IN-SITU VISUALIZATION FOR GLOBAL HYBRID SIMULATIONS

Homa Karimabadi, Patrick O'leary, Mahidhar Tatineni
Burlen Loring, Amit Majumdar and Berk Geveci

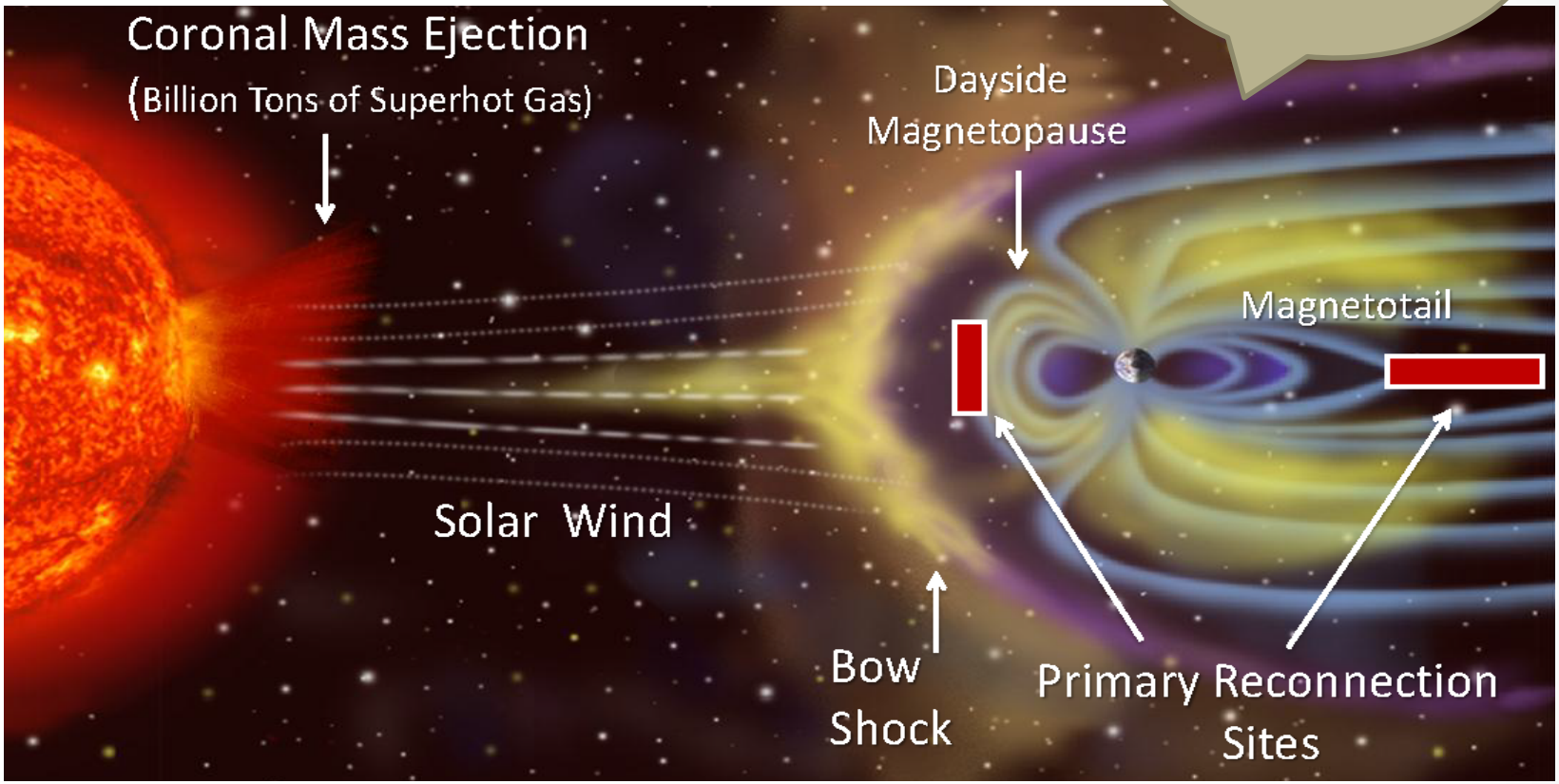
Research Areas

- Plasma Turbulence
- Magnetic Reconnection
- Dynamo
- Exploratory Fusion Concepts
- Space Weather

Computer Performance	
Name	**FLOPS**
yottaFLOPS	10^{24}
zettaFLOPS	10^{21}
exaFLOPS	10^{18}
petaFLOPS	10^{15}
teraFLOPS	10^{12}

Space Weather

Earth's magnetic field provides a protective cocoon but it breaks during strong solar storms



90 million miles or ~ 100 Suns

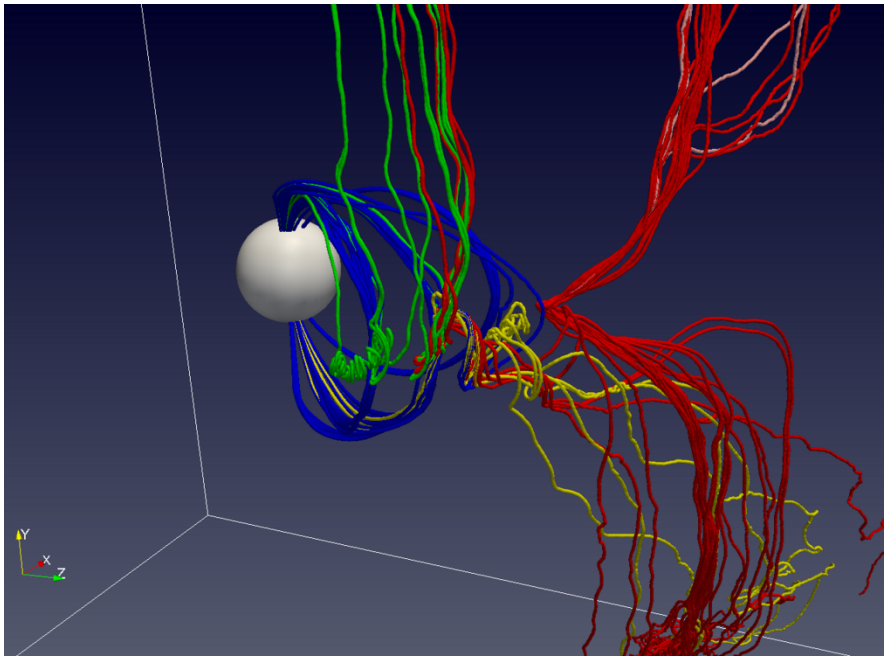
Motivation for Predicting and Preparing for the Impacts of a Strong Solar Storm

