Project Description — GridSolve: A system for Grid-enabling general purpose scientific computing environments

1. Introduction

The emergence of Grid computing as the prototype of a next generation cyberinfrastructure for science has excited high expectations for its potential as an accelerator of discovery, but it has also raised questions about whether and how the broad population of research professionals, who must be the foundation of such productivity, can be motivated to adopt this new and more complex way of working. The rise of the new era of scientific modeling and simulation has, after all, been precipitous, and many science and engineering professionals have only recently become comfortable with the relatively simple world of the uniprocessor workstations and desktop scientific computing tools. In that world, software packages such as Matlab and Mathematica represent general-purpose scientific computing environments (SCEs) that enable users — totaling more than a million worldwide — to solve a wide variety of problems through flexible user interfaces that can model in a natural way the mathematical aspects of many different problem domains. Moreover, the ongoing, exponential increase in the computing resources supplied by the typical workstation makes these SCEs more and more powerful, and thereby tends to reduce the need for the kind of resource sharing that represents a major strength of Grid computing [1]. Certainly there are various forces now urging collaboration across disciplines and distances, and the burgeoning Grid community, which aims to facilitate such collaboration, has made significant progress in mitigating the well-known complexities of building, operating, and using distributed computing environments. But it is unrealistic to expect the transition of research professionals to the Grid to be anything but halting and slow if it means abandoning the SCEs that they rightfully view as a major source of their productivity. We therefore believe that Grid computing's prospects for success will tend to rise and fall according to its ability to interface smoothly with the general purpose SCEs that are likely to continue to dominate the toolbox of its targeted user base.

The *GridSolve* project we propose below aims to address this difficult problem directly:

> The purpose of GridSolve is to create the middleware necessary to provide a seamless bridge between the simple, standard programming interfaces and desktop SCEs that dominate the work of computational scientists and the rich supply of services supported by the emerging Grid architecture, so that the users of the former can easily access and reap the benefits (shared processing, storage, software, data resources, etc.) of using the latter.

This vision of the broad community of scientists, engineers, research professionals and students, working with the powerful and flexible tool set provided by their familiar desktop SCEs, and yet able to easily draw on the vast, shared resources of the Grid for unique or exceptional resource needs, or to collaborate intensively with colleagues in other organizations and locations, is the vision that GridSolve will be designed to realize.

The opportunity to develop such a system arises from a growing consensus in the Grid community about the strategic importance of creating a new *GridRPC* mechanism. GridRPC is conceived as a version of the classic *remote procedure call* (*RPC*) programming paradigm that has been adapted to the basic facts of Grid computing, and yet kept simple and clean enough to serve as a foundation for experimentation and interoperability among different approaches. Now, since our NetSolve software system [2, 3] is not only based on the RPC paradigm, but also uses that model to enable Matlab and Mathematica users to utilize a restricted subset of Grid resources, the existence of GridRPC in a form that was compatible with key Grid technologies and received broad community support will provide an ideal foundation for translating NetSolve into GridSolve. The aim of the GridSolve project is to bring about this new synthesis.

Since a viable and robust GridRPC mechanism is the fundamental building block for the GridSolve project (a role it is expected to play in many other Grid efforts as well), *a main thrust of the project will be to help drive the design of GridRPC within the community and to lead the implementation effort that delivers successive working versions of it as quickly as possible.* As we describe below, our research experience working with NetSolve should enable us to move quickly, and the current version of NetSolve gives us a natural vehicle for early phases of that effort. Given the early agreement that has already begun to form around certain proposed features of GridRPC, to achieve its goals the development of GridRPC must be accompanied by several other tasks, including the following:

?? *Create essential GridSolve client interfaces* — A key aspect of GridRPC interoperability involves providing a standard client API that enables the sharing of client programs among Network Enabled Server (NES) (e.g. NetSolve, Ninf [4], Cactus [5], NEOS [6]) systems that are based on the RPC paradigm [7]. Our experience shows that once a solid C client that conforms to this API is created, Matlab, Mathematica, and Fortran clients can all be built on the C interface. Such clients already exist for NetSolve, and we expect that porting them to GridRPC, i.e. making them GridSolve clients, will be straightforward. Because of the importance to the community of Open Source alternatives, the GridSolve project will also develop Octave (http://www.octave.org/) and/or SciLAB (http://www-rocq.inria.fr/scilab/) clients on the same model.

?? *Bind the GridRPC mechanism to major Grid backends* — Proper development of GridRPC should make it a relatively straightforward matter to bind to systems that are already based on RPC mechanisms. But to serve the goals of GridSolve, which aims to be a next generation NES, GridRPC must also be able to utilize Grid systems based on other models, most especially Globus and Condor. At a minimum, this means supplying separate "adapters" that GridRPC can use to start jobs within the Globus and Condor systems respectively, but it may also involve addressing more complex questions, such as resource discovery and scheduling that must be handled outside the GridRPC mechanism. A major challenge of the GridSolve effort will be to discover relatively clean and efficient ways to address such problems for each of the backend Grid systems that the user community wants to utilize.

?? *Ensure GridSolve access to key grid services* — Various services need to be delivered in a relatively ubiquitous way within Grid computing, and the GridRPC based approach will have to determine how to engage with them. Some services (e.g. Globus Security Infrastructure -GSI, the Network Weather Service – NWS) are nearly defacto standards; those that are essential to resource discovery and scheduling (e.g. Metacomputing Directory Service – MDS and AppLeS Parameter Sweep Template – APST) tend to still be major subjects of experimentation and debate. While the GridRPC mechanism will remain simple and standard, GridSolve may make choices about how to incorporate access to these external services that other GridRPC-based approaches may decide to do differently.

?? *Enable GridRPC and GridSolve support for adequate data logistics* — The importance in Grid computing of the flexible management of data movement and distributed state for the purposes of enhanced performance, bandwidth efficiency and fault tolerance are well known. The size of the matrices and data sets used by the Grid community continues to grow and the importance of access to distributed, but relatively collocated storage and computational resources for good performance in task parallel and parameter sweep applications on the Grid has been amply documented [8]. It is therefore critical that the GridRPC mechanism should be able to address those Grid services (e.g. GridFTP, the Internet Backplane Protocol – IBP) that have been expressly designed to facilitate the management of data transport and storage on the Grid.

Below we lay out our plan to address each of these objectives, make GridSolve a reality, and promulgate its use within the vast community of research professionals.

2. Background

2.1 NetSolve and general purpose scientific computing environments (SCEs)

A straightforward way to understand GridSolve is to see it as a further evolution of the research done on our experimental NetSolve system [2, 3], bringing to fruition the knowledge gained in that effort and using it to help the broad scientific community reap the benefits of Grid computing. Though the origins of NetSolve predate the Grid movement, the NetSolve approach to building a Grid that is able to support various, widely used SCEs has long been recognized as an important strand in the Grid community's effort [9]. The reason for this recognition becomes clear in the light of two factors: The power of general purpose SCEs, such as Matlab and Mathematica, within the Computational Science community, and the role that NetSolve was designed to play in extending their capabilities in critical ways. We address each of these aspects.

