pF3D Simulations of Laser-Plasma Interactions in National Ignition Facility Experiments

Steven Langer, Abhinav Bhatele, and Charles H. Still

Abstract—The laser-plasma interaction code, pF3D, is used to simulate laser-plasma interactions in National Ignition Facility experiments. This paper describes the optimizations performed on pF3D to enable scaling to a million or more processes. We begin with a brief description of NIF experiments. We then describe pF3D with a focus on the features that are needed to get good performance at scale. The scalability of message passing, disk I/O, and code steering are key issues that impact scalability and are discussed in detail. Scaling studies on IBM Blue Gene/L, Cray XE6, and IBM Blue Gene/Q systems are used as examples. The paper concludes with a comparison of the backscattered light measured in NIF experiments and computed in the pF3D simulation.

1 Introduction

The National Ignition Facility (NIF [1]) is a large DOE/NNSA experimental facility that houses the world's most powerful laser. One of the key goals of the NIF is to compress a target filled with deuterium and tritium to a temperature and density high enough that nuclear fusion ignition occurs. Figure 1 shows the laser beams entering a hohlraum through holes at both ends. The beams ionize the gas and propagate through the resulting plasma until they reach the hohlraum wall. The laser beams deposit their energy and the wall becomes hot enough that it radiates x-rays. The x-rays fall on the surface of the capsule at the center of the hohlraum and cause it to implode.

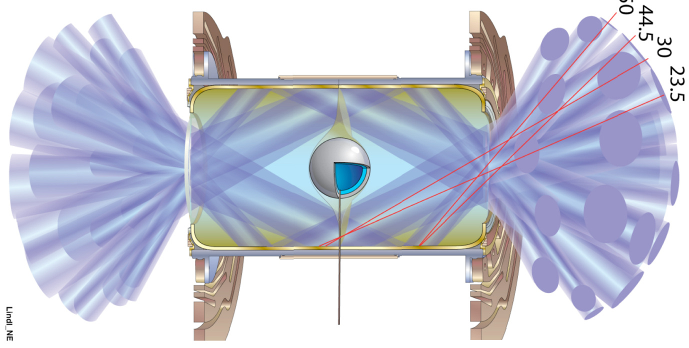

Fig. 1. The figure shows the laser beams entering the hohlraum through holes at both ends and propagating through the plasma until they reach the hohlraum wall. The hohlraum is a "can" used to trap and re-radiate the x-rays emitted by the hot interior walls of the hohlraum.

There are 192 beams in 48 quads (2x2 groups of beams) that heat the plasma to a few tens of millions of degrees. The laser intensity in a NIF hohlraum is high enough that laser-driven plasma instabilities may grow to significant levels. In particular, it is possible for some of the laser light to be backscattered and leave the hohlraum without heating the plasma. Backscatter levels in two experiments with new types of targets were high enough to damage a NIF mirror.

pF3D is a multi-physics code used to simulate interactions between laser beams and the plasma in NIF experiments. pF3D simulations can be used to help select experiments with acceptably low levels of backscatter and to help understand the impact of backscattered light on hohlraum energetics.

pF3D zones are roughly the size of the laser wavelength (0.35 μm) while the plasmas of interest are several mm across. A simulation of the full 7 mm path of a single quad from the entrance hole to the hohlraum wall requires 50 billion or more zones. A simulation of 3 overlapping quads for 2.5 mm requires 600 billion zones. pF3D simulations at this scale may require over 100 million core-hours to complete.

The scale of these simulations suggests that achieving good performance is very important. We have made several changes to improve CPU performance by increasing cache utilization, enabling the compiler to use SIMD instructions, and adding OpenMP directives to the MPI parallelism which was already present in pF3D. That is not the topic of this paper.

We focus on the features required for message passing and parallel I/O to work well with a million processes. pF3D passes large volumes of messaging data in 2D FFTs and light advection. Careful control of how MPI processes are placed on the interconnect topology can yield much higher messaging rates. pF3D also uses checkpoint-restart files to recover from system errors. Saving the state of a pF3D simulation may require up to 190 TB of disk space. High I/O rates are required to keep checkpoint times low. We conclude with a brief discussion of some pF3D results.