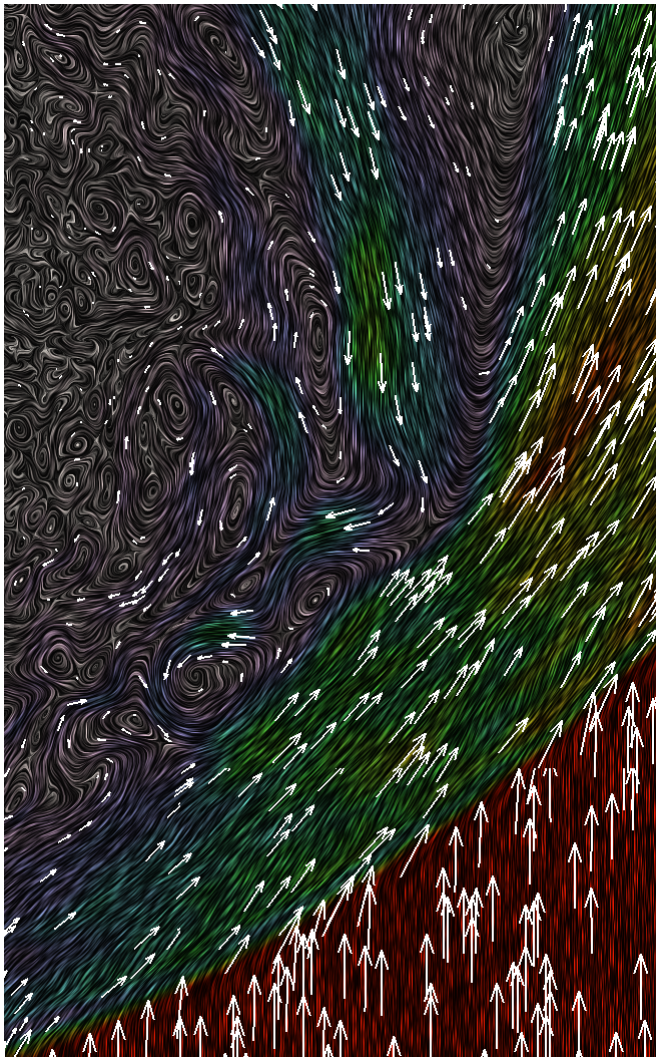
- **Communication**
  Loss of many satellites - no GPS, no cell phones, …
  Has caused over $4 billion in satellite losses
  Crippled communications worldwide

- **Power**
  Catastrophic collapse of power grid over many states that would take months to bring back on line.
  On March 13, 1989, caused a catastrophic collapse of the entire power grid Quebec & widespread blackout along east coast of U.S.

- **Financial**
  A solar storm of the magnitude of the 1859 Solar Superstorm would cause over $2 trillion in damage today.

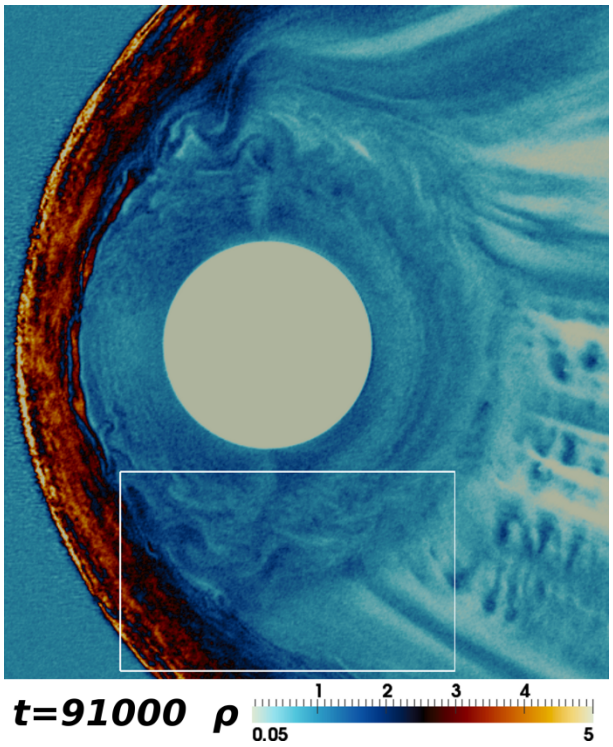
What are the source of vortices inside the Magnetosphere?

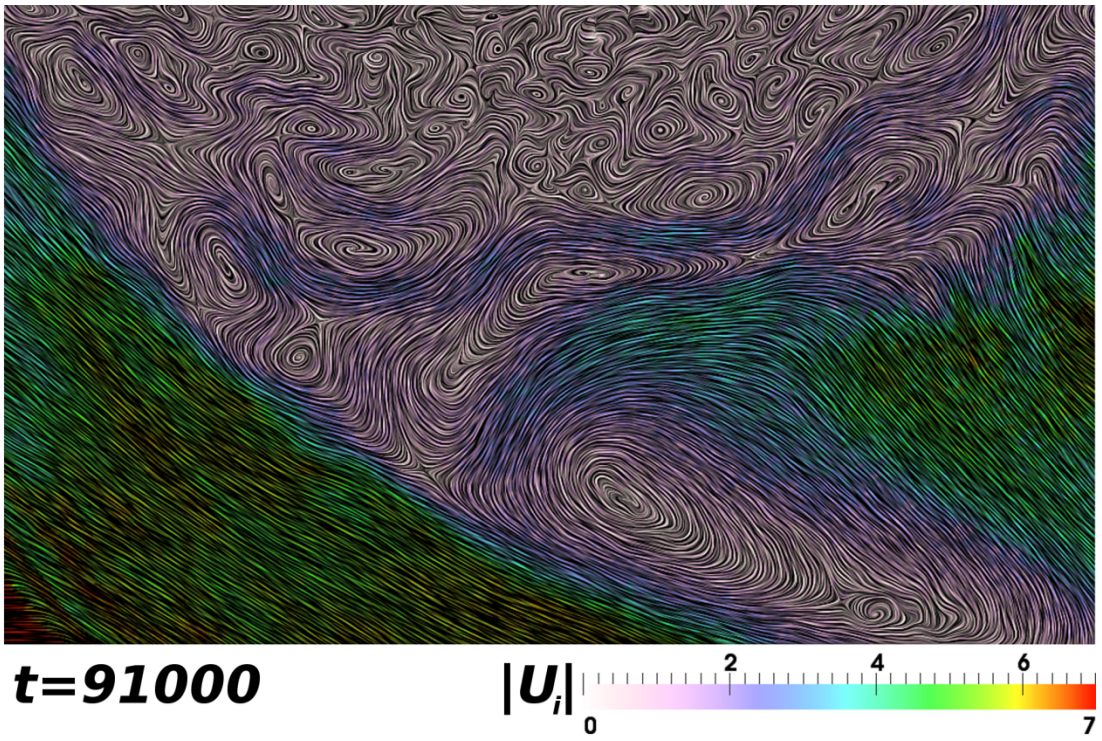




- Confirmed by spacecraft observations
- Vortices at the MP boundary often have FTE's near the vortex core. Are the FTE's causing the vortex? Or…
- Are these driven by Kelvin-Helmholtz instability?

Why In-situ for this study?