The importance of desktop SCEs becomes clear when you consider the case of *Matlab*. Matlab was designed expressly for people doing numerical computation. It began as a "MATrix LABoratory" program, intended to provide interactive access to the famous LINPACK and EISPACK libraries of state-of-the-art numerical routines. These are carefully tested, high-quality general-use packages for solving linear equations and eigenvalue problems. The goal of Matlab was to enable scientists to use matrix-based techniques to solve problems, without having to write programs in traditional languages like C and Fortran. Much of its power as a language stemmed from its ability to *transparently encode* scientific calculations, i.e. encode them in a way that exposes the underlying science to the inspection of the programmer and allows numerical solvers to be applied efficiently. This capacity of Matlab, in combination with its interactive interface, reliable algorithmic foundation, fully extensible environment, and computational speed, has caused it to rapidly replace Fortran as the "*lingua franca* for the exchange of software and algorithms." The addition, over time, of extensions to the language, outstanding graphics and visualization support, and high performance libraries has strengthened that position.

With a user community more than half a million strong spread throughout industry, government, and academia, Matlab is a recognized standard worldwide for technical computing. It is used in a wide variety of application areas, including signal and image processing, control system design, earth and life sciences, finance and economics, and instrumentation. Its relatively simple architecture makes it easy to use Matlab and companion products to explore data and create custom tools that provide early insights and competitive advantages. It is worth noting, however, that Mathematica claims more than a million users worldwide, and would have an analogous story to tell about its established position in the community and its value to its users.

Given this account of the nature and value of these general purpose SCEs, it is natural to wonder why they should require any support from NetSolve or from Grid computing at all. But the simple fact is that the that scientists and engineers continue to confront challenges that push their work beyond capabilities of their individual workstation platforms, requiring them to try to access more computing power, more complex and specialized software libraries, massive or rare data sets, and the unique contributions of their colleagues in other disciplines, locations, and organizations. These are the kinds of factors that tend to drive users to try to access the resources they need remotely, which today means using the shared resources that the Grid computing can supply.

But while Grids can offer tremendous computing power, they also combine and amplify all the well-known complexities of distributed and parallel computing environments, including distributed ownership of the resources, platform heterogeneity, network reliability and performance, and storage availability. All of these issues make it very difficult to use a parallel or distributed computing platform as if it were a single workstation. Worse still, many of the systems that have been built to overcome them tend to, either because of complexity or indifference, *deprive scientific programmers of the very SCE programming environments that have become crucial to their routine productivity*. NetSolve implemented an approach to Grid computing that used a brokered RPC paradigm (a “client-agent-server” model) precisely because it was the approach best suited to creating a Grid computing environment that preserved the viability and user investment in desktop SCEs. It is still remarkable that, to our knowledge, NetSolve is the *only* Grid system ever to support Matlab and Mathematica as native clients for Grid computing.

But though NetSolve has proved valuable as an experimental bridge between the standard toolkit of Computational Science today and the model of Grid computing that is supposed to fuel scientific progress tomorrow, our experience with NetSolve, combined with the development and maturation of Grid computing on a number of other fronts, revealed problems and limitations in NetSolve’s design and implementation that need to be addressed if its user community is to have the kind of next generation working environment that they require. Although NetSolve will continue to serve as a research vehicle, the effort to create GridSolve that we propose here reflects the lessons we have learned from exploring Grid computing through NetSolve in the context of the broader evolution of the Grid computing. Below we discuss this experience in more detail.

2.2 Research experience with Network Enabled Servers as a driver for GridRPC and GridSolve

Since the mid-nineties various experimental systems [2, 4, 6, 10-12] have been created that use a variation of the classic remote procedure call (RPC) paradigm to aggregate hardware and software resources in the fashion of today's Grid. Some of these systems, often referred to as *Network Enabled Servers* (*NES*), predate the development of today's Grid, but have always been considered to be a key part of the movement [9]. NES systems change the RPC model in a number of important ways, some of which involve resource discovery, dynamic problem solving capabilities, load balancing, fault tolerance asynchronicity, security, etc. One of the important outcomes is a "brokering" process that determines dynamically, when the client makes its request, which server out of the server pool is the best one to handle that request.

As illustrated in Figure 1., an NES system consists of a client (or set of clients), a pool of servers, and a set of resource management functions that may or may not be implemented as separate pieces of software. All NES systems can be cast into this model, and the most prominent ones, viz. NetSolve and Ninf, follow it reasonably closely. Of the five main elements that we distinguish [7], only the Client and the Servers (taken singly) interact with each other through the RPC mechanism; the other elements do not participate in but only broker that interaction, i.e. they provide the information and the decision making algorithm necessary to select the individual server to which the client request is to be sent. In NetSolve (current version 1.4), these elements can be described as follows:

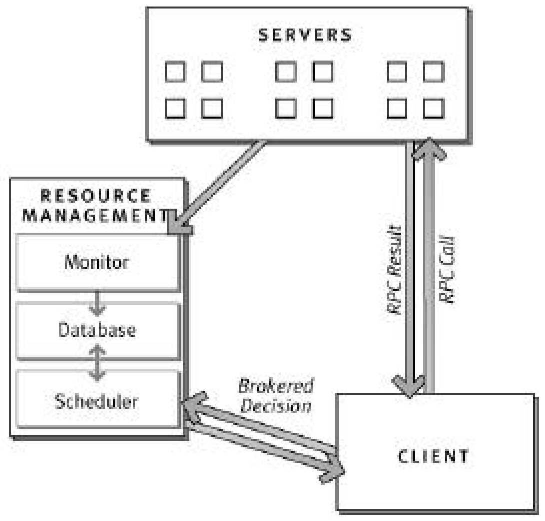

Figure 1: The Basic Network Enabled Server (NES) Architecture

?? NetSolve Client: The NetSolve client library is linked in with the user's application (written for example, in Matlab), which makes calls to NetSolve's application programming interface (API) for specific services. Current clients include C, Fortran, Matlab, and Mathmatica; plans for Octave, Scilab, and Excel clients are underway.

?? NetSolve Servers: The NetSolve server is a daemon process that awaits client requests. The server can run on various kinds of architectures, including single workstations, clusters of workstations, symmetric multi-processors or machines with massively parallel processors. Just as importantly, NetSolve server software includes not only standard numerical libraries (e.g. LAPACK, BLAS) but also more specialized and hard to install packages (e.g. Aztec, PETSC). With NetSolve, users can employ their familiar client interfaces to easily utilize (via an RPC like model) valuable combinations of hardware and software that would otherwise be difficult or practically impossible to access.

?? NetSolve Agent: The NetSolve agent combines both a *Scheduler* that decides which server the client should submit its job too and a Database containing the resource information (e.g. installed software, sever load, available bandwidth, etc.) necessary to make that decision.

?? NetSolve Monitoring: In NetSolve, the monitoring function is spread out among a number of different processes, some running on the servers themselves, others operating as separate middleware services. The current version of NetSolve uses the Network Weather Service (NWS) [13] to provide the most sophisticate information and forecasts of resource status and availability.

