Dedicated Computing on a YMP/C916

Bruce C. Curtis and *Bruce E. Kelly,* National Energy Research Supercomputer Center (NERSC), Livermore, CA

ABSTRACT: *The Special Parallel Processing (SPP) program at NERSC provides the dedicated capabilities of a 16 processor, 256 MW Cray C90 to scientific researchers from a broad range of disciplines. Started in 1993, the SPP program has supplied over 50,000 CPU-hours of computing for dozens of parallel applications in magnetohydrodynamics, quantum chromodynamics, computational chemistry, fluid dynamics, global climate modelling, and other areas. All of the applications sustain an average concurrency of at least 12 on the 16 CPUs, while a number of them exceed 15 and approach 10 GFlop/s. In this paper we describe the structure of the SPP system, discuss difficulties that arise from the dedicated environment, and present performance results for a number of applications.*

1 Introduction

The National Energy Research Supercomputer Center (NERSC) provides high-performance computing and networking services to the energy research community. The primary computing engines at NERSC (see also **http://www.nersc.gov**) currently are a 256 processor T3D, a 16 processor C90, two Cray-2s (eight and four processors), a four processor YMP-EL, and a 12 processor Silicon Graphics Power Challenge.

1.1 Capacity Mode

Historically, the machines at NERSC have been used in *capacity* mode, i.e., a machine's resources are shared among a large and diverse load of jobs. This mode has the advantage that processor utilization and throughput are high. In order to achieve the desired levels of CPU utilization and throughput, memory size limits are placed on all codes. On the 256 MW C90, the memory limit is 80 MW. Without a memory limit, a large serial code could take so much memory that too few other runnable codes would fit in memory, resulting in substantial CPU idle time. The disadvantage of capacity mode, and the memory limit in particular, is that no code can utilize the fullest capabilities of the system. Each code sees a virtual machine with 80 MW of memory and no more than 8-10 processors as a rule of thumb.

1.2 Capability Mode

In the spring of 1993, NERSC started limited operations with the C90 in *capability* mode, as an experiment. The idea was to provide the full capabilities of the system, i.e., dedicated computing, to selected clients. This program was named Special Parallel Processing, or SPP. The success of the program led to expansion of the resources allocated to SPP in each subsequent year.

This paper describes our experiences with the SPP program. Section 2 presents some of the benefits that our clients have reported. We then describe the organization of the SPP system and some problems that appear in Sections 3 and 4. Performance results for several SPP codes are provided in Section 5, and our plans for future work are outlined in Section 6.

2 Benefits of Dedicated Computing

Computing in capability mode provides a more powerful virtual machine to the user. Memory sizes of up to 240 MW are allowed. SPP codes are required to use all 16 processors and are required to sustain a CPU-time-to-wall-time ratio of at least 12, to insure a respectable level of processor utilization. The 3x increase in memory bounds allows users to compute in new regimes. SPP research areas such as plasma turbulence studies [Carreras et al.; Parker et al.; Dimits et al.] typically solve significantly larger problems and employ finer resolution and more realistic parameters than would have been possible without dedicated computing. Quantum chromodynamics codes [Bernard et al.; Kogut] utilize lattices as large as 36x36x36x95, which also would not have been possible without dedicated computing. Thus, the SPP program contributes to the scope and physical realism of the science produced by NERSC's user community.

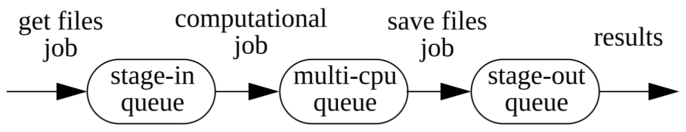
Faster turnaround is another benefit. Long-running jobs, especially those with large memory requirements, take a very long time when executed in capacity mode. A 100-hour 80 MW job typically takes several weeks to complete if it has to compete with our normal workload. In dedicated mode, however, the same job, or even one needing 240 MW of memory, can complete overnight. An ab initio molecular dynamics project [Kendall], for example, completed in early 1995, after two years of SPP access. It would have taken several more years without dedicated resources.

Finally, the SPP program yields two beneficial side effects. One is that it helps prepare NERSC's client base for future parallel systems. In developing efficient codes for dedicated computing on the C90, our users address several of the problems to be faced on massively parallel processors. Besides the educational leverage, there are tangible results, as several SPP codes are written with message passing and run efficiently on MPPs. The other side effect is that C90 utilization during non-SPP hours is enhanced by the presence of well-tuned parallel codes. The users who run large simulations during SPP hours run smaller ones with the same parallel code during normal production hours. With a sufficient number of parallel codes in the workload, a C90 or similar system will rarely suffer high idle time. Approximately 25% of the cycles on our C90 are consumed by parallel codes, due in large part to SPP codes.

