

GRIDL: High-Performance and Distributed Interactive Data Language

Svetlana G. Shasharina, Ovsei Volberg, Peter Stoltz and Seth Veitzer
Tech-X Corporation, 5621 Arapahoe Avenue, Suite A, Boulder, CO 80303
{sveta, volov, pstoltz, veitzer}@txcorp.com

Abstract

Grid technologies provide a valuable solution for distributed scientific applications but are not readily available for the 4th Generation Languages (4GLs) widely used in physics, climate modeling and medicine. Examples of 4GLs are Interactive Data Language (IDL) and MATLAB. There is a need for tools allowing 4GLs to interoperate with the Grid. This paper describes our design and the status of the tools under development, which include IDL bindings for Web Services and tools for running and managing parallel and serial IDL processes on Grids and clusters.

Index Terms— *Interactive Data Language, Grid, Web Service, MPI, MPICH-G2*

1. Introduction

The scientific and engineering communities are increasingly using Grid [1] technologies and Web Services [2] to implement distributed applications and integrate applications from highly heterogeneous computing environments. At the same time, many scientists, engineers, and medical workers are intensive users of the 4th Generation Languages (4GLs) such as the Interactive Data Language (IDL) [3] from Research Systems Incorporated and MATLAB [4].

Unfortunately, there is presently no simple way to integrate the Grid with 4GLs. This motivated us to start developing GRIDL - a set of tools for using IDL within the Grid. GRIDL will enable scientists and engineers to create Web Services clients within IDL in order to access remote Web Services implementations. They will also be able to implement Web Services within IDL. In addition, GRIDL will allow scientists to run parallel (MPI-type) IDL applications on distributed and high performance resources, such as Grids, clusters and supercomputers. This will provide access to more powerful computational resources,

collocate the analysis tools with the analyzed data and allow scientists to tackle more complex problems.

2. GRIDL Design

IDL allows for calling external C/C++ functions. This is achieved by providing wrappers written in C and registering them within IDL. If these C/C++ functions happen to be MPI C-binding functions and we link our application with MPICH, we are then able to run MPI applications written in IDL. If one uses a Globus device underneath MPI (this capability is provided by linking to MPICH-G2 [5]), we are then able to run MPI applications written in IDL on a Grid, rather than on a cluster or a supercomputer

If the external C/C++ functions happen to be client's stub functions, we are then able to create IDL clients for Web services.

In addition, IDL can be called from C/C++ code. If the IDL code happens to be called from a C/C++ Web Service implementation, then we obtain the capability to invoke IDL objects on the server side, i.e. implement IDL Web Services.

3. GRIDL Status

3.1. Running Parallel IDL on a Grid

To prove the feasibility of running parallel IDL applications on a Grid, we first provided several manually created MPI wrappers and then created a simple IDL code using these wrappers. An example of a wrapper is a commonly used `MPI_Comm_size` function:

```
static IDL_VPTR IDL_MPI_COMM_SIZE
(int argc, IDL_VPTR argv[])
{
    IDL_VPTR tmp = IDL_Gettmp();
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    tmp->type = IDL_TYP_LONG;
    tmp->value.1 = size;
}
```

```

return tmp;
}

```

More complex functions require more sophisticated wrapping, related to the need to allocate memory for in and out function parameters, and we do not show them here for the sake of brevity.

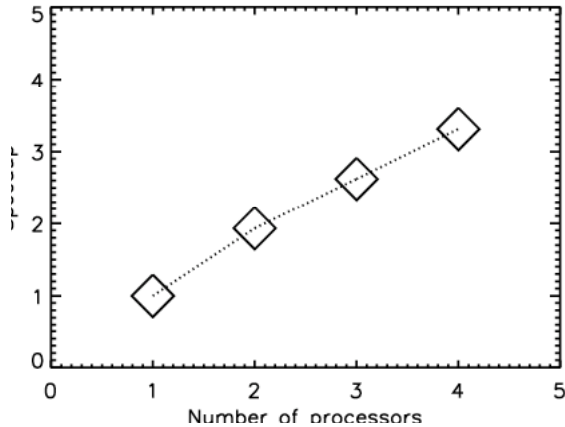


Figure 1. Speed-up obtained in our timing experiments for running parallel IDL on a Grid using 1 to 4 processors.

In this example, the IDL code calculates an estimate of π . Processors with rank 1 to N-1 make an estimate based on random numbers, and send that estimate to the processor number with rank 0, which then averages the estimates. To run this example, we used MPICH-G2 (based on MPICH-G2 v. 1.2.6 and Globus 3.2), gcc3.43 and IDL v.6.1. The software was installed on a Grid consisting of two nodes, each having two processors. Figure 1 shows the speed-up obtained in our timing experiments using 1-4 processor, showing almost linear speedup. The results of this timing example were very encouraging.

3.2. IDL Clients for Web Services

To prove the feasibility of calling remote Web Services implementations from IDL we have implemented a simple Web Service using gSOAP [6] and wrapped its client's stubs into IDL. We used gSoap rather than Globus, because at the time, Globus did not provide C hosting environment. To create this service we used the following header file with the remote method prototype:

```

//gsoap wsidlhello service name: Idlhello
//gsoap wsidlhello service location:Grid.txcorp.com:18083
//gsoap wsidlhello schema namespace: urn:idlhello
int wsidlhello__simprocl(int x,
struct wsidlhello__simproclResponse {} *out);

```

The gSOAP stub and skeleton compiler is a preprocessor that uses the header file to generate the WSDL description of the service. It also generates the necessary stub routines for the client and skeleton routine for the remote method. The simple remote method divides the number π by the factor provided as an input parameter to the IDL program:

```

pro client, x
    simprocl, x
end

```

Additionally, we investigated an alternative way to create IDL clients for a Grid Service. This approach uses the IDL-Java bridge provided by IDL (version 5.6 and higher) and a Java client stub provided by GT3 (version 3.2). We first followed a simplified example provided in [22] and implemented a primitive Grid Service called MathService capable of adding two numbers. Implementing the IDL client turned out to be extremely easy: we just looked at the client written in Java and rewrote it in IDL following IDL syntax.

4. Concluding Remarks

This paper outlines the challenges of bridging Grid technologies and Web Services with the Interactive Data Language. In the next steps of these project we intend to finish the automation tools for wrapping C into IDL, investigate further use of the IDL-Java bridge and move beyond IDL to include other languages like MATLAB.

5. Acknowledgments

The work on this project is funded by DOE under contract # DE-FG03-01ER84100 and by Tech-X Corporation.

6. References

- [1] I. Foster, C. Kesselman, editors, The Grid: Blueprint for a Future Computing Infrastructure, (1999).
- [2] <http://www.w3.org/2002/ws/>.
- [3] <http://www.rsinc.com/idl/>.
- [4] <http://www.mathworks.com/>.
- [5] N. Karonis, B.Toonen, and I. Foster, "MICH-G2: A Grid-Enabled Implementation of Message Passing Interface", Journal of Parallel and Distributed Computing, Vol. 63, No. 5, pp.551-563, May 2003.
- [6] Robert A. van Engelen and Kyle Gallivan, "The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks", in the proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), pages 128-135, May 21-24, 2002, Berlin, Germany.