Code Baseline Completed

Numerical Simulations for Active Tectonic Processes: Increasing Interoperability and Performance

JPL Task Plan No. 83-6791
Milestone E:

E - Code Improvement	***Code Baselines, Scaling Analysis, Performance Analysis completed (generate scaling curves for codes which are already parallel, baseline serial performance for codes which are not already parallel) Documented source code made publicly available via the Web.*** serial code GeoFEST- includes serial iterative solver. 50,000 elements 1000 timesteps - serial implementation serial code Virtual California with N=215 segments for 10,000 time steps, serial implementation on 1 GHz workstation serial code PARK with 15,000 elements for 500 time steps - uses parallel multipole library, but serial main routine.	***07/30/ 02***

The three baseline codes, GeoFEST, Virtual California and PARK, for Milestone E are described in the following sections, including their problem descriptions, scientific significance, simulation and code details, and related references.

The top-level web site for the Active Tectonics task is at http://www-aig.jpl.nasa.gov/public/dus/quakesim/index.html. Our intention is to change this to http://quakesim.jpl.nasa.gov in the coming months. Source code for the three codes may be found at http://www-aig.jpl.nasa.gov/public/dus/quakesim/download.html. Files required for the baseline cases may be found at http://www-aig.jpl.nasa.gov/public/dus/quakesim/milestones.html.

1. GeoFEST Baseline

Synopsis: mesh with over 50,000 tetrahedral elements (55,369) was run on Solaris workstation beginning at system date Tue Jul 30 09:38:55 PDT 2002 and concluding at system date Tue Jul 30 23:22:37 PDT 2002 (total duration 13:43:42 HH:MM:SS) after 1000 time steps each representing 0.5 year of physical time. Each step used an iterative conjugate gradient solver. These steps required iteration counts ranging from 58 to 130, with an average of 85.45.

Input and output files are posted in GeoFEST.MS_E.tar (described below).

Images of the surface vertical deformation at times 0, 100 and 500 years (and the differences) are included as *.gif files posted at http://www-aig.jpl.nasa.gov/public/dus/quakesim/gfgraphics.html. .

1.1 Scientific Significance:

This represents a type of simulation used to compare surface geodetic data (particularly interferometric synthetic aperture radar (InSAR), global positioning system, laser strain meter, and leveling (surveying) data) with probable features of the post-seismic phase of the earthquake cycle. These features include linear and nonlinear rheologic structure, fault friction, interaction with nearby faults, and poroelastic effects (groundwater responding to stress). This simulation includes only linear viscoelatic relaxation, but the finite element methodology can be extended to include the other features.

In the context of current research, this simulation constitutes a validation that layered viscoelastic response to an earthquake can be faithfully simulated with a mesh of highly variable density. The mesh size varies from 30 km at the box boundary to 1.5 km at the edges of the fault. Validation of this kind of adaptive mesh size has been identified as a priority need by the Southern California Earthquake Center Crustal Deformation Group at a recent Deformation Software Workshop at Caltech. Here we take the validation only so far as the level of physical reasonableness; comparisons to other simulations will follow.

The domain is a box 240 x 240 x 100 km in size, centered at the top of a rectangular fault plane representing the blind thrust of the Northridge earthquake event (1994, M=6.7). The parameters of the slip event are the effective uniform-slip rectangular planar fault proposed in Wald et al. (1996), tabulated here:

dip(deg)	strike(deg)	slip(m)	rake(deg)	length(km)	width(km)	depth(km)
40.0	122	1.3	101	14.0	21.0	19.5

The box is vertically stratified into three layers as follows (5, 30, and 65 km thick):

layer	mu	lambda	eta	n	comment
upper	17.0	17.0	17.0	1.0	1 yr Maxwell time
mid-crust	35.0	35.0	0.0	0.0	elastic
mantle	70.0	70.0	7000.0	1.0	100 yr Maxwell time

1.2 Simulation details:

The code is geofest v4.2 (cvs id's shown at end of report) as of 7/30/2002.

The code is compiled on "seismo", with characteristics as follows (obtained with uname -a, psrinfo -v) SunOS seismo 5.6 Generic_105181-30 sun4u sparc The sparc processor operates at 450 MHz, and has a sparc floating point processor.

Compilation flags are CFLAGS = -fast –native with "-lm" on the load line to include standard math library.

