A DYNAMICALLY CONFIGURABLE ENVIRONMENT FOR HIGH PERFORMANCE COMPUTING

Nabil Abdennadher[*], Gilbert Babin[†], Peter Kropf[‡], and Pierre Kuonen[*]

[*] GRIP Research Group - DI - EPFL
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
{nabil.abdennadher,pierre.kuonen}@epfl.ch

[†] Dép. d'informatique
Université Laval
Québec, Canada G1K 7P4
babin@ift.ulaval.ca

[‡] Dép. d'informatique et de rech. op.
Université de Montréal
Montréal, Canada H3C 3J7
kropf@iro.umontreal.ca

KEYWORDS

High Performance Computing, Metacomputing, Web Operating System (WOS™), Automatic configuration, Resource localization.

ABSTRACT

Current tools available for high performance computing require that all the computing nodes used in a parallel execution be known in advance: the execution environment must know where the different "chunks" of programs will be executed, and each computer involved in the execution must be properly configured. In this paper, we describe how the Web Operating System (WOS™) environment may be used to dynamically locate available computers to perform such computations and how these computers are dynamically configured.

INTRODUCTION

Advances in networking technology and computational infrastructure changed the HPC landscape. Tightly coupled, dedicated processors are replaced by loosely coupled independant machines connected via standard local or wide area networks. Centralized High Performance (HP) applications developed with proprietary, closed-source, hardware-dependant environments are more and more replaced by distributed "components" sharing and managing resources spread over a networked environment.

The new HP distributed platforms are accessed from the user's desktop in a uniform and user-friendly manner, such as provided by the Web's interfaces. The network environment combines multiple administration domains, heteregeneous computing platforms and security policies. Sharing and managing the resources spread over this network becomes therefore a cumbersome task. This problem is called the wide-area computing problem (Lindahl *et al.* 1998).

The wide-area-computing problem can be solved in an *ad hoc* manner for each application : scripts and various network tools are provided for this purpose. However, these solutions are very limited, lack scalability, and require a specific knowledge of the architecture of the machines.

From a computer science point of view, the right way to solve the problem is to build a Network Operating System (NOS) for the network. This NOS would provide high level means for sharing and managing complex resources distributed over the network. We think that metacomputing is one promising approach to reach that goal. The purpose of metacomputing is to give the illusion of a single machine by transparently managing data movement, scheduling application components on available resources, detecting faults and insuring that

The WOS™ (Kropf 1999) is a virtual operating system which is suitable for supporting and managing distributed/parallel processing on the Internet. Central to the WOS™ architecture are the communication protocols, which may be seen as the "glue" of the whole environment. Communication between nodes is realized through a generic service protocol and a simple discovery/location protocol (Babin *et al.* 1998).

The service protocol may be versioned to support specialized services. In the present research, we focus on the design of such a version for High Performance computing. This version essentially locates nodes able to execute parallel programs, identifies nodes available for participating in the parallel execution, and sets up an execution environment on the dynamically selected set of nodes.

the user's data and physical resources are protected. Globus (Foster and Kesselman 1997; The Globus Project), Legion (Grimshaw *et al.* 1997; Lindahl *et al.* 1998), and NetSolve (Casanova and Dongarra 1997) are concrete exemples of research projects in this field.

However, requirements for HPC are more than just transparent management and use of resources, distributed over the network; the metacomputing environment selected must meet the performance requirements of the application from a computational and communication standpoint. The metacomputing approaches cited above do support HPC by developing there own, closed-source, HP execution environment. This, in effect, binds the user to the HP execution tools provided by the metacomputing environment selected. We argue that, although this approach favors transparency, it does so at the expense of portability and efficiency. The approach we propose is to use metacomputing tools whenever useful and to avoid them when more efficient tools may be used instead. This way, we can always select the most effective tools for a specific context. Furthermore, we can reuse existing tools, without having to rewrite them for the sake of transparency. This approach is supported by the Web Operating System (WOS™) (Kropf 1999; Plaice and Kropf 1999) which can assist users of HP and, more generally, parallel applications during the configuration stage, and select the most effective execution environment for the execution stage. The WOS™ can be seen as a collection of service classes. Our contribution is to develop a new service class for the configuration and execution of HP applications. This service class assists users during the configuration step by searching and locating the most suitable sites for the execution of their applications.

