Performance issues in simulating Laser-Plasma Interactions in National Ignition Facility experiments

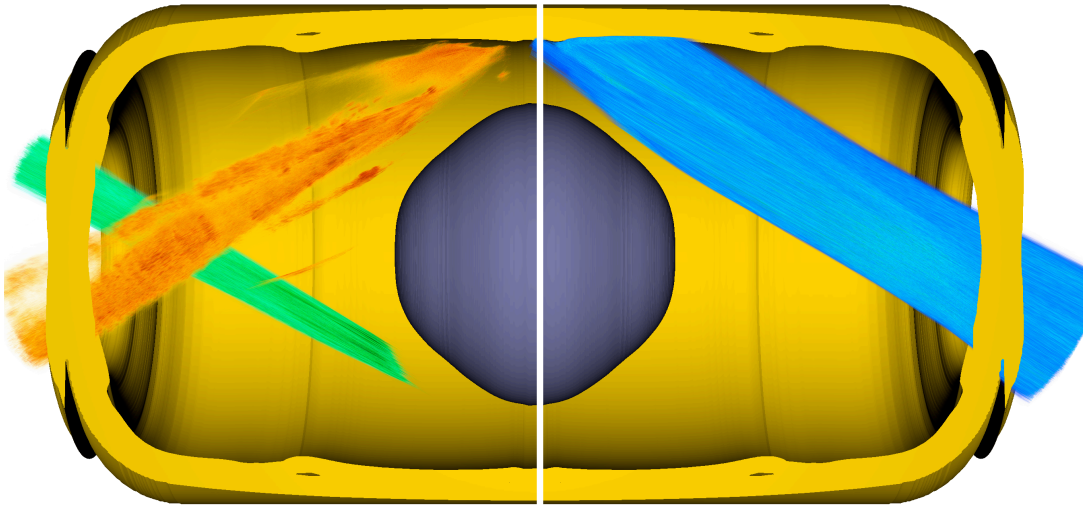
Steven H. Langer, Charles H. Still, Denise E. Hinkel, A. Bruce Langdon, and Edward A. Williams
contact: “langer1” AT llnl.gov

Lawrence Livermore National Laboratory

This work was performed under the auspices of the Lawrence Livermore National Security, LLC, (LLNS) under Contract No. DE-AC52-07NA27344

LLNL-PRES-441933

PF3D simulates high intensity laser beams propagating through ionized gas



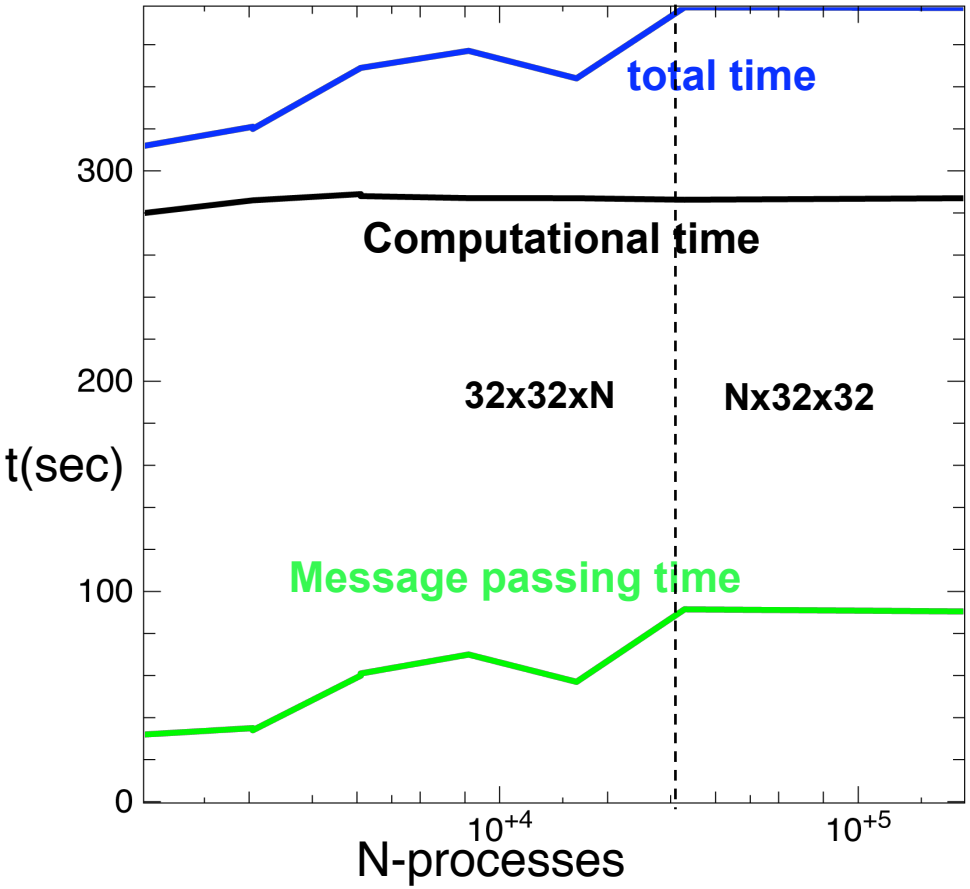
PF3D simulations model the interactions of the laser beam (blue), Raman backscattered light (orange), and Brillouin backscattered light (green).

