In Situ Visualization of Particle Simulations

Will Usher[1], Ingo Wald[2], Aaron Knoll[1], Michael Papka[3], Valerio Pascucci[1]

[1] SCI Institute, University of Utah, [2] Intel Corporation, [3] Argonne National Laboratory

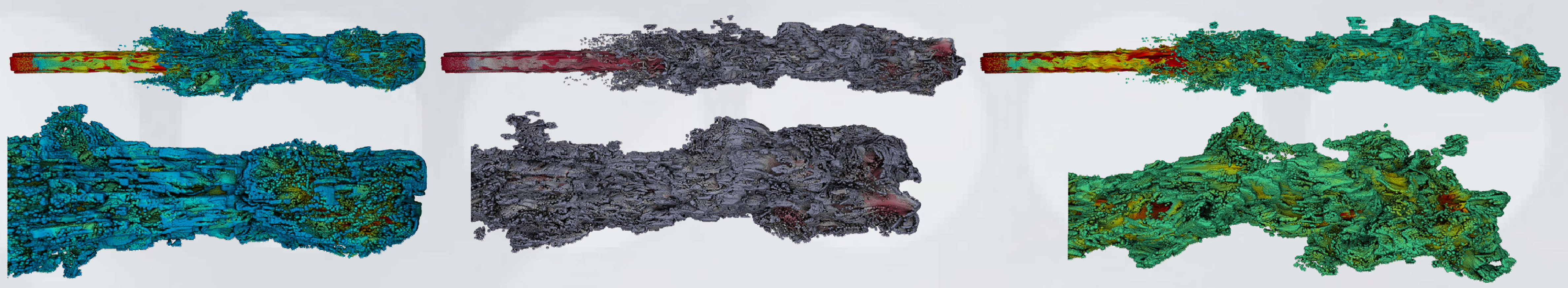


Figure 1. A coal particle combustion simulation in Uintah at three different timesteps with (left to right): 34.61M, 48.46M and 55.39M particles, with attribute based culling showing the full jet (top) and the front in detail (bottom). Using our in situ library to query and send data to our rendering client in OSPRay these images are rendered interactively with ambient occlusion, averaging around 13 FPS at 1920 x 1080.

