Intelligent Networked Scientific Computing

N. Ramakrishnan and E.N. Houstis
Department of Computer Sciences, Purdue University
West Lafayette, IN 47907-1398, USA
email: {naren,enh}@cs.purdue.edu

Amupam Joshi
Department of Computer Engineering and Computer Science
University of Missouri, Columbia, MO 65211, USA
email: joshi@trinetra.cecs.missouri.edu

J.R. Rice and S. Weerawarana
Department of Computer Sciences, Purdue University
West Lafayette, IN 47907-1398, USA
email: {jrr,saw}@cs.purdue.edu

Keywords: Networked Scientific Computing, Resource Selection, Advisory Systems, Software Indexing, GAMS, Intelligent Agents

ABSTRACT

Networked scientific computing systems are becoming increasingly ubiquitous; this represents the future paradigm of a net-centric scenario for computation. However, the network-based paradigm of software usage has not yet become fully automated and completely transparent to the user. Many of the issues impeding this development relate to the provision for intelligent, high-level mechanisms in these systems. In this paper, we present a networked scientific computing scenario in which intelligent mechanisms assist the computational scientist in automatically locating necessary mathematical software for solving his problems in numerical computation. This system is realized by using an algorithm selection methodology to supplement the services of a software indexing system such as GAMS. The idea presented here is demonstrated by implementing such a strategy for elliptic partial differential equation solvers.

INTRODUCTION

Scientific software has progressed from simple FORTRAN codes to being comparable to some of the most complex systems created by mankind. We are witnessing the rapid development of network-based scientific software servers, libraries, repositories and problem solving environments (PSEs), fueled by advances in scalable high performance computing, the Internet and high level environments for doing computational science. In this new scenario, vital pieces of software and information required for a computation are distributed across a network and need to be identified and ‘linked’ together at run time; this implies a “net-centric” paradigm for scientific computing. Several systems are already in place that (i) provide access to software libraries over the network such as Netlib [6] (ii) enable problem solving environments (PSEs) to be utilized over the web (WWW) – Net/ELLPACK [2], NetSolve [5] etc., (iii) index software modules from repositories according to functionality – GAMS [1] and (iv) automate access to databases of test problems, data from performance evaluation such as the Matrix Market [3].

However, from the perspective of the future networked computing scenario, the state-of-the-art is rather primitive. Several challenging research issues remain to be addressed:
• Existing systems assume that the choice of method (algorithm) and platform to solve a given scientific problem is fixed \textit{a priori} (static) and that appropriate code is located, downloaded, compiled and linked to yield static programs. Thus, scientists are very often faced with the onerous task of selecting suitable software and hardware for the problem at hand in the presence of practical constraints on accuracy, time and cost. We refer to this as the \textit{resource selection problem}. This also encompasses issues of how a user specifies problem queries, extracts feature/content information and infers results. This problem becomes even more critical in a networked scenario where the computational resource choices increase several-fold, both in number and heterogeneity. A system to ‘adaptively’ select software and hardware to confirm to the performance requirements set by the user is called for.

• The performance evaluation of scientific software is another important issue that needs to be addressed in this redefined scenario; the ‘network’ can link together several test suites, repositories and make them available as \textit{Software Test Servers}, which provide standardized facilities to conduct software testing. The advantage to creating such systems will be the measurement, control and evaluation of several components or aspects of \textit{software quality}, such as efficiency, accuracy, cost etc. Software test servers will also aid in the data gathering requirements intrinsic to the algorithm selection problem.

In the proposed paradigm, which we call Intelligent Networked Scientific Computing, tools will exist in the PSEs to assist the computational scientist in every stage of the solution process — high level, domain-specific problem modeling and formulation, selection of methods to solve a given problem, and pointing the user (in this case, a scientist) to appropriate sites to obtain and download mathematical software that implement the advised methods. While this vision is yet to be realized in its grand totality, important pieces of its infrastructure are already in place in the form of PSEs such as \texttt{/\textbackslash ELI PACK} \cite{7}. This paper demonstrates results towards achieving the goal of Intelligent Networked Scientific Computing in the context of the resource selection problem.

