

CIG: Computational Infrastructure for Geodynamics

www.Geodynamics.org
California Institute of Technology
Pasadena, California 91125

To: hpc-workshop@teragrid.org

From: Michael Gurnis, Director, Gurnis@gps.caltech.edu

RE: Geophysics Benchmark Suite v. 0.1

Memo: The Computational Infrastructure for Geodynamics (CIG), an NSF Center, is providing this input into the HPC performance requirements on behalf of the geophysics community. CIG is a membership governed organization with 36 domestic member institutions (mostly research universities) and 5 foreign affiliates that develops, supports, and disseminates open source software for the geophysics community.

In solid earth geophysics, the research community uses a wide range of numerical methods to simulate physical processes and interpret data. Simulation codes now in wide use on HPC platforms reflect this diversity. In order to best evaluate future HPC platforms, we recommend the consistent use of a “Geophysics Benchmark Suite”. We believe that the best way to evaluate new HPC hardware is to run the entire Geophysics Benchmark Suite and compare the results from competing and existing systems.

Table 1

Geophysics Benchmark Suite v. 0.1		
Code	Numerical Method	Problem Area
CitcomS	Finite Element	Regional & Global Mantle Convection
SpecFEM3D	Spectral Element	Regional & Global Seismic Wave Propagation
PyLith	Finite Element	Tectonic Deformation: From Dynamic Earthquake Rupture to Earthquake Cycle
SNAC	Finite Difference	Tectonic Deformation: Geologic time-scales, including fault growth

The current CIG Geophysics Benchmark Suite is summarized in Table 1, which includes four codes that are either widely available to the community or are soon to be made

available. We expect that the codes included in this list will change from time to time, with new codes added and others removed. Details of the codes in the GBS are as follows:

CitcomS is a finite element code written in C which solves for thermal convection within a spherical shell. Solving for incompressible, Boussinesq flow with iterative solution schemes (conjugant gradient and multi-grid). The code has been ported on shared memory computers (Sun, Hewlett-Packard, SGI, and IBM), commercial distributed memory machines (Intel, Cray/SGI, IBM BlueGene), and the Beowulf implementations. The code is in wide use by the community.

SPECFEM3D is a spectral-element MPI code written in Fortran90. It solves the seismic wave equation for local, regional, and global scale problems. The package is widely used around the globe and has run on a wide variety of distributed and shared memory machines (linux clusters, IBM, Sun, HP, and Dell clusters, the Earth Simulator, BlueGene, etc.). Typical simulations involve hundreds to thousands of processors. Use of the open source code is based upon a non-commercial licensing agreement.

PyLith is a code for modeling both quasi-static and dynamic deformation in the solid Earth. The code is used to examine crustal deformation at spatial scales ranging from meters to hundreds of kilometers and temporal scales ranging from milliseconds to thousands of years. Problems often include complex geometry and rheologies (elastic, viscoelastic, and/or viscoelastoplastic). It is currently under development and results from a complete restructuring and merging of a quasi-static code (LithoMop which is a descendant of TECTON) and a dynamic code (EqSim). PyLith uses the Pyre simulation framework to drive the simulation workflow and PETSc for its computational engine. Quasi-static problems involve implicit time-stepping whereas dynamic problems involve explicit time-stepping; both types of problems use Krylov solvers.

SNAC is an updated Lagrangian explicit finite difference code for modeling a finitely deforming elasto-viscoplastic solid in 3-D (including both Cartesian and Spherical Shell options) coupled to a thermal field. The code, now coming online, uses the FLAC, fast Lagrangian analysis of continuum, method that has been widely used in 2D codes. Since the code is fully explicit, it allows for easily incorporating different rheologies. It is also much less memory intensive than the CitcomS and PyLith codes, provides an important complement to the rest of the GBS.

CIG currently provides CitcomS to the community and will soon be providing the other three codes as well. The CIG staff will be available to work with computer vendors to port these codes to different platforms and provide all of the input decks for comprehensive benchmarking.