• *Steven Langer, Abhinav Bhatele, and Charles H. Still are with the Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551 USA. E-mail: langer1@llnl.gov, bhatele@llnl.gov, still1@llnl.gov.*

Manuscript received January 24, 2014; revised May 5, 2014

2 Computational Approach

The laser beam propagation code pF3D [2], [3] is a massively parallel, three-dimensional (3D) Cartesian code designed to study laser-plasma interactions (LPI) in experiments such as those being carried out at the National Ignition Facility[1]. The code couples an electromagnetic wave solver in the paraxial approximation (a Helmholtz equation enveloped around the laser wavenumber) for the forward propagating laser electric field to a multi-fluid Eulerian hydrodynamics package (the mesh is fixed: no adaptive refinement is necessary). The code also includes models for Stimulated Brillouin backScatter (SBS), where the forward light scatters off self-generated ion-acoustic waves, Stimulated Raman backScatter (SRS), where the forward light scatters off self-generated electron plasma (Langmuir) waves, electron heat conduction via linearized nonlocal transport or Spitzer-Harm transport, and realistic beam models [2]. The numerical schemes use explicit time step finite-difference or spectral methods with second-order accuracy in space, and have been shown to accurately model experimental results in the regimes relevant to Inertial Confinement Fusion (ICF) experiments [4].

All of the incident and scattered light waves and the Langmuir wave are handled using the paraxial solution technique outlined here. The forward propagating light wave is used as the example where specifics are needed for clarity. The other waves are handled similarly by straightforward modifications to the equations detailed below.

If we take z to be the laser propagation direction (xy-planes are thus transverse to the propagation direction), the forward propagating light vector potential is

$$A = A_0 e^{-i\omega_0 t + ik_0 z} + A_0^* e^{i\omega_0 t - ik_0 z} \tag{1}$$

where $\omega_0(k_0)$ is the laser frequency (wavenumber). The SBS and SRS vector potentials are A_B and A_R, and A_0 couples to the scattered light and density fluctuations by the paraxial wave equation,

$$(-2i\omega_0 \frac{\partial}{\partial t} - 2ik_0 c^2 \frac{\partial}{\partial z} - c^2 \nabla_\perp^2 + v)A_0 = \\ \frac{4\pi e^2}{m_e}(\delta n_f A_0 + \frac{1}{2}\delta n_a A_B + \frac{1}{2}\delta n_L A_R) \tag{2}$$

Here, the electron density $n_e = n_f + n_a + n_L$ is split by a separation of temporal and spatial scales into a slowly varying hydrodynamic part n_f, and axially rapidly varying parts responsible for backward Brillouin n_a and backward Raman n_L. We solve the paraxial equation by an operator-splitting time-advance technique (method of fractional steps) to ensure second-order accuracy. The energy in the light waves is coupled using an action-conserving scheme.

1. http://lasers.llnl.gov

3 Message Passing

The pF3D grid is split up into $np \times nq \times nr$ domains and one domain is assigned to each MPI process. Each process has a coordinate (p, q, r) within the 3D decomposition of the grid. A domain is comprised of $nxl \times nyl \times nzl$ zones. Message passing is used to pass boundary information between the processes. pF3D can spend over 30% of its execution time exchanging messages, so optimizing message passing rates is important.

Each wave propagation (involving the $\nabla_\perp^2$ term in Equation 2) is solved using Fourier transforms in xy-planes. The Fourier transforms require message passing across the full extent of an xy-plane. These messages are sent using `MPI_Alltoall`. There are nq MPI communicators for messages in the x-direction and np MPI communicators for messages in the y-direction within each plane. Propagation of the light requires passing planes of information in the z-direction using `MPI_Isend` and `MPI_Recv`. The higher-order advection scheme in pF3D requires 3 xy-planes from the adjacent domains.

pF3D also solves the hydrodynamic equations. Message passing in this phase involves exchanges of single planes across the six faces bounding each domain using `MPI_Isend` and `MPI_Recv`. The light equations are subcycled because light moves much faster than sound waves. In a typical run, the hydrodynamic equations are solved once every 50 light sub-cycles. pF3D spends most of its time working on light propagation and coupling because hydro steps are so infrequent.

An "xy-slab" is owned by the set of all processes sharing a common value for "r", the z-location within the decomposition. A slab is $nx = np \times nxl$ zones wide by $ny = nq \times nyl$ zones high by nzl zones deep. A separate 2D FFT is performed on each of the nzl planes within an xy-slab. All message passing during a 2D FFT is confined to a single xy-slab. Message passing performed as part of light propagation and hydrodynamics couples different xy-slabs.

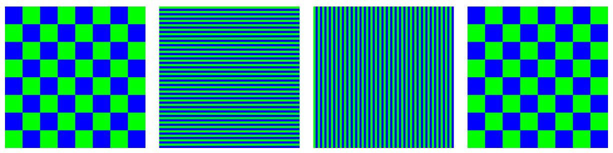

Fig. 2. The 2D FFT message passing scheme.

