

# Installation and Configuration of Gale for Geological Modeling on a Beowulf Cluster

Donald Lewis

*Department of Electrical and Computer Engineering*

*University of Maine*

*Orono, ME, USA*

donald.lewis@umit.maine.edu

## Abstract

The St. Elias Erosion/Tectonics Project (STEEP) is a study into the creation and properties of the St. Elias Mountains, located in southern Alaska and northwestern Canada. The goal of the STEEP is to create a model for the orogeny of the St. Elias Mountain range. The research being done at the University of Maine is primarily concerned with the plate tectonics of the boundary between the North Pacific Plate and Alaska, and how it contributes to the orogeny. To study this, a 2D and 3D computer simulation, Gale, can be used. Gale is a finite element model capable of modeling both 2D and 3D geological simulations. It can model subduction, rifting, orogenesis with the ability to be linked to surface erosion models. For the Beowulf cluster, the use of a direct solver provided the best performance.

## I. INTRODUCTION

The St. Elias Erosion/Tectonics Project (STEEP) is a multi-discipline study of the formation and continued evolution of the St. Elias Mountain range. The St. Elias Mountains are located in southern Alaska and extend into northwestern Canada. There are many geological events that contribute to the evolution of this mountain range, including subduction, crustal convergence and accretion, and glacial erosion. The goal of the project is to create a comprehensive model of the physical processes that drive the evolution of this mountain range. The University of Maine is one of ten institutions working on this project, which covers many disciplines of study. The Department of Earth Sciences of the University of Maine is primarily concerned with the geodynamic aspect of the STEEP.

According to the STEEP project summary, there are a few significant questions that the project seeks to answer. At the University of Maine, research is being done to address the geodynamic aspect of the following questions:

- How is deformation partitioned into lithospheric shortening and uplift versus lateral extrusion of the detached crust, and does intense erosion influence this partitioning?
- Is the orogeny driven primarily by subduction of a buoyant oceanic plateau or by collision of a small microcontinental block attached to allochthonous ocean crust? [1]

Part of this research involves computationally intensive modeling of the zones in 2D and 3D. Models of the Alaska-Pacific plate boundary were created for use in simulations. The program that is used for these types of simulations is Gale.

Gale is a 2D and 3D modeling program targeted at long-term tectonics simulations.[4] Gale is specifically designed to solve problems in orogenesis and subduction, which are important aspects of the research involved in the STEEP project. The

Alaska-Plate boundary is a subduction zone, and participates in the orogeny of the St. Elias Mountains. Modeling this zone using Gale is important for the STEEP project research. Gale is a parallel code, that uses the Message Passing Interface (MPI) and the Portable, Extensible Toolkit for Scientific computation (PETSc) to implement parallel operation. MPI is the standard protocol used for inter-processor communication, and supports C, Fortran, and many other programming languages. PETSc is a collection of parallel solvers for differential equations and other matrix operations. This report will outline the procedures involved in installing and configuring Gale for use on the Beowulf cluster at the University of Maine. The cluster consists of 256 Apple XServe G5 nodes, connected using both Ethernet and a lower latency Myrinet network.

## II. INSTALLATION

Gale requires some libraries in order to build. These libraries include MPI, PETSc 2.3.2, libxml2, and python 2.2.1. For the University of Maine cluster, not all of the required libraries for Gale are installed by default. MPI is already on the supercomputer, but the libxml2 and PETSc libraries must be installed before Gale can be installed and configured. This section will outline in detail the steps that must be performed to get Gale up and running on the University of Maine cluster, and should give a rough outline of how to install it on any Beowulf cluster. The most difficult library to install is the PETSc library, as it can run into some odd errors that will be described later on.

### A. libxml2

The latest version of libxml2 available at the time of installation was version 2.6.32. This version was released on April 8, 2008, and is available for free under the MIT License from XML Soft.[2] The installation of libxml2 is not complicated. Once the source has been downloaded and unzipped, it must be configured using the configure script

available in the directory. Once the installation has been configured, the program can be built using the make and the make install command. Below, the exact commands and sequence are shown for convenience.

```
# tar -xvf libxml2-2.6.32.tar.gz
# cd libxml2-2.6.32
# ./configure
# make
# make install
# make tests
```

If make tests completes without any large deltas or errors, then the installation has completed successfully. Libxml 2 is now ready to use by Gale, and should not require any more configuration or tampering.