In evaluating both the need and the work required to create GridRPC and to transform NetSolve into GridSolve, our experience and the experience of others in the NES community [14] shows that it is important to distinguish between two sets of questions: there are issues that center on how, if at all, the generic RPC mechanism itself needs to be modified or augmented to adapt it to the special conditions of Grid computing; and there is a much larger raft of issues that surround the choice of components, architecture, deployment scheme and configuration of the processes for resource management that must be engaged to provide the "brokering" and other services (e.g.

security) necessary to use the RPC mechanism effectively in the context of the trajectory of Grid computing today. In some respects the latter set of issues is more daunting because it deals with a myriad of different technologies, many of them in different states of development, interoperability, and standardization. In section 3.2 below we will lay out some of the choices we have made or will evaluate, in areas such as scheduling, resource discovery, and security, as part of the process of designing and implementing GridSolve.

The issues that surround the creation of GridRPC are in some degree more basic, since they affect the mechanism that is intended to be a standard for Grid computing generally and at the core of future NES approaches, including GridSolve. For example, NetSolve and Ninf, which are two prominent and widely used NES systems, agree on the way the RPC mechanism needs to be implemented in some respects, but not in others. Both agree in the type of connection protocol they use and that client proxies are necessary to deal with the heterogeneity of Grid backend resources. But they diverge on the crucial question of the Interface Definition Language (IDL), which is used by both the client and the server to marshal input and output data arguments and place actual calls to the appropriate software modules. Likewise they use different wire protocols to communicate between client and server. The result is that, despite several years of active cooperation and mutually beneficial discussion, these leading NES systems do not interoperate in any significant degree.

Certainly these questions must be worked out and settled as part of a community process, and both the NES community (privately) and the larger Grid community (through the Global Grid forum) have begun to participate in this process. We believe that the importance of creating a standard RPC mechanism for the Grid community, along with the maturation of other branches of Grid computing, creates an ideal opportunity to achieve consensus. Happily, early working consensus has already emerged on some questions; for example, that the initial standardization effort should focus on the client API and that it needs to be able to scale as regards the granularity of the calls (<sec to >week) that it can make. But there are three issues — interoperability with other NES systems, ability to use other types of Grid "backends" (e.g. Globus and Condor), and the ability to utilize network storage resources for data logistics (e.g. caching and distributed state management, especially in parameter sweep applications) — which our experience with NetSolve and the AppLeS Parameter Sweep Template shows will be especially important to address in the design of GridRPC. Below, we briefly discuss our experience in each of these areas.

It is natural to ask if one of the industry oriented efforts in this arena might address such problems. One such effort is the Common Object Request Broker Architecture (CORBA) defined by the Object Management Group (OMG). OMG is an independent consortium of vendors; consequently, the standard it defined is open (vendor independent) and has resulted in many independent implementations for a variety of platforms. In order to provide interoperability between these diverse implementations, CORBA defines interoperability mechanisms. A high level, distributed computing model, vendor independence, and a strong interoperability thrust all combined to make CORBA an attractive and popular distributed computing standard; but it has also been recognized as being poorly adapted in some respects to the needs of high performance computing [15]. As such, CORBA meets the necessary requirements to be seriously considered by application developers as part of the Grid infrastructure. Recently, a number of research groups have started to investigate Commodity Grid Kits (CoG Kits) (http://www.globus.org/cog/) in order to explore the affinities of the Grid and commodity technologies. Developers of CoG Kits have the common goal of developing mappings and interfaces between Grid services and a particular commodity technology. We believe that CoG Kits will encourage and facilitate the use of Grid technologies, while at the same time leveraging the benefits of the commodity technology. We will track the work of this group to use these ideas in our effort when possible.

2.2.1 Integration of Grid backends: The NetSolve proxy interface

Since most NES systems began to be developed before the advent of the current Grid model, they tended to closely couple their clients and servers: NetSolve clients worked with NetSolve servers, and similarly for Ninf and other systems. But as the hardware and software systems deployed under other systems that used different, non-RPC programming paradigms, NES systems have worked to adapt in ways that give their users access to a variety of these other "backend" systems. In NetSolve and Ninf, this is accomplished through the use of *client proxies*.

A client proxy is a process that resides on the client host and acts on behalf of the NetSolve client to handle all interactions with a specific kind of Grid backend. Each different backend has its own characteristics and requires its own proxy to shield the client from the details of that interaction. For example, to enable NetSolve to interact with the Globus system we built a proxy for the NetSolve client that knows how to interact with and make use of Globus resources. By using proxies that abstract away the details of the intersystem communication, the NetSolve Client

API can stay consistent while the Grid systems it can utilize can change and grow. With a standard interface between the client and all proxies, it is possible, especially for third party developers, to easily add new language support to the NetSolve system. They simply write libraries that interface the NetSolve proxies from their language of choice, allowing programs of that language to become NetSolve-enabled. The client libraries interact with the proxy thanks to a standard API and the proxy interacts with the Grid system using system-specific mechanisms, i.e. it interacts not just with the computational servers of the system, but also with some or all of the system's own resource management services. This has allowed NetSolve-enabled clients to leverage other Grid resources (e.g. Globus systems) apart from those provided by NetSolve.

But while the Proxy approach has been used successfully by NetSolve and other NES systems (e.g. Ninf), it is not without problems. For example, a NetSolve user who wants to access Globus resources must, according to the Globus security model, have authorization to use all the systems they want to use; but the Globus proxy may neither know nor have an easy way to discover this information. Moreover, interacting with a multiplicity of backends requires a multiplicity of proxies, which can raise problems about how they interact with one another and with the client. In NetSolve, for instance, trying to make the client switch back and forth smoothly among proxies for different systems has proved problematic. Since we want to provide the GridSolve user community with access to the widest possible set of Grid resources, designing an improved proxy interface to address these problems for the major Grid systems will be a high priority.

2.2.2 RPC and the need for data logistics:

While early efforts in Grid computing focused on harnessing distributed computational cycles, applications involving large-scale, data intensive simulation have made it clear that computational and data services, which were once distinguished and addressed separately by distributed systems developers, need to be addressed simultaneously in this context [16]. We use the term *data logistics* to refer to the problem of managing the locality of data so that it is where it needs to be and when in relation to available computational or network resources, in order to be efficiently utilized and with good performance by Grid applications. Our experience with NetSolve led us to explore techniques that exploit the relative cheapness of storage in order to achieve high levels of computational throughput in the presence of large datasets or congested networks through the use of good data logistics [17]. The results of that work, plus the experience of many other middleware and applications groups, are persuasive in pointing to issues of data logistics as a key challenge for Grid computing generally, and therefore as critical to the design of GridRPC and GridSolve.