Job was submitted interactively as date > nr8ex.log ; time nohup ../geofest nr8ex.dat >> & nr8ex.log; date >> nr8ex.log &

An interactive guide to the input file format for a simple case may be found at http://downloads.openchannelsoftware.org/GeoFEST/input/geo_fest_example1.html.

The input file nr8ex.dat has the following characteristics:
10370 nodes
55369 tetrahedral finite elements
29121 equations (displacements are pinned at sides and bottom of domain).
3816 split node records describe the fault slip.

Shape functions are computed once and saved for reuse.
Solver is iterative conjugate gradient.
There is a single fault slip event, at t=0.

Viscoelastic relaxation is simulated for 500 years, with time steps of 0.5 years (hence 1000 time steps).

All displacements and stresses are written to a file at eight times,
2.
10.0
50.0
100.
200.
300.
400.
500.

No checkpointing is used.

CVS tracking data for source code (from "what geofest" command):

CVS id $Id: generat.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2
CVS id $Id: inphase.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2
CVS id $Id: stiff.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2
CVS id $Id: strain.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2
CVS id $Id: solver.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2
CVS id $Id: utility.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2
CVS id $Id: main.c,v 1.1.1.1 2002/07/12 19:05:14 jwp Exp $, Revision: 4.2

Files in GeoFEST.MS_E.tar which demonstrate the benchmark:
nr8ex.dat : the input deck containing mesh and timestep information
nr8ex.log : the standard out log including start and end times
nr.out : the output recording the displacements and stresses at the specified times
cghist.txt : the iteration history
nr8ex_500.out : a slice of nr.out restricted to the final (t=500 years) displacements (created from nr.out by text editor).

These have been made into the following image files of the vertical deformation of the surface in the center 120 x 120 km region of the simulation.

coseis.gif : the surface displacement due to the slip event at t=0 years.
nr8_100.gif: the surface displacement after t=100 years
nr8_100-coseis.gif: the difference of the above two displacements, showing purely postseismic uplift due to viscoelastic deformation.

nr8_500.gif: the surface displacement after t=500 years
nr8_500-coseis.gif: the difference of the t=500 and t=0 displacements, showing purely postseismic uplift due to viscoelastic deformation.
These image files were produced with IDL 5.3. A representative script runs as follows:

```
f500=read_ascii("nr8_500.out",template=ascii_template("nr8_500.out"))
; restrict domain to the surface (where z is near-zero)
sf=where(abs(f500.field05) le 1e-4)
; copy the read-in arrays to simpler-to-use names
x=f500.field03(sf)
y=f500.field04(sf)
z=f500.field05(sf)
dx500=f500.field06(sf)
dy500=f500.field07(sf)
dz500=f500.field08(sf)
; set contour levels, titles and domain to useful defaults
lvls=findgen(16)/20.-0.4
!x.title="km East of fault top center"
!y.title="km North of fault top center"
!x.range=[-60,60]
!y.range=[-60,60]
; pick a nice color table
loadct,23
; do plot in two passes: colored strips, then labeled contour lines where stripes meet
contour,dz500,x,y,/irregular,levels=lvls,/isotropic,/fill,title="nr8 vertical displacement at 500 years"
contour,dz500,x,y,/irregular,levels=lvls,/isotropic,/follow,/noerase
write_gif,'nr8_500.gif',tvrd()
; end of script
```

1.3 References

Wald, D. J., T. H. Heaton, K. W. Hudnut, "The Slip History of the 1994 Northridge, California Earthquake Determined from Stron-Motion, teleseismic, GPS and Leveling Data," Bul. Seis. Soc. of Am., vol. 85, no. 1B., pp. S49-S70, 1996.

2. Virtual California Baseline

This is the Monte Carlo code that generates simulated, realistic earthquakes on an arbitrary fault surface mesh. The topology of the fault mesh is defined by the user. The stress Green's functions are then computed in codes VC_STRESS_GREEN.f and VC_SG_COMPRESS.f and then used as input to the Virtual_California family of codes. The stress Green's functions are then used, together with a user-defined friction model, as input to an initialing code VC_INIT_SER.f. The initializing code is run several times until the user sees all initial transient effects have ceased. Finally, this code, VC_SER.f is used to generate simulated earthquakes.