### B. PETSc

Gale requires PETSc 2.3.2 specifically. This is not the latest version of PETSc, which is currently at 2.3.3, however Gale will not work with the latest version. PETSc evolves rather quickly, and each version is has a lot of changes in it, which tends to hinder backwards compatibility. PETSc 2.3.2 is available from the PETSc website, which currently has all versions as far back as 2.1.6 directly available.[3] The installation for PETSc itself is rather straight forward, and the steps are very similar to the steps required for libxml2. However, after some work with Gale, it becomes apparent that for increased performance, a direct solver is required when scaling jobs to multiple processors. The direct solver that is recommended by Gale is the Multifrontal Massively Parallel sparse direct Solver (MUMPS). Installing this on the cluster is slightly tricky due to some issues with PETSc passing command line parameters from configure to the installation of third party packages.

To start, the tarball must be unpacked. This can be done using tar -xvf command. Once this has been done, an environment variable for the PETSc directory must be created. The easiest way to do this is to export the current working directory to the PETSC\_DIR variable while in the PETSc directory. Once this has been completed, PETSc is now ready to be configured. The exact commands used on the University of Maine cluster are shown below. This is done after the PETSc 2.3.2 code has been downloaded.

```
# tar -xvf petsc-2.3.2.tar.gz
# cd petsc-2.3.2-p10
# export PETSC_DIR=$PWD
```

For the configuration, PETSc requires a Blas/Lapack implementation, which can be downloaded by the configure script with the -download-f-blas-lapack=1 flag. For MUMPS, the configure script can download MUMPS, as well as the other required packages for MUMPS. MUMPS requires ScaLAPACK and BLACS as well, which can all be downloaded and installed by the configure script. The configure script also needs to know where the MPI library and include files are. The command used to configure on the supercomputer is shown below.

```
# config/configure.py
--with-mpi-lib=/usr/lib/libmpi.la
--with-mpi-include=/usr/include
--download-f-blas-lapack=1
--download-mumps=1
--download-scalapack=1
--download-blacs=1
```

This should install all of the necessary packages to use PETSc with MUMPS. On the supercomputer, errors would come up while installing BLACS, as it could not find the MPI include files for the MPI Fortran compiler, mpif77. To fix this, the file Bmake.Inc located in \$PETSC\_DIR/externalpackages/blacs-dev must have the following line modified in the file:

```
F77FLAGS = -I/usr/include -fPIC -g
```

This tells BLACS where the include files are, and fixed the problem. PETSc can then be reconfigured using the script again. This time, instead of telling PETSc to download BLACS, it is given the directory of BLACS, and then PETSc can be built.

```
# config/configure.py
--with-mpi-lib=/usr/lib/libmpi.la
--with-mpi-include=/usr/include
--download-f-blas-lapack=1
--download-mumps=1
--download-scalapack=1
--with-blacs-dir=$PETSC_DIR/externalpackages/
blacs-dev/linux-gnu-c-debug/
# make all
# make test
```

When PETSc is installed, and tested successfully, the environment variable for the architecture as detected by PETSc must be set. The value of this variable is determined by the name of the file make\_log\_PETSC\_ARCH. This value is then exported for Gale to use when it is built.

```
# export PETSC_ARCH=linux-gnu-c-debug
```

### C. Gale

Once the installation of PETSc is complete, Gale is ready to be installed. The source code for Gale is available from the Computational Infrastructure for Geodynamics (CIG) website.[4] Installing Gale should be simple compared to the previous installation. Once the source has been downloaded, it must be unpacked. Gale then must be configured, built, and installed. The commands used to do this on the University of Maine supercomputer are listed below.

```
# tar -xvf Gale-1_3_0.tar.gz

# cd Gale-1_3_0

# ./configure.py -with-petsc-dir=$PETSC_DIR
  --with-petsc-arch=$PETSC_ARCH

# make

# make install
```

If during configuration, Gale is unable to locate the libxml2 libraries or the MPI libraries, they can be pointed to directly. The command `./configure.py -help` offers instruction on how to directly point to the locations of MPI and libxml2.