- The zone size in a PF3D simulation is ~2-3 times the 0.35 μm laser wavelength.
- The gas volume is several mm on a side.
- Billions of zones are required to simulate the full beam volume.
- A multi-week 55 billion zone simulation has been run on a BlueGene/P system at LLNL.

We are moving to a mixture of MPI and OpenMP threads as our parallel programming method

- pf3d is currently a pure MPI code with a 3D spatial decomposition.
- Future systems will probably have less memory per core.
- Large domains have a better compute to message ratio than small domains.
- Threading MPI processes will allow us to have larger domains.
- Preliminary testing suggests that we can efficiently use of order 4 OpenMP threads per process on current systems.
- If future systems have on-chip thread synchronization, we could have more threads per process.

Careful mapping of MPI domains to the interconnect is necessary to achieve good performance



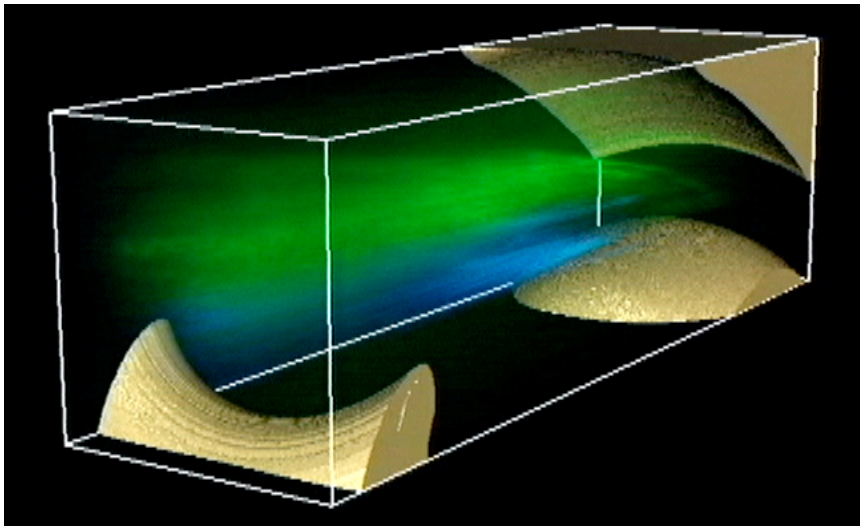
- The LLNL BG/L interconnect was a 96x32x32 3D torus during our full system run.
- We mapped our 32x32x96 domain simulation onto the interconnect in zyxt mode.
- Most messages are passed within 32x32 xy-planes.
- This mapping only uses 2 of the 6 links at a time, but provides excellent scalability to 192K processes.
- Alternative mappings were faster for less than 32K processes but did not scale well.
- We are working on mappings that maintain the scalability of zyxt while using more links.