2.1 Model Physics:

Virtual California uses topologically realistic networks of independent fault segments that are mediated by elastic interactions. (Note that earlier versions had viscoelastic interactions as well, but such are not included in the present code). VC is a "backslip" model, inasmuch as the plate tectonic stress increases is produced by means of applying a negative ("backslip") velocity to each segment whose magnitude is that of the long-term rate of slip on the segment. Since "positive slip" reduces the stress on a fault segment, "negative slip" due to the backslip increases the stress. On each time step, all faults are checked to determine whether the shear stress has reached the failure threshold. Once at least one segment reaches the threshold, the "long time steps" stop, and "short (failure) time steps" (a.k.a. Monte Carlo Sweeps, or mcs) begins. An mcs begins with a check of of each site to determine whether it has failed, followed by a parallel updating of each segment. An update of a segment consists of increasing the sudden seismic slip on each segment so that the stress of the segment, considered in isolation, drops to a residual value, plus or minus a random overshoot/undershoot. The elastic stress on all segments is then recalculated, and another mcs is carried out. This iterative process repeats until all segments are below the failure threshold, at which time the mcs time steps cease and the long plate tectonic time steps begin again. Note that VC also includes a stress-dependent "precursory slip", or "stress leakage" of the type that has been observed in laboratory experiments by TE Tullis (1996) and S Karner and C Marone (2001). The physics of this process is that as the stress on a segment increases, a small amount of stable sliding occurs that is proportional to the level of the stress above the residual. Lab experiments and field data suggest that the parameter called "alpha" (in Rundle et al. 2001) is of the order of a few percent. This parameter is called facsdp() or facrat() or facsd() in this code.

In addition, as described in Rundle (1988), the fault system topology + elasticity + failure law may lead to an unstable, "runaway" dynamics due to the presence of positive eigenvalues, so we introduce a nonlinear, "self-adapting" or "self-correcting" term. The VC initializing code estimates the magnitude of these terms for each segment using a simple but unrealistic dynamics. They are then readjusted by running the VC code with real dynamics until all startup transients (unrealistically large stresses) disappear. The time scale for the transients to disappear depends on the complexity of the problem. Physically, one can imagine that the system evolves on a rough energy landscape, and on average sits at the bottom of a free energy well. Large earthquakes may tend to displace the system from the well allowing non-stationary evolution to occur before the system settles into a new well. The configuration of the well is determined by the magnitude of the positive eigenvalues discussed above. The eigenvalues for each segment are proportional to the derivative of the shear stress with respect to the slip on each fault segment.

2.2 Simulation Details

To run Virtual_California, you will need the following data files:

1. A fault topology data file -- sample: VC_Faults_1999.d

2. A fault friction file -- sample: VC_Friction_1999.d

These can be created in various ways. Note that the fault friction file gets overwritten when you run the actual fault simulator (called VC_INIT_SER.F) on an initial run, so you want to make a copy of this and run the codes with the copy.

The basic fortran source files you will need to compute earthquake histories are:

1. VC_STRESS_GREEN.F -- computes the stress Green's functions

2. VC_SG_COMPRESS.F -- when using viscoelasticity, this code compresses the viscoelastic Green's function using a collocation, or spectral representation

3. VC_INIT_SER.F -- this code uses a phony dynamics (allowing both forward and backward slips on a segment) to produce a dynamics in which all the eigenvalues are negative, corresponding to a stable well on the dynamical energy landscape

4. VC_SER.F -- this is the actual earthquake simulator code, with the real dynamics in it. It includes the stress smoothing (alpha effect) as well as the real nonlinear dynamics, the Coulomb Failure Functions, and so forth.

Note that you also need a file containing array dimensions, included here as EQPARM_4.FOR

The order for running the codes is:

1. Run VC_STRESS_GREEN using fault topology data file, result is a fault output file with the stress Green's functions

2. Run VC_SG_COMPRESS on the fault output file from 1.,to get the final form of the stress Green's function: the viscoelastic Green's functions are compressed

3. Using the stress Green's function file from 2., along with a fault friction file, run VC_INIT_SER to initialize the dynamical variables. The output file from this run should then be repeatedly and iteratively fed back into the VC_Init code several times to refine the dynamical parameters until the number of fault segments failing on any given time step, in both forward and backward slip events, is approximately constant. For the sample fault topology file, this certainly happens after several thousand (~ 5000) time steps. At the moment, the way things are set up, you have to run VC_INIT_SER maybe twice (the code writes an output file that is read as an input file on the next running of VC_INIT_SER).