### III. USING GALE

Gale uses customized XML files as the input to the program. All of the parameters for a simulation are set in these files, which use the XML format along with customized tags to represent different variables, models, and methods. The specifics on how to construct models for simulations are not discussed here. Instead, the variables and settings that affect performance on clusters will be reviewed in detail.

```
<list name=plugins>
```

The values after this tag are the plugins that are loaded and then used by Gale during simulation. This value was of interest because all of the output from Gale can be disabled here. This was used to test the program and determine bottlenecks. By commenting or deleting the line containing `Underworld_VTKOutput`, all of the output gets disabled. With no output, file I/O is significantly reduced, and the significance of disk access in the overall program run time can be determined. If storing the data caused a significant performance reduction, which did occur to some degree on the slower Ethernet network, the rate at which output was recorded can be reduced. The method used to do this will be described shortly.

```
<param name=maxTimeSteps>
```

This parameter controls the length of the simulation. This was used to reduce the run time of simulations so that they could be repeated frequently for performance analysis.

```
<param name=outputEvery>
```

The value of this parameter determined the interval, in terms of time steps, at which Gale output data. If disabling the output sped up the program by a significant amount, then this value could be increased to increase performance. The disadvantage to this is that not all of the output is produced. For long term simulations of 500 steps or more, having output every 25 steps was found to be sufficient.

```
<param name=outputPath>
```

This value, in the example files, is the relative path for the output to be placed in. This works fine for running Gale on a single thread, where MPI is not needed. However, in order to run Gale on the supercomputer, this value must be set to the absolute path. The relative path does not work because the starting directory gets lost when the job is sent to the individual nodes for execution.

```
<param name=checkPointEvery>
```

This value controls how often Gale outputs .dat type data. These files have information about the entire simulation, and require co-ordination between all of the processors. It was found that if this was enabled, the program would crash when trying to write these files. This is likely due to the fact that file locking is used to make sure that all of the processors work together, which is evidently an issue on the cluster. Consequently, this value must be set to zero in order for the program to run. The output data will still be available in individual files that each processor creates independently. The consequence of relying on the individual files is that the output can take up a large amount of disk space.

### IV. RESULTS

#### A. *yielding.xml*

When Gale is installed, it comes with sample input files for testing. These files can also serve as templates for custom, specific models for scientific use. Before using these models, the scalability and performance of Gale on the supercomputer was determined using these provided input files. The primary file used was *yielding.xml*, which is located in the `input/cookbook` directory. To start, it was run as a 10 time step simulation. This is a very short simulation, but allows for a quick glance at how the program is scaling. Figure 1 shows the speedup for running *yielding.xml* for 10 time steps from 1 to 32 processors. By default, Gale uses a Generalized Minimal

Residue (GMRES) solver for calculation. Also, Myrinet was chosen as the network communication because it is faster than Ethernet. The increased communication speed becomes more important as the number of processors increases.

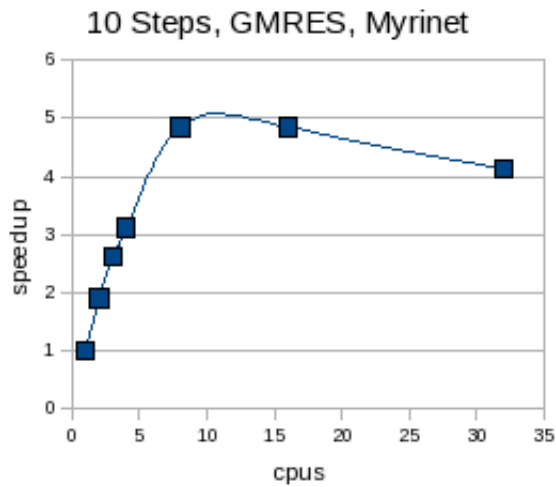


Fig. 1. Speedup - 10 Time Steps, GMRES, Myrinet

From Figure 1, it is evident that 8 and 16 processors offer the same speedup, while 32 processors was slower than 8. While the added overhead of communication does play a role in this, it should not serve to completely overcome the added benefit of the extra processors, especially for such a small simulation. To try and determine if this was an anomaly, a 20 time step simulation was run. The same parameters were used, except for the number of time steps. Figure 2 shows the resulting speedup observed.

