Simplified Grid Computing through Spreadsheets and NetSolve

David Abramson[1], Jack Dongarra[2], Eric Meek[2], Paul Roe[3], Zhiao Shi[2]
[1]*Monash University, Australia*
[2]*University of Tennessee, Knoxville, USA*
[3]*Queensland University of Technology, Australia*
David.Abramson@infotech.monash.edu.au, dongarra@cs.utk.edu, meek@cs.utk.edu, p.roe@qut.edu.au, shi@cs.utk.edu

Abstract

Grid computing has great potential but to enter the mainstream it must be simplified. Tools and libraries must make it easier to solve problems by being simpler and at the same time more sophisticated. In this paper we describe how Grid computing can be achieved through spreadsheets. No parallel programming or complex tools need to be used. So long as dependencies allow it, formulae in a spreadsheet can be evaluated concurrently on the Grid. Thus Grid computing becomes accessible to all those who can use a spreadsheet. The story is completed with a sophisticated backend system, NetSolve, which can solve complex linear algebra systems with minimal intervention from the user. In this paper we present the architecture of the system for performing such simple yet sophisticated grid computing and a case study which performs a large singular value decomposition.

1. Introduction

Many have written about the potential of Grid computing. Yet it is only accessible to the few with expert knowledge on how to use this powerful resource. Real users want to use the Grid through simple and familiar tools such as Matlab™, Excel™, and web browsers.

In this paper we show how an end user tool, spreadsheets, can be used to drive the Grid through sophisticated libraries, in particular NetSolve [2]. We are able to use the Excel spreadsheet application to transparently drive the Grid. No special use need be made of Excel – nor is a special spreadsheet application required. Functions from NetSolve which are available on the Grid may be imported into Excel and used in spreadsheet formulae. Formulae are subject to data parallel evaluation and task parallel evaluation. That is, independent cells and formulae will be evaluated concurrently with no special effort on behalf of the user. Thus spreadsheets may be used to script the Grid. Spreadsheets are popular because they are easy to use and modify, and they support numerical data analysis without programming. Thus the Grid becomes accessible to all those who can use a spreadsheet.

A feature of modern spreadsheets is their capability to be extended; in particular custom functions may be incorporated into them, for example complex simulation functions and computational experiments. For these reasons spreadsheets make an ideal front-end for numeric computing since they support pre and post processing of data without the need for programming.

Existing spreadsheets, such as Microsoft Excel, have inbuilt sequential calculation mechanisms. Thus, they are unable to evaluate multiple cells concurrently using the standard execution mechanism in the calculation engine. To solve these problems we have constructed an architecture for Grid computing using Active Sheets [1,6] and NetSolve.

Active Sheets comprises a plug-in for Excel and a database (job manager) to cache results. It supports the asynchronous evaluation of spreadsheet formulae. This enables jobs to be generated from Excel and results to be received, and cached in the database for subsequent use. Results can be returned in any order and "downstream" calculations only fire when all of the inputs are available. Thus, the spreadsheet can be used to perform quite complex parallel computations. The database also has the side effect of allowing off-line evaluation.

NetSolve users normally have to write code, either in a conventional language like FORTRAN or C, or a

high level mathematic language like MatLab, to perform computations. Thus, the scripting of NetSolve functions needs to be specified programmatically. Using Active Sheets, meta-data which describes NetSolve functions can be downloaded and used to automatically generate proxy code for Excel functions. This enables web services to connect Active Sheets to NetSolve with no programming required. At the backend, the NetSolve system is able to schedule computation across NetSolve servers thereby minimizing any intervention needed by the user to drive the Grid.

This paper describes how a conventional spreadsheet may be evaluated in parallel without modifying Excel's core execution engine. The parallel evaluation mechanism is encoded using custom functions, and works in conjunction with the standard built in sequential evaluation mechanism of spreadsheets. The paper begins with an overview of Active Sheets and NetSolve. We then discuss an architecture for Grid computing using these systems. This is followed by a case study that illustrates the power of such a system.