3 Structure of the SPP System

3.1 Network Queuing System Queues

The SPP system is divided into three phases, each with its own queue. The first phase is the file stage in process, the next phase is the computational part, and the final phase is for file stage out. This is shown in the following diagram.



The user initially submits a job to the stage-in queue,which makes sure all the needed data for the computation is available on local disks. The stage-in queue is currently started three hours before the beginning of an SPP run. During this phase files are retrieved from a local file storage system (CFS), copied from a remote machine or server, and migrated files are brought back on disk. The stage-in queue needs a large disk space limit, but the memory and CPU requirements are not high on this queue. The last thing the stage in job should do is submit the computational job to the multi-cpu queue.

The multi-cpu queue is where the real work gets done. All the resources available are given to each job as it is started. Each job may use 240 MW of memory, 144 hours of CPU time, 16 GB of disk space, and 100 MW of SDS. They must use all 16 processors. Thus, the total CPU time requested can be up to 518,400 seconds, or six CPU-days. The last thing the computational job may do is submit a stage-out job.

This last phase is optional. If the job does not have a large disk requirement, the results might be left on local disks. Otherwise, this queue is very much like the stage-in queue. Jobs should use this phase to store their results.

The stage-in and stage-out queues run while regular NQS production is running. The multi-cpu queue is the only one where the entire machine is dedicated to the job. The stage-in queue is the only queue enabled so users can submit jobs to it. The other two queues are only enabled when necessary. There are also group restrictions on who can submit a job.

3.2 SPP Scripts

SPP is controlled by an AT job that resubmits itself at various time of the day. This script calls on several other scripts to perform all the functions necessary to switch between production NQS and SPP shots. The time at which the SPP script runs is determined by a configuration file.

First the script reads the configuration file, which contains the start and stop times of each SPP shot. This determines which phase of SPP is underway, if any, and when to resubmit the AT script.

If SPP is not scheduled to be run for the rest of the day, then the script is resubmitted for the next day. If there is to be a shot later in the day, the script will resubmit itself when it is time for the stage in phase to begin. In addition, NQS is checked to see if it is in production or SPP mode. If it is still running an SPP shot, then NQS is switched to production mode. Either of these stages could be the normal ending mode for a SPP run. Because this is an AT job, the script could run at any time. So it has to handle any of the possible state transitions.

There are two other possibilities. It could be time to stage in jobs or to run jobs. During the stage in phase, the jobs that are going to run are scheduled. Each scheduled stage-in job is started, one at a time, until the schedule is complete. This guarantees the order in which jobs enter the multi-cpu queue.

The following table indicates when each queue is enabled, disabled, started, and stopped.

Table 1. When Queues are Enabled and Started.

Queue	Production	Begin Stage in	End Stage in	SPP Run
stage-in	enabled stopped	enabled started	enabled stopped	enabled stopped
multi-cpu	disabled stopped	enabled stopped	enabled stopped	enabled started
stage-out	disabled started	disabled started	disabled started	enabled stopped

The scheduling is done by a separate script. The scheduler simply returns the request IDs of the jobs that are to be run. This means the scheduler may be replaced with ease. It could be written as a shell script, a perl script, or a C program, for example.

Currently, the script either does manual or automatic scheduling. For manual scheduling, the list of request IDs is read from a file, which has been entered by NERSC personnel. When scheduling automatically, a user configuration file is examined. The user configuration file has an entry for each user allowed to use the SPP queues. Properties of a user's codes are included in each entry. The scheduler uses these properties to choose jobs to run so as to make the best use of the wall time allocated for the SPP shot.

3.3 SPP Features

A 7 GB file system has been set aside for SPP users. This is known as the fast file system because it resides on a set of disks that are two to four times as fast as our other large disks. The speed of the disks is more important to dedicated jobs than to non-dedicated jobs. If an SPP user wishes to use this file system, a flag is set in the user configuration file. The scheduler compares the disk space request of the job to the space available in the fast file system. If there is enough room the job can be scheduled. Another field in the configuration file is the graceful shutdown bit. If this bit is set it means the user has inserted code that can detect the end of the SPP run and shut down on time. One of these jobs is usually scheduled as the last job to run.

The time limit for the stage-in queue is given an artificial meaning. It does not reflect stage-in CPU time; it is an estimate of the wall time for the dedicated portion. This value is used to determine how many jobs will be scheduled for any SPP run. It is not desirable to stage in more jobs than will run during the shot.

The SPP shots are not entirely dedicated. All NQS jobs are suspended, but interactive use is allowed. This could potentially interfere with the SPP jobs. Several mechanisms have been tried to insulate the SPP jobs from interactive interference. For example, the fairshare scheduler was enabled at the beginning of a shot and turned off again at the end. The fairshare scheduler was not designed to be used this way and resulted in increased interference. The current scheme is that all background jobs are suspended, as well as any foreground jobs that have not written to the terminal in a certain amount of time.