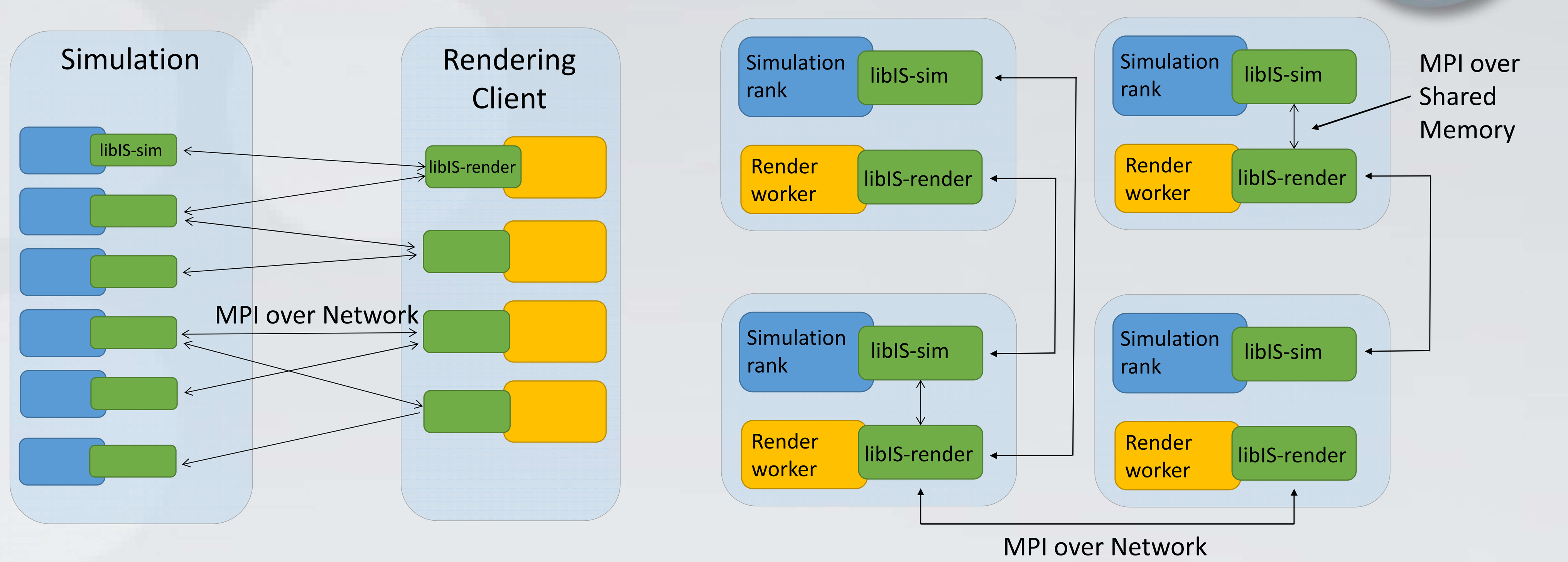


Figure 2. Overview of libIS, showing data forwarded via MPI from the simulation to the renderer.

Introduction

Exascale simulations will produce data beyond what can be effectively archived on parallel file systems. Recent Uintah simulations produce hundreds of gigabytes to terabytes of data and the recent Dark Sky cosmology simulation contains 1.07 trillion particles (32TB/timestep). To explore this large data at full spatio-temporal resolution *in situ* approaches are required.

In situ visualization moves parts of the visualization task into the simulation code or alongside it in a separate process to analyze the data as it's produced, relieving the disk bottleneck for the simulation and analysis.

Lightweight In Situ Library for Particle Data

We introduce libIS, a lightweight library for coupling simulations with in situ applications which allows the client to run on the same nodes as the simulation or different ones and connect/disconnect at will from the simulation. The library comes in two parts, libIS-sim for integrating into the simulation and libIS-render for rendering or analysis clients.

In Situ Data Handling

To query data the simulation sends the world bounds to the clients, which partition the world into a grid. Each client then requests the brick(s) it's been assigned to render from the simulation. This ensures the data layout is suitable for data-distributed rendering with sort-last compositing and allows us to couple to simulations with arbitrary data layouts.