CitcomS

1. Availability: Now available under GPL
2. Code may be obtained at www.geodynamics.org
3. Science enabled as a result of increased capability provided
 - What is the science problem that you would want to tackle if 10X, 100X, 1000x current capability were available?
 - Modeling of planetary mantle convection at high Rayleigh number and resolve multi-scale structures such as plumes, slabs, and broad-scale structure. Also modeling thermochemical mantle convection is important and extremely challenging computationally.
4. Programming: C, Python.
5. Libraries used (e.g. BLAS, SCALAPACK, etc.): none
6. Parallelization use: MPI.
7. Input file size(s) (typical run, and largest envisioned in your research , and number of files (constant, or proportional to number of processes):
Only one input file is needed for most applications, and the file is of order of 10s of Kbytes. However, in case of restarting, a complete set of outputs for a specific time step is required. For sizes and number of those files, see the question 8.
8. Output file size(s) (typical run, and largest envisioned in your research, and number of files (constant, or proportional to number of processes):
Raw outputs are ascii/binary. Each processor writes its own outputs of temperature, velocity, pressure et al. The size of outputs per time step depends on grid size and is $N \times M$ words, where N is the number of physical quantities and M is the number of grid points.
9. What are the requirements for parallel I/O? N/A.
10. What percentage, if any, of the runtime is typically dedicated to I/O?
Don't know for sure, but not much and perhaps less than 1%.
11. Which, if any, of the following features does your application use or have?
 - FFT or multi-grid: Yes, multi-grid.
 - Particle-based or lattice-gas algorithms: Yes, particles.
 - Continuum equation solvers: Yes.
12. Ported to what architectures/systems
 - What type of system do you prefer to run this application on and why? Linux-based distributed memory machines.
 - What type of validation tests do you use after porting the application to a new system? Simple test problems to check if the ported program gives the same solution with the one acquired from an already working platform.
 - What is the typical time (FTE months) to port your application to a new system and test the port? 0.25
13. Do you routinely use more than one type of system to run the applications that are of interest in your research? No.
14. What aspect of computer system performance tends to be most limiting when running your code (e.g., memory bandwidth, memory latency, network bandwidth, network latency, network topology, I/O bandwidth) and how was this determined?
We do not fully know yet the limiting factors, because for most of applications that I am aware of using CitcomS, the number of processors is ~ 100 or less and the code

performs excellently (90% efficiency). But I can envision as # of processors exceeds 1000, network latency will become an issue.

15. Minimum number of processors required. 1.

16. Maximum number of processors scaled to and why. ~512 for testing but could go higher if having machines. For the testing, the efficiency of the code is still very good.

17. Minimum amount of RAM/proc required. For reasonable application runs, 500 Mbytes/proc is needed.

18. What type of scaling is most important for this application, Strong Scaling (where the same computation can be run on few or many processors), or Weak Scaling (where the problem is adjusted to maintain constant work per processor as the number of processors increases)? Weak scaling.

What is the communications to computation ratio? (Don't know yet. Guess that it is machine-dependent)

How does the ratio behave as the application is scaled up? (Don't know yet.)

If the number of degrees of freedom is increased, by what factor do the resources (e.g. memory, I/O traffic, cycles, wall-clock time, ...) needed by the problem scale? Roughly linearly, as CitcomS uses multigrid.

19. Does your application use check-pointing? If so how often is check-pointing performed (per number of time-steps or similar)? Yes. As often as writing outputs.

20. Are you willing to provide support (e.g. answer questions about the application)? Yes.

21. What other codes do you know of that are used for the same type of science problem? Are they algorithmically similar or different? Are they sensitive to the same sort of architectural characteristics? Terra, STAG3D, and some other codes: Most use multigrid. Sensitivity to architecture is unknown.