Finally, the nschedv parameters are adjusted at the beginning of the dedicated run to give the SPP code every advantage.

3.4 SPP Runs

SPP runs are divided into production and test runs. The test runs have the same features as the production runs and are used for short runs and testing. Test runs are done for one or two hours in the early morning and utilize their own set of queues. The production runs are on Friday, Saturday, and Sunday nights and last up to nine hours. This totals 32 hours of SPP time per week.

4 Difficulties Arising from Dedicated Mode

Our efforts to provide a useful dedicated environment on the C90 have revealed a number of problems. In this section we discuss a few of these.

4.1 User Perspective

4.1.1 Code Development

The fact that resources available during SPP runs (principally memory) are greater than those normally available is a disadvantage when debugging. A code larger than 80 MW (the limit for normal production) cannot be run under the debugger. Thus, the user either debugs via print statements or tries to duplicate the bug with a smaller version of the code, often in vain. SPP runs are batch jobs only, executed in the middle of the night. Turnaround time is overnight, but can be longer depending on the jobs in the SPP queue. As an environment for code development, therefore, SPP is markedly aggravating. Knowing this, and choosing only stable, well-developed codes for SPP is only a partial solution. Sometimes anomalous behavior only appears with the larger problems or finer resolutions that are not possible except through SPP runs.

4.1.2 Code Optimization

Optimization in dedicated mode requires a change in the mindset of most users. For non-dedicated computing, the user typically attempts to minimize the CPU time (NERSC charges for CPU time only). For dedicated mode, the user should instead minimize the wall time. Therefore everything should be parallelized, even if poorly. When a user struggles to improve the parallel performance of an SPP code, it is often the case that a serial section has been overlooked because of its relative unimportance. Discovering such serial regions is nearly always by inspection, since the available tools for optimizing parallel performance (e.g., ATEXPERT) do not reveal the serial sections in the source code.

Tuning the major parallel regions is more exacting in dedicated mode than normal production. The two primary issues are load balancing and the granularity of the parallelism. Scientific codes often have an abundance of parallelism expressed at different levels. Thus, there can be several degrees of freedom in structuring the parallel processing. The choice of the level of parallelism to exploit, while important in non-dedicated mode, is most critical in dedicated mode. A too fine grained parallel region may perform at an adequate MFlop/s rate in normal production, but run at half that rate in dedicated mode, due to a large increase in CPUs waiting for access to the shared registers. A too coarse grained parallel region may cause idle CPUs in dedicated mode, since many processors may have a long wait for the last CPU to finish its work. This form of load imbalance is not so damaging in non-dedicated mode because the inactive CPUs would be released to work on other codes. While a parallel code developed for normal production may perform poorly in dedicated mode, fortunately the reverse is not true. A well-tuned SPP code performs well in normal production.

NERSC's SPP mode adds another wrinkle to the optimization problem. The environment is not always dedicated, because interactive computing is not disabled. Over the course of months, this interactive interference consumes an average of less than one half of one CPU, but a given SPP job can suffer several "missing" CPUs. It can be ruinous, therefore, for SPP codes to optimize for 16 processors. If the work is divided into 16 chunks, and one CPU is missing, the parallel performance doesn't just drop to 15, it drops to eight!

Experience on the C90 provides a rule of thumb for optimizing parallel regions for SPP mode. The amount of time one chunk of work takes should be 3,000 clocks or more. The number of chunks in the region should be a at least a moderate multiple of 16. Sixty-four chunks will suffice, but more will improve the load balance, especially when there is interactive interference.

I/o must be optimized in order to utilize dedicated resources efficiently. Processors sit idle in dedicated mode when synchronous i/o is performed, while in non-dedicated mode they would be applied to other jobs. Many of the SPP researchers depend on visualization to analyze their simulations, resulting in considerable i/o requirements. I/o must be pruned as much as possible, and the fastest available devices must be employed for the remaining data. Asynchronous i/o should be utilized as much as possible.

These and other optimization issues were the subject of a workshop for SPP users given by NERSC in December 1994. Copies of the viewgraphs from the workshop are available from the authors.

4.1.3 Job Monitoring

It follows from the structure of the SPP system that one user's job can be affected by another user's job. For example, if several jobs are scheduled in a particular shot, the later jobs depend to a certain extent on the earlier jobs. If the earlier jobs take more wall time than expected, there may be no time left for the later jobs. When that happens, any jobs that did not start during the shot must be deleted and resubmitted, to avoid having their files purged or migrated. Another case that has occurred is when a job gets hung for some reason in the stage-in queue (say, repeated attempts to read a file that doesn't exist), the rest of the jobs scheduled for that shot are also stuck and cannot submit their jobs for dedicated computing.