The problem of algorithm/software selection in a networked setting, as described above, is an important one and crucial to the current scientific computing landscape. Simply stated, this entails selecting a good algorithm (software module) to solve a given scientific problem with performance constraints on the solution process. At Purdue, we have focused on building automated advisory systems (under the PYTHIA project) that perform algorithm (software) selection within problem solving environments for scientific computing. Our initial case studies focus on the areas of partial differential equations (PDEs) and numerical quadrature. The results in this paper will be presented in the context of the former domain—elliptic partial differential equations. PYTHIA currently uses a collaborative methodology to determine a good solver for a particular PDE problem. It however, doesn’t direct attention to where these routines/algorithms can be obtained. A popular location for obtaining these is the GAMS (Guide to Available Mathematical Software) system \cite{1} at the National Institute of Standards and Technology (NIST) which provides a virtual repository and cross-index of mathematical software for all major areas of numerical and scientific computation. While it enables the scientist to locate software routines appropriate for his needs, the system, however, doesn’t provide advice on which of these routines is the best or ‘most reasonable’ one to use for the given problem at hand. In this paper, we will demonstrate how the GAMS and the PYTHIA systems can be integrated to provide complete software selection for a class of problems.

The rest of the paper is organized as follows: We first describe the GAMS system. The collaborative PYTHIA methodology to select solvers for elliptic PDEs is described in detail next followed by an elaboration of the mechanisms used to interface PYTHIA with the GAMS system. Pointers for future work and conclusions are provided last.

**THE GAMS SYSTEM**

As mentioned before, GAMS \cite{1,4} is a virtual mathematical software repository system that provides
Figure 1: Architecture of the GAMS System. The components that serve to interface to PYTHIA are also shown.

convenient access to thousands of software modules physically distributed among several Internet repositories. It promotes easy access and also encourages software reuse in the current distributed computing environments. It supports search for appropriate modules by problem, keyword or name. Objects that can be downloaded include abstracts for the routines, documentation, examples and source code. The reason why it is referred to as a virtual repository is that it provides an extensive cross-index of other repositories in scientific computing. This should be contrasted to maintaining all the software modules in-house.

In addition to providing the above mentioned services, GAMS's main contribution to mathematical software lies in the tree structured taxonomy of mathematical and software problems used to classify software modules. This taxonomy extends to seven levels and provides a convenient interface to hone in on appropriate modules. For example, the problem class I refers to modules catering to differential and integral equations, I2 indexes to modules about PDEs, I2b caters to elliptic boundary value problems, I2b1 refers to the linear kind of these problems and so on. This taxonomy indexes about 10,000 problem solving modules from over 100 software packages maintained at four software repositories. Much of this software is in FORTRAN format for problems from linear algebra, differential equations, number theory, optimization, statistics, interpolation, approximation, symbolic computation, geometry and most other areas in computational science. Among the packages cataloged are the collected algorithms of the ACM (TOMS), IMSL, NAG, PORT, and SLATEC libraries, BLAS, EISPACK etc.

Several interfaces to the GAMS system are available, including a HTTP gateway, a command line interface, a X11 graphical user interface and a Java powered WWW client. The main structural components of the GAMS system are as follows:

1. The GAMS Database, a relational database based on the FORTRAN RIM database management system that provides the necessary support for the GAMS repository model.

2. A Relational Database Interface to provide the standardized database management calls,

3. The GAMS Interface that implements a set of procedure calls to achieve the repository functionality,

4. The GAMS Server, which is a stateless network server implementing the repository services using simple protocols,

5. GAMS Client Interfaces that access the GAMS Server, in addition to providing the functionality of the GAMS Interface and

6. User Interfaces that provide the management at the user level.

This can be easily visualized as shown in Fig. 1 (The remaining components in the figure are explained later). GAMS functions in an interactive mode, guiding the user from the top of a classification tree
to specific modules as the user describes the problem in increasing detail. During this process, many features of the problem are determined, indirectly from the user. However, at the ‘leaves’, there still exist several choices of algorithms for a specific problem. The entire GAMS tree has around 750 nodes, and is quite elaborate.

THE COLLABORATIVE PYTHIA SYSTEM

PYTHIA [9] is a knowledge based system that selects scientific algorithms to achieve desired tasks in computing. It determines a near-optimal strategy (i.e., a solution method and its parameters) for solving a given problem within user specified resource (i.e., limits on execution time and memory usage) and accuracy requirements (i.e., level of error). While the ideas behind PYTHIA are quite general, our current implementations operate in conjunction with systems that solve (elliptic) partial differential equations, such as the ELLPACK and //ELLPACK PSEs [7] developed at Purdue. The methodology of PYTHIA is to gather performance information about PDE solvers on standardized test problems and use this data plus feature information about PDE problems to determine good algorithms to solve the PDEs. It can be immediately observed that the efficacy of this method is dependent on the methods and the problems set used in the performance evaluation stage.

