earth Sci)	5.16.5	GMAO global data, 1344 MPI ranks
nu-WRF	SMD (Earth Sci)	v8-3.71	MERRA-2, 1680 MPI ranks

SBU2 Benchmark Performance



Performance of CFD codes



Performance Study: Intel Xeon Phi

Goal: **Evaluate potential of new architectures for NASA applications**

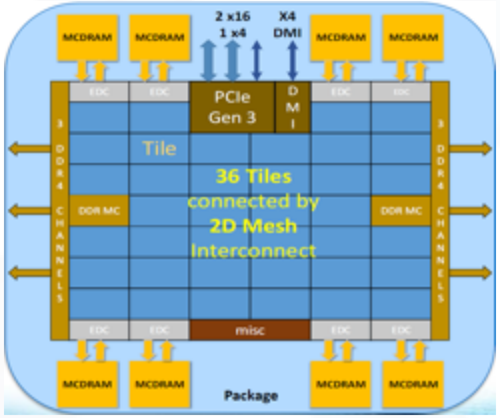
Approach: Use microbenchmarks, NAS parallel benchmarks, full-scale applications

Areas of Interests:

- Architecture
- Hierarchical memory
- Comparison with Xeon processors (Haswell, Broadwell)
- Application porting effort
- Compiler and tools
- Code optimization
- Data layouts and structures

Intel Xeon Phi (Knights Landing-KNL) Processor

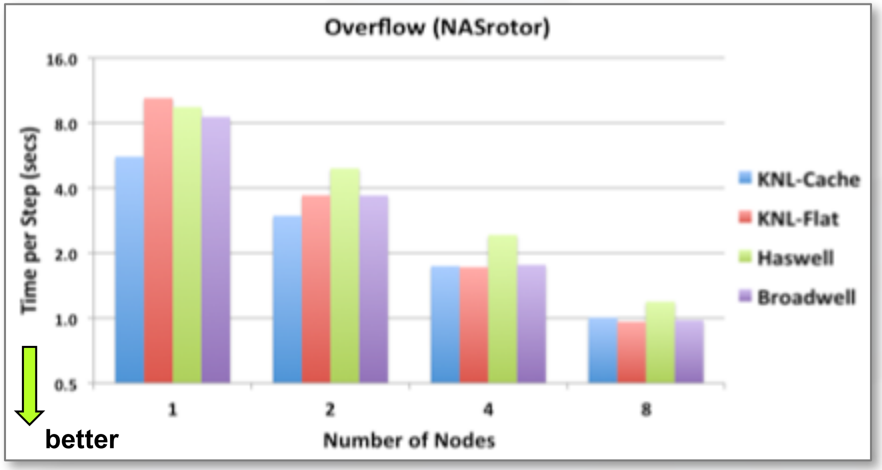
- Self-boot, Intel Many-Integerated Core (MIC) architecture
- Binary compatible with Xeon ISA
- 2 wide (512-bit) vector processing units
- Integrated on-chip high bandwidth memory (MCDRAM)
  - can be used in several modes: cache, flat memory, hybrid



Xeon Phi Performance

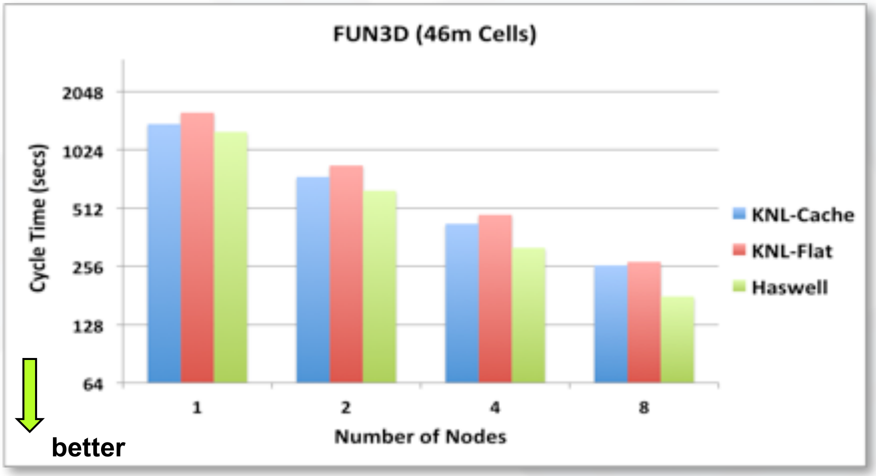
Overflow

- NASrotor: 91 M grid points, 45 GB memory
- KNL-cache mode 20-40% better on 1, 2 nodes as case doesn't fit in MCDRAM
- On 4, 8 nodes no difference between cache and flat modes on par with Broadwell



FUN3D

- 46M cell, 70 GB memory
- KNL-cache mode better upto 4 nodes as case doesn't fit in MCDRAM
- Haswell better as MPI impedes scaling on KNLs



- Easy initial porting of code – no changes required
- Optimization needed for memory hierarchy in cache mode / NUMA effects in flat-memory mode
- Codes that are vectorized and cache-optimized will perform better