Figure 2 shows a high-level overview of the message passing involved in performing a 2D FFT. `MPI_Alltoall` calls are used to perform four transposes during an FFT. The first step is to transpose from the "checkerboard" decomposition to a row decomposition. Each process then does an ordinary single processor 1D FFT of length nx on its rows. The next step is to transpose back to the checkerboard (not shown) and transpose again to a column decomposition. A 1D FFT of length ny is then performed and there is a final transpose back to the original checkerboard state. The x-phase of the FFT uses a total of $nq \times nr$ x-communicators and the y-phase uses a total of $np \times nr$ y-communicators.

3.1 Task Mapping on Blue Gene systems

The mapping between MPI processes and the associated physical domains and their location on the interconnect can have a large impact on the message passing performance on systems with mesh or torus-based interconnects. pF3D must run on very large numbers of processors, so the first requirement for a good mapping is that it should be highly scalable. The majority of the message passing occurs during 2D FFTs, so it is important to choose a mapping that gives very good performance for FFT messages. The mapping must also give good performance for messages passed across faces to adjacent domains, but this is a somewhat lower priority.

Jobs running on IBM Blue Gene/L (BG/L) and Blue Gene/P (BG/P) systems are assigned partitions of the interconnect that are a private 3D torus. Jobs running on IBM Blue Gene/Q (BG/Q) systems are assigned partitions that are a private 5D torus. There is no contention between messages from different applications because the applications run on different tori.

The regular and balanced message passing of pF3D and the private tori on a Blue Gene system make it possible to generate mappings from MPI rank to torus location which deliver highly scalable message passing rates. The default behavior on Blue Gene systems is to assign MPI processes to nodes in MPI "rank order". Several permutations of this rank order are supported. On a BG/P using XYZT mode, the x-coordinate on the interconnect varies fastest and the "thread" coordinate, t, varies slowest as the MPI rank increases. In ZYXT mode, the z-coordinate on the interconnect varies fastest and the thread coordinate varies slowest as the MPI rank increases. Both of these mappings scatter physical planes across multiple interconnect planes unless the dimensions of the decomposition match the dimensions of the interconnect. These default mappings tend to produce message passing rates that drop significantly as the number of nodes increases in a weak scaling study.

Figure 3 shows per-process message passing rates as a function of the number of processes for weak scaling studies on the LLNL BG/L, Cielo (Cray XE6 at LANL), and Sequoia (IBM BG/Q at LLNL). Different test problems were used on the different clusters so the rates are not directly comparable. The rates do give a rough indication of the relative performance of the various interconnects.

The BG/L runs had a $32 \times 32 \times N$ decomposition. The XYZT mapping delivers good rates for small simulations, but is very slow for 192k processes. The ZYXT mapping has nearly constant messaging rates from 32k to 192k processes and is over $2\times$ faster for 192k processes. The ZYXT mapping scales well because a 32×32 xy-slab maps onto a 32×32 plane of the interconnect. An x- or y-communicator uses a single row or column

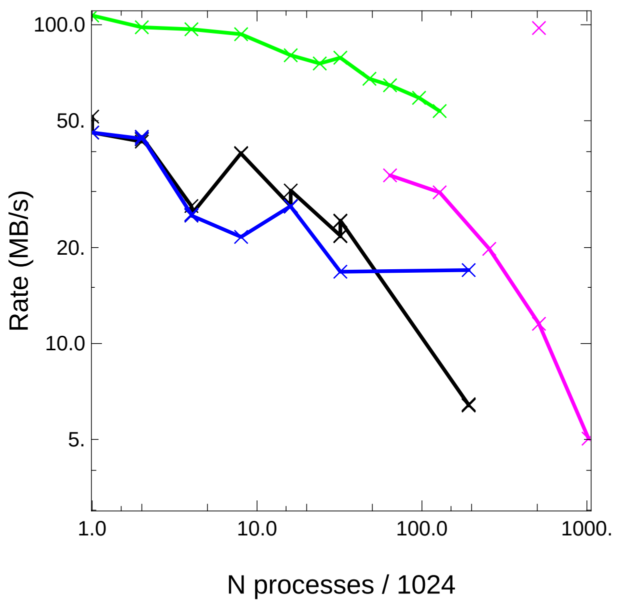

Fig. 3. Aggregate message passing rates from weak scaling studies on BG/L with the the default XYZT mapping (black), BG/L with ZYXT mapping (blue), Cielo (green), and Sequoia with TEDCBA mapping (magenta) are shown as a function of the number of processes. The XYZT mapping (black) works well on BG/L systems for small process counts but is very slow for 192k processes. The rate for ZYXT is constant from 32k to 192k processes. The rate for Cielo drops steadily as the number of processes increases. The rate for Sequoia drops rapidly for 256k or more processes. The magenta point for 512k processes used a custom mappings from MPI rank to torus location and other modifications to achieve a much higher bandwidth on Sequoia.

of the interconnect and does not contend with any other communicator for links. Hop counts remain constant as xy-slabs are added.

