1997

Network Servers for Multidisciplinary Problem Solving

Anupam Joshi

Sanjiva Weerawarana

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

John R. Rice
Purdue University, jrr@cs.purdue.edu

Shahani Markus

Report Number:
97-023

NETWORK SERVERS FOR MULTIDISCIPLINARY PROBLEM SOLVING

Anupam Joshi
Sanjiva Weerawarana
Elias N. Houstis
John R. Rice
Tzvetan T. Drashansky
Shahani Markus

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR 97-023
April 1997

Network Servers for Multidisciplinary Problem Solving

Anupam Joshi, Sanjiva Weerawarana, Elias N. Houstis, John R. Rice,
Tzvetan T. Drashansky and Shahani Markus

Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398.

Abstract

The evolution of the Internet into the Global Information Infrastructure (GII) is impacting many institutions of life in general, and, the way we view computing in particular. This future computational and communication infrastructure will allow computing everywhere. Learning and training simulators will be part of any classroom and laboratory. The very concept of classroom, laboratory and individual workplace will change; they will become virtual places based on an array of multimedia devices [1]. We are developing the software architecture of such an environment and are working on related research issues. The advent of such environments will affect the process of prototyping, which is part of every scientific inquiry, product design, and learning activity. The new economic realities require the rapid prototyping of manufactured artifacts and rapid solutions to problems with numerous interrelated elements [2]. This, in turn, requires the fast, accurate simulation of physical processes and design optimization using knowledge and computational models from multiple disciplines in science and engineering. Thus, the realization of ***rapid multidisciplinary problem solving*** *or* ***prototyping*** *is the new grand challenge for Computational Science and Engineering (CS&E) [3]. We refer to a software realization of multidisciplinary prototyping as a* ***Multidisciplinary Problem Solving Environment*** *(MPSE).*

We describe in this paper research that is needed (and that is on-going at Purdue) to formulate and develop a mathematical and software framework for MPSEs including the tools, enabling technologies, and underlying theories needed to support physical prototyping in the classroom, laboratory, desk, and factory. The MPSE will utilize the GII facilities. It will be adaptable and intelligent with respect to end-users and hardware platforms. It will use collaborating software systems and agent based techniques to build demonstration MPSEs which run on heterogeneous, networked platforms. It will allow wholesale reuse of software and provide a natural approach to parallel and distributed problem solving.

The advent of optical, ATM, and wireless hardware communication technologies and distributed computing infrastructure like the Web (WWW) will make the concept ***"The Network is the Computer"*** *a reality. We assume that* ***"The Net"*** *interconnects computational units (ranging from workstations to massively parallel machines) and physical instruments. Moreover, we view the WWW infrastructure as an object-oriented operating system kernel that allows the development of an MPSE as a distributed application utilizing resources and services from many sources.*

1 Introduction

The growth of computational power and network bandwidth suggests that computational modeling and experimentation will continue to grow in importance as a tool for big and small science. In this scenario, the design process will operate at the scale of the whole physical systems with a large number of components that have different shapes, obey different physical laws and manufacturing constraints, and interact with each other through geometric and physical interfaces. We consider multiagent problem solving systems in scientific computation and, in particular, strategies that allow scientific computing systems to solve problems cooperatively over the network. Scientific computation involves numerical models of real world phenomenon, and these models are becoming increasingly complex. Heretofore, scientific computing systems have been developed as standalone systems targeted to a particular class of applications modeled in a somewhat homogeneous, generic way. However, the real world consists of physical objects that interact with each other. The overall behavior of a physical system is a result of these interactions. Consider an automobile engine. Its design involves the domains of thermodynamics (gives the behavior of the gases in the piston-cylinder assemblies), mechanics (gives the kinematic and dynamic behaviors of pistons, links, cranks, etc.), structures (gives the stresses and strains on the parts) and geometry (gives the shape of the components and the structural constraints). The optimal engine design emerges from the interaction of these different domain-specific analyses. The different domains share common parameters across interfaces but each has its own parameters and constraints. We refer to these multi-component based physical systems as multidisciplinary applications.

Realizing this scenario requires the development of new algorithmic strategies, as well as software for managing the complexity and harvesting the power of the expected high performance computing and communication (HPCC) resources. It requires technology to support programming-in-the-large and to reduce the overhead of HPCC computing. Our research aims to identify the framework for the numerical simulation of multidisciplinary applications and to develop the enabling theories and technologies needed to support and realize this framework in specific applications. Our design objective is to allow the "natural" specification of multidisciplinary applications and their simulation with interacting software components through mathematical and software interfaces across networks of heterogeneous computational resources.