4. Using the data file that eventually comes out of 3., finally run VC_SER. When you first run this, you will observe another instability develop. You will have to run VC_SER several times, until all transients are gone, and you see that no fault has remained stuck for a time period of 10,000 years or longer (watch output to screen); the number of fault segment failures basically stabilizes, but with fluctuations. Typical time steps would be 1 year, or perhaps .2 year. If needed, the time steps can be increased to 10 years or larger to help eliminate unwanted transient effects more quickly. Another thing to look at is the values of the Coulomb Failure Stress, as well as the fraction of unstable slips, and the values of the cumulative slips themselves. When all segments seem to have turned on and slipping at reasonably regular intervals, you have reached a statistically steady state, in which the dynamics has found a new local free energy minimum, and the point in phase space corresponding to the system state is fluctuating near the bottom of this new potential well.

5. When an output data file is felt to have acheived a statistically steady state, the output data can be viewed with the VC_Visualize code, which is a series of scripts that runs on an IDL base.

To compute and plot the horizontal surface deformation corresponding to a give earthquake data file, you need to do the following. Note that you also need a file, included here, or array dimensions, DEPARM.FOR

1. First run VC_DEF_GREEN.f to compute the deformation, or kinematic Green's functions. These are computed at a square grid of points centered on the x-y midpoint of the fault system.

2. Second, run VC_DEFORM.f to compute the deformation difference between two instants of time. This code can be kept running at an interactive level while plotting is going on.

3. Third, use VC_VISUALIZE to make .ps plots of surface deformation, either as arrows, or as InSAR fringes.

The code VC_Visualize.pro is a set of scripts written on an IDL base (http://www.rsinc.com/idl/index.asp) that includes the following capabilities and procedures:

1. An x-y (km) map of the model. Uses procedure: model.pro
2. A seismicity time-distance plot. Uses procedure: timdis.pro
3. A plot of slip as a function of fault location. Ues procedure: slipdis.pro
4. A map of the event superposed on the x-y map of the model. Ues procedure: event.pro.
5. 3-dimensional map view of the model segments. Ues procedure: model_3d.pro
6. A 3-dimensional map of event slip superposed on the fault map. Uses procedure: slipmap_3d.pro
7. A 3-d color-coded fault friction map. Uses procedure: frictionmap_3d.pro
8. A map of horizontal deformation arrows superposed on fault map. Uses procedure: displ_arrow.pro
9. An image of wrapped InSAR fringes superposed on fault map. Uses procedure: insarfringe_wrapped.pro

In the following, we report performance results of benchmarking tests for major code elements of *virtual_California* codes. This family of codes perform realistic earthquake simulations on topologically realistic fault systems subject to realistic evolutionary dynamics based on quasi-static elastic fault interactions, together with friction laws observed in laboratory experimentation. We benchmarked two codes on two workstation-class serial machines.

Machine descriptions:
I. A workstation running Slakware Linux, "**Boltzmann**"
CPU -- 696.98 MHz Pentium III (Coppermine)
Memory -- 256 MB
Memory Speed -- 133 MHz
II. A workstation running Slakware Linux, "**Julia**"
CPU -- 997.5 MHz Pentium III (Coppermine)
Memory -- 256 MB

Memory Speed -- 256 MHz

We ran two of the Virtual California codes for purposes of baseline timing:

1. **VC_Stress_Green.f** Code computes the $(N_2 + N)/2$ elastic interaction coefficients for the fault network specified. We timed the main loop over all main 215 fault segments in the 1999 southern California fault system model.

2. **VC_Ser.f** Code is the main earthquake simulation code, uses interactions, together with friction and failure dynamics to evolve the fault system stress field through time and compute simulated earthquakes. We timed the main time-stepping loop for 10,000 1-yr time steps for the 215 fault segments in the 1999 southern California fault system model.

Results: The results we obtained from benchmarking the codes on the two systems are shown in the table below.

System Code	**Boltzmann**	**Julia**
VC_Stress_Green.f	10 m 1.64 s	5m 37.74 s
VC_Ser.f	29 m 22.16 s 15 m	48.33 s

2.3 References

Rundle, JB, A physical model for earthquakes, 2. Application to Southern California, J. Geophys. Res., 93, 6255

Rundle, JB, Linear pattern dynamics in nonlinear threshold systems, Phys. Rev. E, 61, 2418 (2001)

Rundle, PB, JB Rundle, KF Tiampo, JSS Martins, S McGinnis, and W Klein, Phys. Rev. Lett. 87, 148501 (2001)

Rundle, JB, PB Rundle, W Klein, JSS Martins, KF Tiampo, A Donnellan and LH Kellogg, GEM plate boundary simulations for the plate boundary observatory: Understanding the physics of earthquakes on complex fault systems, Pure. Appl. Geophys., in press (2002).