The Sequoia (BG/Q) rates were generated from a problem with a $32 \times 16 \times N$ decomposition and a 20480×4608 zone xy-plane. The run used 2 MPI processes per core. One of the BG/Q default mappings, TEDCBA, was used.The letters A through E denote the 5 torus directions and T is the hardware thread number. The MPI ranks were mapped to the torus in "rank order". The hop counts with an x-, y-, or z-communicator increase as the number of processes increases. Contention for links is also likely to increase. The result is that the message passing rate falls off for 256k or more processes. This result is consistent with results on BG/L and BG/P systems - pF3D message passing rates do not scale well with default rank to torus mappings unless they happen to line up with the problem decomposition. The magenta point shows a 97.7 MB/s message rate for 512K processes using custom mappings from MPI rank to torus and other recent optimizations to the message passing.

The scalability of the ZYXT mapping is attractive, but it delivers message rates lower than XYZT delivers for small runs. We set out to develop custom mappings which would have the scalability of ZYXT but deliver higher rates. These schemes divide the partition into equal sized blocks and map an xy-slab to each block. Blocks for adjacent xy-slabs are placed next to each other on the torus. Hop counts within blocks do not change as the number of blocks increases, the distance to adjacent slabs does not change, and messages passed during a 2D FFT do not interfere between blocks. The ZYXT mapping with 32×32 process xy-slabs is a special case of these scalable mappings. Mappings using N-dimensional blocks instead of planes perform better than ZYXT because an FFT can use all three pairs of links on a BG/L or BG/P and all five pairs on a BG/Q.

The number of possibilities that need to be investigated when searching for a near optimal mapping is high, so we developed a tool called Rubik which can be used to quickly generate many scalable mappings [5]. Rubik can take an N-dimensional partition and split it up into equal sized N-dimensional blocks. Rubik can "scramble" the processes within a block to improve link utilization. Rubik mappings have delivered excellent scalability and good performance on BG/P systems. Recent investigations have shown that Rubik mappings can produce a 4X speedup on BG/Q systems.

3.2 Message passing on a Shared Torus

The characteristics of Blue Gene systems have allowed us to deliver scalable message passing with high message rates. Cray XE6 systems have an interconnect that is a 3D torus, so they seem similar to Blue Gene systems at first glance. In practice, they are significantly different. Some of the nodes on the torus are used for I/O and some are in use by other jobs. The shape of the set of nodes assigned to a simulation is irregular and will change from one job submission to the next. Message contention between applications can occur because they all run on the same torus. Regular mappings from physical domain to location on the torus are not possible due to the irregular shape of the node set assigned to a job.

On the XE6, the network is built out of Gemini routers with two compute nodes connected to each router. The default scheme on LANL's Cielo is to allocate in units of 2 Gemini routers (or $2\times2\times1$ nodes). pF3D used a $16\times16\times N$ decomposition, so x-communicators were confined to a 16-core node and most y-communicators will map to a $2\times2\times4$ chunk of the torus. Some communicators will be larger because of nodes that are in use for I/O or by other jobs. An xy-plane had 10240×3072 zones.

Figure 3 shows the message passing rate averaged over all communication phases as a function of the number of processes for a weak scaling study on Cielo. The Cielo runs used the default mapping between MPI rank and location. The message passing performance drops slowly from 105 MB/s to 55 MB/s as the number of processes increases from 1k to 128k. The message passing rate varies from 80 MB/s to 100 MB/s from one job submission to the next for a 32k simulation that we completed on Cielo. This variation is nearly half the size of the variation due to the number of processes in the scaling study. This suggests that much of the variation in both cases may be due to the placement of processes on the Cielo torus.

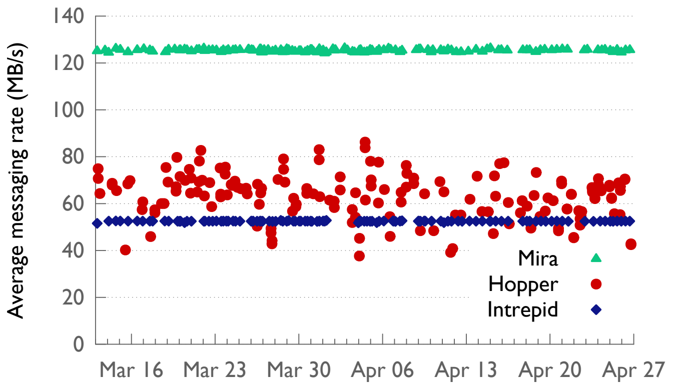

Fig. 4. Average messaging rates for pF3D batch jobs running on IBM Blue Gene/Q (Mira - green), a Cray XE6 (Hopper - red), and an IBM Blue Gene/P (Intrepid - blue). The slowest message rate on Hopper is half the fastest rate.