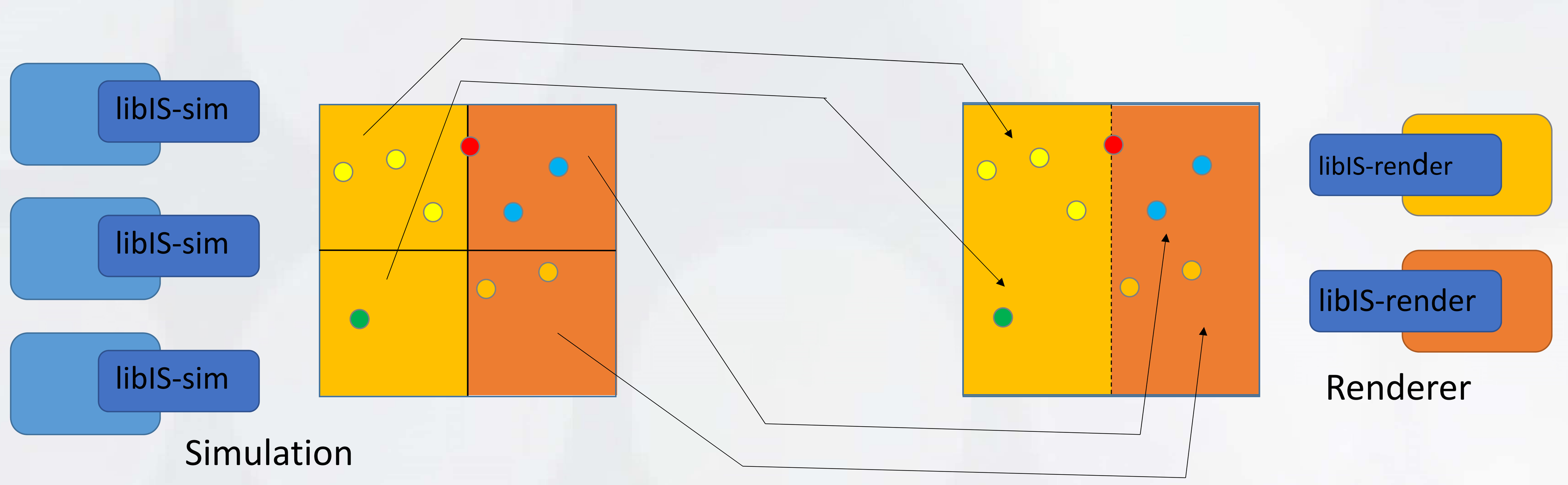
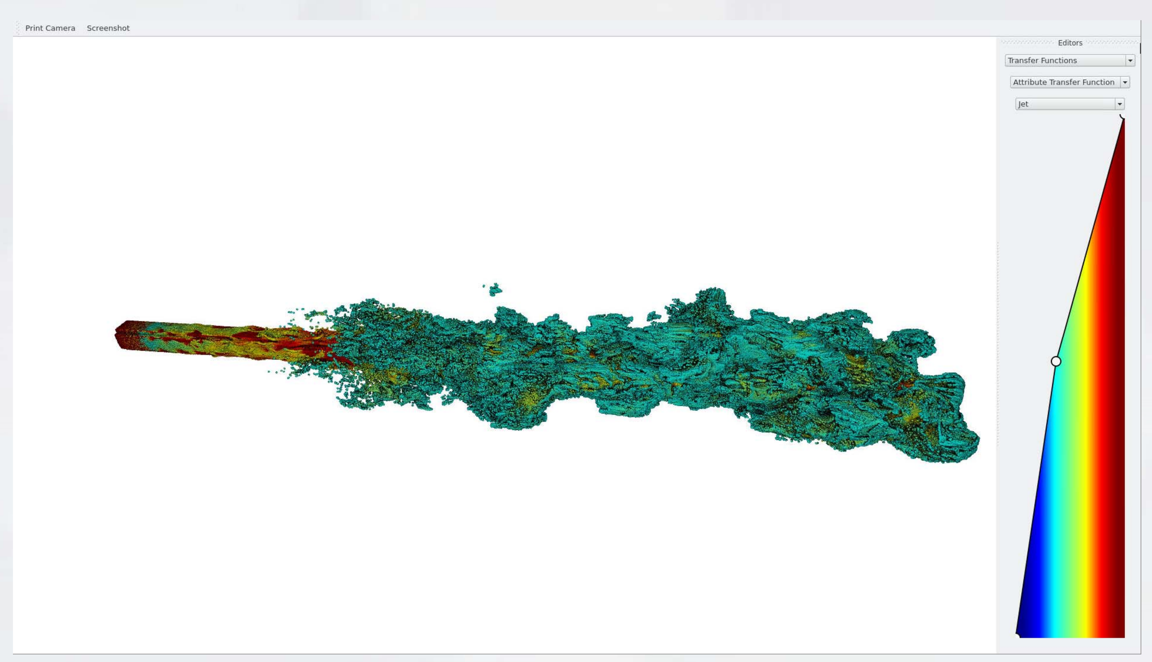


Figure 3. Data query with 3 simulations ranks coupling to 2 renderer ranks

Rendering Client in OSPRay

We implement a distributed particle rendering client in OSPRay which periodically requests new timesteps from the simulation, and allows for interactive camera movement and transfer function editing to cull particles by attribute.



Results

We evaluate libIS and our rendering client on Stampede and Maverick at TACC in separate and shared configurations, rendering data in situ from Uintah (Fig 1.) and LAMMPS (Fig 9.) simulations.

Separate Nodes

Running the renderer and simulation on separate nodes allows for improved rendering and simulation performance since the nodes aren't as overloaded, ideal for exploring the simulation over a long period. We evaluate with Uintah on Stampede and LAMMPS on Maverick.