Typical forms of data logistics include caching, replication, and prestaging, and since all these forms and more are relevant to Grid computing, they will also be so for GridRPC and GridSolve. Our experiments with NetSolve focused primarily on "supply side" caching, in which the application at the client layer of the system is the supplier of data to the computational servers on the Grid, and the data is cached near those servers in a set of storage "depots," called a *Distributed Storage Infrastructure* (*DSI*), for reuse in servicing successive client requests. For our experiments, we place a NetSolve client application at the University of California, San Diego and experimented with requests to a pool of computational servers and DSI depots at the University of Tennessee. The most notable result was that, for a file size that is modest by Grid standards (2.68MB), only two accesses were needed before the overhead added by the DSI was outweighed by the reduction of network activity caused by the reuse of cached data, which reaches an asymptotic level when the cache enhanced system shows three times (3X) speed up over the unenhanced system. More encouraging still, the limit in speed up was reached because of server overload, not a bottleneck in bandwidth or latency [17].

The DSI API we created for NetSolve to support this effort is designed to be general in at least two ways. First, the DSI API and collateral modifications to the NetSolve object model were designed to allow the NetSolve community to use DSIs to build applications that make use of a variety of data logistical techniques on the Grid, but without modifying the standard NetSolve functions for computational requests. Preserving such strategic flexibility in data logistics while continuing to support general purpose SCEs will be an important element of GridSolve.

Second, attempts to deal with data logistics in a general way must confront the fact that the kind of ubiquitous and application independent access to transmission bandwidth that the Internet supplies does not exist for the storage resources that sit on top the network fabric. Several experimental DSIs are now under study, including the Internet Backplane Protocol (IBP) [18], GridFTP [19], Global Access to Secondary Storage (GASS) [20], and NeST [21]. Unless there's a logical reason for doing so, I would remove italics here.For our NetSolve experiments, we used IBP, which is middleware designed and developed by PIs Beck and Plank in order to manage distributed storage for logistical purposes in large scale, distributed systems such as NetSolve. By providing a uniform, application-independent interface to network storage that can be scalably shared, IBP makes it possible for applications of all kinds to use logistical networking to exploit data locality and more effectively manage buffer resources. The goal is to allow distributed applications that need to manage storage for data logistics to benefit from the kind of standardization, interoperability, and scalability currently enjoyed by IP-based network technologies. Although our research actually used an IBP DSI, the DSI API was carefully designed and implemented so as not to be dependent on the underlying DSI. Then, as now, the area of data logistics and DSIs are active areas of research, with various promising technologies (e.g. IBP, GridFTP, NeST, GASS) under active development. Consequently the data logistics strategy we plan for GridSolve/GridRPC, discussed below, will have the same kind of DSI independence possessed by the DSI interface for NetSolve.

2.2.3 Task parallelism on the Grid and the problem of co-scheduling transport and storage: APST

The APST project [22] started in 1999 and builds on the findings and techniques developed as part of the Application-Level Scheduling (AppLeS) project [23]. AppLeS focused on the design and development of Grid-enabled high-performance schedulers for distributed applications. The first generation of AppLeS schedulers demonstrated that simultaneously taking into account application- and system-level information makes it possible to effectively schedule applications on the Grid. However, each scheduler was embedded within the application itself and thus difficult to re-use for other applications. The next logical step was to consider classes of applications that are structurally similar and to develop independent software frameworks, i.e. templates, which can target all applications within a specific class. This led to the development of APST for Parameter Sweep Applications.

APST's goal is twofold: (i) Investigate scheduling strategies for PSAs; (ii) provide transparent application deployment on Grid resources. The work in [8] details novel adaptive scheduling approaches used in the current APST implementation. Transparent deployment is achieved with a modular design that allows APST to use many Grid middleware back-ends simultaneously. The philosophy here is that the user should be able to use a single application execution environment for all the resources he can access. For instance, APST can launch application tasks using Globus [24], NetSolve, Condor [25], or even via a default SSH mechanism. In terms of Data Grid middleware, APST provides support for GASS, GridFTP, IBP, and can default to FTP or Scp. SRB [26] support is underway. APST's user interface is extremely simple and the user can describe his/her application and Grid resources via XML files. APST is currently used in production and evaluation by applications in several research institutions. It is gaining momentum in the scientific communities as it provides disciplinary scientists with a straightforward way to benefit from the Grid. The software was demonstrated both at SuperComputing'99 and SuperComputing'01 and version 1.1 was publicly released in February 2002.

3. Proposed Work

3.1 GridRPC as basic middleware for Grid-enabled SCEs

A reasonable way to draw the distinction between GridRPC and GridSolve is to say that GridRPC is a basic mechanism while GridSolve is a full-blown NES. More precisely, the GridRPC layer focuses on the mechanisms that are necessary for the correct execution of a remote procedure call on the Grid, with careful attention to keeping it as simple and generic as possible; this ensures that GridRPC is usable for a wide spectrum of applications and that it can be integrated easily into many existing projects. GridSolve, by contrast, encompasses the other middleware required — proxies, schedulers, monitors, security, monitors, etc. — to enable desktop SCEs to use utilize Grid resources as easily and efficiently as possible. The diversity of approaches that are now being explored on many of these fronts by the Grid community makes the issues and choices for GridSolve more complex. For example, consider the case of scheduling. Many Grid projects that can benefit from GridRPC (NetSolve, Ninf, AppLeS, APST, Nimrod [10]) address the question of scheduling in very diverse ways. NetSolve, the ancestor of GridSolve, relies on an external entity, the NetSolve agent, in order to select appropriate resources for task executions. In its first incarnation, GridSolve will rely on a similar strategy. Therefore, the GridSolve client, before issuing each RPC call via the GridRPC client, will query an external agent in order to obtain a resource to which the call should be sent. This approach has the advantage that the decision making process stays external to GridSolve, that it has a single point of entry into GridSolve, and that it can therefore be easily replaced and modified. But part of the

GridSolve effort will be experimenting with other approaches (e.g. the approach of APST) for incorporation into GridSolve in its subsequent phases.

Ongoing discussions within the NES community and through the Global Forum have produced certain rough consensus on the development path for the GridRPC mechanism itself [27]. Leaving aside the difficult issues of the IDL and the wire protocol for the second and third phases of the effort, the initial focus will be on creating a standard client API. The basic goals would be to allow different NESs to share client programs and to provide the basic building blocks for writing task parallel programs on the Grid, whether through an NES or independently, e.g. through the APST. The list of generally agreed upon client features include the following: 1) Medium to coarse grained calls (duration <1sec to > week); 2) Task-parallel programming on the Grid (Asynchronous calls, 1000s of scalable concurrent calls); 3) Dynamic RPC: Resource discovery and scheduling; 4) Large matrix data and file transfer (Call-by-reference, shared memory matrix arguments, scientific IDL); 5) Grid-level security (Ninf-G with GSI Implemented with Globus toolkit); 6) Simple Client-side programming and management (no client-side stub programming or IDL management); 7) Server job monitoring and control; 8) Make it very bandwidth efficient.