2. NetSolve

NetSolve is a client-server system that enables users to solve complex scientific problems remotely. The system allows users to access both hardware and software computational resources distributed across a network. Some of the goals of the NetSolve project include ease-of-use for the user, efficient use of the resources, and the ability to integrate any arbitrary software component as a resource into the NetSolve system. Figure 1 shows the infrastructure of NetSolve and its relation to the application that use it. The shaded parts of the figure represent NetSolve system. We can observe that NetSolve acts as glue layer that brings application or user together with the hardware and/or software it requires to complete its useful work, which is the reason why systems like NetSolve are sometimes called grid middleware.

The major components of the NetSolve system are the *NetSolve agent*, an information service and resource scheduler, *NetSolve server*, a networked resource that serves up computational hardware and software resources, and the *NetSolve client* libraries, which allows users to instrument their applications with NetSolve calls for remote computational services. NetSolve agent is the gateway to the system. As an information service, the agent maintains a database of NetSolve servers along with their capabilities (including hardware performance and available software) and dynamic usage statistics. The agent attempts to find the server that will service the request the quickest using the information it possesses. It also balances the load amongst its servers and keeps track of failed servers.

The NetSolve server is the computational backbone of the system. A computer host becomes a NetSolve server when it is configured to run the NetSolve serve daemon. The server can run on single workstation, clusters of workstations, symmetric multi-processors or machines with massively parallel processors. NetSolve problem description file (PDF) is used to describe a computational problem formally. The PDF specifies basic information about a problem such as problem name, inputs and outputs. It also contains the information of libraries that implements any underlying functions or services being interfaced by NetSolve. In essence, the PDF defines a wrapper that NetSolve uses to call the function being incorporated

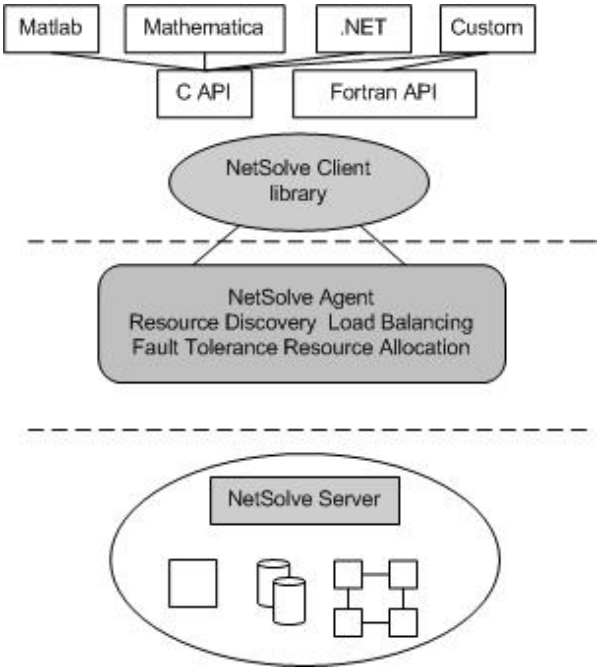


Figure 1. An architectural overview of NetSolve system

At the top tier of Figure 1, the NetSolve client library is linked with the user application. NetSolve supports program development in a variety of programming environments. Interactive environments such as Matlab [7] and Mathematica

[8] are easy to use and often relieve the user from details of variable declaration, memory allocation and other routine tasks. The programming interfaces for which NetSolve APIs have been developed are FORTRAN and C. Currently a native .NET interface is being developed, which will allow all native .NET applications to leverage the power of grid computing using NetSolve.

NetSolve continues to emerge as one of the leading programming paradigms for the Grid. Its light-weight and ease of use make it an ideal candidate for middleware and as it continues to evolve, the NetSolve system will be extended to become applicable to and even wider range of applications.

3. Active Sheets

Active Sheets features a component based spread sheet interface for specifying computational experiments. Figure 1 gives an overview of the Active Sheets vision. This allows computational experiments to be simply and easily driven from a spreadsheet.