Consequently, users must monitor the progress of their SPP jobs more closely than their regular NQS jobs.

4.2 System Administration

Since dedicated computing is not supported by UNICOS, the implementation is locally developed and maintained. Some of the system administration issues are discussed below.

4.2.1 Suspending Jobs

There is a basic system administration problem with NQS. How big should swap and spool be made so that there is sufficient space at the highest usage? In the case of SPP the only concern is swap space. Swap space has to be large enough to accept all running jobs at one time. The processes from normal production are out on disk when SPP starts. In addition, 240 MW to hold the swap image of the SPP job should be available.

A dedicated job may not finish in the time slot and is suspended at the end of the SPP shot. The suspended process and all its files sit unused until the next shot, and even worse, the files may migrate off disk. In order to minimize the resource congestion due to suspended SPP codes, NERSC has written simple library routines to promote *graceful shutdown.* The user calls a routine to inform the library how much warning time is needed before the end of the shot and to provide a variable for the library to update. The user code polls the variable at strategic intervals. At the requested time, the library sets the variable and the user code knows to save the state of the computation and exit.

Another problem is that suspending NQS jobs does not always work. This can leave several processes running along with the SPP job. The reasons for this are not all known, but one instance is that the suspend call can time out waiting for the process to complete its i/o. The system will not suspend the process until the i/o completes. Thus, we must periodically check for processes that did not suspend and try again.

4.2.2 Poor Utilization

Monitoring each shot is essential. User errors during dedicated time can waste resources at an alarming rate. Interactive interference could lead to poor utilization by causing the SPP job to swap to disk, for instance.

An AT job runs every ten minutes during dedicated time to try to diagnose and remedy periods of poor utilization. The difficulties in automatically determining the causes of poor utilization currently require operator intervention in most cases.

5 Performance Results

The floating-point performance and parallel performance of several SPP projects are shown in Table 2. Below, five research efforts are highlighted, listed by their project titles, principal investigator, and affiliation.

5.1 Lattice Boltzmann Approach to Turbulence in Divertor Plasmas (Vahala, William and Mary University)

Neutral fluid turbulence effects in magnetically confined plasmas are studied using the relatively new Lattice Boltzmann approach. The TLBE (thermal lattice Boltzmann equation) code solves a linearized Boltzmann equation on a lattice.

The most computationally intensive section of this 2D code (soon to be extended to 3D) is ideal for the C90. The outer loop, corresponding to the X direction of the mesh, is parallelized via autotasking, providing a well-balanced 512 chunks of work or more. The inner loop, the Y direction, also has 512 or more iterations, providing plenty of work per parallel chunk. The inner loop also contains an unrolled loop of length six, characterized by multiply operations that chain into add operations.

The advection subroutine takes only about one percent of the run time, but it consists of 12 serial loops. Fortunately, the loops are independent. The largest of these were split, so that the resulting 16 loops could be executed in parallel.

Before optimization, this code ran at 50 Mflop/s on a single CPU. The low rate was due to the vector length of size. After optimization, it has run at 623 Mflop/s per CPU, and attains an average of 15.52 CPUs (CPU-time-to-wall-time ratio), or 9.7

Gflop/s overall. That represents two-thirds of the peak performance of the C90.

5.2 Direct Numerical Simulation of Natural Transition to Turbulence in a Waveguide Mixing Layer (Bell, Lawrence Livermore National Lab)

Spatially developing mixing layers are simulated by selectively refining portions of the computation region. This Adaptive Mesh Refinement (AMR) method was developed as part of the Grand Challenge Project in Computational Combustion and Fluid Dynamics.

The parallelization of this code is accomplished by allowing each processor to compute asynchronously (implemented with macrotasking) on independent grid patches. These grid patches comprise a large range of shapes and sizes, thus it would be inefficient to process them in lock-step.

The i/o requirements, due to the writing of large files to generate movies, requires special attention. For each 2.5 hours of wall time, 10 GB of data is written. The only file system at NERSC with sufficient free space uses a DD-42 disk, which is relatively slow. The solution is to buffer data in main memory, then write it to SDS. When an SDS buffer fills, it is flushed asynchronously via the “back door” channel to disk. Other i/o, such as standard output, is directed to a different file system, so that it won’t block while the huge SDS segment transfers to the DD-42.

While the numerically intensive core, written in Fortran, runs at 500 Mflop/s per CPU, the overall code, mostly written in C++, runs at 382 Mflop/s and sustains an average of 13.85 CPUs, including i/o. The code uses 110 MW of main memory and 100 MW of SDS. A frame from a movie of the simulation is shown in Figure 1.