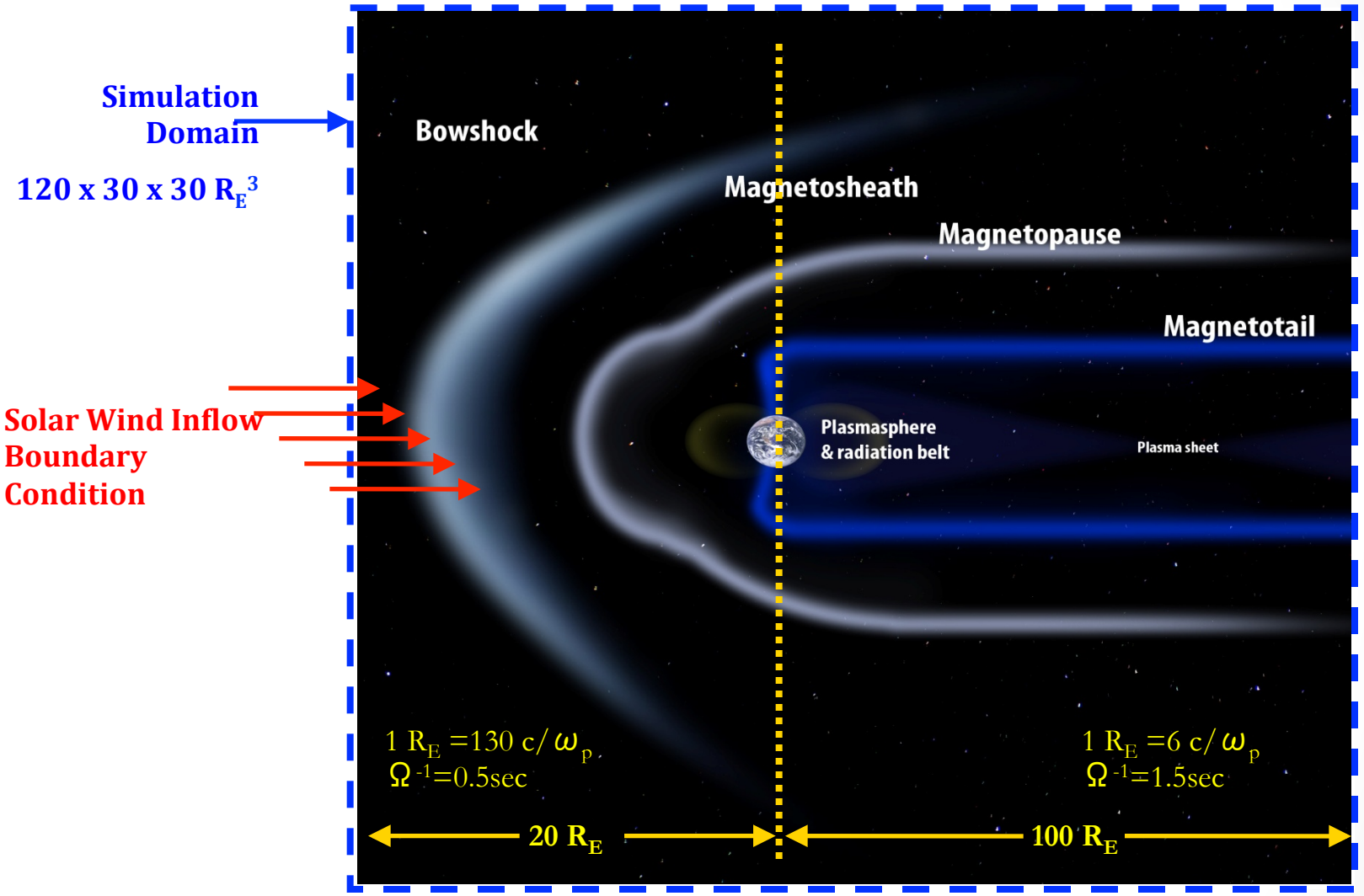




- Simulation is I/O bound and runs are limited by disk quotas
- With in-situ we reduce I/O costs by using “extracts”
- And increase temporal resolution
- Removes need for post processing – saves disk and computational time.

Challenge: Extreme Multi-scale Nature of the Physics

Numerology & the Computational Challenge



Challenges in Global Simulations

- Multi-Scale Coupled System
  - Spatial scales vary from ~km to 1.2×10^6 km
  - Temporal scales vary from minutes to days
  - Kinetic effects have global consequences
- Multi-Physics
  - Electron Physics – plays a critical role in reconnection
  - Ion Physics - dominates formation of boundaries and transport
  - Global features and dynamics - magnetotail/energetic particles
  - Coupling to the ionosphere

Three Approaches to Global Simulations

- MHD
  - Used extensively
  - Does not resolve important ion physics
  - Not suitable for studies of boundaries & discontinuities
- Hybrid
  - Fluid electrons, kinetic ions
  - The next stage of advance in global simulations
  - Resolves ion spatial scales (ion inertial length) and ion temporal scales (gyroperiod)
  - Requires use of resistivity model for modeling reconnection
- Full Particle
  - The only approach where physics of reconnection is fully captured
  - But has a very high computational cost

Our Approach

Hybrid Code

- Nonuniform Mesh + Multi-time zones

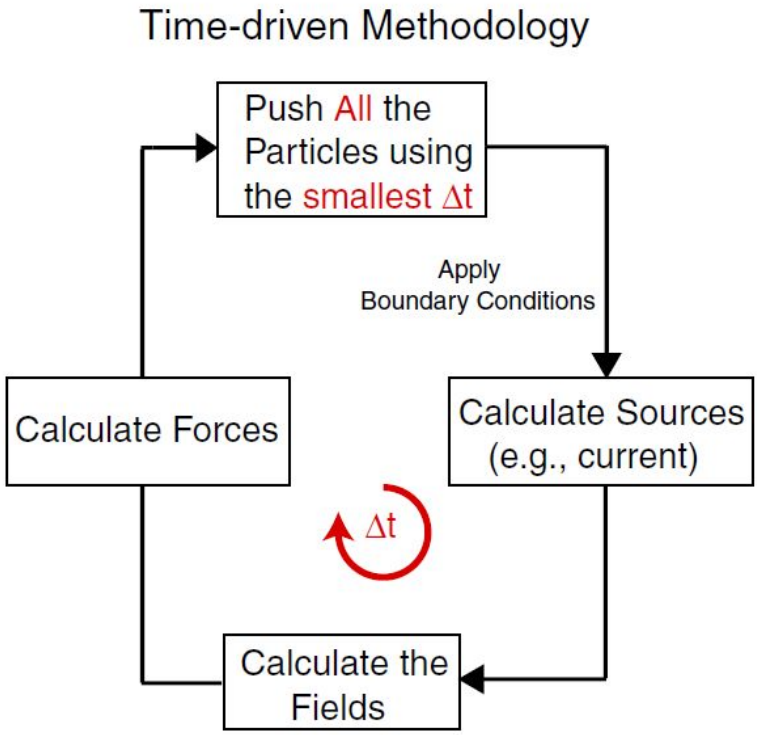
- Discrete Event Technique (Omelchenko's talk 10:55)
  - robust (stable) and efficient (no idle computation)
  - works for arbitrary meshes
  - Predict local Δt for each variable f ("state") based on its estimated trajectory, $f=f_E(t)$ and a given Δf_E (selected based on local CFL/ reaction conditions)