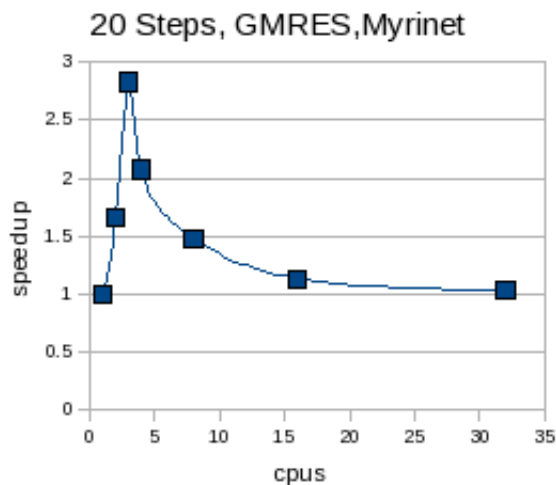


Fig. 2. Speedup - 20 Time Steps, GMRES, Myrinet

From this data, the optimum speedup was achieved with only 3 processors. This is not the desired behavior, and was not expected. The 50 time step simulation provided the following speedup, shown in Figure 3.

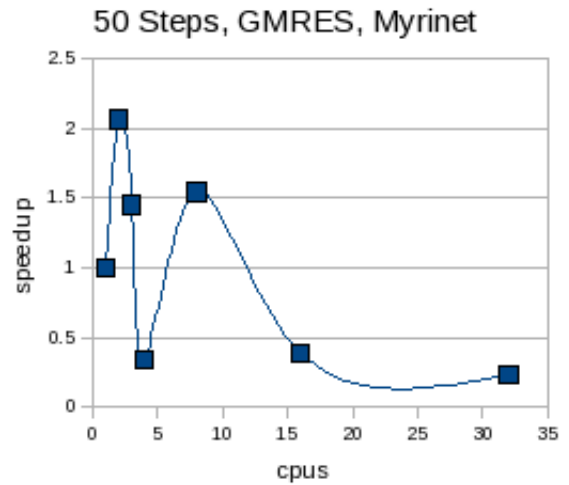


Fig. 3. Speedup - 50 Time Steps, GMRES, Myrinet

This simulation size ran slower on 4, 16, and 32 processors than it did on 1 processor. This data shows that Gale was not going to scale very well to processor counts of 128 and higher. Research into the operation of Gale resulted in the conclusion that this apparently random behavior was due to the nature of the GMRES method.

The GMRES method is an iterative numerical solver. Its basic operation consists of performing calculations until the error, or residual is below a defined threshold. This method is very efficient for large sets of data, as directly solving the equations takes much longer for large sets of data. It is also very inefficient for small sets of data. Direct solvers have a run time that scales exponentially with the amount of data. This property also means that the run time scales down exponentially fast as the data set decreases. This results in the direct solver being much faster for large processor counts, as the data gets split into many small sets.

In order to test and verify this theory, the MUMPS direct solver, which was mentioned earlier in the installation section, was used to provide a parallel direct solver method. Using the direct solver with Gale was relatively simple. To use it, a few flags had to be added to the Gale program call. The required flags were `-mat_type aijmumps -ksp_type preonly -pc_type lu`. Once the direct solver was working, the same simulations were tested. Figure 4 shows the results from the 10 time step simulation.