5.3 Gyrokinetic Simulation of Tokamak Turbulence (Lee, Princeton Plasma Physics Lab)

The research under way here is to directly simulate magnetically confined plasmas in tokamaks (torus-shaped devices). This will provide better understanding of the turbulent transport phenomena observed experimentally. A 3D (5D in phase space) toroidal gyrokinetic particle simulation code has been developed. The code, parallelized with autotasking, sustains a speedup of 14.5 and performs at a rate of 3.7 Gflop/s on a typical run. Runs generally use a 128x128x64 grid with 4 million particles, requiring 115 MW of memory and two to three hours of wall time. Much of the run time is concentrated in two areas: particle pushing, where each particle’s position and velocity are advanced; and the accumulation of the ion density at grid points (gather/scatter).

Figure 2 shows a time sequence of the exponential growth of a coherent linear mode structure in the poloidal cross section. In this visualization it is apparent that the radial mode structure elongates during nonlinear saturation and then rips apart.

Table 2. Performance of eight SPP Projects on the 16-processor Cray YMP/C916

Principal Investigator and Affiliation	Area of Research	Floating Point Performance per CPU (Mflop/s)	Parallel Performance (CPU time/wall time)	Aggregate Floating Point Performance (Gflop/s)
Vahala William and Mary	Plasma Physics	623	15.5	9.7
Kogut Univ. Ill	Quantum Chromodynamics	584	15.6	9.1
Hack NCAR	Global Climate Model	440	14.6	6.4
Soni Brookhaven Nat Lab	Quantum Chromodynamics	385	15.9	6.1
Bell Lawrence Livermore	Fluid Dynamics	382	13.9	5.3
Cohen Lawrence Livermore	Plasma Physics	378	13.8	5.2
Leboeuf Oak Ridge Nat Lab	Plasma Physics	358	14.0	5.0
K. Lee Lawrence Berkeley	Electromagnetic Imaging	327	15.1	4.9

Agreement between experimental measurements and these large-scale nonlinear simulations is very encouraging.

5.4 Hadronic Matrix Elements of Heavy-Light Mesons (Soni, Brookhaven National Lab)

This quantum chromodynamics research project is to determine weak matrix elements of *B*-mesons, which are crucial inputs for deducing irreducible parameters of the Standard Model of Elementary Particle Interactions.

Two codes are used, one for configuration generation, and the other for quark matrix inversion. The configuration generation code runs at 385 Mflop/s and has attained as much as 15.87 CPUs for a six wall hour run when there was negligible interactive interference. An example from the matrix inversion code follows. This nest of loops has a similar structure to a dozen other nests, which together represent about 80% of the run time.

```
integer c,y,z,t,zp1
complex i, ctmp1, ctmp2
complex v(4,3,0:ns-1,0:ns-1,0:nt-1)
complex mv(4,3,0:ns-1,0:ns-1,0:nt-1)
complex u(3,3,0:ns-1,0:ns-1,0:nt-1)
do 70 z=0,ns-1
zp1 = mod(z+1,ns)
do 70 y=0,ns-1
do 70 c=1,3
do 70 t=0,nt-1
ctmp1 =
u(c,1,y,z,t)*(v(1,1,y,zp1,t)+i*v(3,1,y,zp1,t
))+u(c,2,y,z,t)*(v(1,2,y,zp1,t)+i*v(3,2,y,zp
1,t))+u(c,3,y,z,t)*(v(1,3,y,zp1,t)+i*v(3,3,y
,zp1,t))
mv(1,c,y,z,t) = mv(1,c,y,z,t) + ctmp1
mv(3,c,y,z,t) = mv(3,c,y,z,t) - i*ctmp1
ctmp2 =
u(c,1,y,z,t)*(v(2,1,y,zp1,t)+i*v(4,1,y,zp1,t
))+u(c,2,y,z,t)*(v(2,2,y,zp1,t)+i*v(4,2,y,zp
1,t))+u(c,3,y,z,t)*(v(2,3,y,zp1,t)+i*v(4,3,y
,zp1,t))
mv(2,c,y,z,t) = mv(2,c,y,z,t) + ctmp2
mv(4,c,y,z,t) = mv(4,c,y,z,t) + i*ctmp2
70 continue
```

The inner loop, unfortunately, runs over the last dimension in the arrays. The lattice sizes to be considered have factors of two in NS (e.g. NS=16, NT=39 for a small one, and NS=36, NT=95 for a large one), thus the stride is a large power of two and the vector length is not advantageous. The outer loops, while all potentially parallel, do not provide a desirable number of chunks of parallel work. So, while there is abundant parallelism contained in the nest, exploiting it efficiently takes some work.