Monitoring Power Usage of Applications

Goal

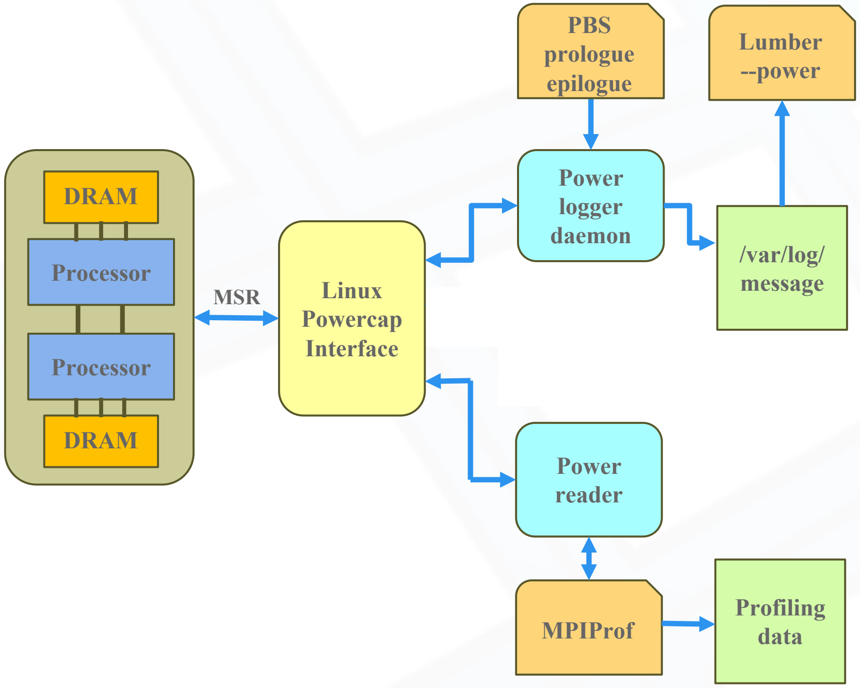
- Analyze correlation with application characteristics
- Understand and improve resource utilization of applications

Infrastructure built on Intel RAPL MSR

- Accessing via the Linux powercap interface
- Energy usage data for processors and DRAM

Approach

- Per-application monitoring
  - for focused analysis
- Per-job monitoring
  - for system-wide resource analysis



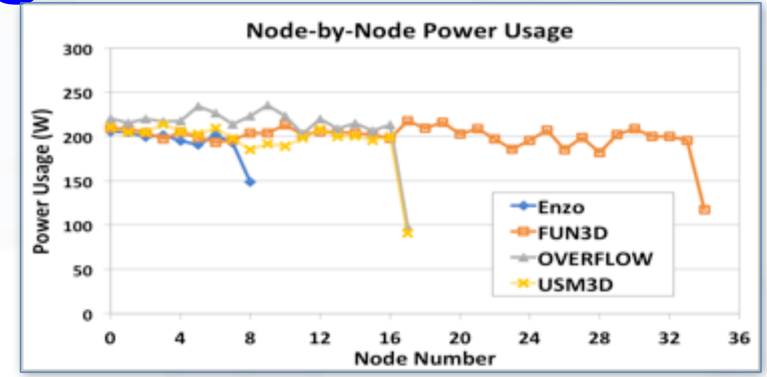
RAPL – Running Average Power Limit, MSR – Model Specific Registers

Lumber – a tool for real-time data-mining of system log-files for sophisticated job and system behavior analysis.

Power Usage Results

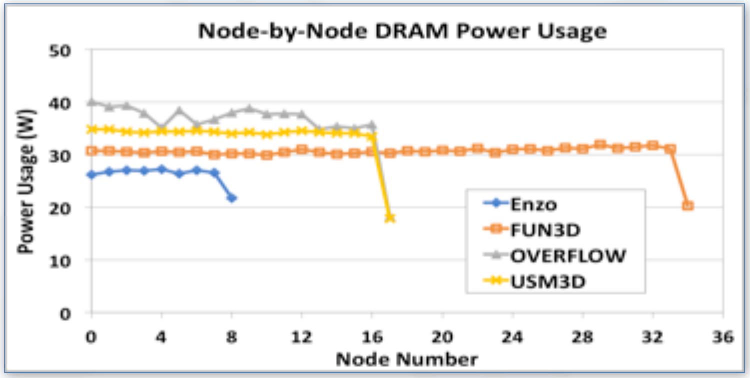
Processor power usage comparison:

- Similar across applications
- Drop at the last node related to less workload on the node



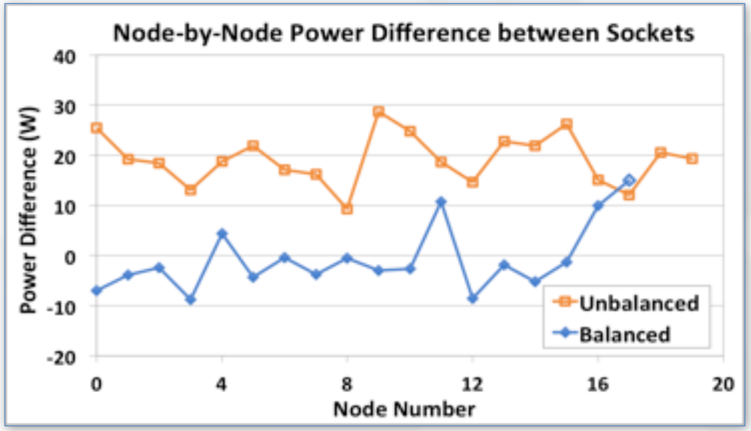
DRAM power usage comparison:

- Shows correlation with different applications
  - Most with OVERFLOW, least with Enzo