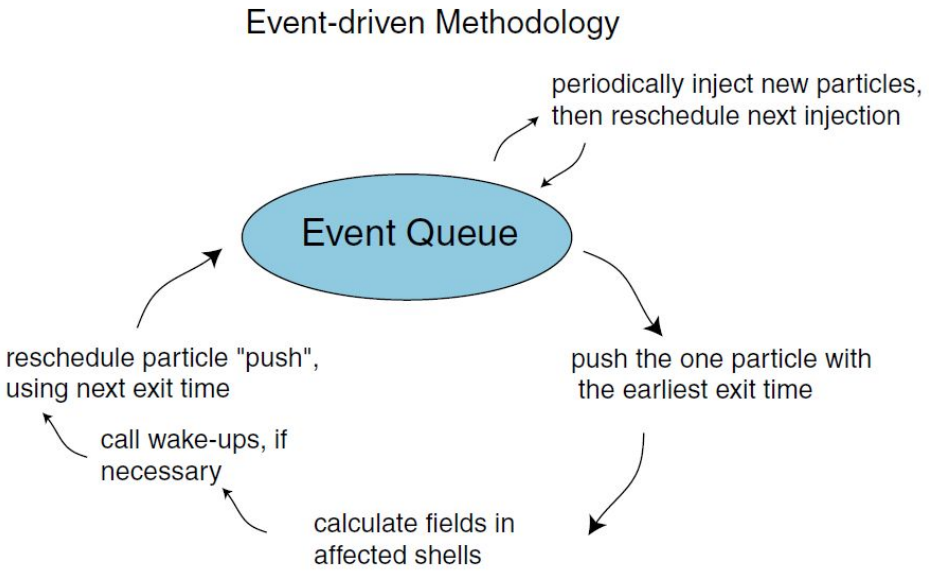
Fully Kinetic Code

- 2D global for capturing the microphysics of reconnection in a global setting

Particle-In-Cell Plasma Codes

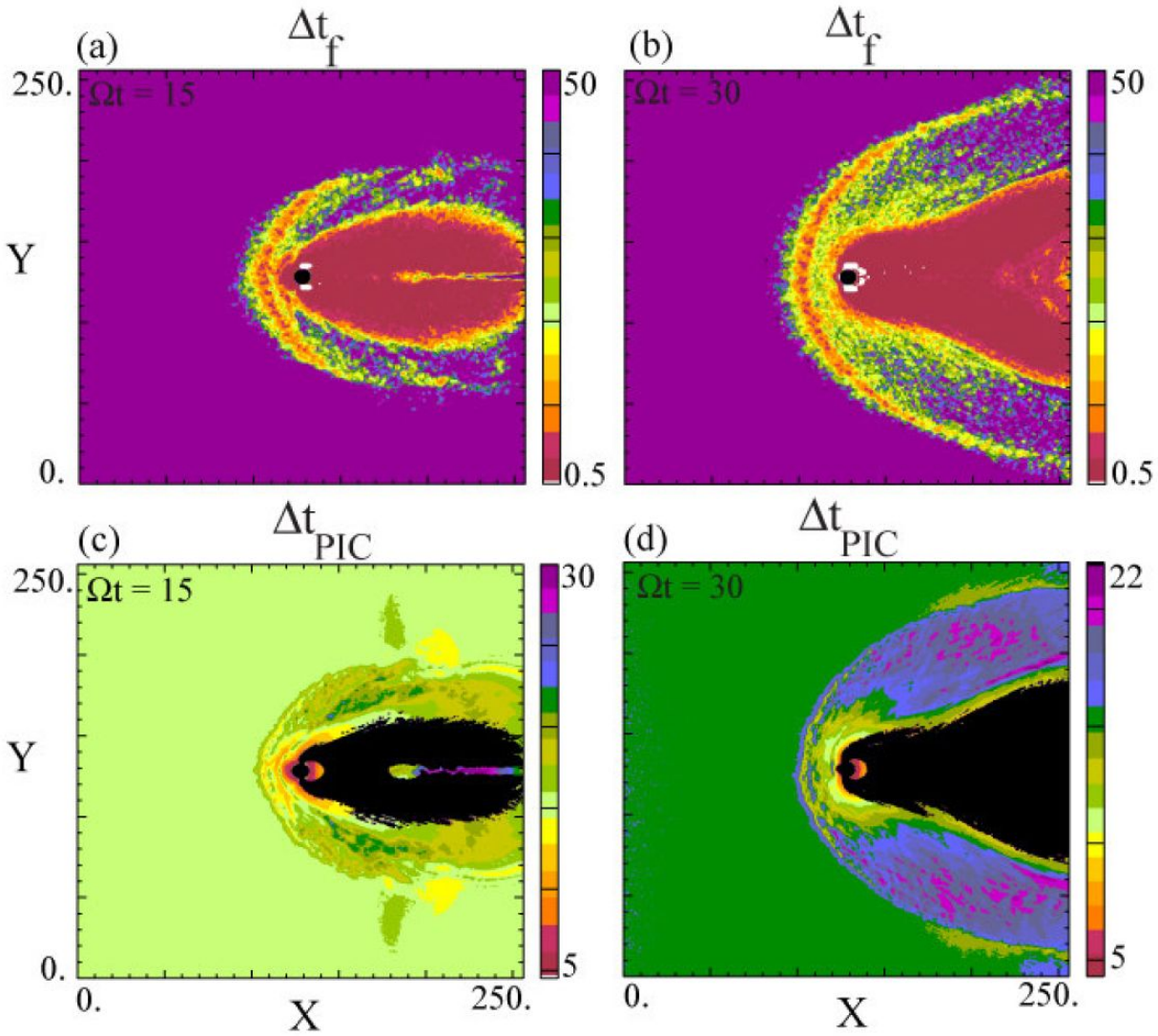
- Fully kinetic (electrons and ions are treated as particles)
- Hybrid (electron fluid, particle ions)