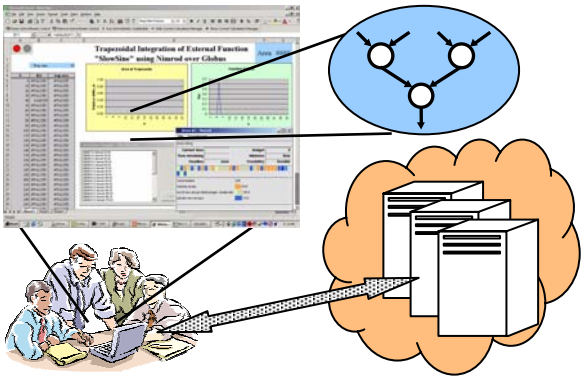
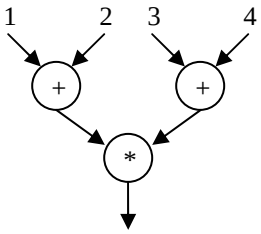


Figure 2: Active Sheets Overview

Backend computations are expressed through spreadsheet formula. These are automatically scheduled according to functional dependencies in a dataflow style, see Figure 3:. In addition operations themselves may be data parallel, implemented through Excel array formulae.



	A	B	C	D
1	1	2	3	4
2	=A1+B1		=C1+D1	
3	=A2*C2			

Figure 3. Data flow and spreadsheet equivalent

We have chosen to use Microsoft Excel as the base spreadsheet application, and have used Microsoft .NET plug-in to control Excel. The plug-in uses .NET remoting to connect to a local job manager, which caches results and manages connections to back-end machines such as NetSolve. Figure 4 shows the Active Sheets architecture.

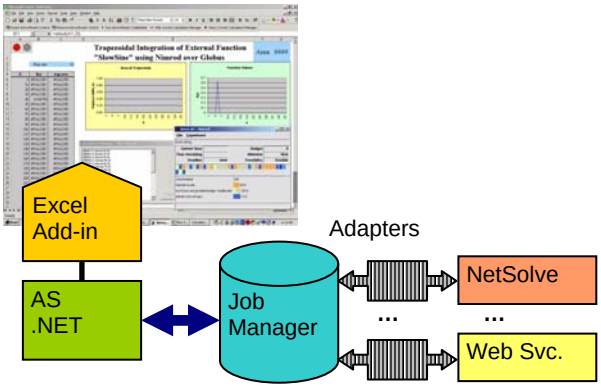


Figure 4. Active Sheets Architecture

The job manager caches results and hence decouples Excel from backend servers. This allows Excel to be used in a disconnected mode, and for the backend computations to be undertaken without Excel running. The job manager also supports a plug-in architecture for adapters; this enables different backend machines to be supported through corresponding adapters. The adapters are .NET assemblies which are dynamically loaded by the job manager.

4. Driving NetSolve from Active Sheets

We have constructed two options for connecting Active Sheets to NetSolve. The first is to produce a custom NetSolve adapter for Active Sheets which understands the native NetSolve communications protocol. This protocol operates over various TCP ports, and is in widespread use. The disadvantage of this is that it requires the designated TCP ports to be open and so does not always work across the Internet. The second approach is to create a web service wrapper for NetSolve using Microsoft .NET. This allows Active Sheets to drive NetSolve from any network so long as HTTP (port 80) is available. This is shown in Figure 5.

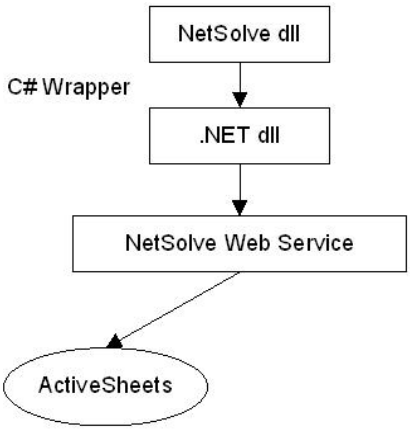


Figure 5. Active Sheets NetSolve Architecture

The system can be monitored at all points. Active Sheets has an option to show the jobs which are outstanding in the spreadsheet. The Job manager has a user interface which allows the status of all jobs, cached values, backend connections, and loaded adapters to be viewed. Finally a GUI tool enables the status of the NetSolve back-ends to be viewed.