22. Size of the user community, if the code is a community code? ~50

23. Optimum continuous scheduled run time as a function of number of processors or other limiting factor? About a week with 100-1000 processors.

What is the range of runtimes and resources needed to most effectively solve the problems? One day to two weeks. 100 to 1000 processors.

24. No on-demand requirements

25. Workflow and workflow programming:

To start, the main program is executed with an input file. Workflow is internally managed and no additional workflow programming is needed. When restarting from some intermediate time step, the code will read in some pre-existing files. Otherwise there is no difference.

The code (CitcomS.py) is also embedded within the Pyre Python modeling framework and this allows: (1) regional or small scale simulations to be embedded within large-scale or global simulations, or (2) the coupling with other codes, such as SNAC which can simulate the visco-elastic crust/lithosphere with CitcomS being used to simulate the viscous mantle.

PyLith

1. Open-source (GPL)
2. www.geodynamics.org. Beta versions currently available, but version anticipated Summer 2006.
3. Science. Reducing the uncertainty in seismic hazard estimates requires that we make significant progress in understanding the physics of earthquake ruptures and crustal deformation that spans spatial scales from meters to thousands of kilometers and temporal scales from milliseconds to thousands of years.
 - a. What is the science problem that you would want to tackle if 10X, 100X, 1000x current capability were available?

Current problems require narrowing the spatial and temporal scales to a narrow range (3-4 orders of magnitude). Expanding current capabilities by 100-1000x would allow simulating a much larger fraction of the relevant spatial and temporal scales, thus permitting the study of direct links between the physical processes underlying earthquake ruptures and large-scale system behavior. Additionally, it would facilitate use of Pyre's more advanced features, including coupling physics codes to solve problems that span multiple disciplines (such as crustal deformation driven by mantle convection), and running many instances of solvers in inverse problems.
4. What programming languages: Python, C++, f95
5. Libraries: PETSc (BLAS/LAPACK), HDF5
6. Parallelization used:MPI
7. Input file size(s) (typical run, and largest envisioned in your research , and number of files (constant, or proportional to number of processes):

Problem specification: 1 MB (constant)
Input (finite-element mesh): 10 MB (per processor)
Supporting info (data): 10 GB (constant)
8. Output file size(s) (typical run, and largest envisioned in your research, and number of files (constant, or proportional to number of processes):

File sizes: 100 MB (per processor)
files: 10 (constant)
9. What are the requirements for parallel I/O?

Simulations write to files using MPI I/O for parallel I/O. All processors write to the same file with file position coordinated among the processors without the need for shared file pointers (i.e., permitting the use of PVFS2). Check pointing using parallel I/O requires the greater I/O bandwidth (1 GB per processor).
10. What percentage, if any, of the runtime is typically dedicated to I/O? 5-10%
11. Which, if any, of the following features does your application use or have?

Sparse linear algebra
Implicit or explicit time stepping (depends on problem)
Global scatter-gather (few per timestep)
parallel I/O (once per every few timesteps)
12. Ported to what architectures/systems

1. What type of system do you prefer to run this application on and why?

We prefer to run the application on distributed memory systems including Beowulf clusters, because they are widely available and the code can be ported to new systems with little, if any, modification.

2. What type of validation tests do you use after porting the application to a new system?

We run community defined benchmarks to validate the code after porting the application to new systems.

3. What is the typical time (FTE months) to port your application to a new system and test the port?

Typical time to port to a new system is <0.2 FTE months.

13. Do you routinely use more than one type of system to run the applications that are of interest in your research?

We often run simulations on a variety of systems, basing the decision on the available resources and problem size. For example, small problems may be done on SMP workstations or small Beowulf clusters, medium problems on Beowulf clusters or older supercomputers (HP V-Class), and large problems on Beowulf clusters.

14. What aspect of computer system performance tends to be most limiting when running your code (e.g., memory bandwidth, memory latency, network bandwidth, network latency, network topology, I/O bandwidth) and how was this determined?