As part of this project, we will help lead the NES and Grid Community effort to create a standard client and to do a robust reference implementation of it. By building GridSolve around this common foundation, we hope to set the stage for further agreement on the thornier issues of a standard GridRPC (e.g. the IDL), and at the same time assure that the SCE user community will be investing in a technology that will exhibit reasonable longevity. But a review of several of the items in the desired feature list for the client (viz. task parallel grid programming, large data sets, call by reference, efficient bandwidth usage) shows that they all relate in one way or another to issues of data logistics, which our previous research has identified as critical RPC-based programming on the Grid. Below we describe an innovative idea we plan to explore in that arena.

3.2 Stages in the evolution from NetSolve to GridSolve

Since the purpose of GridSolve is to enable a multitude of general-purpose SCE users to tap into the power of the Grid, it is important in our view to engage with their experience early and often during the development of GridSolve and the GridRPC middleware on which it depends. Achieving this intermediate goal requires the rapid implementation of Grid middleware adhering to the GridRPC API (as it stabilizes), the development of a generic GridRPC client, the creation of Matlab and Mathematica clients by wrapping the GridRPC client, and the promulgation of this early prototype within the Computational Science community. The existence of NetSolve software and of the growing NetSolve community provides an ideal vehicle for engaging quickly with this large and important class of potential Grid users. The ultimate goal of to create a three-part system that conforms to the NES model, but uses GridRPC and consolidates what the NES community has learned about Grid computing and what it can anticipate for its future. Figure 2 presents a picture of GridSolve in it's fully developed form.

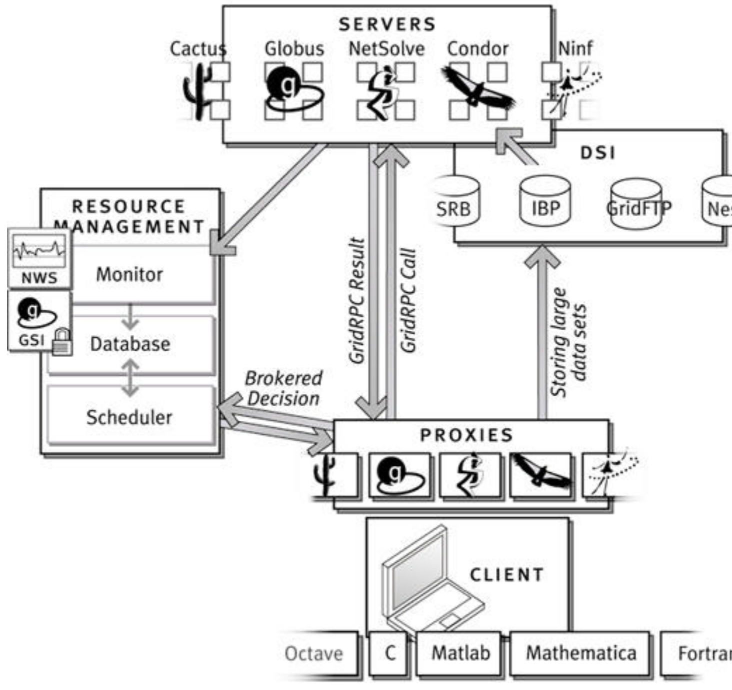

Figure 2: GridSolve in its mature form. Not pictured here is the use of the exNode structure to provide data logistics (sec. 3.4)

Our plan for developing GridSolve falls into 3 stages. At the first stage, we will work with the NES and Grid communities to specify and develop the GridRPC API quickly, and then we will do an initial implementation of it inside the NetSolve framework. Although it is not strictly necessary to use NetSolve as the prototype middleware, doing so will greatly accelerate the process of getting a working version of GridSolve/GridRPC into the hands of its intended user community. NetSolve provides various components that will be helpful in the initial stages of implementation. It has Matlab and Mathematica interfaces that can be re-worked in terms of the GridRPC API. Also, NetSolve has proxies that know how to "speak" to other back-end servers (Globus, for example). Finally, the NetSolve agent has resource management capabilities that an NES requires to make GridRPC useful on the Grid immediately. Initially, the early implementation of GridRPC would consist of a wrapper around the NetSolve C interface. Most of the calls in the GridRPC, in its current outlines, can be easily mapped to the NetSolve interface and the others could be straightforwardly implemented. This work would also allow us to begin to address potential limitations to the GridRPC API somewhat, regarding the lack of certain features required by SCEs, such as the ability to specify the data layout (row major vs. column major). It will be necessary to write a Matlab interface to GridRPC, but since we have the NetSolve Matlab interface to consult, this should not prove difficult. With this initial implementation, the client may only be able to interact with a single back-end since it is unclear whether the proxy limitations can be easily resolved (the limitation is that only one proxy can be used in any one session). By wrapping other elements of NetSolve around the initial implementation of GridRPC, the Grid application community will have a usable prototype of GridSolve in their hands quickly.

In the second phase we will work to get the existing proxies for different Grid back-ends working seamlessly. This could include developing a plugin interface that allows a developer to easily insert support for other back-ends into the middleware without touching the middleware code itself. The most critical problem in this regard is how we address issues of resource discovery, selection, and scheduling. This is a complicated issue for a variety of reasons, not the least of which is the fact that it is an active area of research among groups trying to realize the original vision of the Grid [28]. While we have already done some preliminary experimentation in this area (with the APST for example [22]), before selecting a design and the requisite components, we plan to explore several technologies. To illustrate, here are two likely candidates:

?? Globus Metadata Directory Service (MDS) – MDS provides for static resource allocation and advanced reservation of resources. Resource capabilities are advertised in MDS. The user searches MDS and submits jobs to the appropriate resource. Coarse-grained user directed scheduling.

?? Condor Class-Ad mechanism — Provides mechanisms for resources to advertise available capabilities and for jobs to advertise wanted capabilities. Provides a matching mechanism to pair jobs and resources. Also allows migration of jobs to other matching resources should the current resource be needed for other work. Finer grained automatic scheduling.

Given the established capabilities of the Globus toolkit, it could provide many of the functions we envision needing. We already plan to ensure that GridSolve utilizes the Globus Security Infrastructure [29]. Given the importance of security, we describe this separately below. On a separate note, the use of the Network Weather Service (NWS) [13] for resource monitoring is also and obvious early choice because of it is both capable and increasingly ubiquitous.

For the third stage of the project our goal is to forge key agreements within the NES community on the IDL and wire protocol for GridRPC. Once the back-ends support that protocol, the proxy/plugin becomes superfluous since the client can communicate with any GridRPC-compliant back-end. At this point, we would need to develop a new GridRPC implementation because the current implementation would, presumably, not conform to the agreed upon protocols. We may also develop a GridRPC plugin that would allow clients to call both the new GridRPC-compliant back-ends and non-compliant back-ends. The higher-level services built on top of the NetSolve-based GridRPC would not need modification.

