SchedSP: Providing GRID-enabled Real-World Scheduling Solutions as Application Services

G. Goulas, E. Housos

Computer Systems Laboratory, Department of Electrical and Computer Engineering, University of Patras
Rio, Patras, 26500, Greece

Abstract

In this paper, an Application Service Providing framework for the provision of real world scheduling applications as services, named SchedSP, is presented. The main assumption of this paper is that the main business tendency is for distant users to have relatively small computational capabilities with web access being the main remaining capability. Since scheduling applications are computationally intensive and the demand for such applications is growing, it appears that a platform as SchedSP will be a good model for the computer applications environment of the future. The architecture of SchedSP is presented and analyzed. Its interaction with GRID resources and basic scheduling applications background is also presented.

KEYWORDS: XML Protocol and HTTP messaging; Scheduling Applications; Application Service Provision; XML Web Services, GRID

1. INTRODUCTION

Real world scheduling applications require powerful computers due to their combinatorial nature and the immense set of possibilities that need to be examined for every given situation. The initial installation, maintenance and upgrading of the various software and hardware components is only possible for very large corporations with significant IT staff. In order for these tools to be utilized by smaller enterprises, which might indeed be quite distributed, a new computational model is needed. This requirement together with the fact that the Internet has expanded in quality and popularity led to the creation of the Application Service Providing computational model in order to allow the distant user to have access to powerful applications without the need of heavily investing in hardware, software and human resources. The Application Service Provider (ASP) offers applications over the Internet to its clients for use on its computational resources using a flexible licensing, credit and/or payment scheme. The goal of the ASP approach is to reduce the computer requirements of the clients to just a web browser.

In this paper, a software framework for use as an Application Service Provider that emphasizes in real world scheduling solutions, named SchedSP, is presented. For the computational needs of SchedSP, local computational resources organized in a GRID manner are utilized, and when this is not enough, GRID resources of another organization can be utilized, in the model of Computational Infrastructure Service Provisioning (CISP) [1]. SchedSP is designed to allow the system to grow both in terms of concurrent user support and in terms of the addition of new scheduling applications. The modular design of the architecture provides support for the creation and modification of input data, and view of the results of the scheduling solutions provided in a manner similar to the traditional use of a scheduling application that resides on a local computer system.

Initially, the plan is for SchedSP to provide the scheduling applications, which have been developed over the years in the Computer Systems Laboratory of the University of Patras, as application services over the Internet. These applications include several particular domains of real world scheduling problems, like university timetabling [2], high school timetabling [3, 4], personnel scheduling [5], bus driver scheduling [6], and airline personnel scheduling [7, 8, 9, 10].

Several published papers discuss various methodologies for the organization and architecture of a service provisioning framework. In Vinci [11], a service-oriented architecture for supporting the development of web

applications is proposed, using XML [12] document exchange, which is in several ways similar to the SchedSP philosophy. Middleware for Method Management presents a web based system for the sharing of statistical computing modules [13, 14, 15]. The use of XML for invocation of remote procedures has been proposed in several documents, reports and standards like the XML RPC [16] and the XML Web Services stack of SOAP [17], WSDL [18] and UDDI [19]. Products that utilize the XML Web Services specifications include Microsoft .NET framework [20] and the Sun ONE [21] architecture.

After a quick introduction on scheduling applications, the CISP concept is presented. In the following section, detailed requirements of the basic SchedSP characteristics are presented, followed by an extensive discussion of the Service Providing Components architecture of SchedSP.

2. A QUICK INTRODUCTION TO SCHEDULING APPLICATIONS

A real-world scheduling application takes in most cases input files with resources that need to be matched with other resources, which are in most cases humans and machines, in an optimal manner and obeying a set of rules and constraints. The human resources can be for example teachers in the case of school timetabling, or personnel in the case of more general personnel scheduling. The other resources in the above examples would be classes and classrooms or work shifts. The main characteristic of the problems solved by most scheduling applications is the complexity of the solution process which is of exponential order of the problem size. This complexity is referred to as combinatorial explosion and for this reason the results obtained from most real-world scheduling problems are near optimal. A mathematical expression that evaluates the quality of the present allocations of the various resources, often referred to as the objective function, is repeatedly calculated during the solution process. Many aspects of the overall quality of the result are directly dependent on the expressions that constitute the objective function.