We believe that memory bandwidth is the limiting factor because the linear solve, which uses Krylov solvers, dominates the computation time.

15. Minimum number of processors required.

Small problems can be run on 1 processor.

16. Maximum number of processors scaled to and why.

We have run simulations using up to 32 processors. This number of processors resulting in the fastest turn around time for simulation results based on the problem size and the architecture and queue configuration of the machine.

17. Minimum amount of RAM/proc required.

Runtime is fastest when using around 1 GB of RAM per processor on current Beowulf systems (i.e., systems with 1-2 GB of RAM per processor with ~1-2 GHz 64 bit processors).

18. What type of scaling is most important for this application, Strong Scaling (where the same computation can be run on few or many processors), or Weak Scaling (where the problem is adjusted to maintain constant work per processor as the number of processors increases)?

Weak scaling is most important as problem sizes can vary dramatically, depending on the physical process and temporal and spatial scales being studied.

a. What is the communications to computation ratio?

The communications to computation ratio is usually 0.1-0.15.

b. How does the ratio behave as the application is scaled up?

This ratio increases slightly as the problem size increases.

c. If the number of degrees of freedom is increased, by what factor do the resources (e.g. memory, I/O traffic, cycles, wall-clock time, etc.) needed by the problem scale?

In general, memory, I/O, cycles, and wall-clock time scale with the number degrees of freedom. However, in many cases larger problems also involve decreasing the size of smallest scale, so larger problems usually require a smaller time step; in these cases cycles and wall-clock time will often scale closer to the square of the number of degrees of freedom while memory and I/O continue to scale with the number of degrees of freedom.

19. Does your application use check-pointing? If so how often is check-pointing performed (per number of time-steps or similar)?

Application uses check-pointing (usually about once per 1000 time steps).

20. Are you willing to provide support (e.g. answer questions about the application)? CIG staff will provide support.

21. What other codes do you know of that are used for the same type of science problem? Are they algorithmically similar or different? Are they sensitive to the same sort of architectural characteristics?

There are a wide variety of other codes used for similar problems. Some are algorithmically similar with very similar architectural characteristics, whereas others use different discretization schemes (finite-difference or boundary integral) resulting in the use of different algorithms and architectural characteristics. For example, simulations using finite-difference discretizations require much less memory but many more cycles to solve the same problem.

22. Size of the user community, if the code is a community code?

We anticipate the user community will grow from ~10 to ~100 in the next 5 years.

23. Optimum continuous scheduled run time as a function of number of processors or other limiting factor?

?

a. What is the range of runtimes and resources needed to most effectively solve the problems?

A wide range of problems that have many different runtime and source demands can be solved by this application.

24. There are no on-demand requirements at this time.

25. Please give a brief description of the workflow associated with running the types of jobs needed for your research. What are the requirements (data movement, etc., synchronization of multiple executables, etc.)? What workflow programming is used? (E.g., does the application exist as mixed f90+/C++ binaries dynamically loaded into a python script that communicates with a perl program to manage the workflow?)

The high-level portions of the code are written in Python and make use the Pyre simulation framework. The Python scripts dynamically load C++/f95 dynamic libraries as necessary. During initialization, the basic input files are read by all processors albeit independently and large data files are read as necessary by each processor independently. The main loop may be either implicit or explicit time-stepping (depending on the problem). Computed fields are saved every few time steps via parallel I/O. In simulations that couple different codes, synchronization of fields between solvers will be required. This may require significant data movement among certain processors at a small fraction of the timesteps or small data movement among certain processors every few timesteps.