5. An application: SVD

A SVD application has been developed to demonstrate the unique combination of NetSolve and Active Sheets. Singular Value Decomposition (SVD) [9] is a useful mathematical tool for finding and removing information stored in matrix form based on its significance to the rest of the data. SVD is widely used in the area of, digital signal processing [10], information retrieval [11] and analysis of gene expression [12].

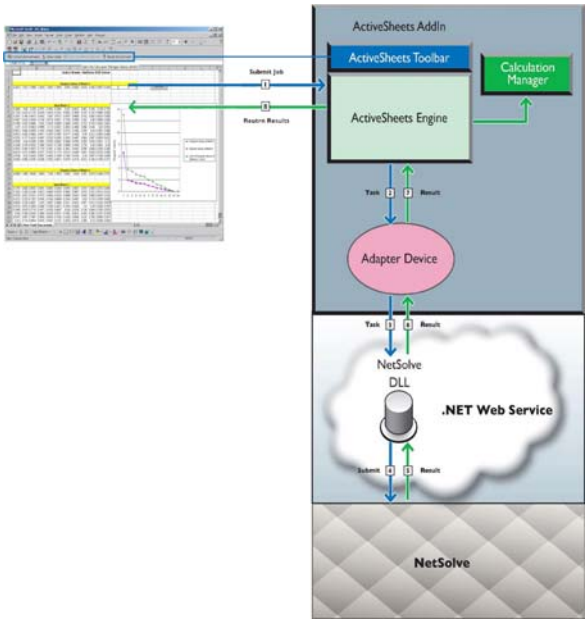


Figure 6. Usage Scenario of NetSolve-Active Sheets

Figure 6 shows the execution scenario of an SVD application using the Active Sheets/NetSolve system. Two matrices are laid out as separate tables in an Excel spreadsheet. When the user then submits a problem, Active Sheets dispatches and manages the tasks from the Excel spreadsheet to the back-end grid computing server(s). It also handles the completion of the tasks by ensuring the data is returned to the correct cell(s) in the spreadsheet. A plug-in adapter is used by Active Sheets to encapsulate the connection to the computational grid. When the user submits a problem and selects NetSolve as the computational grid, the Active Sheets engine will use the NetSolve adaptor to handle communication with NetSolve. The NetSolve adaptor will then forward all the user requests from the spreadsheet to the NetSolve servers in parallel. After the problems have been solved simultaneously and the results are ready, the NetSolve adaptor will retrieve the results from the NetSolve server(s).

The NetSolve adaptor contacts NetSolve through .NET web services [13] that have been implemented via SOAP over HTTP. The NetSolve web services are created for the sake of better interoperability and avoidance of firewall issues. Currently, the prototype

NetSolve web service implements *dgesvd* (a singular value decomposition routine from the LAPACK [14] mathematical library) in a non-blocking fashion. Invoking the *dgesvd* web method creates a new thread and immediately returns. The web service employs a .NET DLL which is built from an unmanaged NetSolve DLL. The unmanaged DLL takes care of all the underlining communication with NetSolve. The problem is then submitted to NetSolve using the *dgesvd* problem and solved. The result is returned to Active Sheets through the adapter and put into the correct cells. Finally, the singular values returned from the NetSolve server can be merged and placed in a results table in Excel.

If C was used to perform the same operation, a high level of programming proficiency would be required. For example, several hundred lines of nontrivial C code would be necessary to import the data, submit the data to NetSolve and finally merge the results. This is very complicated compared to the simple importing of data into, and the problem submission from, an Excel spreadsheet.