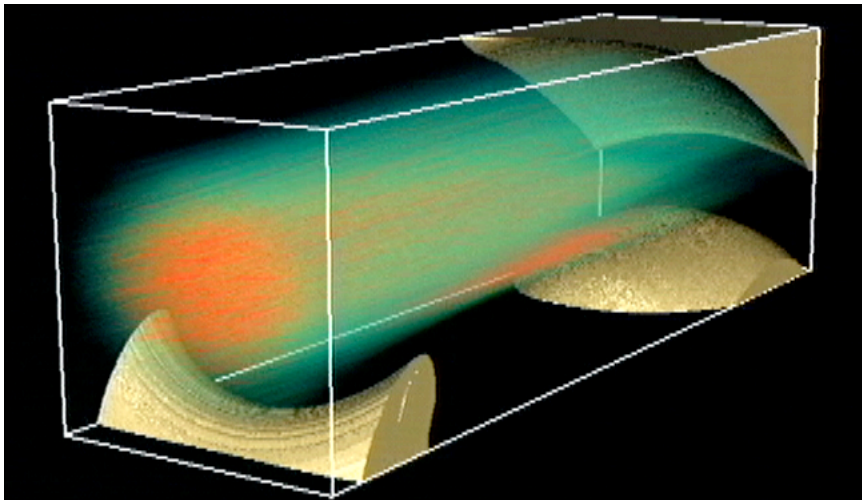
I/O performance will continue to be an issue on large parallel computers

- pf3d writes multiple files using Posix I/O as opposed to a single file written using MPI-IO.
- The processes are split into I/O groups (typically 8-32 processes) that share a file for visualization, history, and spectral dumps. These dumps are moderate sized and are written frequently.
- Group members send their data to the group leader via MPI messages.
- The group leader writes the data for all group members into a single file.
- Restart dump sets are written with one file per MPI process.
- Our scheme has worked well on Lustre file systems (with or without caching) for several years.
- We recently modified the I/O scheme to deal with distributed metada operations and the 64K file per directory limit of Intrepid's GPFS.
- We have achieved write rates of 17 GB/s for a 46 TB data set on Intrepid.
- In mixed MPI/OpenMP runs our plan is to have MPI processes (not individual threads) handle file I/O.

Post-processing will require new or enhanced parallel tools

- We used VisIt to make a volume visualization movie of a 55 billion zone simulation. It required almost all of the 2 TB of memory in LLNL's prism visualization cluster.
- We are currently running a 112 billion zone simulation on Intrepid. The Eureka visualization cluster has 3.2 TB of memory which is probably not enough to make a movie the way we did for the 55 billion zone simulation.
- We need to either simplify the visualization or reduce the memory used by VisIt.





FFT performance is being improved

- pf3d currently uses FFTs that we compile from the source code. We can use either FFTW or a FFT due to Paul Schwartztrauber of NCAR.
- We are modifying pf3d to access the hand tuned FFTs in Intel's MKL and IBM's ESSL libraries. Both have FFTW wrappers for their FFTs, so they can be used as a drop-in replacement for the FFTW we compiled.

pf3d needs to make better use of short vector hardware

- BG/L and BG/P systems have a “double hummer” floating point unit. The performance of a code can be doubled if all operations are performed on aligned, two element double precision vectors.
- Intel and AMD x86_64 processors have a 128 bit SSE2 unit. pf3d typically performs computations using 32 bit float data. The performance on an x86_64 system can be quadrupled if all operations can be performed on four element vectors.
- pf3d currently has a few kernels where calls to vector library functions in MKL or MASSV have been inserted by hand and some loops are vectorized by the compiler.
- Vectorization currently provides much less than a 2X or a 4X speedup.
- Efforts are underway to make it easier for the compiler to recognize vectorizable loops.