SNAC

1. Availability: (soon to be) publicly available
2. If available, site where code may be obtained: www.geodynamics.org
3. Science enabled as a result of increased capability provided
What is the science problem that you would want to tackle if 10X, 100X, 1000x current capability were available? Modeling of multi-scale deformation in the earth sciences, such as mountain building, formation and evolution of faults and development of sedimentary basins.
4. Programming: C, Python, C++.
5. Libraries used (e.g. BLAS, SCALAPACK, etc.): none
6. Parallelization use: MPI.
7. Input file size(s) (typical run, and largest envisioned in your research , and number of files (constant, or proportional to number of processes): order of 10s of Kbytes. 1, constant. However, in case of restarting, a complete set of outputs for a specific time step is required. For sizes and number of those files, see the question 8.
8. Output file size(s) (typical run, and largest envisioned in your research, and number of files (constant, or proportional to number of processes): Raw outputs are binary. Each processor writes its own outputs for 16 variables. Typically, the total size of outputs per time step is about 1 Mbytes in binary.
9. What are the requirements for parallel I/O? N/A.
10. What percentage, if any, of the runtime is typically dedicated to I/O? Don't know yet. Only partially estimated as 5% at this time
11. Which, if any, of the following features does your application use or have?
Global scatter-gather operations (if yes, how frequently?): Yes. Twice per time step.
Dynamically evolving coordinate grids: Yes.
Continuum equation solvers: Yes.
12. Ported to what architectures/systems
What type of system do you prefer to run this application on and why? Linux-based distributed memory machines.
What type of validation tests do you use after porting the application to a new system? Simple test problems to check if the ported program gives the same solution with the one acquired from an already working platform.
What is the typical time (FTE months) to port your application to a new system and test the port? 0.25
13. Do you routinely use more than one type of system to run the applications that are of interest in your research? No.
14. What aspect of computer system performance tends to be most limiting when running your code (e.g., memory bandwidth, memory latency, network bandwidth, network latency, network topology, I/O bandwidth) and how was this determined? Network latency because SNAC's algorithm requires frequent communications between processors but the size of exchanged message is usually very small.
15. Minimum number of processors required. 1.
16. Maximum number of processors scaled to and why. >1000. With 100 processors, it takes about a week to solve a problem for a reasonable period of model time. To increase the spatial resolution by an order of magnitude but keep the computation

time at the same level, about 1000 of processors would be needed. Models with 10,000 processors are envisioned in several years.

17. Minimum amount of RAM/proc required. 50 Mbytes.

18. What type of scaling is most important for this application, Strong Scaling (where the same computation can be run on few or many processors), or Weak Scaling (where the problem is adjusted to maintain constant work per processor as the number of processors increases)? Strong Scaling.

What is the communications to computation ratio? (Don't know yet.)

How does the ratio behave as the application is scaled up? (Don't know yet.)

If the number of degrees of freedom is increased, by what factor do the resources (e.g. memory, I/O traffic, cycles, wall-clock time, ...) needed by the problem scale? About 1, but not sure.

19. Does your application use check-pointing? If so how often is check-pointing performed (per number of time-steps or similar)? Yes. As often as writing outputs.

20. Are you willing to provide support (e.g. answer questions about the application)? Yes.

21. What other codes do you know of that are used for the same type of science problem? Are they algorithmically similar or different? Are they sensitive to the same sort of architectural characteristics? FLAC3D: Algorithmically similar. PyLith: Algorithmically different. Sensitivity to architecture is unknown.

22. Size of the user community, if the code is a community code? ~100

23. Optimum continuous scheduled run time as a function of number of processors or other limiting factor? About a week with 1000 processors.

What is the range of runtimes and resources needed to most effectively solve the problems? One day to a week. 100 to 1000 processors.

24. No on-demand requirements

25. Workflow and workflow programming: To start from the initial step, the main program is executed. An input file name should be given as an argument. When restarting from some intermediate time step, it is necessary to post-process the outputs. This might require data movement from distributed disks to a common place. Once the files for restarting are ready, the main program is executed with the restart option turned on in the input file. Workflow is internally managed and no additional workflow programming is needed.

In addition, the code is entirely integrated into the Pyre, Python framework such that this visco-elastoplastic code can be coupled with the CitcomS mantle convection code.