3.3 Security: GrisSolve and GSI

During its first phase, GridSolve will automatically inherit NetSolve's current security model, which is based on the ability to generate access control lists that are used to grant and deny access to the NetSolve servers. The NetSolve developers opted for using Kerberos V5 services because it is one of the most trusted and popular infrastructures for authentication services. The server implements access control via a simple list of Kerberos principal names that usually consist of a name (often a UNIX username) and a realm, which defines the Kerberos "domain". This list of principals is maintained by the NetSolve server administrator and is kept in a private text file, which is consulted by the server. A request to a NetSolve server must be made on behalf of one of those principal names. If the principal name associated with the Kerberos credentials in the request appears in the list, and the credentials are otherwise valid, the request will be honored. Otherwise, the request will be denied.

The NetSolve system supports the interoperation of Kerberized and non-Kerberized components. In either case the client sends a request to the server, and the established protocol dictates that, if required, the server must send an explicit request for authentication. At this point, the client can either abort the transaction (knowing it does not have the proper credentials) or attempt to authenticate itself to receive proper servicing. Currently, there is no mechanism to allow the client to insist on authentication of the server - a Kerberized client will happily talk with either Kerberized or non-Kerberized servers. At this time, the Kerberized version of NetSolve performs no encryption of the data exchanged among NetSolve clients, servers, or agents, nor is there any integrity protection for the data stream.

With the proposed GridSolve activities we will investigate the design and implement a security model based on the Globus Toolkit's Grid Security Infrastructure (GSI) for enabling secure authentication and communication over an open network. We view GSI support as essential in giving GridSolve's target user community full access to the growing resources of the Grid; such support will therefore be one of the main features that distinguishes GridSolve from NetSolve.

The primary motivations behind the GSI are:

?? The need to provide for mutual authentication and single sign-on

?? The need for secure communication (authenticated and perhaps confidential) between elements of a computational Grid.

?? The need to support security across organizational boundaries, thus prohibiting a centrally-managed security system.

?? The need to support "single sign-on" for users of the Grid, including delegation of credentials for computations that involve multiple resources and/or sites.

GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation. The Globus Toolkit's implementation of the GSI adheres to the Generic Security Service API (GSS-API), which is a standard API for security systems promoted by the Internet Engineering Task Force (IETF).

3.4 Support for data logistics in GridRPC and GridSolve: the exNode

A key part of our work with GridRPC and GridSolve are data logistics. The temptation, to keep the interface and software simple, is to keep references to data simple and falling into one of three classes:

?? Direct inclusions of data. These are the arguments marshalled into a serialization format and passed to the server. All RPC engines have to include basic functionalities for direct inclusion of data.

?? Actively transported files. Often it is easier for the server to use standard file primitives and be passed a pointer to a file that is actively transported with the RPC arguments. This allows a highly optimized file transport mechanism such as GridFTP to perform the data movement.

?? Lazily transported files. As an optimization, the client may desire to pass the server a pointer to a file that may or may not exist at the client. The server may then find and download the file at its leisure, again potentially using highly optimized transport mechanisms. This allows for third-party interactions apart from the client and server and also for the server to use data that is close to it.

Each of these classes have difficulties: direct inclusion/serializations can be cumbersome or even impossible for data structures that exceed the size of the receiver's available storage; and file pointers may not allow the receiver to take advantage of locality by moving portions of the data close to it itself, or may not scale across loosely connected sites and different administrative domains. Thus the combination of these three classes gives a rather rich suite of data logistic alternatives, we view them as too limited. We propose an alternative to lazily transported files: exNodes.

exNodes are XML-based serializations that allow one to compose remote allocations of data into something like a file. However, there are key differences between exNode "files" and standard files. The main difference is that a file is composed of disk blocks local to the machine that houses the file. An exNode "file" is composed of allocations (typically IBP because of its great flexibility [30], but others are possible) that can be dispersed across the wide area, and in fact across the world. Additionally, exNodes allow for flexible aggregation of storage allocations in its compositions. For example, exNodes allow for arbitrary replication and partitioning of data.

The exNode is an approach to the representation of data aggregates that allows the use of network storage resources to implement logistical strategies. The exNode represents aggregation and redundancy in a manner that is akin to a file descriptor such as the Unix inode. The exNode thus allows the implementation across the wide area network of logistical strategies including aggregation to increase capacity, caching and staging to improve performance through locality, and redundancy for error detections and correction.

The use of the exNode to implement logistical algorithms means that it is possible to pass an aggregate across the network by simply serializing and sending the exNode data structure rather than the data it represents. In that way, it is like sending lazily transported files. However, unlike the lazily transported files, exNodes *expose* the state of the logical-to-physical mapping that enables logistical data storage and movement and allows it to be transferred between widely distributed network nodes.

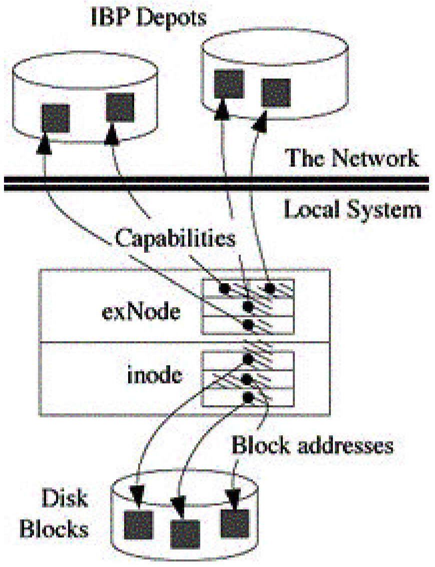

Figure 2: The exNode compared to a Unix inode. While IBP depots are pictured, the exNode data structure supports URIs for a variety of types of storage, such as GridFTP, NeST, and SRB.

Exposing the logistical state of a linear aggregation is a powerful tool for sharing of data without the limitations of either direct inclusion or standard files. However, there is an additional challenge: the logistical management of linear data aggregates in the network requires not only the ability to map linear extents to various storage resources, but it also requires the application of algorithms and additional state information to correctly and efficiently map the logical data structure onto those resources. In the case of direct inclusion, the algorithms and state are stripped off through serialization and rebuilt during deserialization (or, in the case of Java serialization, both are serialized and sent with the data itself as byte code, creating performance difficulties); in the case of file pointers, these are either eagerly transported in their entirety to the client or remain totally at a remote site.

In the case of the exNode, additional state information can be passed as part of the exNode serialization in the form of XML-encoded metadata, but algorithms that interpret that state information must be implemented at each network node. In the case where the state information is highly specialized to the application domain, it may be necessary to employ user-level libraries to interpret it.

We plan to develop user libraries for a class of vector and matrix data representations for data stored in the network that have representations as exNodes with additional metadata that describes the mapping from numerical data structures to storage resources. A number of logistical operations will be implemented, ranging from some very generic operations that apply in non-numeric applications to others that are closely tied to the underlying linear algebra operations.

By its nature as a Grid RPC system for the HPC community, GridSolve will be heavily focused on linear algebra. The data files will often be large, and will need to be staged to the computational backends provided by GridSolve. The most economical way to achieve data staging in a late-binding environment such as GridSolve is to store data files on the network rather than locally. The growing number of network storage solutions dictates that you be able to store data to the network and talk about its location in an architecture-independent manner. The exNode is the answer to this problem.