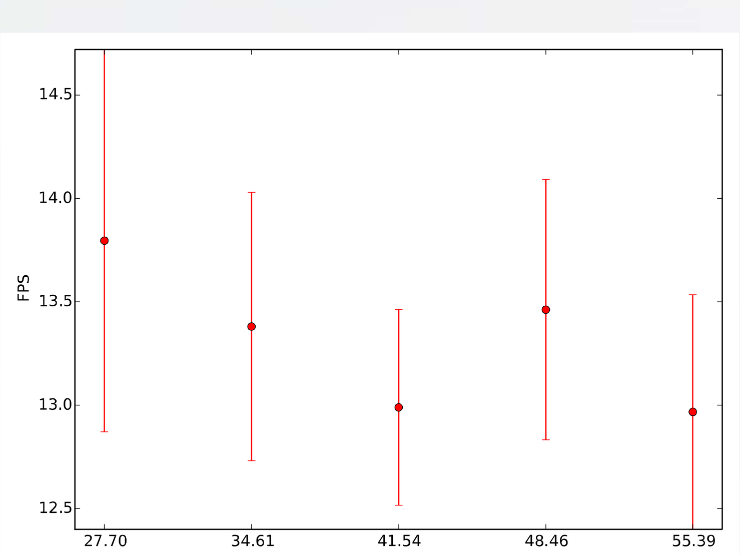


Figure 4. Framerate of separate run with 64 Uintah ranks sending to 12 OSPRay ranks

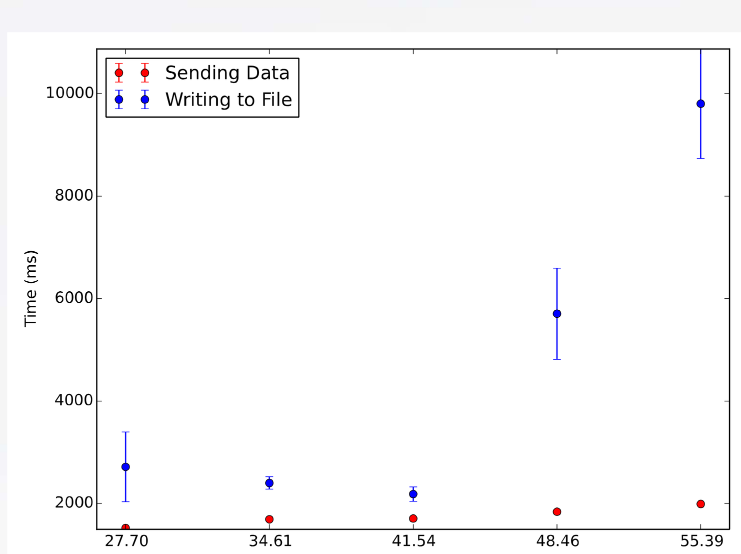


Figure 5. Sending data vs. writing files for 64 Uintah ranks sending to 12 OSPRay ranks

Shared Nodes

Running on the same nodes as the simulation impacts both simulation and renderer performance, but only requires a single interactive node to display the viewer. This mode provides a non-intrusive method for quickly checking in on the state of a long running simulation. In this configuration there's also the possibility that MPI will transfer data using shared memory, however this is not guaranteed by our system. We compare both data send time and rendering performance with LAMMPS on Maverick and compare to the separate node configuration, Fig 6-8.

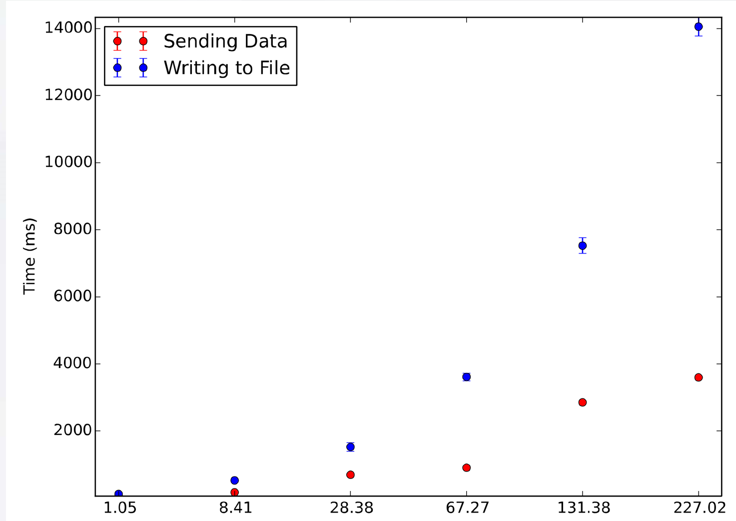


Figure 6. Sending data vs. writing files for 80 LAMMPS ranks sending to 16 OSPRay ranks.