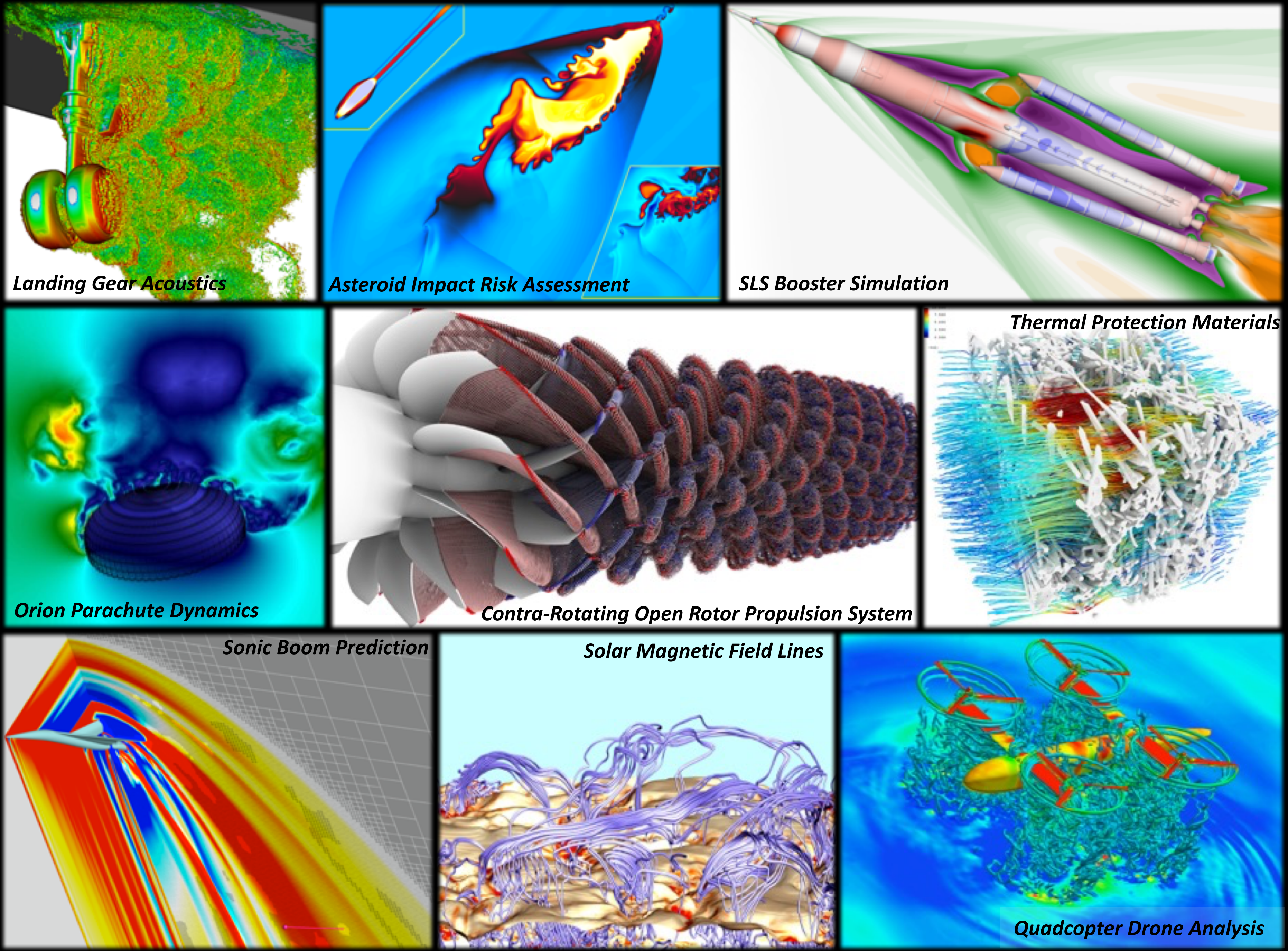


OVERFLOW runs (y-axis power diff between sockets):

- Unbalanced run: Cores fully populated on the first socket but not on the second socket showing upto 30% difference



Modeling & Simulation @ NAS


Landing Gear Acoustics
Asteroid Impact Risk Assessment
SLS Booster Simulation
Orion Parachute Dynamics
Contra-Rotating Open Rotor Propulsion System
Thermal Protection Materials
Sonic Boom Prediction
Solar Magnetic Field Lines
Quadcopter Drone Analysis

CFD Technologies @ NAS

- ***Cart3D***
  - ***Michael Aftosmis***, *Marian Nemec, David Rodriguez, George Anderson, Marsha Berger (NYU)*

- ***eddy***
  - ***Scott Murman***, *Laslo Diosady, Anirban Garai, Corentin Carton de Wiart, Patrick Blonigan, Dirk Ekelschot*

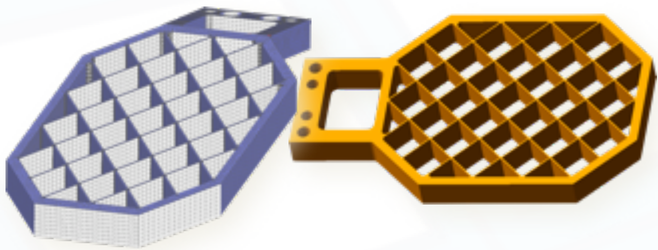
- ***LAVA*** (Launch, Ascent, and Vehicle Aerodynamics) Framework
  - ***Cetin Kiris***, *Jeff Housman, Mike Barad, Joseph Kocheemoolayil, Emre Sozer, Francois Cadieux, Gerrit Stich, Marie Dennison, James Jensen, Jared Duensing*