One solution is to reorder the array dimensions so that the t dimension is first; compound the t and y loops to form a longer inner loop; and compound the c and z loops to form a longer outer loop to be parallelized:

```
complex v(0:nt-1,0:ns-1,0:ns-1,4,3)
complex mv(0:nt-1,0:ns-1,0:ns-1,4,3)
complex u(0:nt-1,0:ns-1,0:ns-1,3,3)
do 70 cz = 0,3*ns-1
z = cz/3
c = mod(cz,3) + 1
zp1 = mod(z+1,ns)
do 70 ty = 0,ns*nt-1
ctmp1 =
u(ty,0,z,c,1)*(v(ty,0,zp1,1,1)+i*v(ty,0,zp1,
3,1))+u(ty,0,z,c,2)*(v(ty,0,zp1,1,2)+i*v(ty,
0,zp1,3,2))+u(ty,0,z,c,3)*(v(ty,0,zp1,1,3)+i
*v(ty,0,zp1,3,3))
mv(ty,0,z,1,c) = mv(ty,0,z,1,c) + ctmp1
mv(ty,0,z,3,c) = mv(ty,0,z,3,c) - i*ctmp1
ctmp2 =
u(ty,0,z,c,1)*(v(ty,0,zp1,2,1)+i*v(ty,0,zp1,
4,1))+u(ty,0,z,c,2)*(v(ty,0,zp1,2,2)+i*v(ty,
0,zp1,4,2))+u(ty,0,z,c,3)*(v(ty,0,zp1,2,3)+i
*v(ty,0,zp1,4,3))
mv(ty,0,z,2,c) = mv(ty,0,z,2,c) + ctmp2
mv(ty,0,z,4,c) = mv(ty,0,z,4,c) +i*ctmp2
70 continue
```

The original nest runs at 55 Mflop/s (for the NS=16, NT=39 case) and 7.54 CPUs. The transformed nest runs at 388 Mflop/s and achieves 15.83 CPUs for the same case. The matrix inversion code overall sustains 14.84 CPUs at 335 Mflop/s, or 5 Gflop/s.

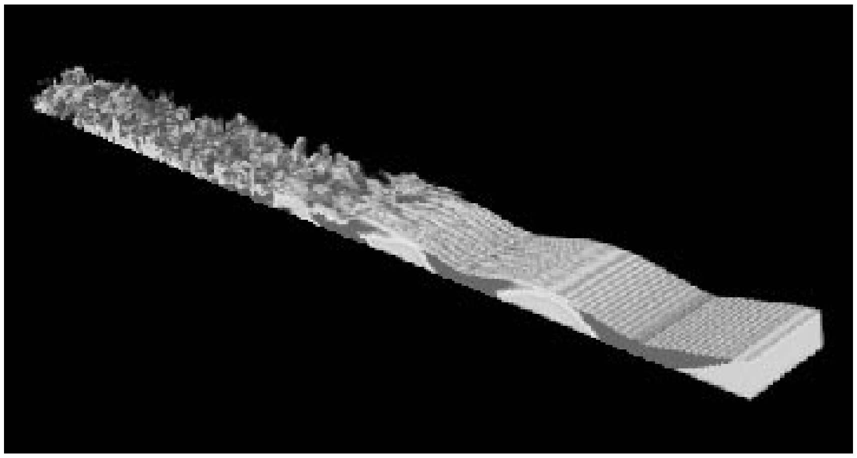


Figure 1: A frame of a movie of the direct numerical simulation of natural transition to turbulence in a waveguide mixing layer.

5.5 High Resolution Gyro-Landau Fluid Plasma Turbulence Calculations at the Core of Tokamaks (Leboeuf, Oak Ridge National Lab)

KITE, a nonlinear plasma fluid turbulence code, is used to study resistive pressure gradient driven turbulence in magnetic fusion devices. Experimentally observed transitions from a low mode of plasma confinement to a high mode of confinement has been accounted for by theory. It is theorized that the confinement is improved by the nonlinear suppression of turbulence by sheared poloidal flows. KITE has been used to verify the theory [Lynch et al.] using realistic equilibrium and dissipation profiles and parameters corresponding to the Advanced Toroidal Facility torsatron. Figure 3 shows the transition from a low mode of confinement (left half) to a high mode of confinement (right half) for ion density (top) and potential (bottom).

The high-resolution SPP runs requires 192 MW of memory, and averages 14 processors running at 358 Mflop/s, for an aggregate rate of 5 Gflop/s. The runs are configured to take 3.5 wall hours each, with the state of the calculation written to disk to be resumed in the next run. Since the code detects the end of the SPP shot and writes a restart file, it always completes during a shot. Furthermore the stage-out portion of the job submits the stage-in script for the next run, thus the calculation is self-submitting. KITE frequently consumes 150 CPU hours per weekend.

KITE employs Fourier expansion in the poloidal (the short way around a torus) and toroidal (the long way around) directions and a finite difference grid in the radial direction. Parallelization of the most time-consuming portions is accomplished by autotasking the outer loops over the number of modes, up to 4605 modes. The vectorized inner loops range over the number of radial grid points.