The message passing rates in the figure include X messages which are quite fast because they are on node. The message passing rate for Y messages summed over all 16 processes is probably about 1 GB/s per node. That is significantly less than the roughly 5 GB/s rate that MPI benchmarks achieve.

Hopper is a Cray XE6 at NERSC. It runs more and smaller jobs than Cielo. The default policy on Hopper is to allocate in units of nodes, so two jobs may share a single Gemini router. The default y-communicator for our test runs is $1 \times 2 \times 8$. Figure 4 shows the average message passing rate from many batch jobs for a modest size pF3D test problem on three different architectures. The variability is very low on both the BG/P and BG/Q. The variability is large for the runs on the Cray XE6. In runs that get low messaging rates, pF3D spends more time passing messages than computing. Our investigation has shown that the most important contributor to this variability is contention with other jobs which are passing messages that traverse links in use by pF3D [6]. Variability is lower on Cielo where the allocation unit is larger and there are a few large jobs instead of many smaller ones. Currently, we do not have any way to reduce this variability. Changing process allocation schemes so that applications have private blocks of the torus would help reduce message rate variability, but might also reduce node utilization due to packing difficulties.

TABLE 1
Choosing the correct parallel I/O scheme can make a large difference to performance. The GPFS file systems on Intrepid and Mira require the files in a dump set to be split across many directories to give good performance. The other systems used Lustre file systems. The file system on Dawn delivers much better performance with custom I/O groups that are spread uniformly across the I/O nodes. There were 2 MPI processes per core on Mira and Sequoia and 1 on other systems.

system	no. of processes	file system	group type	priv dirs	hst GB	hst rate GB/s	rst TB	rst rate GB/s
Dawn	144K	Lustre	unif.	no	25.4	0.124	15.93	14.66
Dawn	32K	Lustre	BG	no	4.77	0.165	2.08	6.39
Intrepid	16K	GPFS	BG	no	3.43	0.014	1.04	0.59
Intrepid	16K	GPFS	BG	yes	3.43	0.374	1.04	5.56
Intrepid	160K	GPFS	unif.	yes	6.23	0.061	48.00	16.23
Cielo	64K	Lustre	unif.	no			64.00	57.0
Mira	256K	GPFS	unif.	no	29.0	0.259	18.70	26.13
Sequoia	64K	Lustre	unif.	no	73.0	0.241	16.36	6.82
Sequoia	128K	Lustre	unif.	no	49.0	0.263	32.73	10.04
Sequoia	256K	Lustre	unif.	no	56.0	0.302	65.46	19.72
Sequoia	512K	Lustre	unif.	no	79.0	0.323	130.91	40.28
Sequoia	1024K	Lustre	unif.	no	96.0	0.678	191.97	38.83

4 Parallel I/O

Good I/O rates on parallel file systems require writing large data streams to disk in linear order. pF3D uses yorick's I/O package [7], [8] which includes buffering for input/output. We set the buffer size to match the block size (typically 4 MB) of the parallel file system. The yorick I/O package naturally writes bytes sequentially to disk. HDF5 in its default mode writes alternately between a symbol table and a data section. This results in flushing partial blocks to disk unless special buffering code is added. The I/O rates for pF3D are higher than those achieved by codes using HDF5 in default mode.

pF3D uses a file-per-process strategy for restart dump files. This approach works well because each restart file is large enough (at least 100 MB) that it can be written efficiently. Some file systems do not perform well with large numbers of files being written simultaneously, so pF3D has an option to write restart files in multiple phases so that only a fraction of the processes write at one time. pF3D's I/O package has been tested on several large systems at DOE/NNSA labs. Intrepid (160K cores, ANL) and Dawn (144K cores, LLNL) are IBM Blue Gene/P systems, Cielo (139K cores, LANL) is a Cray XE6, and Mira (768K cores, ANL) and Sequoia (1536K cores, LLNL) are IBM Blue Gene/Q systems.

History (hst) files contain a few spatially decomposed slice planes for key variables and are saved at regular intervals. The dump set size is under 1% of the size of a restart dump set. If we tried to write a file per process, it would take longer to create the files than to write the data. To get better performance, we form I/O groups. Each process writes its share of the data to an in-memory file then sends the data as a single message to its I/O group leader. The group leader writes the data from its group members to their shared file. This results in significantly better performance than using a file-per-process approach.

Blue Gene systems have I/O nodes which are dedicated to a specific set of compute nodes. When writing hst dumps it is important to make sure that the group leaders are spread uniformly across the I/O nodes. pF3D has an option to compute I/O groups at run time based on each process's location relative to the I/O nodes.

