SInRG Application HOWTO

Jeff M Larkin, Innovative Computing Laboratory, UT
<larkin@cs.utk.edu>

Revision History
Revision 1.0 2004, June 25
First Public Release

This paper is intended to help developers port their applications to run on the SInRG platform. It defines a few basic tips that can be used to evaluate how an application can use the grid resources. These tips will serve as a first step to making an application run on SInRG.

What is SInRG?

The Scalable Intracampus Research Grid (SInRG) project deploys a research infrastructure on the University of Tennessee, Knoxville (UTK) campus that mirrors the underlying technologies and the interdisciplinary research collaborations that are characteristic of the emerging national technology grid. SInRG's primary purpose is to provide a technological and organizational microcosm in which key research challenges underlying grid-based computing can be attacked with better communication and control than wide-area environments usually permit. Knowledge acquired in this smaller environment helps in improving the national grid infrastructure. SInRG resources are available to all faculty, staff, and students at the university and individuals from every discipline are encouraged to explore the benefits these resources can provide to their applications. The project is supported by a grant from the National Science Foundation. Additional information, including a frequently asked questions, can be found at http://icl.cs.utk.edu/sinrg/.

What types of applications can run on SInRG?

Although SInRG consists of several cluster computers, its resources are not intended for running typical cluster applications. Because the project aims to explore grid computing, applications that run on SInRG must be more than just distributed applications, but also grid applications. In order for an application to be considered a grid application, it must be loosely coupled (little or no interprocess communication) and must use a grid middleware for scheduling and distribution. Although MPI is installed on most SInRG machines, users are discouraged from using MPI in their applications and encouraged to use NetSolve (http://icl.cs.utk.edu/netsolve/). The grid computing model used in NetSolve provides a simple interface to distributing work among many processors, but eliminates the possibility of inter-process communication. Each NetSolve process is completely independent from each other process, using separate data and only able to communicate its results back to the calling process. Not all applications are suited for this model, but many applications that, on the surface, appear incompatible with this model can still be adapted to use NetSolve. Applications that are embarrassingly parallel in nature lend themselves very well to the NetSolve grid computing model.

How can my application use SInRG/NetSolve?

Commonly Used Functions

NetSolve can be interfaced from several popular programming languages and environments. These languages include C, Fortran, Matlab, Mathmatica, and Octave. NetSolve servers, installed on SInRG machines, have many commonly used functions or "problems" that they are able to solve. Those who wish to use SInRG in their application should first check if they are using one or more common functions that

NetSolve is already able to solve. For example, if an application uses LAPACK's dgesv routine, which NetSolve is able to solve, the application developer will only need to change their code slightly to call NetSolve, rather than calling LAPACK directly. Here is an example:

Example 1. A C NetSolve Example

Without NetSolve:

```
dgesv(n,1,a,lda,ipiv,b,ldb,&info);
```

With NetSolve:

```
status = netsl('dgesv()',n,1,a,lda,ipiv,b,ldb,&info);
```

As a first step to adapting an application to use NetSolve, application writers should look at the list of functions available on NetSolve (either via the NetSolve website or the NS_problems utility, provided with NetSolve). If a function that is available through NetSolve is being used in the application, it can be sent to NetSolve to be solved on a SInRG machine. Before proceeding in this way, the developer should make sure that the size of the input warrants such an action. If the time to send the data to and from the server is greater than the time required to solve the problem, then using NetSolve will not make any sense and will result in a slower application. If an application uses some commonly available library that is not currently installed in NetSolve, contact the Jeff Larkin `<larkin@cs.utk.edu>` about adding the library to NetSolve. Using SInRG resources through NetSolve to perform core calculations in an application is the simplest option for a developer and provides a good starting point.

Adding Functions to NetSolve

Just because an application does not use any of the core functions provided by NetSolve, does not mean that the application is unable to use NetSolve and SInRG. It could be the case that parts of the application can be pulled out of the application and installed on NetSolve servers. Developers should look in their applications for pieces of code that are being executed in loops (or threads) where each iteration (or thread) is performing calculations that are completely independent of each other iteration (or thread). A simple example appears below:

Example 2. A Simple NetSolve Loop Example

```
for (i = 0; i < 1000; i++)
{
  somefunction(data, i);
}
```

The above example is somewhat contrived, but the idea to notice is that the same calculation is being performed many times with slightly different parameters and the calculations are not dependent on each other. The function being called in the above example could easily be added to NetSolve. To do so, application developers should first extract all such functions to a static library and then contact Jeff Larkin `<larkin@cs.utk.edu>` for help adding the functions to the SInRG grid.

My application does not fit the above cases.

The above two cases will help most developers interested in using SInRG/NetSolve in their application, but not all. If the above two sections are not enough to help an application developer, they should feel free to contact Jeff Larkin <larkin@cs.utk.edu> for additional consultation. It is still likely that the application developer will be able to use SInRG resources in some way.

Data Distribution and Storage

Data distribution and storage may seem to be a barrier to bringing some applications to the grid, but a solution to this problem has been developed here at the UT computer science department. Part of the research being performed to support the SInRG infrastructure provides a solution to large data distribution and storage through logistical networking. Storage can be placed on the network with the Internet Backplane Protocol (IBP), which can be accessed simply with the Logistical Runtime System (LoRS). Logistial networking provides many benefits that are beyond the scope of this document, but more information is available at the LoCI Lab website (http://loci.cs.utk.edu/).

Additional Help

Since one of the goals of the SInRG project is to empower researchers of many different disciplines, we recognize that not all potential users are programmers. For this reason additional e-mail and one-on-one support is available. If you require additional help, please contact Jeff Larkin <larkin@cs.utk.edu>.

Additional Reading

- SInRG FAQ [http://icl.cs.utk.edu/sinrg/faq/index.html]
- NetSolve Documentation [http://icl.cs.utk.edu/netsolve/custom/index.html?lid=54&slid=81]
- LoCI Lab Web Site [http://loci.cs.utk.edu/]