Furthermore, it will be necessary to assert things about the data referenced by an exNode in a generic way. In particular, we will want to make assertions about input matrices. Enter the first application-specific exNode: the matNode. A matNode is an exNode that is required to have a certain amount of matrix specific metadata. This metadata might include matrix structure, storage orientation (row-major, column-major), dimensions, precision (double, single, arbitrary) and so on. Conceptually, the matNode becomes a generic way to move matrices around the grid in a storage-independent manner.

3.5 GridRPC and APST: A foundation for task parallel applications

Providing a first implementation of GridRPC for this project has two major impacts: (i) it will be key for evolving from the NetSolve model into GridSolve, with more flexible/extensible design as well as more robust implementation; (ii) it will allow other projects to benefit from GridRPC. Among those projects are Grid application execution environments that allow for the transparent scheduling and deployment of applications on Grid resources. One such environment, the AppLeS Parameter Sweep Template (APST) project (http://grail.sdsc.edu/projects/apst) developed by one of the PIs, targets applications consisting of many independent computational tasks [22]. APST uses sophisticated scheduling techniques [8] with the goal of co-locating application data with compute resources in a view to reducing application execution time. Given the structure of the target applications, APST needs an RPC programming model for launching application tasks on many Grid resources.

The overall APST philosophy is to focus on scheduling issues and re-use software provided by the Grid community for all deployment concerns. In its current incarnation, APST "emulates" RPC access to Grid resources by implementing many software layers between its scheduler and several mechanisms that can be used to launch remote processes. Currently APST supports Globus, NetSolve, Condor, and Ssh. Having to implement interfaces to all those systems greatly adds to the complexity of the APST software. Furthermore, that emulation of the RPC programming model is tied to the APST implementation and cannot be easily re-used for other projects. In that sense, APST needs a component that provides an "RPC abstraction".

GridRPC will provide that abstraction. In the first phase of this project we will add GridRPC support to APST and perform validation experiments. The first round of experiments will be qualitative, in order to evaluate whether the GridRPC API is amenable for integration within APST. This will allow us to evolve the GridRPC API with hands-on experience with a Grid application execution environment. We will also characterize the reduction in APST software complexity if GridRPC were the only mechanism needed for launching remote processes. As the proposed GridRPC development makes progress, we will also perform quantitative experiments. The goal will be to measure the overhead incurred when going through the GridRPC layer. We will repeat those experiments for each incarnation of our proposed GridRPC implementation and report on the reduction in overhead.

In the last year of the proposal's time-frame we will package, document, and release a GridRPC-enabled version of APST. That version of APST will allow us to gather user feedback about GridRPC for hardening the implementation, as well as for collecting usage statistics and measure the impact of our work. Finally, our APST/GridRPC integration will allow for APST to seamlessly benefit from future GridRPC development.

3.6 GridSolve as a educational tool for simulation science

Given the community that Gridsolve seeks to serve, an educational strategy is centrally important to its success. As computationally intensive modeling and simulation become staples of scientific life across every domain and discipline, the difficulty of acquiring and sustaining the necessary expertise in scientific computing is becoming increasingly acute for the broad rank and file of students and professionals. While access to necessary computing and information technology has improved dramatically over the past decade, the efficient application of scientific computing techniques still requires specialized knowledge of numerical methods and their implementation in mathematical software libraries that many students, scientists, and engineers working beyond the already strenuous demands of their particular field, must struggle to achieve. To address this problem, we have initiated a project that combines NetSolve and Netlib, the on-line repository of choice for numerical software for science and engineering, to create an active collection of mathematical software for science and engineering education. GridSolve will build on this foundation.

The purpose of this system, called *Active Netlib*, is to provide the kind of rich, highly interactive, and inquiry-based learning environment needed to enable students and application scientists to attain the confident mastery of numerical methods and software libraries their work in this new era requires. While base content for the collection is supplied by Netlib, NetSolve will provide users access to Active Netlib and will make Netlib’s numerical software directly usable by students on servers over the network from Matlab, Mathematica, Fortran, and C interfaces, without requiring them to download and install the software themselves. NetSolve’s *adaptive solver interface*, which guides users in selecting appropriate software in setting parameters correctly, and in interpreting numerical results, is being further extended to provide more detailed feedback to users about the heuristics it uses and the decisions it makes in selecting Netlib software to solve a particular problem. GridSolve will immediately leverage the work that is now being done on NetSolve to provide an active interface to the Netlib collection that supports direct, hands-on learning with its contents. It will also provide new interfaces (e.g. Octave and Scilab) and improved access to a wider array of Grid resources.

3.7 Standards and Community Effort

Standards will play an increasingly important role as Grid computing begins to gain wider acceptance. The standards development process for GridRPC has already begun in the Global Grid Forum and we plan to continue it there. The Global Grid Forum (GGF) is a community-driven set of working groups that are developing standards and best practices for distributed computing ("Grids" and "Metacomputing") efforts including those specifically aimed at very large data sets, high performance computing, and increasingly, those efforts that industry is calling "Peer-to-Peer." GGF represents a merger of three technical communities: those in North America (originally called "Grid Forum"), Asia Pacific, and the European Grid Forum (eGrid).

We will engage the Grid users' community in developing the framework described in this proposal. This will be accomplished by using a similar mechanism as was used for the MPI standardization process. We will hold workshops, we will disseminate the resulting documents and software, and we will engage the user community in the concepts and prototype implementations of the design.

4. Related Work

As a Network Enabled Server (NES) system designed for desktop Scientific Computing Environments (SCE), the GridSolve project embraces a wide range of related topics, from distributed computing generally, to problem solving environments, to distributed storage. A wide variety of RPC systems are currently in experimental or production deployments within the computational science community, including NetSolve [2, 3], Ninf [4], Cactus [5], and NEOS [6]. Some RPC based Grid systems also focus on task parallel and parameter sweep applications, including APST and Nimrod. Only NetSolve, however, has provided significant support to the general purpose SCE community in the form of Matlab and Mathematica clients, and GridSolve will carry on and expand this heritage. Also, while NES systems generally are experimenting with different approaches to distributed storage and data logistics, we expect that the exNode [31] will prove an exceptionally capable and general mechanism for GridSolve and GridRPC to use.

The current effort in Grid computing [28] embraces important systems that do not use the RPC mechanism and are not NES systems. Leading projects that fall into this general category include Globus [24], Legion [32], Condor [25]. All of these projects build on current systems for the definition and management of storage and computing functions, layering their own tools on top of them. A key goal of the GridSolve project is to enable the SCE community to utilize these systems and their resources.

As regards distributed storage infrastructure for managing data logistics, a number of systems have been developed for Grid computing as a layer on top of existing file management systems, such as the Unix file system, HPSS [33] and DPSS [34]. Examples of these systems are Global Access to Storage Services (GASS) [20], the Storage Resource Broker [26], and GridFTP [35]. GASS uses URLs as a global namespace and provides copy-on-open caching of files. SRB maintains its own cross-platform file directory and provides a uniform API for accessing and caching files. Neither of these system provides fine-grained control over data placement, lightweight allocation or unbrokered sharing of storage resources that the combination of IBP and the exNode can supply. But the exNode can work with URIs from any of those sources and represents a general solution.