The GPFS file system uses each I/O server as a metadata controller. The server must obtain a lock on a directory before creating a file. Access to the lock becomes a serious bottleneck if many files are written to the same directory. pF3D has an option to spread the files in a given dump set across many sub-directories. Lustre file systems have a centralized metadata server and performance tends to be good with many files in a single directory.

Table 1 shows the time spent by pF3D writing history ("hst") and restart ("rst") dumps for several runs. Blue Gene/P specific dumps (group type "BG") improve performance on both Lustre (0.124 $\rightarrow$ 0.165) and GPFS (0.061 $\rightarrow$ 0.374). Directory per process dumps ("priv dirs") greatly improve performance on Intrepid (0.014 $\rightarrow$ 0.374). The restart write rates of 16 GB/s on Intrepid's GPFS file system, 15 GB/s on Dawn's Lustre file system and 57 GB/s on Cielo's Lustre file system are close to the IOR benchmark rate for all 3 file systems. pF3D's rate on Intrepid was the highest of any science code running on that system.

The I/O package is still being tuned on BG/Q systems. The main intrinsic bottleneck for BG/Q file systems (either Lustre or GPFS) is the throughput of the I/O nodes. The aggregate restart write rate increases nearly linearly with the number of processes from 64k to 512k processes. The aggregate rate is the same for 512k and 1024k processes. This could be due to saturation of the file system, but we suspect it is due to a few remaining yorick functions which run in time linear in the number of processes.

5 Code Steering

pF3D has been implemented as a set of compiled C routines connected to the yorick interpreter [7], [8]. Yorick handles I/O, graphics, input decks, and code steering during a run. Yorick has a package called mpy that permits interpreted functions to pass messages between tasks using MPI. mpy is used to coordinate I/O and to implement the time step loop. Using yorick's interpreted language for problem setup and I/O greatly enhances our productivity and gives us a lot of flexibility. Some effort is required to keep the time spent in yorick low when using a million cores.

Each pF3D process reads 20 yorick dot-i files during startup. These are similar to the dot-py files read by a Python-based application. When using the original version of mpy, the file system sees a large number of independent requests to read the same set of 20 files. File systems do not handle this sort of I/O activity well, so startup times for pF3D became long when using more than 100,000 processes. The second version of mpy (mpy2) adds an include function which is a collective operation. The MPI rank zero process reads the input file and broadcasts it to all other processes. The load on the file system is no higher than for a single process run and startup is quick.

The first version of mpy (mpy1) starts parallel functions by having rank zero contact each process directly. mpy2 starts parallel functions using a broadcast tree. There are many calls to functions in the time step loop. Switching to mpy2 reduced the time spent in mpy per time step from 1146 to 308 seconds for a 64k process BG/Q test. This run spent 2183 seconds in physics routines and 640 seconds passing messages, so the time spent in yorick was excessive before these changes were made. The mpy2 time grows to 1442 seconds for 1M processes, which is higher than we would like. There are still a few places where all processes communicate directly with rank zero. Work is underway on additions to mpy2 which will allow those functions to switch to broadcast tree based methods. At that point, the time spent in mpy should remain well under 10% of the total time for runs with 2 million or more processes.

6 NIF Simulations

Figure 5 shows a volume visualization of the light due to Stimulated Raman Scatter (SRS) light in a pF3D simulation of three interacting quads in a NIF experiment. The SRS comes in bursts a few picoseconds (ps) long - a time shorter than the detectors can resolve. At this point in the burst, SRS from individual quads can be seen. Later in a burst, SRS comes from a smooth band including all three quads. Understanding "cooperative backscatter" between multiple quads is a key goal of these simulations.

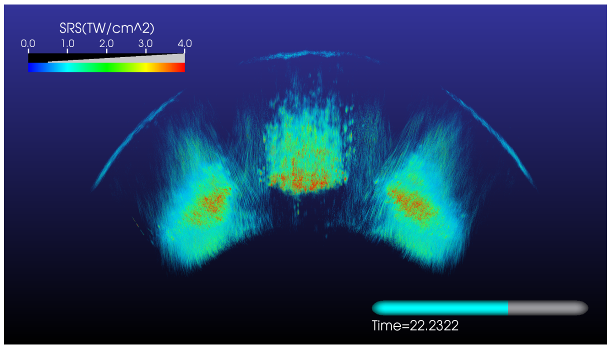

Fig. 5. The SRS comes in bursts. At this time, SRS is generated independently in each quad. Later in the burst, SRS is generated in the region where the quads overlap.