From this discussion, it is obvious that the scheduling problem is transformed to a mathematical model that is solved using various optimization techniques. A common need of these techniques involves the solution of linear or integer optimization problems with constraints. For this problem, several libraries and tools can be utilized. If the scheduling application depends on such tools, they need to be installed on the host computer. In addition, scheduling applications typically require considerable computational and memory resources. These computational necessities along with the fact that such scheduling applications in many instances are not used every day in order to justify the maintenance and support of private solutions qualifies these applications as good candidates for use through the newly developed application service providing deployment scheme.

3. GRID-ENABLED COMPUTATIONAL INFRASTRUCTURE SERVICE PROVISIONING

Computational Infrastructure Service Provider (CISP) is a special application service provider case whose purpose is to deliver at a fee its computational resources to various interested parties, over the Internet. The CISP concept is further illustrated in Figure 1. The Application Service Provider on the middle of the figure might have its own resources for serving its customers, but for various reasons the additional computational power of a CISP can be also utilized. The benefits of the CISP concept are that the CISP approach often results in an organization structure that best utilizes the computational resources by making them seamlessly available to larger set of users. In addition, this model appears to be less demanding in terms of system administration requirements. It appears easier for an ASP to cooperate with a CISP in order to share risk and minimize the required investment.

FIGURE 1: An ASP and CISP co-operation

A prototype system that uses of the CISP concept tightly linked with the GRID type organization of computational resources has been recently developed under the name of "PLEIADES" [1, 22]. An overview of PLEIADES is shown in Figure 2. The cornerstone functionality of PLEIADES was to provide an iNOW over a Web-based interface and a Computer-to-Computer Interface. iNOW stands for Internet-based Networks-Of-Workstations, which is a more generalized view of a local area network based NOW [23]. In the iNOW situation processors located around the globe can be utilized and form such a virtual parallel/distributed machine with the only requirement being their continuous connection to the Internet.

FIGURE 2: PLEIADES system overview

The iNOW functionality of PLEIADES is based on the voluntary donation of machines concept. If the users would like to share their resources in the exchange of the right to use other resources when they need them, resources in volumes that it could be impossible to have on their own can be attained. The CISP part of PLEIADES is an XML-based interface where messages are transferred over the HTTP protocol. The CISP interface is capable of serving most of the functionality that is available through the PLEIADES Web GUI. Since the ASP is able to create directories, files and applications in the PLEIADES file space.

The PLEIADES system provides the services shown in Figure 3. The first three services of Figure 3 include the submission of a job to PLEIADES for execution and monitoring. In order to store the input and output files of

the job, the file management service is used. The Cross compile is a service that creates executables for any of the PLEIADES available platforms.

FIGURE 3: PLEIADES core services

4. SCHEDSP REQUIREMENTS AND FEATURES

The main goal that initiated the SchedSP project was the design and implementation of a framework for the generation of real-world scheduling solutions as application services over the Internet. Having this as the primary goal of this work all the requirements and application features of this paper should follow. SchedSP will act as a special type of middleware between the computational resources that compute real-world scheduling solutions and their corresponding client interfaces. The particular client interfaces should be based on the thin client idea in the sense that most of the computational work should be planned for execution on the server side. SchedSP should provide the ability to seamlessly execute remote scheduling applications using a set of input and output files. In addition, it also should be able to create a terminal emulation environment in order to control the application, as a user would do using the keyboard, and gather its output from the screen. More over, since scheduling applications require significant processing and memory resources, it appears reasonable to utilize GRID based computational resources in order to provide and sustain a satisfactory service level for its users. SchedSP must be able to utilize the CISP services of the PLEIADES system.

The SchedSP interface that a thin client sees must be kept simple in order to facilitate the creation of new clients and complete scheduling services. A well-defined programmatic interface using open standards that are both widely adopted and programming language-neutral is required. The transfer of message requests and responses is carried out by HTTP. The choice of the HTTP has been made because it is the only protocol able to overcome firewalls and proxies. The client use SchedSP is not very different in concept from the interaction of a user with a web-based application, and that denotes that when HTTP activity is allowed or denied, SchedSP activity should be treated the same way. The messages transferred using HTTP should be easily understood and modified by any language without any compatibility problems. Compatibility on the data format level between different platforms can be accomplished either by the use of special libraries like XDR or by the use of Java for the whole application. Another technique that guarantees compatibility that appears to gain momentum in recent years assumes that the XML data format standard is used for all interactions among the various components of the overall SchedSP application. The use of the new XML based SOAP standard [17] will be examined in the future although it appears that XML alone is sufficient for the particular application.