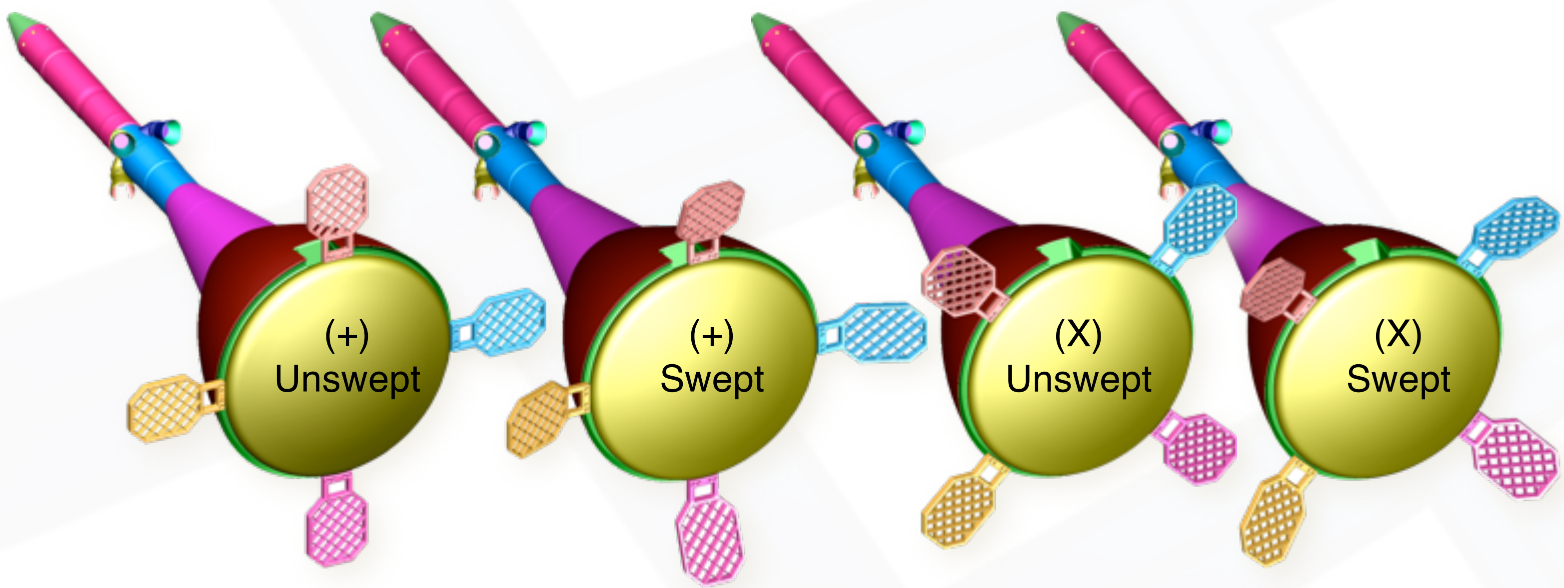
- Designed for analysis and design of complex aerospace vehicles.
  - Automated meshing – insensitive to geometric complexity
  - Inviscid analysis with automatic solution verification
  - Includes surface modeling, mesh generation, data extraction
  - Automatic & robust error control with quantitative error bounds
- Applications
  - Aerodynamic database generation - Including case management
  - Parametric and trajectory studies
  - Preliminary design - includes gradient-based design framework
- Most common use is populate aerodynamic performance databases for arbitrarily complex vehicles
  - Routinely run O(10^3-10^4) individual cases on complete configurations
  - All cases use adjoint-based mesh adaptation and include mesh convergence studies with error estimates for outputs of engineering interest
  - Widely used throughout NASA, DoD, and industry. NASA use includes HEOMD (Orion MCEV, SLS), ARMD (CST, LBFD, AATT), SMD (ATAP)
- HPC
  - Typical problem size of 10^7-10^8 cells on 1000 cores
  - *Near ideal scalability on distributed and shared memory systems (documented up to 8k cores)*

Cart3D: Typical Application

Aero-performance database of Grid-Fins equipped Launch Abort Vehicle

- Geometrically complex vehicle designs
- Database of ~10^7 cases examining performance similar to Orion-MCEV
- Wide range of flight conditions from low subsonic to supersonic