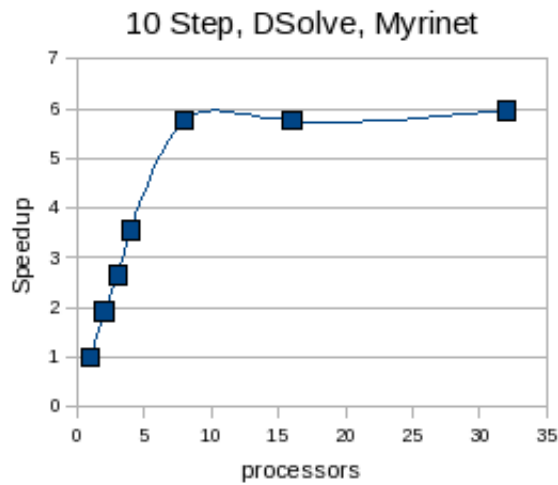


Fig. 4. Speedup - 10 Time Steps, DSolve, Myrinet

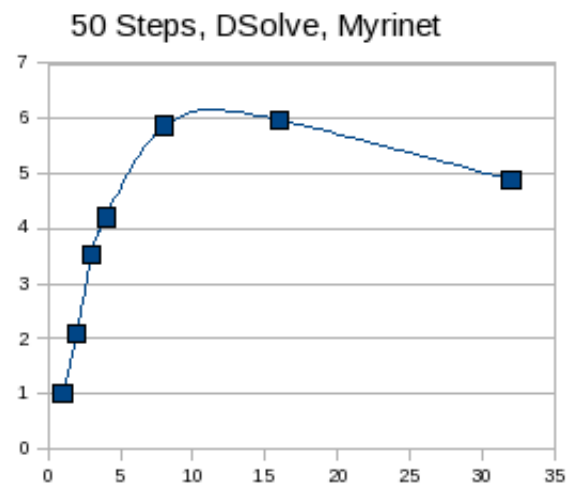


Fig. 6. Speedup - 50 Time Steps, DSolve, Myrinet

For this simulation, the 32 processor job was the fastest, but not by a large amount over the 8 processor job. This performance is easily explained by communication overhead. However, this chart is similar to the GMRES version, as shown in Figure 1. Below, Figure 5 shows the speedup for 20 time steps.

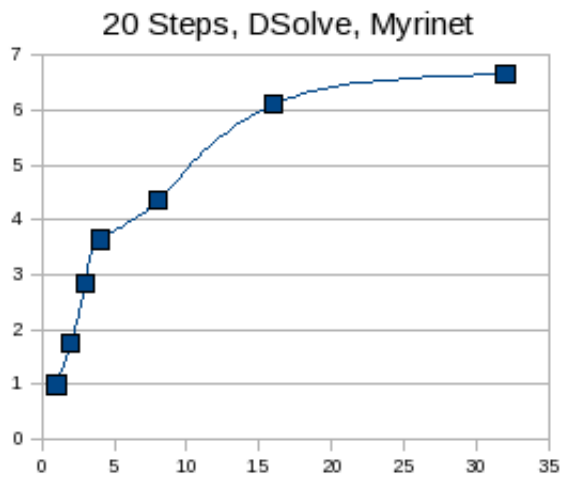


Fig. 5. Speedup - 20 Time Steps, DSolve, Myrinet

This chart shows true scalability for Gale, with more processors always leading to more speedup. The gain from each additional processor is constantly decreasing, and levels off quickly. This behavior is much different than the GMRES method for 20 time steps. From this, it is apparent that the direct solver makes Gale much more scalable and efficient. For further analysis, the speedup for 50 time steps is shown in Figure 6.

From Figure 6, a much smoother curve for speedup is shown, as compared to the GMRES method. Network overhead explains the decline in speedup for more than 16 processors. Overall, the direct solver produced faster results. To inspect the efficiency, Figures 7 and 8 show the efficiency for the 50 time step simulation.