Since scheduling applications require the use of input and output files, a method for supplying such sets of files to the applications is needed. A virtual file space service should be provided by SchedSP in order for the users to create and edit the necessary files. It is a strict requirement of the SchedSP that the inputs and outputs files of a particular problem solution remain connected for future reference and consultation. This is achieved by having these sets of files stored in the same directory. The transfer of the whole files and directory structures over the Internet would be redundant, and a way of adding, deleting and modifying specific portions of the files, is required. To resolve this problem, the files should be stored in XML format, and SchedSP should provide services for partial modifications of these files. The XML format also ensures that the files are not corrupted during the transfer, and conform to the specifications of the applications.

5. SCHEDSP SERVICES AND ARCHITECTURE

The services required to perform the previously described functionalities of SchedSP are shown in Figure 4. The Execute and Monitor services allow the use of a real world scheduling software on the solution of a particular real world problem and the monitoring of its progress. The file space service allows various operations on the files stored in the SchedSP environment. The XML transformations service allows for operations on the content of the XML files stored in the virtual file space. The Executive service is the control service of SchedSP because it performs user authentication, authorization, and service selection and organization. The MultiServe service provides the means to automatically perform multiple requests between the clients and SchedSP.

FIGURE 4: SchedSP Services

FIGURE 5: SchedSP Communication layers

The 3-layer communication stack as presented in Figure 5 provides the interface between SchedSP and its clients. For the lower layer the HTTP protocol is used. The SchedSP layer, which follows the XML standard, is responsible for user authentication and session handling. The SchedSP Service Provisioning layer contains the particular application domain knowledge and through this layer the actual scheduling services are provided to the SchedSP clients and their users.

FIGURE 6: Overview of the SchedSP architecture

An abstract overview of the basic SchedSP architecture and its components is shown in Figure 6. The requests are transferred between the SchedSP clients and the core of SchedSP using HTTP through the Gateway and Executive components. The set of components shown on the right of the figure are called Service Provisioning Components (SPC) and their role is to serve the Service Provisioning layer requests of the SchedSP clients. The Executive analyzes these requests and directs them to the appropriate SPC component. The SPC components respond to the clients, in the opposite direction, using an application standard XML response. All the SPC components have a common programmer interface, and they are accessed via COM [24] in the current SchedSP prototype. The Gateway component implements the HTTP layer of the communication stack, the Executive is responsible for the SchedSP layer and the SPC components serve the Service Provision layer requests. Special SPC components are used for the communication between the SchedSP and the GRID substrate, which hosts the required scheduling applications. This will further elaborated in Figure 7, where the overall view of SchedSP and its interaction with the PLEIADES GRID is shown. A short presentation of all of the components and the other architectural and implementation issues are presented in the remaining of this section.

FIGURE 7: The SchedSP Architecture Prototype

The SchedSP Gateway component is responsible for the lowest layer of the SchedSP communication stack, which as mentioned before is the HTTP protocol. In this component web server security techniques could be applied. Gateway has been implemented as a Microsoft ActiveServer Pages [25] script, and it utilizes several external COM components including the SchedSP Executive.

The role of the Executive component is central in ESP in terms of organization. Its role is to implement the SchedSP layer of the communication stack, authenticate and authorize the users using a session management approach, and then select the appropriate service provisioning components that will serve the requests. For the authorization to resources, an Access Control List approach is used, as described below in this section.

Both the Executive and the SPC components use a set of helper components in order to provide uniform behavior in common operations. For the current prototype, they are the SchedSP-Settings and the Access Control List. The SchedSP-Settings helps the sharing of common information between the Executive and the SPC components. Its role is to retrieve and provide to the various SchedSP components the needed customization settings, like paths, details for database connections, URLs. It also maintains information that is derived when a user logs on, like his personal details, session ids and other. Besides these, the SchedSP-Settings component provides some uniform methods for translating resources from the aliases used by the SchedSP clients to the appropriate resources. The access control mechanism to these resources is provided by the Access Control List component. When an SPC or the Executive needs to access a resource, it asks this component first and depending on its response it continues the operation or halts because of access denial. This component conforms to the SPC specifications and is accessible from the SchedSP clients for its services, but its main use is internal in the SchedSP core.