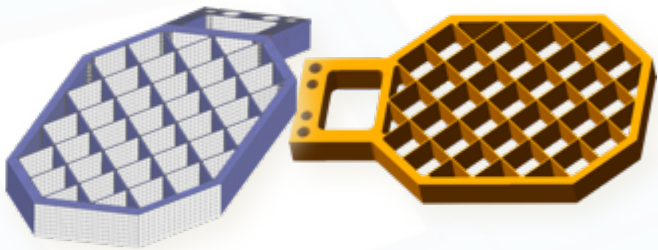


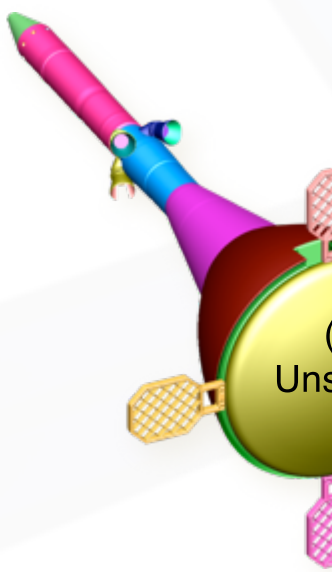


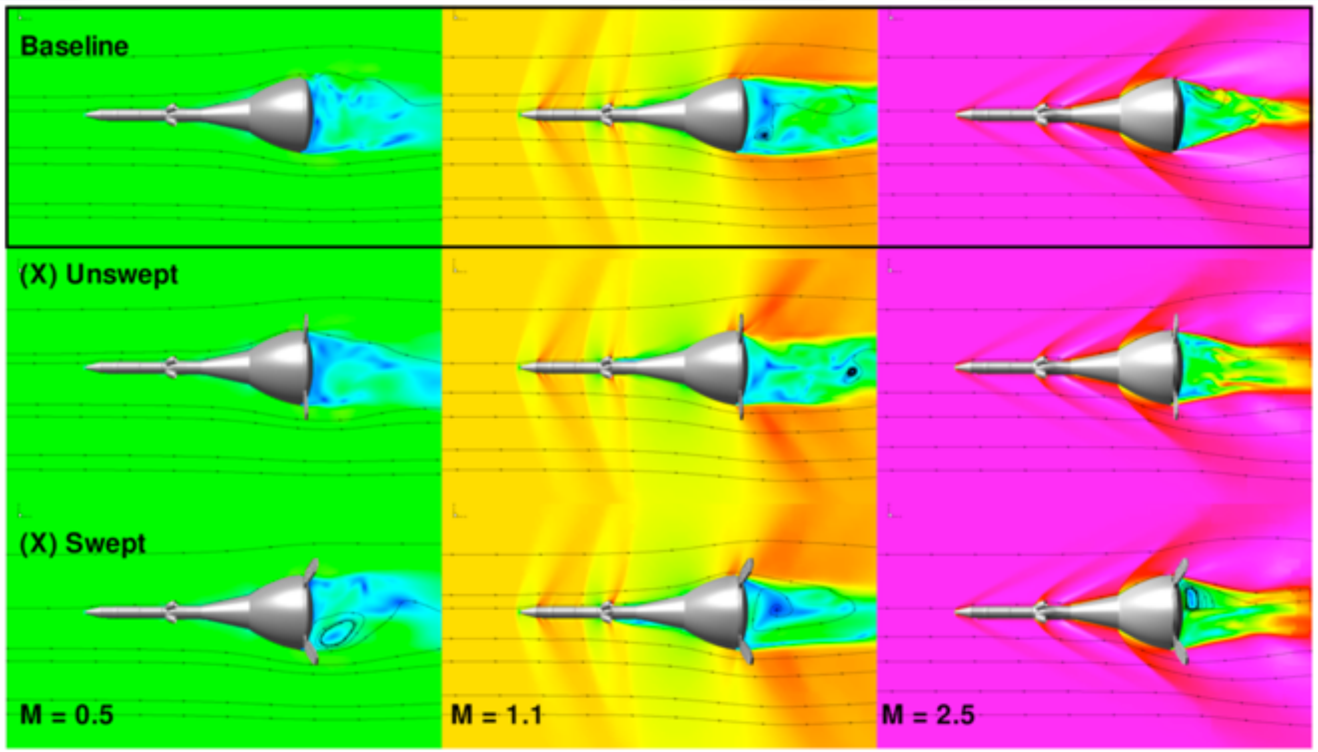
Cart3D: Typical Application

Aero-performance database of Grid-Fins equipped Launch Abort Vehicle

- Geometrically complex vehicle designs
- Database of ~10^7 cases examining performance similar to Orion-MCEV
- Wide range of flight conditions from low subsonic to supersonic





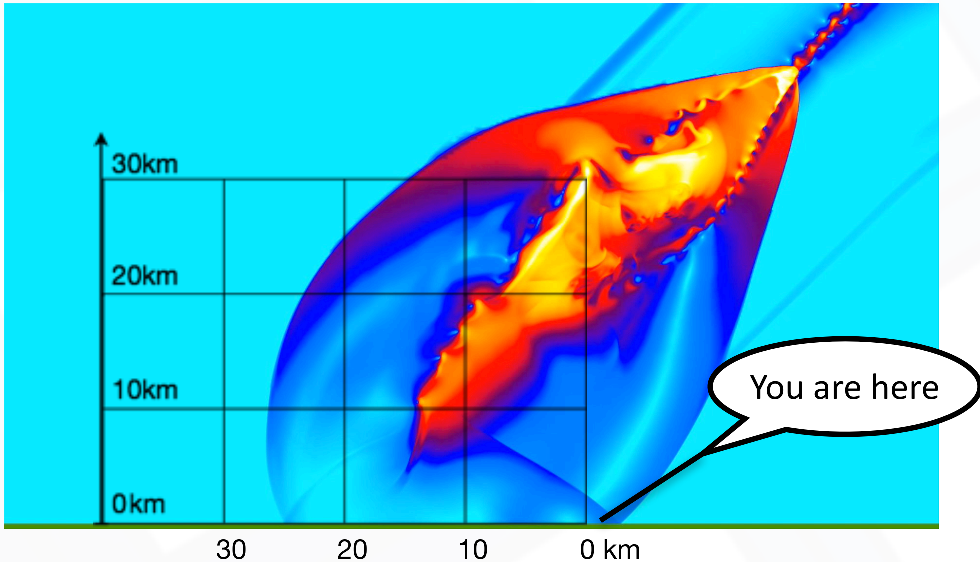


Cart3D: Recent Application

Evaluate threat due to asteroid entry into Earth's atmosphere

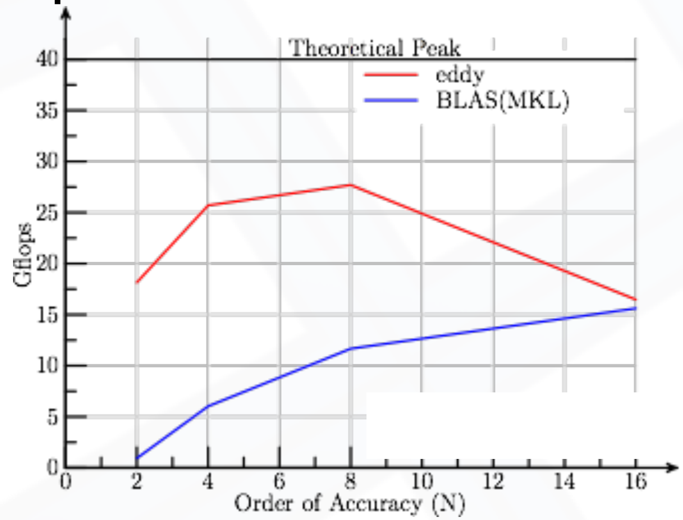
Calculate overpressure and wind speeds when asteroid hits the ground to evaluate damage