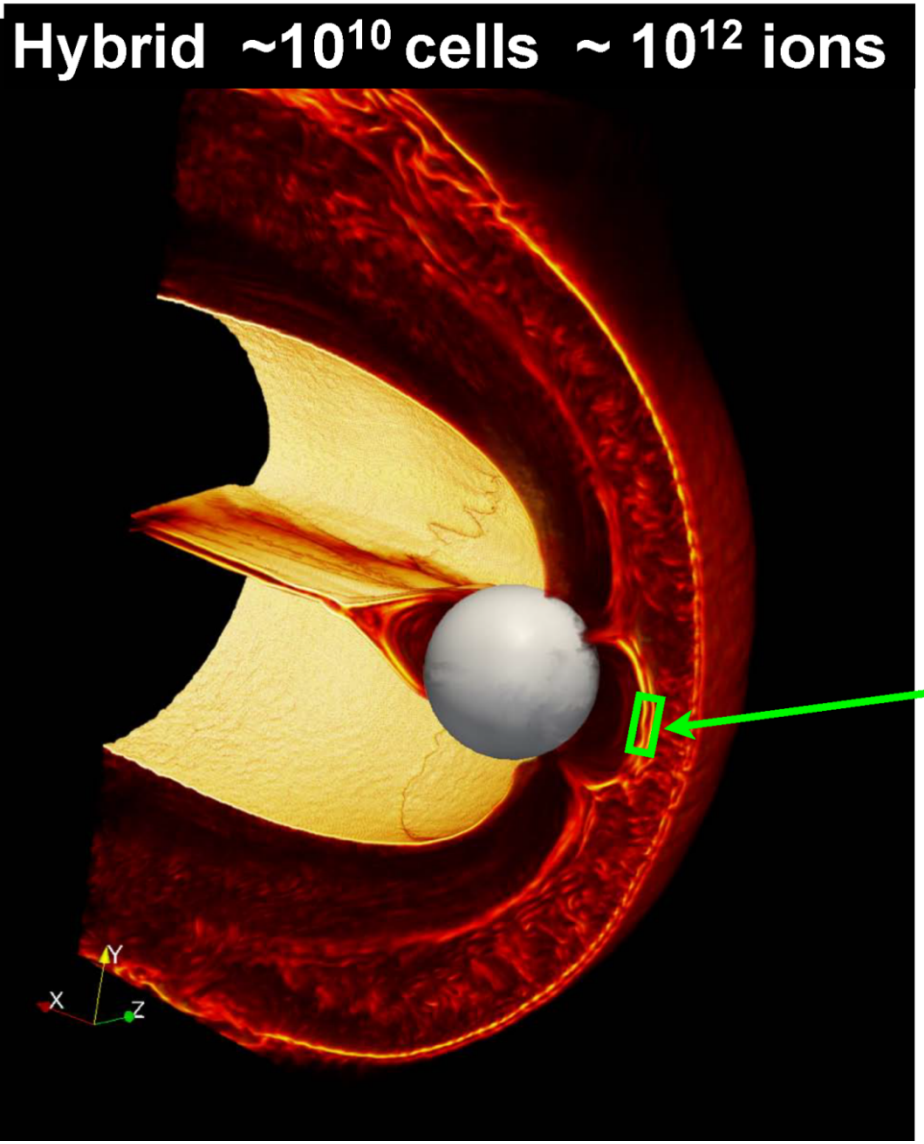
Karimabadi et al., JCP, 2005

Event-Driven Simulations



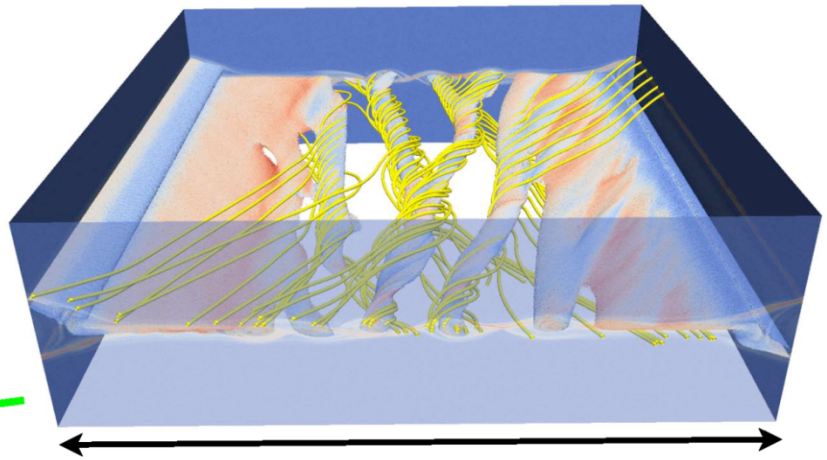
Omelchenko and Karimabadi, JCP, 2011

What simulations are feasible at the petascale?



Fully Kinetic

$\sim 10^{10}$cells $\sim 10^{12}$ particles



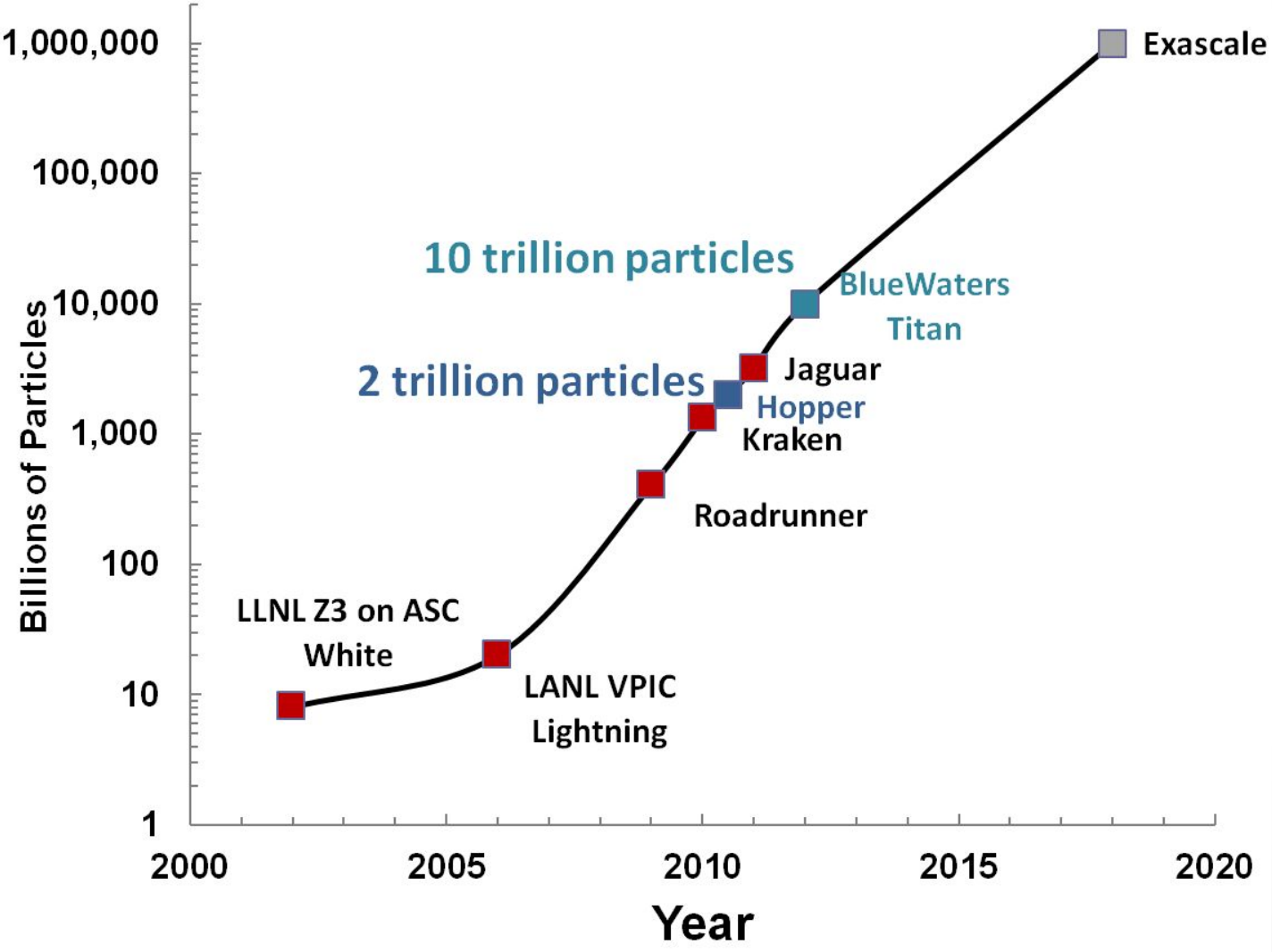
$\sim 100 d_i \sim R_E$

$3D \rightarrow m_i/m_e = 100 - 400$

$2D \rightarrow m_i/m_e = 400 - 1836$

Progress in Particle Simulations

(measured in terms of number of particles)



Simulation Characteristics – Single Run

- 8 variables/particle =32 bytes
- $N_p = 10^{12}$ particles on 300 K cores
- N_b = Buffer factor ~ 2
- $N_c = N_{cores}/300{,}000$
- Memory = $32*N_p * N_b * N_c /10^{15}$ ~ **$0.65\ N_c$ PB**
- N_{check} = # of checkpointing files = 2
- Disk ~ **$1.5*N_c$ PB**

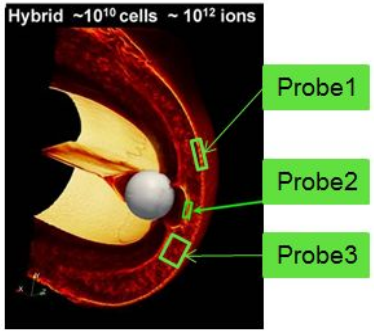
Run time required on full scale machine ~ 72 hours

CPU Hours ~ $72*N_{cores}$~**$21.6*N_c$ Million Hours**

In Situ Visualization

- For many of our analysis, we need high time resolution data dumps. However, this is not practical since each data dump can take over 10 TB.
- In situ visualization provides a possible solution.