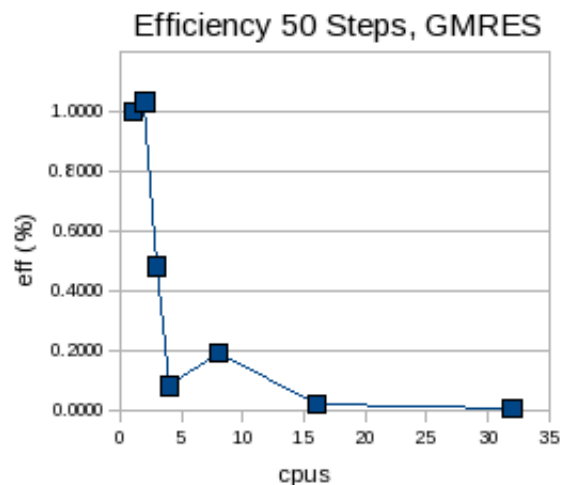


Fig. 7. Efficiency - GMRES (50 steps)

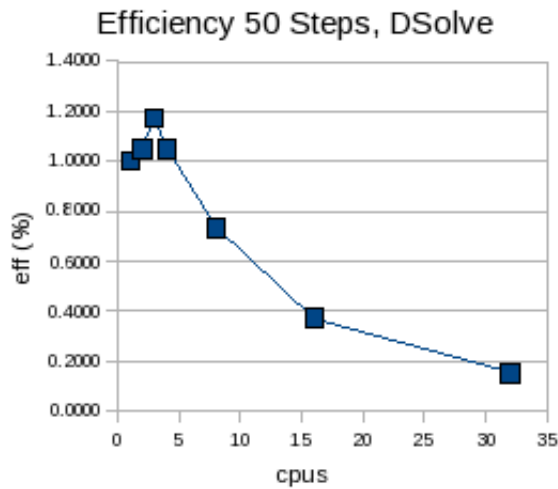


Fig. 8. Efficiency - Direct Solver (50 steps)

The efficiency is much higher for the direct solver, when compared to the GMRES method. The efficiency for GMRES drops off at 3 processors. All simulations running on more than 4 processors had an efficiency below 20%. For the direct solver, the only simulation that was under 20% efficiency was the 32 processor simulation. The direct solver is both faster and more efficient when compared to the GMRES method.

#### B. AK2D.xml

One of the models created for research at the University of Maine was configured in a Gale compatible format. This file, AK2D.xml, was tested using both the GMRES method and the direct solver. A small, 50 time step simulation was run for testing purposes. The resulting speedup is shown in Figure 9.

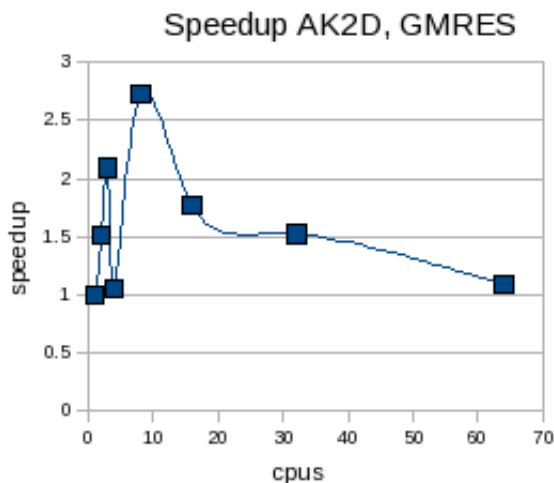


Fig. 9. Speedup - AK2D, GMRES, 50 Steps

This behavior was similar to that observed in the 50 time step yielding.xml simulation. To correct this behavior, the direct solver was used. Figure 10 shows the speedup achieved using the direct solver.

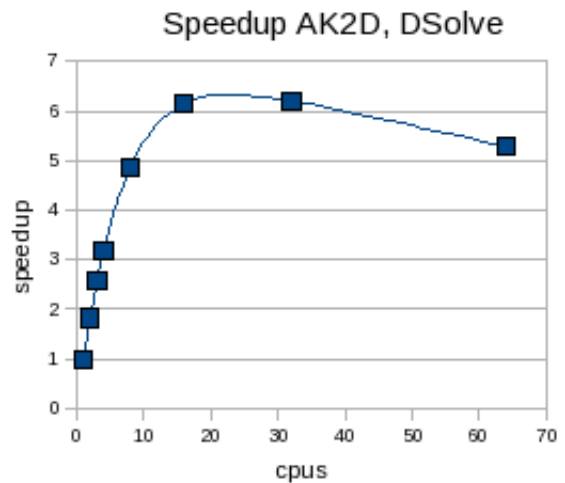


Fig. 10. Speedup - AK2D, Direct Solver, 50 Steps

The direct solver had the same affect on the AK2D simulation that it did on the test code. As a result, for optimum scalability, Gale must be run on multiple processors using the direct solver.