3. PARK Baseline

3.1 Problem description:

Compute the history of slip, slip velocity, and stress on a vertical strike-slip fault that results from using state-of-the-art rate and state frictional constitutive laws on the fault for a specific geographic setting at Parkfield, California. The boundary conditions are those appropriate for Parkfield and the distribution of constitutive properties on the fault

zone are as realistic as our ability to characterize the subsurface properties of the fault there allow. The methods developed in solving this problem can be generalized to other geologic settings in which the fault geometry, the boundary conditions are not so simple and multiple faults are involved.

3.2 Algorithm and Code description:

The main program is a boundary element program that determines the stress on every element of the fault surface due to slip on every other element, using a Greens function approach. The fault constitutive law is used to determine what the slip velocity will be for that stress and this velocity multiplied by the time step gives the slip to be used to calculate the stress in the next time increment. This involves the forward time integration of coupled ordinary differential equations. The integration is done with a fifth order Runge-Kutta scheme with adaptive step size control. Because the time-steps range over ten orders of magnitude, depending on whether the fault is slipping very slowly in the interseismic period or very fast during an earthquake, the adaptive step-size control is an essential element in the solution.
The main program calls a variety of subroutines and the one of these subroutines that calculates the derivatives used in the forward time integration itself calls a Fast Multipole library that is suitable for such Green's functions problems. The Multipole approach allows a number of computations to scale as N log N rather than N2 as would otherwise be the case. The particular Fast Multipole approach being used allows determination of the degree of grouping of the remote cells based on an analytical approximation to the Greens function. In order to reduce computation time it also renumbers the elements so that those that are near in space are also near in memory.
The main program and most of its subroutines are written in Fortran. At the time of the Baseline Milestone these programs are not written in parallel and so the Baseline run is done as a serial job. However, the Fast Multipole library already is written in parallel using MPI.

3.3 Scientific Significance:

Achieving the Baseline Milestone is significant because it represents a large amount of consolidation, reorganization, and documenting of code that has been developed over a period of more than ten years by a variety of workers. For the first time it presents to the scientific community codes that allow one to create a simulation of the entire earthquake cycle in a 3D model that uses the most accurate description of fault friction, rate and state friction, and the quasi-dynamic radiation damping approximation to full elastodynamics. A significant advance over previous implementations of the fault code is the inclusion of a Fast Multipole library that opens up the potential for greatly increasing the number of elements that can be included in the model.

At this stage of development, because the fault code is only serial, the number of elements that can be used is too small. This means that the sizes of the elements that must be used to represent a fault of any reasonable dimensions are too large to represent

properly the behavior of a continuum. They are also too large to allow occurrence in the model of earthquakes with a large range of sizes. The attainment of future milestones involve converting the fault code to run in parallel using MPI and increasing the efficiency of the code in other ways. This will not only represent an advance in our computational ability to simulate earthquakes, it will allow us to understand the earthquake process better by creating data sets that can be compared with data on real earthquakes.

3.4 Simulation details:

Code and documentation can be found in two places, this web site and on turing, an SGI Origin 3000 at NASA Ames. This web site is the only public location. The machine turing is the front end to the machine chapman on which the Baseline Milestone was run. For the purposes of NASA's verifying that the baseline run is as described in the Milestone_Certification_Data file and repeating the run if desired, it may be easier to use the version of the documentation, files, etc. that is located on turing since everything has been uncompressed, untared, compiled, etc. Verification can also be done by the public or by NASA officials by using materials on this web site.

Within the appropriately named subdirectories under the BaselineMilestones directory in which this file is found can be found all the necessary material that describes the Baseline Milestone and gives instructions that would allow one to duplicate it. The materials there include:

Milestone_Certification_Data - a file that give the time required for the Baseline run and describes various parameters of the run.

README-FIRST.txt - a file that explains complications involved with what machine the baseline was done on.

README-setting_up_input_files.txt - a file that tells one how to understand the input files including an explanation of how the elements are created from the input files.

README-Compile.txt - a file that tells how to create both the multipole library and the PARK fault files using the appropriate Makefiles.

Baseline_Input_Files - a directory that contains the input files that were used in the Baseline run.

Baseline_Output_Files - a directory that contains the output files that were generated in the Baseline run.