In Situ Visualization Using Intelligent Probes



Features of Intelligent Probes

Targeted data collection
Ability to mimic spacecraft-like trajectories
Creation of instrument-like products
Generation of high resolution data products

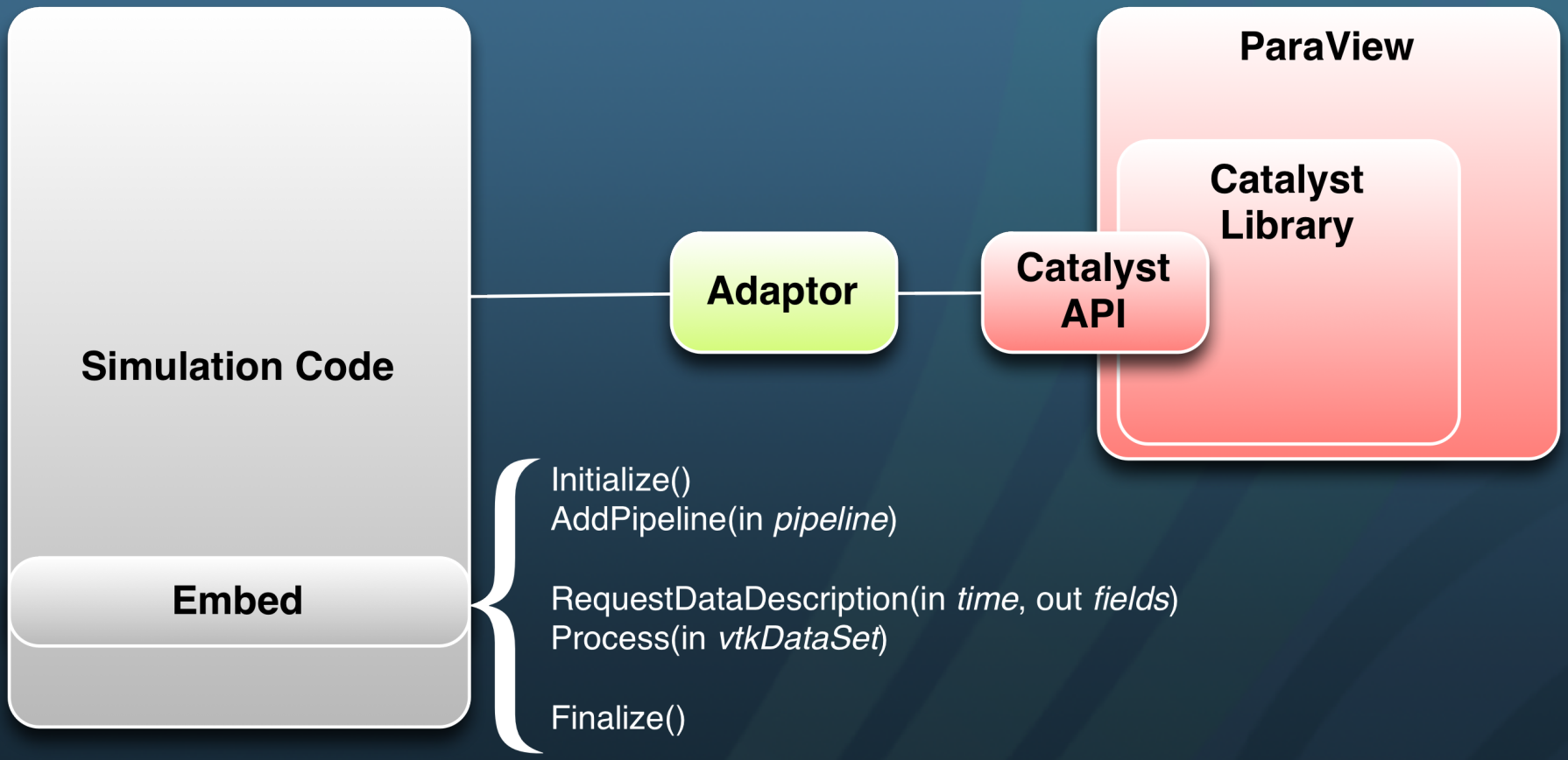
Selective Outputs

Images and visualizations
Data from probes (less raw data)
Statistics

Enables

Direct comparison with spacecraft data
Orbit optimization
Instrument design and testing

ParaView Catalyst Architecture



Augment Script
in
Input Deck

Simulation Code

Embed

Adaptor

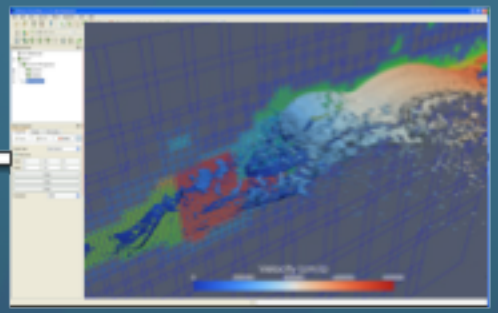
Catalyst API

ParaView

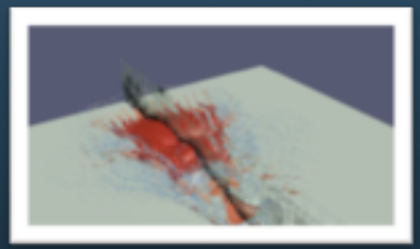
Catalyst Library

```
# Create the reader and set the filename.
reader = servermanager.sources.Reader(FileName=guts)
view = servermanager.CreateRenderView()
repr = servermanager.CreateRepresentation(reader, view)
reader.UpdatePipeline()
datainfo = reader.GetDataInformation()
pDinfo = datainfo.GetPointDataInformation()
arrayInfo = pDinfo.GetArrayInformation("displacement0")
if arrayInfo:
  # get the range for the magnitude of displacement0
  range = arrayInfo.GetComponentRange(-1)
  lut = servermanager.rendering.PVLookupTable()
  lut.RGBPoints = [range[0], 0.0, 0.0, 1.0,
                   range[1], 1.0, 0.0, 0.0]
  lut.VectorMode = "Magnitude"
  repr.LookupTable = lut
  repr.ColorArrayName = "displacement0"
  repr.ColorAttributeType = "POINT_DATA"
```

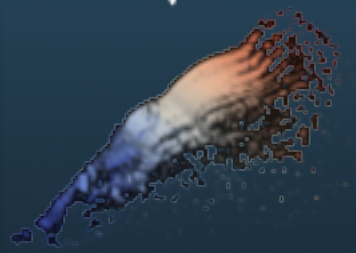
or
Create a Script



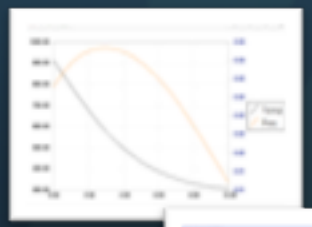
Export a Script



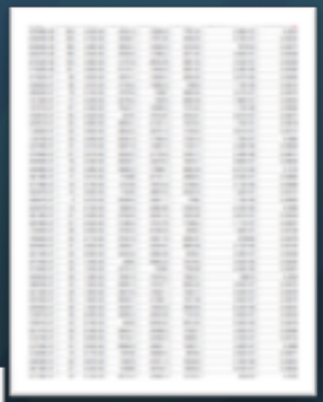
Rendered Images



Polygonal Output
with
Field Data



Series Data



Statistics

Kitware

Results – Memory Overhead of In-Situ Viz

- The first aspect studied was the memory overhead of the in-situ visualization on the simulations. Table 1 shows results for varying particle counts.
- The results show that the memory overhead is fairly constant at around 360MB/core.

Table 1. : Memory overhead comparison with fixed grid size (128x128x128) and varying particle counts.