- Extreme range of velocity, length, and time scales
  - Velocity: Entry Mach = 40-70, into $M_\infty = 0$ atmosphere
  - Length: Domain extends hundreds of kilometers, but desire loads on human-scale structures
  - Time: Strong shock propagation requires small time steps, but must propagate hundreds of kilometers ; Shock requires over 5mins to travel 100km, but entry requires time steps $\Delta t = \mathcal{O}(10^{-3}) \rightarrow \mathcal{O}(10^5\text{-}10^6)$
- Typical cases have 200-300 M cells
- Usual run is on 4-8k cores (NAS Pleiades system)
- Planned improvements:
  - Add terrain and structures
  - Mesh adaptation
- Similar to a broad spectrum of unsteady problems – this problem can be run parallel in space but is sequential in time as opposed to aero-database applications which are "embarrassingly parallel"
  - Requires extreme parallelization of all stages to gain overall efficiency



Asteroid Entry – 10 Megaton airburst, Diam = 54m, 20km/sec, 45° entry angle eddy

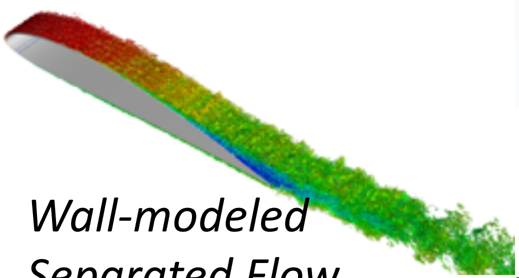
- Develop next-gen tools for scale-resolving simulations with a focus on exascale computing
- Develop new technology, not re-use existing algorithms, models, etc.
  - Entropy-stable high-order solver, dynamic variational multiscale method, metric-based adaptation, chaotic adjoint shadowing, …
- Use exascale computing to open new possibilities for
  - Multi-physics, robust error estimates, …
  - Certification by simulation
- Optimized for next-gen exascale hardware
  - 75% of machine peak in core tensor-product factorization routines

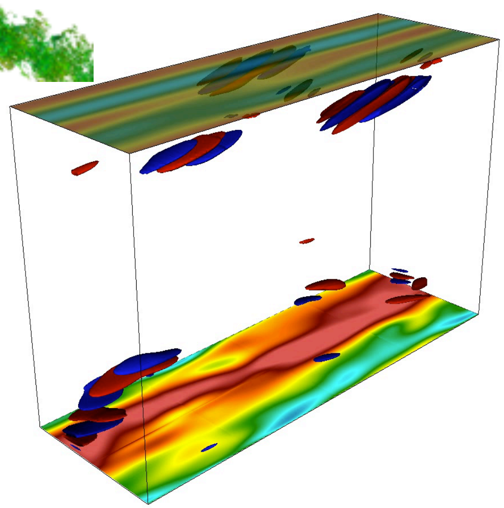


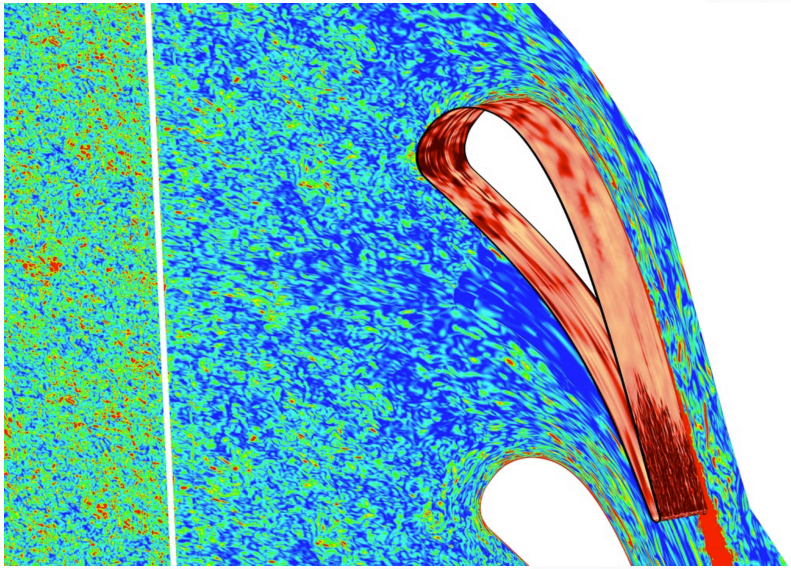
Intel Broadwell

- Recent work extending to novel monolithic multi-physics solver
  - Aeroheating, jet interactions, chemistry, …
  - Rotating turbomachinery, combustion, …
- Four presentations at SciTech 2018