A physical system in the real world normally consists of a large number of heterogeneous components. The behavior of each component is modeled mathematically by a system with various formulations for the geometry, interface/boundary/linkage and constraint conditions in many different geometric regions. It is difficult to imagine creating a monolithic software system to model accurately a complicated real problem with hundreds of diverse parts and dozens of physical phenomena. Therefore, one needs a software framework which is applicable to a wide variety of practical problems and allows for software reuse. Most physical systems and manufactured artifacts can be modeled as a mathematical network whose nodes represent the physical components in a system or artifact. Each node has a mathematical model of the physics of the component it represents and a solver agent [4] for its analysis. Individual components are chosen so that each node corresponds to a simple mathematical problem defined on a regular geometry. There exist many standard, reliable solver systems that can be applied to these local node problems. Some nodes in the network correspond to interfaces that model the interaction of the parts in the global model. To solve the global problem, the local solvers collaborate with each other to resolve the interface conditions. An interface controller or

mediator agent [4] collects parameters and constants from neighboring subdomains and adjusts these to better satisfy the interface conditions. This "network" abstraction of a physical system allows us to build a software system which is a network of well defined collaborating software parts using interfaces. Some of the theoretical issues of this methodology have been addressed in [6] for the case of collaborating partial differential equation (PDE) models. The results obtained so far verify the feasibility and potential of network-based prototyping.

Network-based computing [7] is likely to be the paradigm of the future for solving complex scientific problems. This viewpoint leads naturally to distributed, collaborating, agent based methodologies. This, in turn, leads to very substantial advantages in both software development and quality of service as follows. Servers export to the user's machine an agent that provides an interactive user interface built on top of the standard services of the Net. The bulk of the software and computing is done at the server's or third party sites using software tailored to a known and controlled environment. The server site can, in turn, request services from specialized resources it knows, e.g., a commercial PDE solver, a proprietary optimization package, a 1000 node supercomputer, an ad hoc collection of 122 workstations, a database of physical properties of materials. Each of these resources is contacted by an agent with a specific request for problem solving or information service. Again, all this collaboration is built on standard Network services. All of this can be managed without involving the user, without moving software to arbitrary platforms, and without revealing source codes. This approach also allows software reuse for easy software update and evolution, things that are extremely important in practice. For example, each new automobile engine normally results in a new software system. Recreating such a system could easily take several months or years. In contrast, the execution time to perform the required computation might only be a few days. Also, a new engine design often incorporates parts from old designs. Notice that such a physical change corresponds to replacing, adding, or deleting a few nodes in the network with a corresponding change in interface conditions, and can be easily done. In such applications each physical component is viewed both as a physical object and as a software object. In addition, this mathematical network approach is naturally suitable for parallel and distributed computing as it exploits the parallelism in physical systems. One can handle issues like data partition, assignment, and load balancing on the physics level using the structure of a given physical system. We believe that this networked multiagent approach is natural and direct. It is facilitated by the existence of a multitude of standalone scientific problem solving agents that can effectively model and solve for the behavior of fairly simple, homogeneous physical phenomenon. Some of these agents are no more than subroutine libraries in the classical sense, others are very much larger and more sophisticated Problem Solving Environments (PSEs) [8]. We believe that this approach will allow locally interacting problem solving agents to decompose a complex computation into a distributed collection of self contained computations. It also allows high scalability.

2 Research Issues

2.1 Domain Specific PSEs

Even in the early 1960s, scientists had begun to envision problem solving environments not only powerful enough to solve complex problems, but also able to interact with users on human terms. The rationale of our research is that the dream of the 1960s will be the reality of the 1990s: high performance computers combined with better algorithms and better understanding of computational science have put PSEs well within our reach.

A PSE is a computer system that provides all the computational facilities needed to solve a target class of problems. These facilities include advanced solution methods, automatic selection of appropriate methods, use of the application's language, selection of appropriate hardware and powerful graphics, symbolic and geometry based code generation for parallel machines, and programming-in-the-large. The *scope* of a PSE is the extent of the problem set it addresses. This scope can be very narrow, making the PSE construction very simple, but even what appears to be a modest scope can be a serious scientific challenge. For example, a PSE for a specific application has a narrow scope, but is still a complex challenge as it requires us to incorporate both a computational model and an experimental process supported by physical laboratory instruments. We are also creating a PSE called PDELab for partial differential equations. This is a far more difficult area than a specific application problem and the resulting PSE will be less *powerful* (less able to solve all the problems posed to it), less *reliable* (less able to guarantee the correctness of results), but more *generic* (more able to "parse" the specifications of many PDE models). Nevertheless, PDELab will provide a quantum jump in the PDE solving power delivered into the hands of the working scientist and engineer.

A substantive research effort is needed to lay the foundations for building PSEs. This effort should be directed towards i) a PSE kernel for building scientific PSEs, ii) a knowledge based framework to address computational intelligence issues for PDE based PSEs, iii) infrastructure for solving PDEs, and iv) parallel PDE methodologies and virtual computational environments. The description of these proposed tasks can be found in later in this document.