Cores	Grid Size	Millions of Particles	GB RAM / Core	
			non-in-situ	in-situ
32	64^2 x 256	33.5	0.97	1.34
64	64 x 256^2	67	0.98	1.34
128	128^3	134	0.98	1.34

Results – Memory Overhead of In-Situ Viz

- The first aspect studied was the memory overhead of the in-situ visualization on the simulations. Table 2 shows results for varying grid and particle counts.
- The results show that the memory overhead is fairly constant at around 360MB/core.

Table 2. : Memory overhead comparison with varying grid and varying particle counts.

Cores	Grid Size	Millions of Particles	GB RAM / Core	
			non-in-situ	in-situ
32	64^2 x 256	33.5	0.97	1.34
64	64 x 256^2	67	0.98	1.34
128	128^3	134	0.98	1.34

Results – Memory Overhead of In-Situ Viz

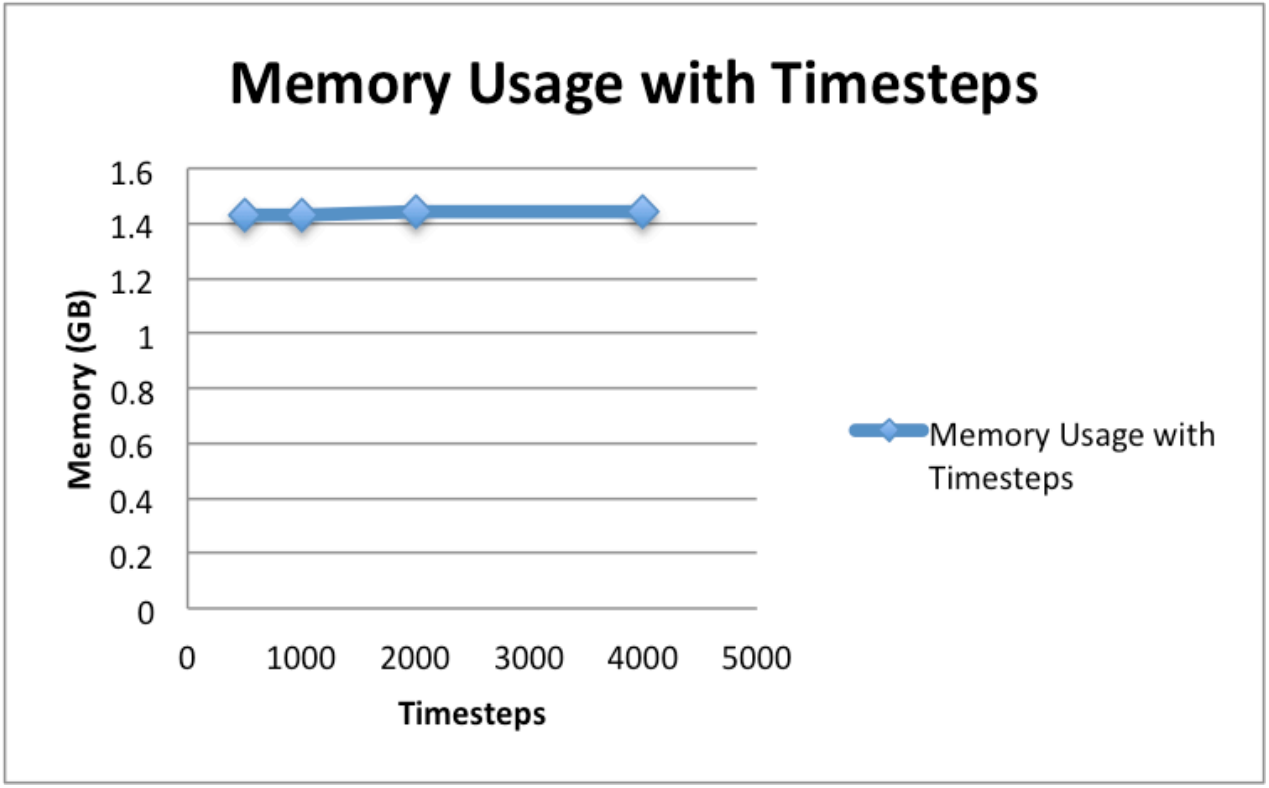


Figure 3: Variation of memory usage with timesteps for the in-situ visualization test case.

Performance Impact of In-Situ Viz

- Test runs conducted with In-Situ Viz output *every timestep* (extreme case). Run times compared with normal case where output is every 500 steps.
- First set is weak scaling with particle counts and a fixed grid size. 20-30% impact on run time with 8 cores/node.

Table 3: Wall clock time (for 500 timestep run) variation with different core counts, with a fixed grid size (128x128x128), and varying particle counts.

Cores	Millions of Particles	Wall-clock time (sec.)	
		non-in-situ	in-situ
32 †	33.5	388.05	459.20
64 †	67	290.53	386.49
128 †	134	311.91	401.88
128 ‡	134	317.49	456.27
†— 8 cores per node, ‡ — 16 cores per node			

Performance Impact of In-Situ Viz

- Test runs conducted with In-Situ Viz output *every timestep* (extreme case). Run times compared with normal case where output is every 500 steps.
- Second set is weak scaling with varying particle counts and grid sizes. 20-35% impact on run time with 8 cores/node. Performance impact consistent up to 1k cores.

Table 4: Wall clock time (for 500 timestep run) variation with different core counts (fixed 8 cores/node), with a varying grid size, and varying particle counts.

Cores	Grid Size	Millions of Particles	Wall-clock time (sec)	
			non-in-situ	in-situ
32	$64^2 \times 128$	33.5	280.35	340.64
64	64×128^2	67	289.43	380.45
128	128^3	134	311.91	401.88
256	$128^2 \times 256$	268.4	318.64	430.58
512	128×256^2	536.9	354.36	453.75
1024 ‡	256^3	1073.74	441.03	546.77
‡ — 16 cores per node				

Performance and Memory Impact of In-Situ Viz

- Test runs conducted with In-Situ Viz output *every timestep* (extreme case). Run times compared with normal case where output is every 500 steps.
- Strong scaling with problem size fixed at 64x128x128, 67M particles.
- Memory overhead is almost constant and performance impact is reasonable given the extreme case considered.

Table 5: Wall clock time and memory overhead (for 500 timestep run) variation with different core counts (fixed 8 cores/ node). Grid size fixed at 64x128x128 and with 67 M particles.

Cores	Wall-clock time (sec)		GB RAM / Core
	non-in-situ	in-situ	
32	506.09	621.61	0.37
64	289.43	380.45	0.36
128	185.20	278.28	0.36

Performance and Memory Impact of In-Situ Viz

- Test runs conducted with In-Situ Viz output *every timestep* (extreme case). Run times compared with normal case where output is every 500 steps.
- Strong scaling with problem size fixed at 128x256x256, 536.9M particles.

Table 6: Wall clock time and memory overhead (for 500 timestep run) variation with different core counts (fixed 8 cores/node except for 1024 core case). Grid size fixed at 128x256x256 and with 536.9M particles.