The Filespace SPC component provides space in the disks of the SchedSP host machine mainly for storing inputs and results from the scheduling solvers. The files and directories that appear in this service are actual files on the host. File operations include creating and deleting directories, creating and deleting files, viewing the contents or storing new ones in a file, and validating an XML file. There are no operations to modify the content of the existing files provided by this SPC, since similar operations are more easily performed by the use of the XML transformations service.

The XML transformations SPC allows most of the transformations that could be needed for an XML file. These transformations include the use of the Extensible Stylesheet Language Transformations (XSLT) filters [26]. The remaining modifications of an XML file include the addition, removal and exchange of an element, an attribute and an XML sub tree. The element that needs to be modified in all of the above cases is expressed using the XPath syntax [27], a standard used for addressing parts of XML documents. The XML transformations SPC component allows multiple transformations to be applied at once, but the whole operation will success only and only if the resulting XML file is valid, or well-formed in case there is no corresponding schema. This SPC helps applications to avoid uploading the input files at whole but only the portions that have changed.

The Execute and Monitor SPC components allow the execution of scheduling applications on the computational resources that are available to SchedSP. Execute can use either the PLEIADES CISP GRID services or utilize local GRID facility managed by the Condor Resource Manager for Microsoft Windows NT / 2000 [28, 29, 30]. A required extension to this service involves the creation of a load balancing strategy that would utilize the local GRID resources of SchedSP when they can guarantee a minimum quality of service otherwise would utilize the PLEIADES or some other GRID Resources. Since some scheduling applications have special requirements like the existence of certain libraries or special components on the host computer, the Execute SPC has knowledge on what is required and how to express these constraints to the CISP or to the Condor Resource Manager.

The MultiServe SPC component allows for the automatic execution of several functionalities in a single request to the SchedSP core from a SchedSP client. Since the main part of the MultiServe component is also a part of the Executive, this common functionality has been placed for reusability and support purposes in a separate component. MultiServe uses an XML based scripting language to coordinate the whole process that the client defines. In case of errors, they can be ignored or there can be specified the appropriate rollback procedures. The benefits from the use of the MultiServe service arise from the fact that combined actions are performed in a safe and efficient manner.

6. CONCLUSIONS - FURTHER WORK

In this paper, the possibility of utilizing existing scheduling software for real world problems as application services available over the Internet is presented. The goal is to allow distant and possibly computationally disadvantaged users to utilize both the software and the hardware capabilities of an Application Service Provider type service provided either by a private enterprise or by a central computational center of the particular application domain. Ease of use and support for these scheduling applications is one of our primary driving goals.

In order to achieve this, an architecture based on Service Provisioning Components is proposed and implemented. The architecture is designed such as to allow for growth both in terms of users using a particular application as well as for the addition of new scheduling application service. The SPC components and their capabilities were designed so that the user can create input data, change previously entered data and view the results in a manner similar to a traditional local system configuration. Our goal is to maintain the essence of such applications although the distant user is required to only maintain a light computer with minimal web browser capabilities.

If the interaction between the user interface and the core of a particular application is not continuous and there are significant computational steps that take place, the SchedSP framework should be capable to handle them. More experiments in this direction will be performed in the future.

Another central weakness of the service providing concept that needs to be further examined and elaborated involves the security risks of the whole process. Many researchers all over the world are presently tackling these issues and it appears that we will all have a safer Internet interaction possible in the near future.

The use of the XML Web Services protocols, SOAP, WSDL and UDDI will also be investigated although it appears that for this class of applications that requires special purpose local clients the XML protocol suffices. However, if SchedSP was to fully adopt the XML Web Services specifications, the development of new application specific SchedSP clients would be further simplified.