2.2 MPSEs for Prototyping of Physical Systems

In simple terms, an MPSE is a framework and software kernel for combining PSEs for tailored, flexible multidisciplinary applications. A physical system in the real world normally consists of a large number of components which have different shapes, obey different physical laws and manufacturing/design constraints, and interact through geometric and physical interfaces. Mathematically, the physical behavior of each component is modeled by a PDE or ODE system with various formulations for the geometry, PDE, ODE, interface/boundary/linkage and constraint conditions in many different geometric regions. In the case of complicated artifacts such as the automobile engine, which has literally hundreds of odd shaped parts and a dozen physical phenomena, it is difficult to imagine creating a monolithic software system to model accurately such a complicated real problem. Therefore, one needs an MPSE mathematical/software framework which, first, is applicable to a wide variety of practical problems, second, allows for software reuse in order to achieve lower costs and high quality, and, finally, is suitable for some reasonably fast numerical methods. Most physical systems and manufactured artifacts can be modeled as a ***mathematical network*** whose nodes represent the physical components in a system or artifact. Each node has a mathematical model of the physics of the component it represents and a *solver agent* [4] for its analysis. Individual components are chosen so that each node corresponds to a simple PDE or ODE problem defined on a regular geometry.

The network abstraction of a physical system or artifact allows us to build a software system which is a network of collaborating well defined numerical objects through a set of interfaces. Some of the theoretical issues of this methodology have been addressed in [5][6] for the case of collaborating PDE models. The results obtained

so far verify the feasibility and potential of network-based prototyping.

MPSEs must exploit and build on the new technologies of computing. By the time MPSEs are operational, the advances in computing power and the communication infrastructure will allow ubiquitous high performance computing, i.e., every where by every one. The designs for MPSE must be application and user driven. An MPSE must simultaneously minimize the effort and maximize the solution power delivered to researchers, engineers and scientists, students, and trainees. We do not restrict our design just to use the current technology of high performance computers, powerful graphics, modular software engineering, and advanced algorithms. We see MPSE as delivering problem solving services over *the Net*. This viewpoint leads naturally to collaborating, agent based methodologies. This, in turn, leads to very substantial advantages in both software development and quality of service as follows. We envision that a user of MPSE will receive at his location only the user interface. Thus the MPSE server will export to the user's machine an agent that provides an interactive user interface built on top of the standard services of *the Net*. The bulk of the software and computing is done at the server's site using software tailored to a known and controlled environment. The server site can, in turn, request services from specialized resources it knows, e.g., a commercial PDE solver, a proprietary optimization package, a 1000 node supercomputer, an ad hoc collection of 122 workstations, a database of physical properties of materials. Each of these resources is contacted by an agent from the MPSE with a specific request for problem solving or information service. Again, all this collaboration is built on standard Network services. All of this can be managed without involving the user (if s/he so desires), without moving software to arbitrary platforms, and without revealing source codes.

3 Conclusion

Realizing this MPSE technology requires research advances both in the general structure and implementation area and in more specific areas from the applications we consider. For example, we must design and create the tools that allow the MPSE agents to collaborate over *the Net*. We must create a flexible and general methodology for interfacing large and heterogeneous software systems. We must advance the computational models currently used in application areas and map them to networked virtual parallel environments. For example, in on-going work, we are defining and creating a library (PDEPack) of PDE solving modules including classical numerical linear algebra routines and modern distributed mesh generators and solvers. We must advance the practice of identifying high level characteristics of partial differential equations and their use for selecting solutions methods and computing platforms. We must integrate the physical (wet) laboratory equipment with simulated mathematical models to create a unified software/hardware environment for studying application areas. We believe that by 2001 the vision of MPSEs on the Net can be realized by addressing the problems outlined in this paper.

4 References

[1] W.R. Johnson, Jr., Anything, Anytime, Anywhere: The future of networking, in *Technology 2001: The future of Computing and Communications*, D. Leebaert, editor, MIT Press, Cambridge, MA, (1992).

[2] J.T. Vesey, Speed-to-Market distinguishes the new competitors, *Research Tech. Mgmt.*, 34 (1991), 33-38.

[3] R. G. Voigt, Requirements for multidisciplinary design of aerospace vehicles on high performance computers, ICASE Report No. 89-70, Sept. 1989, 9 pages.

[4] T. Drashansky, A. Joshi and J.R. Rice, *SciAgents-* An Agent Based Environment for Distributed, Cooperative Scientific Computing, CSD-TR-95-029, Department of Computer Sci-

ences, Purdue University. Submitted for publication.

[5] S. McFaddin, *An Object Based Problem Solving Environment for Composite Partial Differential Equations*, Ph.D. thesis, Department of Computer Sciences, Purdue University, 1992.

[6] M. Mu and J.R. Rice, Modeling with collaborating PDE solvers – Theory and practice. *Contemporary Mathematics*, **180**, (1994), 427–438.

[7] S. Weerawarana, E.N. Houstis, J.R. Rice, M.G. Gaitatzes, S. Markus, A. Joshi, "Web //ELLPACK: A Networked Computing Service on the World Wide Web", CSD-TR-96-011, Department of Computer Sciences, Puurdue University, 1996. Submitted for publication.

[8] E. Gallopoulos, E.N. Houstis, and J.R. Rice, Computer as thinker/doer: Problem solving environments for computational science. *IEEE Comp. Sci. Engr.*, **1** (1994), 11–23.