Orion Parachute


Wall-modeled Separated Flow

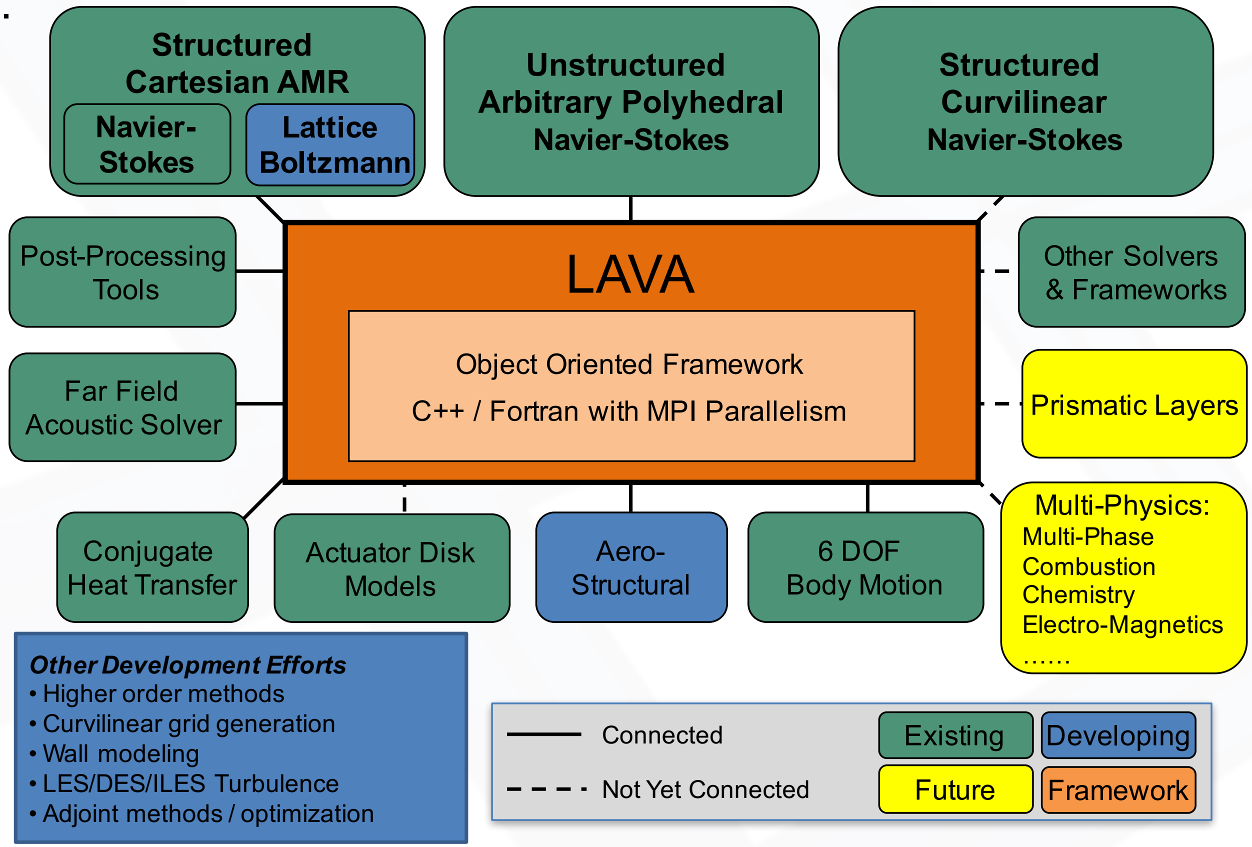

Adjoint of TKE Channel Flow


HPT Bypass Transition

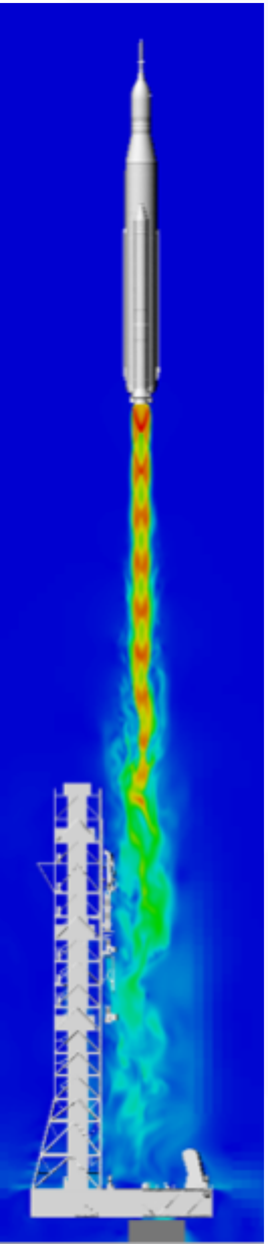
LAVA Framework

A flexible, modular framework supporting multiple computational grid paradigms

- Provides development opportunity for unsteady separated flows as well as aeroacoustics applications.
- Explores revolutionary approaches to reduce computational time to reach converged statistics.