6 Future Plans

More work is needed to improve the robustness of the SPP system. We plan to strengthen the mechanism by which running SPP codes are monitored. If it can be automatically determined that an SPP code is failing (e.g. idling CPUs, or continuously generating exceptions) and hold the job, allowing the next job run, then valuable resources will be conserved. The current SPP monitor only looks for the simplest failures; operator intervention is required in most cases.

We also plan to detect instances where a job has been prevented from running by other jobs. A message would be sent to the victim so that he will know to resubmit the job.

The job scheduler can also be improved to make more complete utilization of the wall time allocated for SPP. One promising improvement would be for the job monitor to invoke the scheduler when a failing code has been detected. The scheduler could look for an appropriate replacement for the held job, such as one that doesn't need a stage-in period.

NERSC has one MPP system and is procuring more, but also is very interested in SMP systems. The SPP program will likely be carried forward on SMP systems should they become a reality at NERSC.

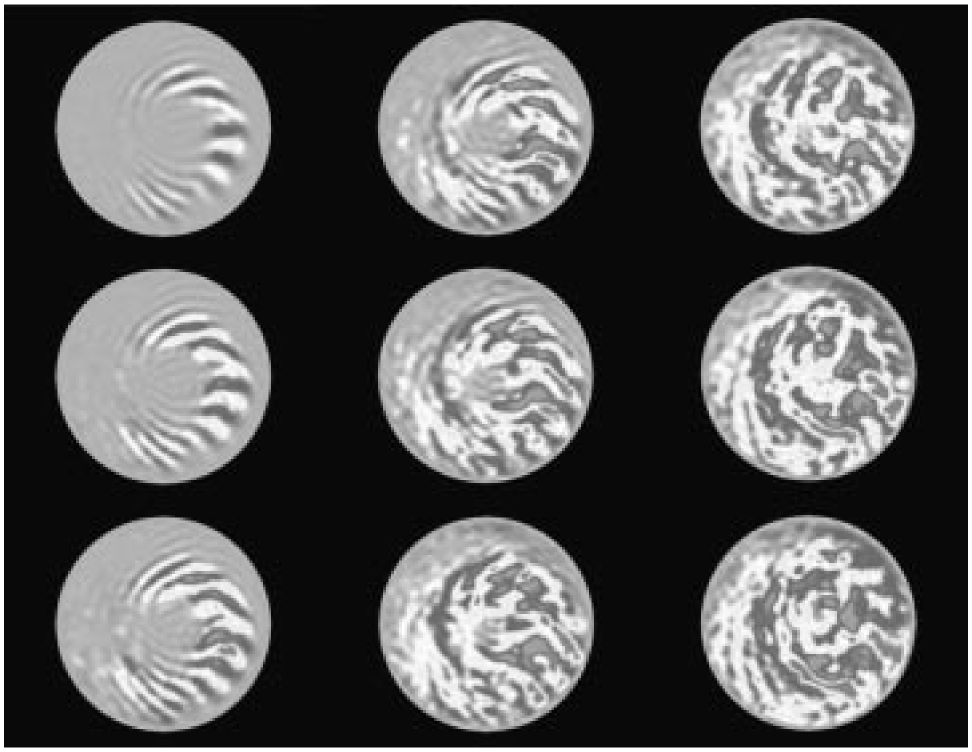


Figure 2: A time sequence of the exponential growth of a coherent linear mode structure in the poloidal cross section.

Acknowledgments

The SPP program could not have been successful without the support of a number of people. We would like to acknowledge the contributions of Clark Streeter, formerly of NERSC, our supervisor Morris Jette, Steve Luzmoor of CRI, and Colin Henry, formerly of CRI. The full support of NERSC management, especially Bruce Griffing, as well as that of Tom Kitchens of DOE has been greatly appreciated. We thank the NERSC consulting group for their help, and finally, many thanks to all the SPP users for their hard work and patience.

References

C. Bernard, J. Labrenz, and A. Soni, "A Lattice Computation of the Decay Constants of *B* and *D* Mesons", Phys. Rev. D 49, 2536 (1994).

B.A. Carreras, V.E. Lynch, L. Garcia, and P.H. Diamond, "Dynamics of Second-order Phase Transitions in Resistive Pressure-gradient-driven Turbulence", Phys. Plasmas 2, 2744 (1995).

A.M. Dimits, T.J. Williams, J.A. Byers, and B.I. Cohen, "Implications of Gyrokinetic Simulations on the Role of ITG Turbulence in Tokamak Transport", Proc. 1995 International Sherwood Fusion Theory Conference.

J.A. Greenough, W.Y. Crutchfield, and C.A. Rendleman, "Numerical Simulation of a Wave-Guide Mixing Layer on a Cray C90", AIAA Paper 95-2174.