References

- [1] Kouloupoulos D., Papoutsis K., Goulas G. and Housos E. (2002) PLEIADES: An Internet-based parallel/distributed system. *Software: Practice and Experience*, 32, 1035-1049
- [2] Papoutsis K., Valouxis C., Housos E. (2002) A Column Generation Based Approach for the University Timetabling Problem. *Computers and Operations research* (under review)
- [3] Papoutsis K., Valouxis C., and Housos, E. (2002) A Column Generation Approach for the Timetabling Problem of Greek High Schools. *Journal of the Operational Research Society* (accepted for publication)
- [4] Birbas Th., Daskalaki S., Housos E. (1997) Timetabling for Greek High Schools. *Journal of the Operational Research Society*, 48
- [5] Valouxis C., Housos E. (2000) Hybrid optimization techniques for the workshift and rest assignment of nursing personnel, *Artificial Intelligence in Medicine*, 20, 155-175
- [6] Valouxis C., Housos E. (2002) Combined bus and driver scheduling. *Computers & Operations Research*, 29, 243-259
- [7] Alefragis P., Sanders P., Takkula T. and Wedling D. (2000) Parallel Integer Optimization for Crew Scheduling. *Annals of Operations Research*, 99 (1), 141-166
- [8] Goumopoulos C., Alefragis P., Housos E. (1998) High Performance Airline Crew-Pairing Optimization. *Proceedings of European Parallel and Distributed Systems Conference (Euro-PDS98)*, July 1-3, 1998 Vienna, AUSTRIA, pp.~337-243, ISBN 0-88986-266-4, IASTED/ACTA Press
- [9] Goumopoulos C., Alefragis P., Housos E. (1998) Parallel Algorithms for Airline Crew Planning on Networks of Workstations. *Proceedings of International Conference on Parallel Processing (ICPP-98)*, Aug 10-15, 1998, Minneapolis, Minnesota, IEEE CS PR08650, pp.~70-78, ISBN 0-8186-8650-2, IEEE Computer Society
- [10] Alefragis P., Papoutsis K., Valouxis C., Housos E. (2002) A Parallel Column Generation Approach in Solving the Crew Pairing Problem. *Annals of Operations Research* (under review)
- [11] Agrawal R., Bayardo R., Gruhl D., Papadimitriou S. (2002), Vinci: A Service-Oriented Architecture for Rapid Development of Web Applications, *Computer Networks*, 39 (5), 523-539
- [12] the XML Standard: <http://www.w3c.org/xml>

- [13] Jacobsen H.A., Günther O., Riessen G. (2000) Component leasing on the World Wide Web, *Netnomics Journal* 2, 191-219
- [14] Günther O., Müller R., Schmidt P, Bhargava H., Krishnan R. MMM: A web-based system for sharing statistical computing modules, *IEEE Internet Computing*, May-June 1997, 59-68
- [15] Jacobsen H.A., Günther O. (1999) Middleware for Software Leasing over the Internet. *ACM Conference on Electronic Commerce*, ACM EC99, pages 87-95, Nov 3-5 1999, Denver, Colorado, USA
- [16] XML-RPC specification: <http://www.xmlrpc.com/spec>
- [17] the SOAP Specification: <http://www.w3c.org/TR/SOAP>
- [18] the WSDL Specification: <http://www.w3.org/TR/wsdl>
- [19] the UDDI Specification: <http://www.uddi.org/specification.html>
- [20] .NET framework, from Microsoft : <http://www.microsoft.com/NET>
- [21] SunONE, from Sun Microsystems: <http://www.sun.com/software/sunone/>
- [22] Koulopoulos D., Goulas G., Papoutsis K., and Housos E. (2002) A Parallel/Distributed Platform for University Computational Infrastructure Service Provisioning, *Proceedings of the CSIT Workshop 2002*, Patras, Greece, 18-20 September 2002
- [23] Anderson T., Culler D., Patterson D.A. (1995) A case for NOW (Networks of Workstations). *IEEE Micro*, 38, 1
- [24] Microsoft corporation, COM home page: <http://www.microsoft.com/com/>
- [25] Microsoft ActiveServer Pages:
<http://msdn.microsoft.com/library/default.asp?url=/nhp/default.asp?contentid=28000522>
- [26] XSL and XSL-T: <http://www.w3.org/Style/XSL/>
- [27] The XPath specification: <http://www.w3.org/TR/xpath>
- [28] The Condor Project homepage: <http://www.cs.wisc.edu/condor/>
- [29] Basney J., Livny M. (1999) Deploying a High Throughput Computing Cluster. *High Performance Cluster Computing*, Rajkummar Buyya (Editor), Vol. 1, Chapter 5, Prentice Hall PTR
- [30] Livny M., Basney R., Raman R., Tannenbaum T., (1997) Mechanisms for High Throughput Computing. *SPEEDUP Journal* 11, 1

© G. Goulas, E. Housos 2002