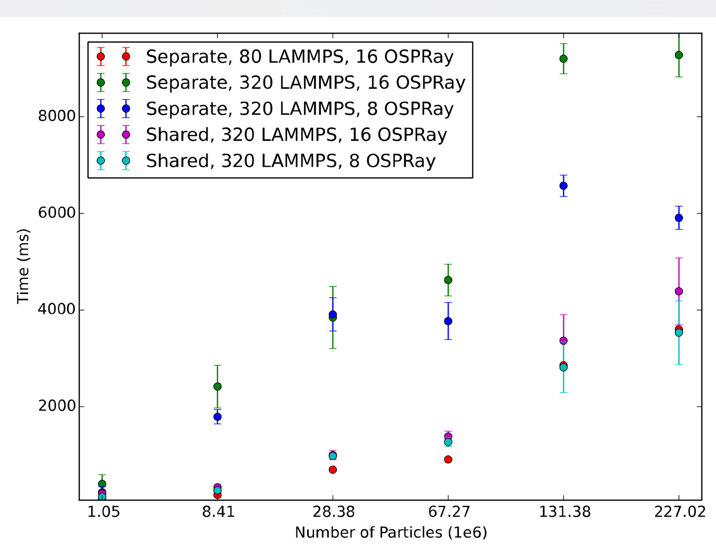


Figure 7. Data send time of separate vs. shared runs with LAMMPS.

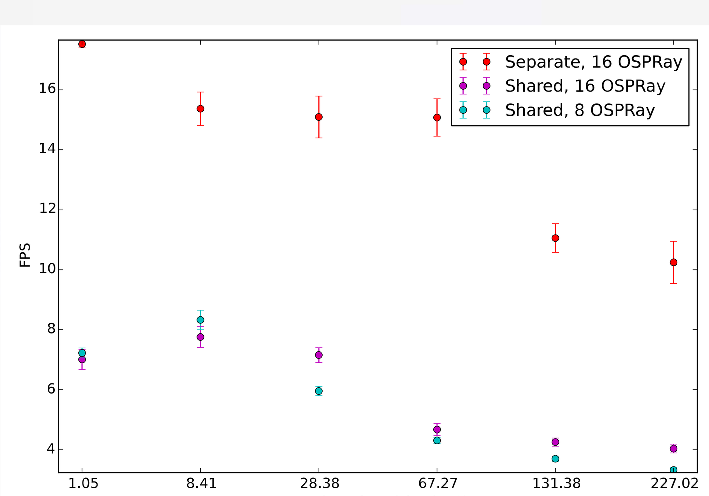


Figure 8. Framerate of separate vs. shared runs with LAMMPS.

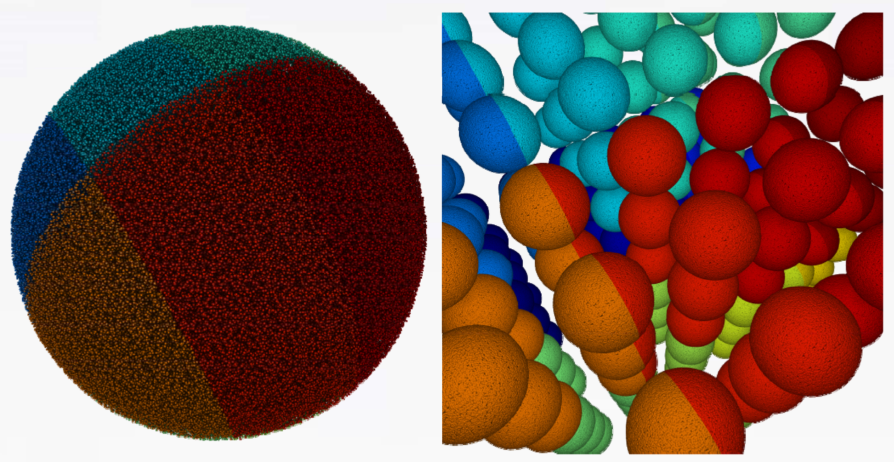


Figure 9. LAMMPS single carbon nanosphere and 6^3 replicated grid of nanospheres.

Acknowledgements: This work was supported in part by NSF: CGV: Award:1314896, NSF CISE ACI-0904631, DOE/Codesign P01180734, DOE/SciDAC DESC0007446, CCMSC DE-NA0002375, and PIPER: ER26142 DE- SC0010498. Additional support comes from the Intel Parallel Computing Centers program and the Argonne Leadership Computing Facility. This material is also based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002375.