J. Kogut "Quenched QCD at Finite Baryon Density", Phys. Rev. D51, 1282 (1995)

V. E. Lynch, B.A. Carreras, and J.N. Leboeuf, "The Performance of Fluid Codes on Parallel Computers", Bull. Am. Phys. Soc. 39, 1725 (1994).

V.E. Lynch, B.A. Carreras, J.N. Leboeuf, and D.E. Newman, "Numerical Calculations of Improved Modes of Confinement in Magnetic Fusion Devices", NERSC Buffer 19, 8 (1995).

S.E. Parker, J.C. Cummings, W.W. Lee, R. A. Santoro, "Gyrokinetic Simulation of Tokamak Plasma Turbulence", NERSC Buffer 19, 5 (1995).

G. Vahala, P. Pavlo, L. Vahala, H. Chen, "Effects of Velocity Shear on a Strong Temperature Gradient - a Lattice Boltzmann Approach", Phys. Lett. A 202, 376 (1995).

G. Xie, K.H. Lee, and J. Li, "A New Parallel 3D Numerical Modelling of the Electromagnetic Fields", to be presented at the SEG annual meeting Oct. 1995.

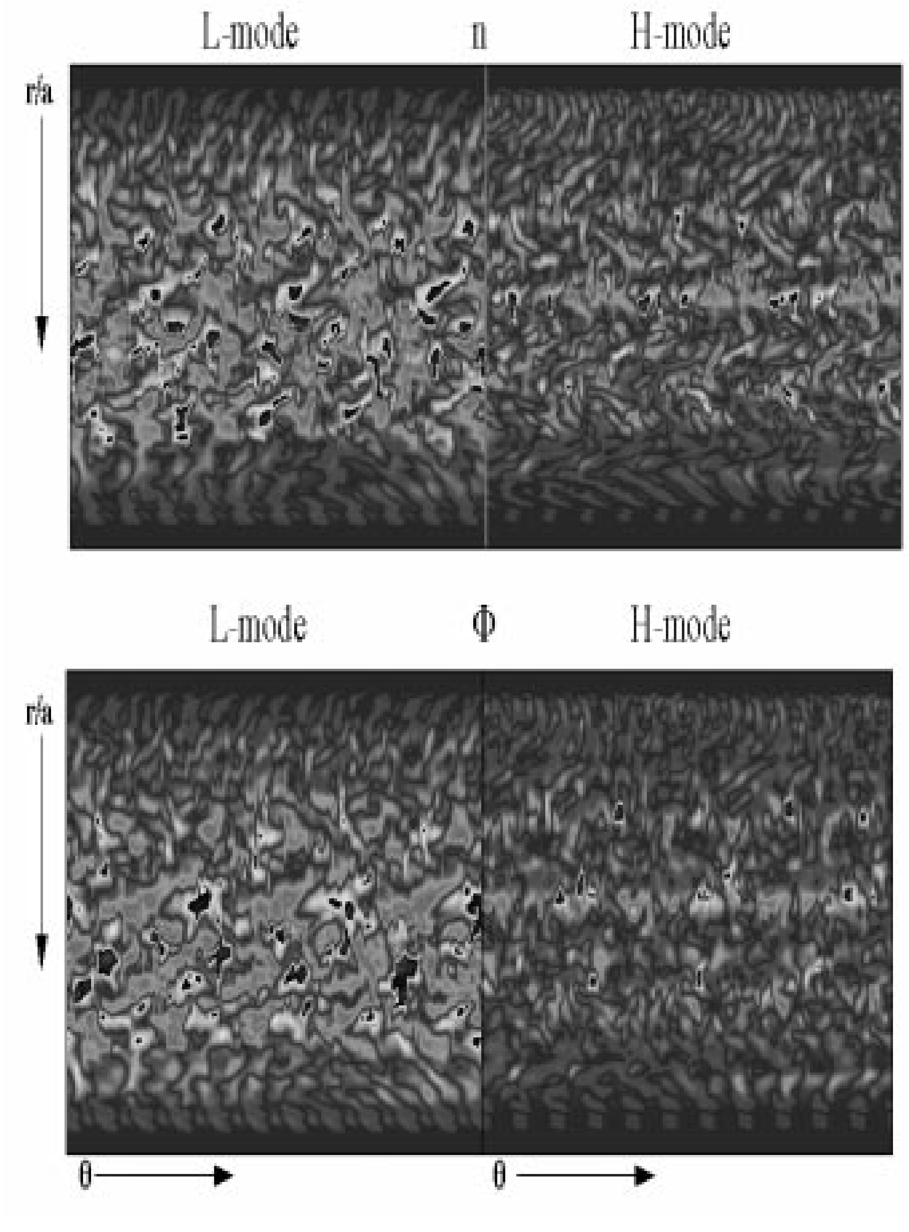


Figure 3: The transition from a low mode of confinement (left) to a high mode of confinement (right) for ion density (top) and potential (bottom).