5. Goals and Planned Deliverables

Year 1	?? Preliminary design of system for binding NetSolve to Globus and Condor as well as integration of NetSolve with other metacomputing paradigms, including resource balancing using Network Weather Service and data transfers using exNode ?? Study and include preliminary interoperations with Matlab, Mathematica, Octave, and SciRun as well as investigate interoperations of NetSolve with other GRPC systems like Ninf, NEOS, Punch, Cactus ?? Establish common set of concepts for interoperations with GridRPC systems ?? Engage the participation of MathWorks and Mathematica in the development of this framework ?? Study and devise solution to NATS

Year 2	?? Hold a workshop with participation from industry partners, government labs and agencies for the development and adoption of these ideas. ?? Develop and deploy interfaces for Globus, Condor, NWS, and the exNode as well as distribute a working version of NetSolve that will interface with Matlab, Mathematica, Octave, SciRun ?? Interact with the Common Component Architecture community as well as engage the Grid community to develop standards for Grid remote procedure calls ?? Develop software toolkit for including additional component into NetSolve ?? Work with the NPACI and the Alliance to foster the use of these enabling technologies in their applications
Year 3	?? Experimentation on a number of applications from the NSF PACI and other computational science projects ?? Implementation and evaluation of fault-tolerance and migration in NetSolve ?? Demonstration of technology to use in various systems like Matlab, Mathematica, etc, and work with these companies for including Grid concepts into their platforms ?? Develop Grid/industry standard for GRPC and interoperability of systems ?? Hold tutorials at conferences like SC'XY, All-Hands meeting for the Alliance and NPACI, and Users Groups for Globus, Condor, MathWorks, and Mathematica

Many of the positive benefits associated with these activities will be derived directly from external visibility. In addition to the normal academic venues (publication in journals and conferences, external talks, and software distribution), we will pursue a number of opportunities to raise its public profile.

?? We will maintain a central Web site, produced and maintained the the NetSolve staff that will include information on all aspects of this activity (research, education, outreach, and tech transfer).

?? We will host technical workshops for the scientific community on topics related to this area of research. These workshops will bring the community together with potential users, collaborators, and other researchers from outside of our activity to discuss problems and progress in the area.

6. Results of Prior

6.1 Jack Dongarra, University of Tennessee

New Technologies (now Advanced Computational Research)

NSF ACI-9876895, NetSolve — An Enabling Environment for Fault Tolerant, High Performance Network Computing

The advances in computer and network technologies that are fueling the evolution of the global information infrastructure are also producing a new vision of how that infrastructure will be used. The concept of a *Computational Power Grid* has emerged as a way of capturing the vision of a network computing system that provides broad access not only to massive information resources, but to massive *computational* resources as well. Such computational power grids will use high-performance networking to connect hardware, software, instruments, databases, and people into a seamless web that supports a new generation of computation-rich scientific computing environments for scientists and engineers. ***NetSolve*** is a software environment for network computing that addresses the complexities of such systems. Its main purpose is to enable the creation of complex applications that harness the immense power of the grid, yet are simple to use and easy to deploy. NetSolve uses a modular, client-agent-server architecture to create a system that is very easy to use. Moreover, it is designed to be highly *composable* in that it readily permits new resources to be added by anyone willing to do so. In these respects, NetSolve is to the Grid what the World Wide Web is to the Internet. But like the Web, the design that makes these wonderful features possible can also impose significant limitations on the performance and robustness of a NetSolve system. This project is exploring the design innovations that push the performance and robustness of the NetSolve paradigm as far as possible without sacrificing the Web-like ease of use and composability that make it so powerful.

Students Supported: Graduate: Sudesh Agrawal, Dorian Arnold, Kiran Sagi, Sathish Vadhiyar, Nathan Garner, Yan Huang

Publications: Published papers: [7, 17, 36-41]

Software release: NetSolve, version 1.4,was released July 2001. The software can be downloaded free of charge from the web at: http://icl.cs.utk.edu/netsolve/. Enhancements and new features include: Distributed Storage Infrastructures, Mathematica Interface, Sparse Matrix Data Structure, Sparse Solver Services, Dense Solver Services, NWS Integration, Globus proxy, Request Sequencing/Data Persistence, Security via Kerberos, Problem Description File Generator, User's Guide/Tutorials, and Transaction Logging Facilities

6.2 James S. Plank and Micah Beck, University of Tennessee

Next Generation Software

NSF EIA-9975015, Logistical QoS through Application-Driven Scheduling of Remote Storage

The major contribution of this work is the definition of an area within computer science called "logistical scheduling," where the placement of data within (perhaps widely) distributed applications is as important a scheduling parameter as the processor and network speeds. As demonstrated by the interest generated in the NetStore 99 Conference (an entire conference dedicated to storage as a network resource), this is an area of research whose time is coming. The focus of this particular NGS grant is to apply these principles in Grid applications. Although the work is young, the preliminary results (as published in the "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments" paper), show that logistical scheduling should indeed be pursued further in the area of Grid computing.

Students Supported: Graduate — Ilwoo Park; Haihang You; Kim Buckner; Eric Wing, Sathish Vadhiyar; Ganapathy Raman; Tinghua Xu; Wael R Elwasif; Xiang Li; Ling Wo; Zheng Yong Zheng; Erika Fuentes. Undergraduate —Stephen Soltesz.

Publications: [8, 17, 22, 30, 39, 40, 42-48]

Software release: Software components of this project (IBP, NetSolve, the Network Weather Service, AppLes) have been made available to the computer science community at http:loci.cs.utk.edu.

6.3 Henri Casanova, University of California San Diego

Information Technology Research

Virtual Instruments: Scalable Software Instruments for the Grid

This project addresses the significant computer science problems that arise from the need to support *steerable* scientific simulations in large-scale Computational Grid environments. To investigate those issues, we are designing and prototyping a *Virtual Software Instrument*. Context for this work is provided by the MCell computational neuroscience application [49]. The virtual instrument will present the user with an intuitive graphical user interface, 3-D rendering capabilities, a computational steering feature, and will transparently use the tremendous amount of computational and storage power that is becoming increasingly available in Computational Grids. At the core of the virtual instrument is a **scheduler,** which takes into account application characteristics, user behavior, available static and dynamic resource information, and computational requirements to assign tasks to resources. The project will have an impact on the Grid computing community as well as the computational neuroscience community by leading it to a new set of disciplinary results. The current research performed by Casanova, Berman, and their research team has led to accomplishments along two fronts. First, they have finalized the design and implemented a prototype of the core Virtual Instrument software. The prototype is being tested and experiments will start in early 2002. Second, Casanova, Berman and their graduate students have developed novel scheduling strategies that are crucial to scheduling scientific simulations such as MCell on the Computational Grid.

Students supported: Graduate — Marcio Faerman, Alan Su

Publications: [7, 47, 50, 51]