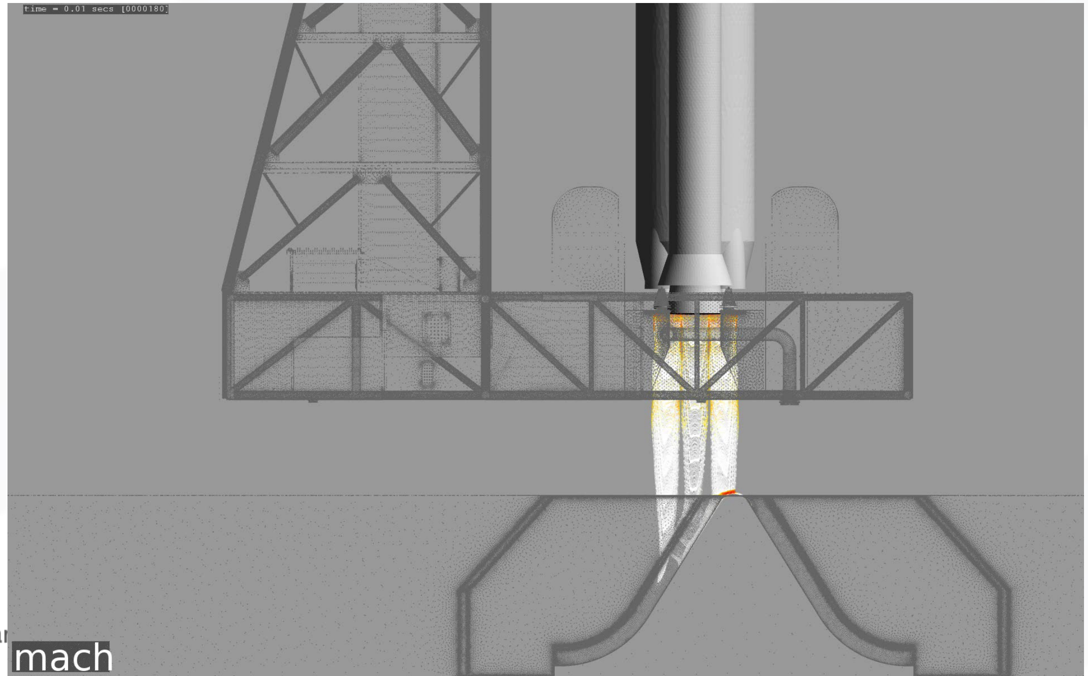


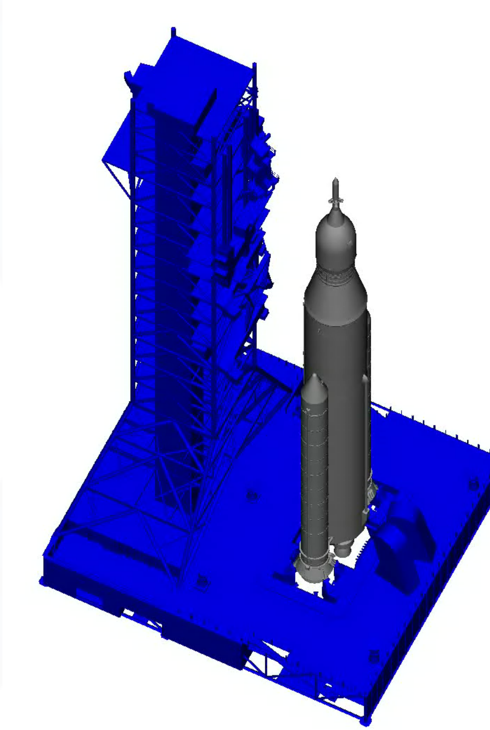
LAVA: Launch Environment



Predictive analysis of launch environment (trench and mobile platform)

- Pressure and thermal analysis of plume impingement on main flame deflector
- Containment analysis of plume in flame trench
- Numerous vehicles were analyzed on the pad, including SLS and commercial vehicles
- Drift analysis with plume impingement:
  - unsteady CFD with fixed vehicle
  - time-averaged SLS plume swept past pad and tower following 4000 trajectories

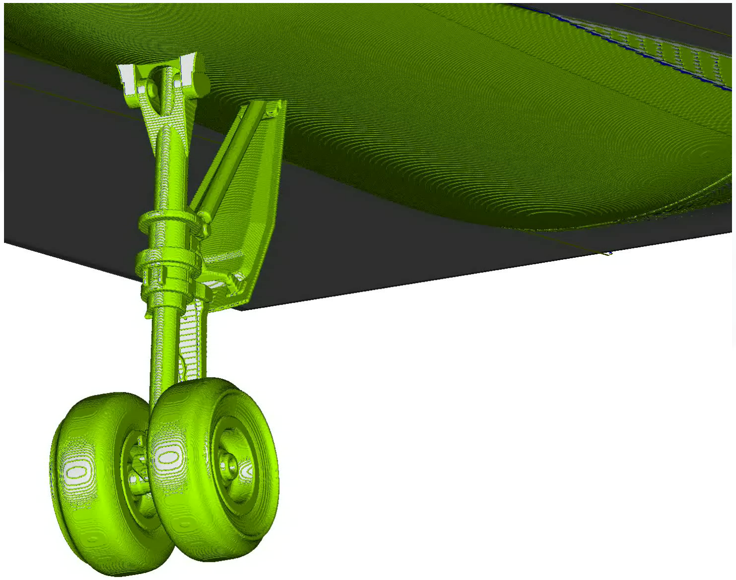




Challenges in Computational Aero-Acoustics

Computational Requirements

- Resources used for Cartesian Navier-Stokes examples:
  - Launch Environment: ~200 million cells, ~7 days of wall time (1000 cores)
  - Parachute: 200 million cells, 3 days of wall time (2000 cores)
  - Contra-Rotating Open Rotor: 360 million cells, 14 days (1400 cores)
  - Launch Abort System: 400 million cells, 28 days of wall time (2000 cores)
  - Landing Gear: 298 million cells, 20 days of wall time (3000 cores)
- Space-time resolution requirements for acoustics problems are more demanding.
- LAVA Cartesian infrastructure re-factored to add Lattice Boltzman Method (LBM)
  - Utilized existing LAVA Cartesian data structures and algorithms



Lattice Boltzman Landing gear: vorticity colored by Mach number

LAVA Performance

Method	CPU Cores (node type)	Cells (million)	Wall Days to 0.19 sec	Core Days to 0.19 sec	Relative SBU Expense
NS-GCM	3000 (ivy)	298	20.5	61352	12.1
NS-IIM	9600 (has)	222	6.1	58490	15.3
LBM	1400 (bro)	260	2.25	3156	1

- For a comparable mesh size, LBM is 12-15 times faster (in CPU utilization) than Navier-Stokes with immersed boundaries, and is equally accurate.

- Performance details:
  - Both Cartesian Navier-Stokes and LBM are memory-bound (not compute-bound) algorithms, the latter much more so than the former.
  - Non-linear, LBM collision operation (bulk of the computation) is entirely local. This data locality is critical to the computational efficiency of LBM relative to high-order Cartesian NS codes.

HPC Challenges

- Intra-node performance
  - Increasing number of cores
  - Cache/Memory hierarchies and bandwidth
  - Vectorization
  - Hybrid architectures
  - Code optimization and “smarter” algorithms
- Inter-node performance
  - Load balance
  - Communication optimization
  - Latency hiding
- Fault tolerance/resiliency particularly at scale
- I/O
  - I/O optimization
  - Infrastructure to support a wide variety of usage patterns
- Data analysis and visualization of extremely large dataset

Acknowledgements

Performance Characterization:

Henry Jin, Bob Hood, Application Performance Group

Modeling & Simulation:

Mike Aftosmis, Cetin Kiris, Scott Murman, Seokkwan Yoon

Visualizations:

Data Analysis and Visualization Group

Thanks!

piyush.mehrotra@nasa.gov

www.nas.nasa.gov