Baseline_Code - a directory that contains the PARK and related fault application files used in the Baseline run.

SmallerValidation_IO - a directory that contains the input and output files that were generated in a smaller run that can be used to check if a new user's code is giving the same answers as does the code used in the validation run.

Downloads - a directory that contains four tar files that allow one to generate the files needed for the Baseline run. There are tar files that will create the:
1) Multipole library
2) source files for the PARK fault application
3) input files for the baseline run
4) input and output files that were used in the Smaller Validation run.

These 4 were created on turing in the following way, all while in the BaselineMilestone directory:

1) For the Multipole library, but after having first moved the libsw.a and the mpmy_seq.o files from the t17-7/Objfiles/IRIX32 directory to another temporary location:

tar -cvf MP_lib.tar t17-7

2) For the PARK files, but having first temporarily moved the object files and the executable files to another temporary location:

tar -cvf PARK_source.tar src-bin

3) For the Baseline input files:
tar -cvf input.tar in

For the Smaller Validation input and output files:
tar -cvf SmallValid.tar SmallValidation

The Baseline Milestone run was as follows:

It had 500 time steps and 15000 elements. It was run on a single processor of the SGI, chapman, at NASA Ames and finished on Wed Sept 18, 2002 at 14:28:14. It was run as a special batch job in order to have enough wall clock time to complete. It was run using the PBS-script-chapman.15003 script file in the /u/tullis/BaselineMilestone/in directory of turing and thsi website. The run includes preceding the executable name with the command

time

in order to report the time of execution. At the end of the standard output, the time command returned the following:

real 7h53m8.23s
user 7h52m19.37s
sys 0m3.00s

This can be seen at the end of the
stdout_of_script.chapman.15003 file in the
BaselineMilestone/out Directory of turing and this web site.

The PARK programs and the Fast Multiple library were compiled
with -O2 as the Makefile in the BaselineMilestone/src-bin
directory shows for the PARK earthquake model and the
BaselineMilestone/t17-7/Make-common/Make.IRIX64 file shows for
the Fast Multipole library.

Another way of seeing some of the specs on the job are to
look at the Job Resource Usage Summary for the run that is
copied here from that same stdout_of_script.chapman.15003 file
referred to above:

Job Resource Usage Summary for 40552.chapman.nas.nasa.gov

CPU Time Used : 07:51:18
Memory Used : 0kb
Virtual Memory Used : 0kb
Walltime Used : 07:53:16

Memory Requested : 3920mb
Number of CPUs Requested : 16
Walltime Requested : 24:00:00

Nodes Assigned : 4
CPUS Used : 32to47
Execution Queue : special
Execution Host : chapman.nas.nasa.gov
System Billing Units : 126.204444
Charged To : g12560

Job Stopped : Wed Sep 18 14:28:14 2002

The Baseline Milestone-E run was also done on the IBM SP at Brown University. Details follow:

It had 500 time steps and 15000 elements. It was run on a single processor of the IBM SP at Brown University and began on Fri Jul 26 at 16:18:49 2002. It was run as a batch job in the 1n-pwr3 class. It was run by preceding the executable name with the command

time

and at the end of the standard output, the time command returned the following:

89719.390u 10.970s 25:01:09.75 4.2% 239+239k 0+0io 92pf+0w

This can be seen at the end of the std_output.MP.15003 file in the BaselineMilestone/Baseline_on_SP/out directory.

It was compiled with -O3 as shown in the Makefile in the BaselineMilestone/Baseline_on_SP/src-bin directory.

Another way of seeing some of the specs on the job are to look at the output of the LoadLeveler that runs the batch jobs:

From: LoadLeveler

LoadLeveler Job Step: control.cascv.brown.edu.21657.0
Executable: in/tetbatch.prk.15003
Executable arguments: State for machine: f4n49.cascv.brown.edu LoadL_starter:
The program, tetbatch.prk.15003, exited normally and returned an exit code of 0.
This job step was dispatched to run 1 time(s).
This job step was rejected by Starter 0 time(s).
Submitted at: Fri Jul 26 16:18:49 2002 Started at: Fri Jul 26 16:18:49 2002
Exited at: Sat Jul 27 17:20:01 2002
Real Time: 1 01:01:12
Job Step User Time: 1 00:55:19 Job
Step System Time: 0 00:00:11
Total Job Step Time: 1 00:55:30
Starter User Time: 0 00:00:00
Starter System Time: 0 00:00:00
Total Starter Time: 0 00:00:00