The collaborative PYTHIA methodology [8] recognizes the fact that there are many different types of PDEs and most scientists tend to use only from a limited subset of them encountered in their application domain. Hence, the approach taken is to create several different PYTHIA ‘agents’ each of which has information about a certain class of PDE problems and can predict an appropriate solver for a given PDE of the class.

A brief discussion about the internal functioning of a single PYTHIA agent is in order here. PYTHIA ranks various solvers on test problems based upon different constraints on accuracy and time. The strategy of PYTHIA, then, is to compare a given problem to the ones it has seen before, and then use its knowledge about the performance characteristics of prior problems to estimate those of the given one. Together with a good method to solve a given problem, PYTHIA also provides a factor of confidence that it has in the recommended strategy. For example, a typical PYTHIA output for a PDE problem would be: “Use the 5-point star algorithm with a 200 x 200 grid on an NCube/2 using 16 processors: Confidence = 0.85”. In its current preliminary implementation, a PYTHIA agent accepts as input the description of an elliptic PDE problem, and produces the method(s) appropriate to solve it.

The PYTHIA project web pages at http://www.cs.purdue.edu/research/cse/pythia provide information about this collaborative PYTHIA methodology and facilities to invoke it remotely. At the outset, there is a facility to provide feature information about a PDE problem. In particular, there are forms that enable the user to provide details about the operator, function, domain geometry and boundary conditions. Once these details are given, the information is submitted to a ‘central’ PYTHIA agent called ‘C-PYTHIA’ that performs further processing. It first classifies the given PDE problem into several categories of problems. These classes are defined based on the properties possessed by the solutions of the PDEs and are ‘Solution–Singular’, ‘Solution–Analytic’, ‘Solution–Oscillatory’, ‘Solution–Boundary–Layer’ and ‘Mixed–Boundary–Conditions’. Having classified the problem into one or more of these classes, the PDE is taken to an appropriate PYTHIA agent for this class of problems, which in turn predicts an optimal strategy and reports back to the user.

PYTHIA, thus proves to be very useful in providing assistance about what method (and associated parameters) to use to solve a given PDE. It however, does not provide information about where to obtain the software modules for this purpose.

THE PYTHIA–GAMS INTERFACE

As would be obvious from the above discussion on PYTHIA and the GAMS system, their services are
complimentary and can be quite neatly combined. In this section, we describe an interface between the collaborative PYTHIA multi-agent system and the GAMS repository. For the purposes of the discussion here, it suffices to note that the GAMS categories \( 12b1a1a, 12b1a3 \) are the ones pertinent to the software routines that PYTHIA has information about. Some of these routines are the methods “9”, “9,jorder=2” etc. More information about these can be obtained by browsing the GAMS web pages [1] about the ELLPACK modules for the above mentioned classes of problems. It should be observed that we have made extensive use of this GAMS-indexed software during the performance evaluation stage of designing the PYTHIA agents. Thus, the interface between these systems is a very pertinent one.

As shown in Fig. 1, the connection between the PYTHIA system and GAMS is achieved by means of a proxy program that is configured for this purpose. If this program is properly installed, any requests sent to the GAMS server about methods for categories such as \( 12b1a1, 12b1a3 \) would be transferred over to the C–PYTHIA agent by the proxy. Thus the proxy serves to enforce acceptability criteria on the user’s input. At this point, C–PYTHIA in turn requests input from the user about the PDE problem characteristics and predicts a method to solve the given PDE. In addition to displaying this information, it also provides a hyperlink back to the GAMS system to download the appropriate software modules. It is to be noted that at every point of time, the system provides the facility to return back to the original GAMS hierarchy if the user is not interested in performing resource selection for a class of problems. This serves as a convenient scheme to switch back and forth between the GAMS and PYTHIA systems.

**CONCLUSION**

In this paper, we presented an interface between PYTHIA, an automatic advisory system and GAMS — a cross-index of mathematical software. This connection is but a natural one and it is our vision that such schemes will help in realizing the dream of ‘Intelligent Networked Scientific Computing’. There are several directions in which this work could be extended. The GAMS system now has a new Java-enabled interface and it is possible to customize the combined system further to exploit this feature. Also, from software selection, one could take the paradigm further to provide automated services for software delivery. In other words, the GAMS server could be used to provide up-to-date copies of relevant software on an as-needed basis.

**References**