6. Related work

There are a few commercial products for performing parallel Monte Carlo simulation using spreadsheets. However these are restricted to Monte Carlo simulation applications constructed using the supplied modeling functions. Palisade has a product called @RISKAccelerator™; this supports the parallel evaluation of @RISK simulations [4]. Decisioneering has a similar product called Crystal Ball Turbo™ which parallelises their Monte Carlo simulation addin for Excel (Crystal Ball) [3]. Platform computing also have an Excel add-in for performing the data parallel evaluation of cells in its Platform Symphony™ software [5]. However, none of these systems supports the general parallel evaluation of spreadsheet functions, as does Active Sheets, nor do they support pluggable backend Grid computers.

7. Conclusions

This paper has shown how Grid computing may be simplified through an architecture which enables spreadsheets to use sophisticated numeric libraries through a web service enabled NetSolve. A mechanism for parallelizing traditional spreadsheets has been defined. The mechanism is based on the idea of dataflow and is particularly useful in that it does not require modification to spreadsheet applications. The parallel spreadsheet tool interfaces with NetSolve through both custom TCP protocols and web services. The latter uses a web service wrapper to web enable NetSolve. On going work concerns how to use large datasets resident outside Excel. For further information on Active Sheets and NetSolve see [1,2], both systems are available for download and use.

8. Acknowledgements

This research was funded in part by Microsoft research and by the Applied Mathematical Sciences Research Program of the Office of Mathematical, Information, and Computational Sciences, U.S. Department of Energy under contract DE-AC05-00OR22725 with UT-Battelle, LLC. We wish to thank Gavin Cheuk for his contribution towards the development of Active Sheets, and the DSTC (www.dstc.edu.au) for their continued support.

9. References

[1] Abramson, D., Roe, P., Kotler L and Mather, D., "*Active Sheets: Super-Computing with Spreadsheets*". 2001 High Performance Computing Symposium (HPC'01), Advanced Simulation Technologies Conference, April 22-26, 2001, pp 110 – 115, Seattle, Washington (USA). http://www.citi.qut.edu.au/ActiveSheets/

[2] *NetSolve: Past, Present, and Future - a Look at a Grid Enabled Server*, S. Agrawal, J. Dongarra, K. Seymour, and S. Vadhiyar, Grid Computing: Making the Global Infrastructure a Reality, F. Berman, G. Fox, and T. Hey, eds., John Wiley & Sons, April 2003, ISBN 0-470-85319-0. http://icl.cs.utk.edu/netsolve

[3] Decisioneering Inc. http://www.decisioneering.com/

[4] Palisade Corporation, http://www.palisade.com/

[5] Platform Computing, "*Adapter for Excel*" http://www.platform.com/products/symphony/adapter/index.asp

[6] Abramson D., Sosic R., Giddy J. and Hall B., "*Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations*", The 4th IEEE Symposium on High Performance Distributed Computing, Virginia, August 1995.

[7] The MathWorks Inc. *Using MATLAB Version 5*. The MathWorks Inc., 1992.

[8] Wolfram S. The Mathematica Book(3rd edition). Wolfram Media Inc. and Cambridge University Press, 1996.

[9] Golub G., Van Loan C. F., *Matrix Computations*. The John Hopkins University Press. ISBN 0-8018-5414-8. 1996.

[10] Deprette, E. F, ed. *SVD and signal processing, Algorithms, Applications and Architectures*. North Holland. 1988.

[11] Furnas G.W., Deerwester S., Dumais S. T., Landauer T. K., Harshman R. A., Streeter L. A., and Lochbaum K. E. *Information retrieval using a singular value decomposition model of latent semantic structure.* In Proceedings of the Eleventh International Conference on Research & Development in Information Retrieval, pages 465-480, 1988.

[12] Wall, M.E., Dyck P.A., Brettin T.S. SVDMAN – singular value decomposition analysis of microarray data. Bioinformatics 6:566-568. 2001.

[13] Microsoft Web Services Developer Center: http://msdn.microsoft.com/webservices/

[14] *LAPACK Users' Guide, 3rd Edition*, E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammaring, A. McKenney, and D. Sorensen, SIAM Publication, Philadelphia, 1999, ISBN 0-89871-447-8.