#### V. CONCLUSION

The use of the direct solver makes Gale very scalable. For large simulations, on the order of 500 time steps or more, the direct solver was found to be at least 6 times faster. During the testing of this project, a 500 time step 3-D simulation was run using the direct solver. The simulation took 13 hours, while with the GMRES method it was not completed in a reasonable amount of time. The GMRES method ran for 78 hours before the cluster killed the job due to wall time errors. The direct solver is required for large simulations.

One issue to be looked into in the future is output. The 500 step 3-D simulation produced 9.2 gigabytes of output. This is a large amount of data that needs to be moved from the cluster to the Geodynamic FTP server. The supercomputer only supports SFTP, so this will require the Geodynamic FTP server to be compatible with SFTP. Once this issue is overcome, using Gale should be trivial. Some sample PBS queue files are included in Appendix A. To run a simulation, the only thing that needs to be done is for the model to be put into the XML format.

With Gale, the University of Maine should be able to create unique and important models required for the STEEP project. Gale is relatively simple to use for anyone, and should make 2-D and 3-D modeling much easier.

#### VI. ACKNOWLEDGEMENTS

- This research was co-funded by the ASSURE program of Department of Defense in partnership with the National Science Foundation REU Site program under Grant No. 0754951.
- Special thanks to Dr. Peter Koons and Benjamin Hooks of the Department of Earth Sciences at the University of Maine.

## REFERENCES

- [1] (2008, Jul.) St. elias erosion/tectonics project(steep). [Online]. Available: <http://www.ig.utexas.edu/steep/>
- [2] (2008, Jul.) The xml c parser and toolkit of gnome. [Online]. Available: <http://xmlsoft.org/downloads.html>
- [3] (2008, Jul.) Petsc: Download. [Online]. Available: <http://www-unix.mcs.anl.gov/petsc/petsc-2/download/index.html>
- [4] (2008, Jul.) Gale. [Online]. Available: <http://www.geodynamics.org/cig/software/packages/long/gale>

## APPENDIX

### A. *time\_grab*

This perl script was used to parse the output from Gale to determine the runtime for the program.

Listing 1. "time\_grab"

```
#!/usr/bin/perl
$file = './' . $ARGV[0];
open(DATA, $file);
$start = 0;
@lines=<DATA>;
$index = 0;
while( $lines[ $index ] )
{
    if( $lines[ $index ] =~ /EDT/ && !$start )
    {
        @st = split (/ \s+/, $lines[ $index ] );
        @st = split (/:/, $st[3]);
        $start = 1;
    }
    elsif( $lines[ $index ] =~ /EDT/)
    {
        @en = split (/ \s+/, $lines[ $index ] );
        @en = split (/:/, $en[3]);
    }
    $index ++;
}

# calculate total seconds taken
$sec_st = $st[0]*3600 + $st[1]*60 + $st[2];
$sec_en = $en[0]*3600 + $en[1]*60 + $en[2];
$total_sec = $sec_en - $sec_st;

print("$total_sec\n");
```

### B. *PBS Script*

This is the PBS script used to submit the 500 time step 3-D Alaska model on 128 processors.

```
#!/bin/bash
#PBS -l nodes=128:ppn=1:myrinet
#PBS -A REU
#PBS -l walltime=10000:00:00
#PBS -q linux-batch
#PBS -o o.128.3ak3d
#PBS -e e.128.3ak3d

echo `date`
/usr/bin/mpirun -np 128 --mca btl gm,self ~/SuperMe/Gale/Gale-1_2_2/bin/Gale
~/SuperMe/Gale/Gale-1_2_2/ak3d/AK3D.xml -mat_type aijmumps -ksp_type preonly -pc_type lu
echo `date`
```