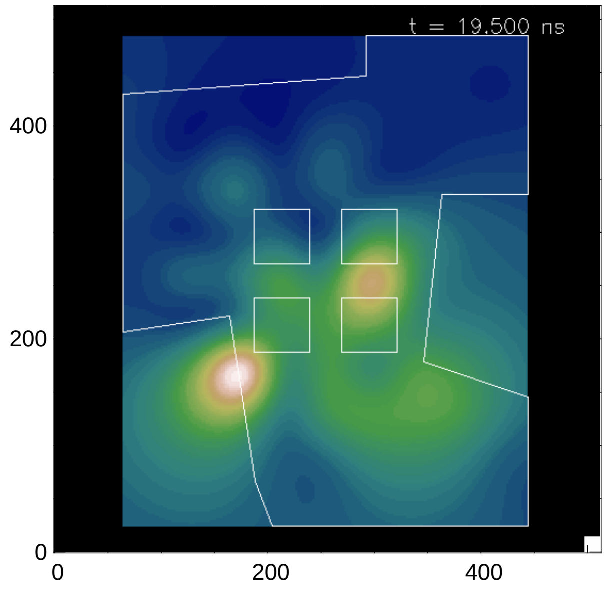

Fig. 6. The NBI detector measured the SRS in a NIF experiment performed on Dec. 4, 2009. This data is from a 30 degree quad at a time of 19.5 ns. The rectangles show the location of the 4 lenses. The other lines show the border of the NBI scatter plate. The SRS covers an area much larger than the lenses. (The image was normalized to use the full color table.)

The Full Aperture Back Scatter (FABS) diagnostic measures the light scattered back through the lenses and was the first backscatter diagnostic on NIF. pF3D simulated SRS from a single quad in a NIF Early Light experiment in 2003 and showed that only 25-35% would pass back through the FABS. This provided an incentive to quickly deploy the Near Backscatter Imager (NBI) [9], a detector that measures the SRS surrounding the lenses.

Figure 6 shows the measured SRS intensity pattern at the NBI. The SRS is spread across an area significantly larger than the lenses. A smooth fit to the data fills in the parts not covered by the scatter plate. The NBI cannot resolve SRS bursts as short as those seen in the pF3D simulations.

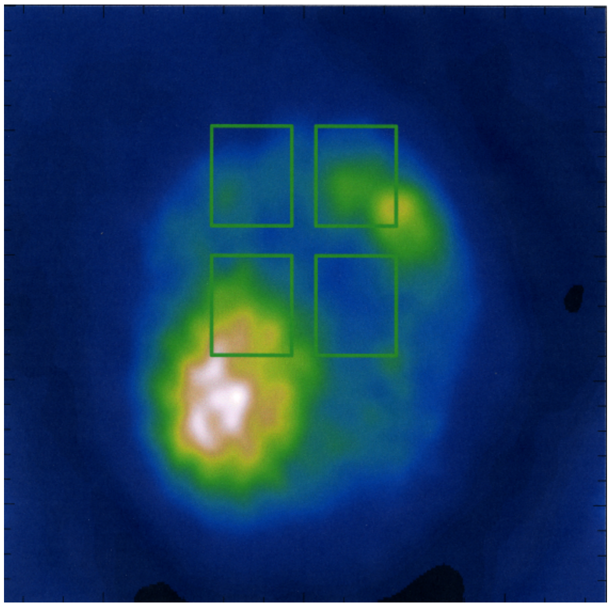

Fig. 7. The SRS simulated by pF3D has bursts with a period of a few ps. This figure shows the SRS pattern at the NBI near the peak of a burst (17.4 ps). The rectangles show the location of the lenses. The NBI cannot resolve bursts as short as those seen in the pF3D simulations. (The image is normalized to use the full color table.)

Figure 7 shows the simulated SRS intensity pattern at the target chamber wall near the peak of an SRS burst. A time-averaged image is similar to this image. In both the NBI data (Figure 6) and the simulations, the brightest spot is below and to the left of the lower left lens. The lens boundary can be used to determine the relative scale of Figures 6 and 7.

The simulated backscatter fraction is 23.7% at 17.39 ps, 14.4% at 18.11 ps, 8.0% at 18.83 ps, and 4.4% at 19.54 ps. Figure 8 shows the simulated SRS intensity pattern at the target chamber wall near the end of an SRS burst. The brightest spots are now at the location of the upper right lens, close to the second brightest spot in the experiment. The location of the brightest spot in the images moves throughout the burst. Volume visualizations show that the location in the plasma where the SRS is generated also moves over the course of a burst. Future work will look at this complex temporal and spatial variability in more detail.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-648736). The authors would like to thank John Moody of LLNL for the NBI data.