Cores	Wall-clock time (sec)		GB RAM / Core
	non-in-situ	in-situ	
128	1017.10	1072.97	0.37
256	569.82	643.63	0.37
512	354.36	453.75	0.36
1024 ‡	280.53	406.81	0.36
‡ — 16 cores per node			

Rendering without GPUs

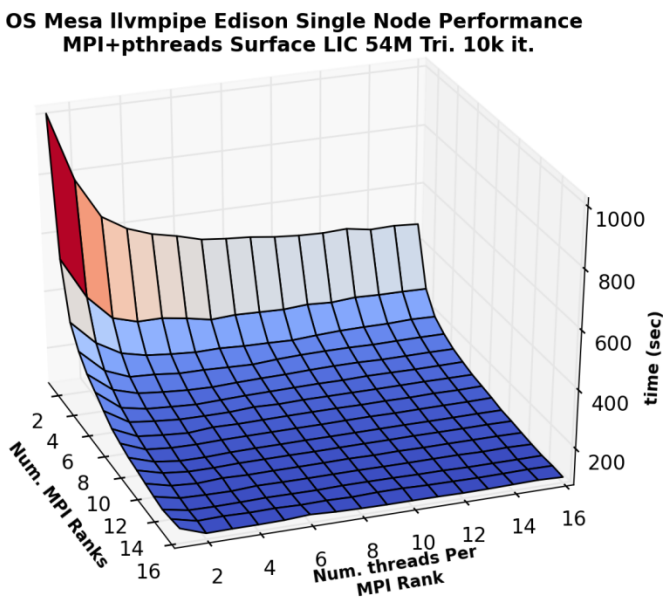
- Mesa an open source OpenGL implementation Used ubiquitously with Linux (http://www.mesa3d.org)
- Provides a fully software based off-screen renderer called “OS Mesa”
- Often OS Mesa is the only option at HPC sites
- Supports up to OpenGL 3.1
- Mesa Gallium llvmpipe renderer is threaded and uses LLVM for JIT compilation of GLSL shaders

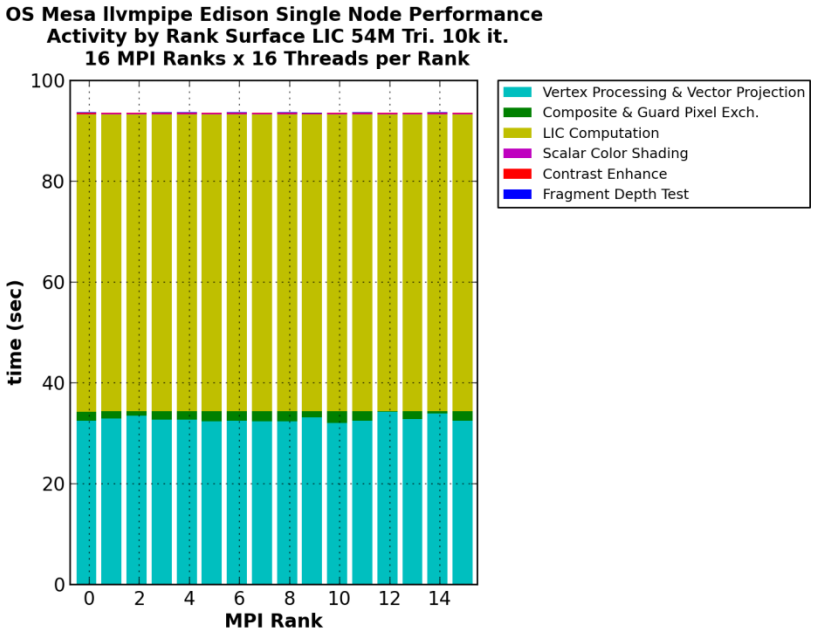
Hybrid-Parallel Rendering w/ Gallium llvmpipe

- Fastest most feature rich software rendering Mesa option
- It's under development, we tested a pull from Mesa's Git repo
  - GL_VERSION: 2.1 Mesa 9.2.0 (git-062317d)
  - GL_RENDERER: Gallium 0.4 on llvmpipe (LLVM 3.2, 128 bits)

- How many rendering threads per MPI rank?
- Does hyper-threading help?

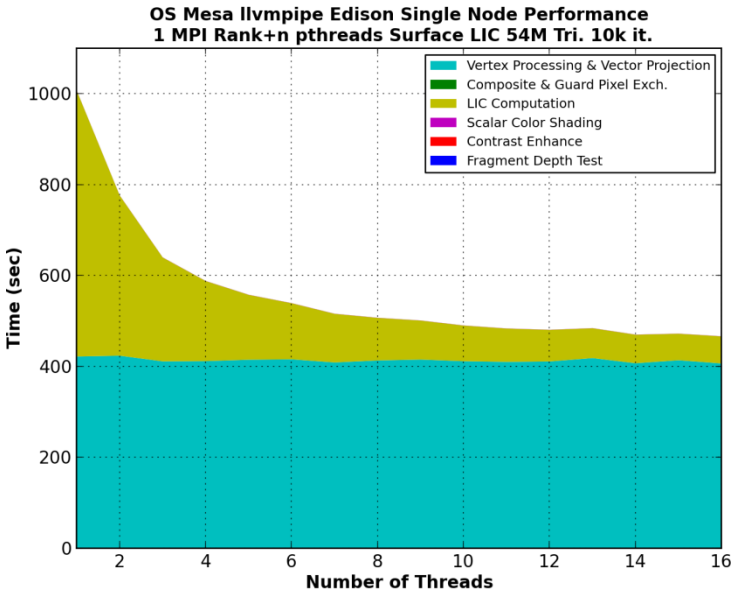
Single node rendering performance

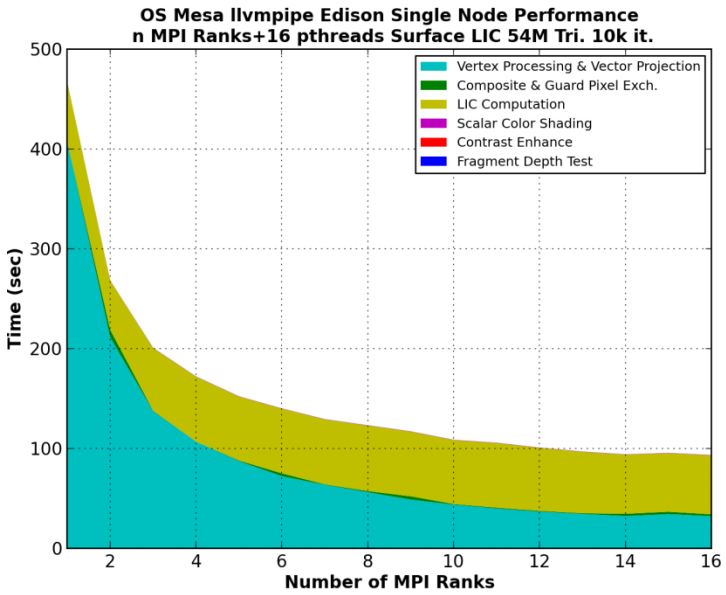




- NERSC Edison Cray XC30, two 8 core Xeon E5-2670 CPUs and 64GB of DDR3 ram
- 256 runs varying number of MPI ranks per node and rendering threads per rank
- Used a large dataset to push the node to it's limits
- Run time dominated by vertex processing and LIC computation, not guard-pixel IPC
- more rendering threads seems to be better but…

Single node rendering performance





- Left : vary number threads on a single rank
  - vertex processing is not threaded !
  - Mesa developers are working on it for a future release.
  - Benefit of threading depends on rendering algorithm's demand for fragment processing
- Right : vary number of MPI ranks with fixed number of threads per rank.
  - After the total number of threads per node increases beyond the number of virtual cores the threading performance does decrease as expected but…
  - Because vertex ops aren't threaded using more MPI ranks per node helps so much that it more than offsets the losses
  - Hyper-threading delivers better performance

Summary

- In Situ Visualization implemented for H3D code using Paraview Catalyst architecture.
- Performance and memory overhead tests run on Gordon for both weak and strong scaling cases.
- Results show memory overhead is reasonable and stays constant as the core count is increased.
- Performance impact is 20-30% for extreme case of viz output every timestep. For practical cases it is expected to be <5%.