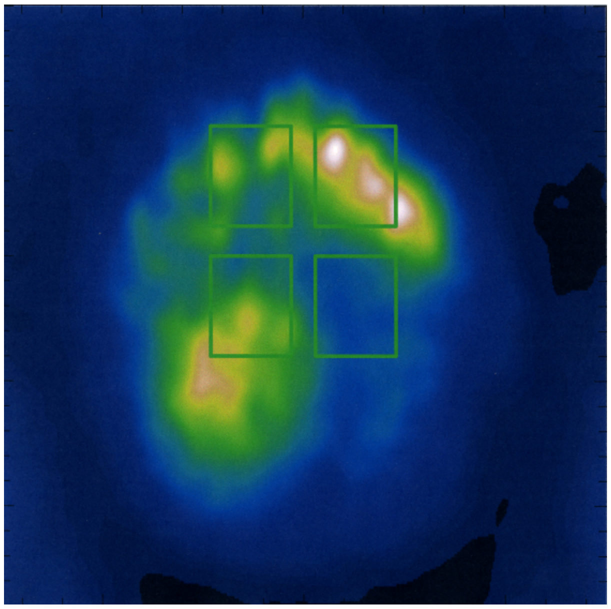

Fig. 8. This figure shows the SRS pattern from pF3D at the target chamber wall near the end of a burst (18.8 ps). The location of the brightest spots has shifted significantly towards the upper right relative to the peak of the burst. (The image is normalized to use the full color table.)

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

Biographies

Steven H. Langer is a physicist at Lawrence Livermore National Laboratory. His research interests include the physics of inertial confinement fusion, high performance computing, and adapting physics simulation codes to new architectures. Langer has a PhD in applied physics from Stanford University. He is a member of the AAS, APS division of plasma physics, and ACM. He can be reached at langer1@llnl.gov.

Abhinav Bhatele is a computer scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. His research interests include performance optimizations through analysis, visualization and tuning and developing algorithms for high-end parallel systems. Bhatele has a PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the ACM and IEEE. He can be reached at bhatele@llnl.gov.

Charles H. Still is an ASC Code Group Leader at Lawrence Livermore National Laboratory. His research interests include the impact of advanced architectures on hydrodynamics and energy transport methods, performance metrics for advanced computer architectures, and laser-plasma interactions. Still has a PhD in mathematics from the University of South Carolina. He is a member of the APS divisions of plasma physics and computational physics. He can be reached at still1@llnl.gov.

References

[1] E. I. Moses, R. N. Boyd, B. A. Remington, C. J. Keane, and R. Al-Ayat. The national ignition facility: Ushering in a new age for high energy density science. *Phys. Plasmas*, 16(041006):1–13, April 2009.

[2] C. H. Still, R. L. Berger, A. B. Langdon, D. E. Hinkel, L. J. Suter, and E. A. Williams. Filamentation and forward brillouin scatter of entire smoothed and aberrated laser beams. *Physics of Plasmas*, 7(5):2023–2032, 2000.

[3] R. L. Berger, B. F. Lasinski, A. B. Langdon, T. B. Kaiser, B. B. Afeyan, B. I. Cohen, C. H. Still, and E. A. Williams. Influence of spatial and temporal laser beam smoothing on stimulated brillouin scattering in filamentary laser light. *Phys. Rev. Lett.*, 75(6):1078–1081, Aug 1995.

[4] D. H. Froula, L. Divol, R. A. London, P. Michel, R. L. Berger, N. B. Meezan, P. Neumayer, J. S. Ross, R. Wallace, and S. H. Glenzer. Pushing the limits of plasma length in inertial-fusion laser-plasma interaction experiments. *Phys. Rev. Lett.*, 100:015002, Jan 2008.

[5] Abhinav Bhatele, Todd Gamblin, Steven H. Langer, Peer-Timo Bremer, Erik W. Draeger, Bernd Hamann, et al. Mapping applications with collectives over sub-communicators on torus networks. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’12. IEEE Computer Society, November 2012.

[6] Abhinav Bhatele, Kathryn Mohror, Steven H. Langer, and Katherine E. Isaacs. There goes the neighborhood: performance degradation due to nearby jobs. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’13. IEEE Computer Society, November 2013. LLNL-CONF-635776.

[7] D. H. Munro. Using the yorick interpreted language. *Computers in Physics*, 9(6):609, 1995.

[8] D. H. Munro. The yorick home page. http://yorick.github.com.

[9] J.D. Moody, P. Datte, K. Krauter, E. Bond, P. A. Michel, S. H. Glenzer, L. Divol, C. Niemann, L. Suter, N. Meezan, B. J. MacGowan, R. Hibbard, R. London, J. Kilkenny, R. Wallace, J. L. Kline, K. Knittel, G. Frieders, B. Golick, G. Ross, K. Widmann, J. Jackson, S. Vernon, and T. Clancy. Backscatter measurements for nif ignition targets. *Rev. Sci. Instrum.*, 81:10D921, Oct 2010.