This paper is organized as follows : Section 2 provides a definition of metacomputing. Section 3 analyzes metacomputing functionalities in the context of HP applications. Sections 4 and 5 respectively describe the architecture of the WOS™ and the proposed HP service class. Section 6 provides additional information regarding the status of the project and concludes the paper.

A VISION OF METACOMPUTING

We share Buyya's (Buyya 1999) definition of metacomputer : a set of computers sharing resources and acting together to solve a common problem. Our vision of a metacomputer comprises thousands of computers and terabytes of memory in a loose confederation, tied together by a network. The user has the illusion of a single powerful computer; he manipulates objects representing data resources, applications or physical devices.

At this point, it is important to distinguish between a parallel computer and a metacomputer. The main difference is the behavior of the computational nodes. A metacomputer is a dynamic environment that has some informal pool of independant nodes, each relying on its own complete operating system, and which can join or leave the environment whenever it desires. According to this definition, some parallel computers, such as the IBM SP series or the Swiss-T1 machine (Kuonen and Gruber 1999) can be considered as local metacomputers, which is not the case for the Cray T3D. In addition, a metacomputer is distinguished from a simple collection of computers by a software layer (middleware) which transforms a collection of independant resources into a single, virtual and coherent machine.

To better understand what metacomputing is, we first introduce the concept of Grid Computing, which may be compared to the electricity grid (Foster and Kesselman 1999) : when we power up our computer or television, we don't care about the location of the electricity generator that effectively provides energy to the concerned appliance. In the same manner the national electricity grid routes electricity across hundred of miles, the grid computer, i.e. a set of connected supercomputers, should allow the transparent execution of a program by searching and allocating the resources it requires. It is the grid computer's responsability to support transparent security, scheduling, data displacement, fault tolerance, conversion, etc. Metacomputing is a generalization of Grid Computing where the supercomputers may be replaced by off-the-shelf computers.

A metacomputer should provide four basic services (Lindahl *et al.* 1998) :

1. *Transparent Remote Execution*. By using a metacomputer, a user should be able to execute his application by simply typing a command line. The system should select the appropriate node(s) among those the user is allowed to use, transfer binary code and, launch execution. The transfer of the input data and the storage of the output data should be done

transparently by the metacomputer. Finally, the user should not know about the queuing system of the selected executing node(s).

2. *Transparent Access Distributed File System*. A metacomputer should allow transparent access to a file, regardless of its location. NFS is a well known exemple of a distributed file system (Levy and Silberschatz 1990). However, NFS requires super-user configuration. The World Wide Web is another well known distributed filesystem limited to read-only access.

3. *Wide-Area Parallel Processing*. The goal is to execute a single parallel application by using multiple remote resources (parallel machines). The application should tolerate the latency involved by the transfert of data between the remote sites. Problems, well known in parallel processing, such as task scheduling and assignment, load balacing and fault tolerance should be taken into account by metacomputing.

4. *Meta-Applications*. Meta-Applications are composed of a set of connected legacy applications that were previousely executed as standalone applications. The output of the first application is the input of the second, etc. The meta-application can thus be represented by a graph, where nodes are stand-alone applications and edges are data communication links between nodes. Beside the scheduling problem already encountered above, there is another issue : data are geographically distributed and it is the metacomputer's responsability to decide when to migrate computation to the data or vice versa.

METACOMPUTING FOR HIGH PERFORMANCE COMPUTING

The approach we propose is to take advantage of some services provided by metacomputing environments during the configuration phase : evaluation of the workload on each node, localization of the resources required by the different modules of the parallel program, assignment and scheduling of the different tasks on the nodes, fault tolerance configuration, etc. These fonctionalities are provided by the Wide-Area Parallel Processing service.

For this purpose, we propose a versioned metacomputing environment based on the Web Operating System (WOS™) which will assist users during the configuration stage of the parallel program execution. The WOS™ provides the user with the necessary infrastructure to search for the most adequate resources required by the parallel program. The HP version of WOS™ is able to select, among all the WOS™ nodes, the ones that satisfy the constraints imposed by the application as expressed by the user : workload threshold, network and machine architecture, software resources that must be available on the node (JAVA, PVM, DBMS, etc.), date and time of execution, etc. The execution environment to use is also managed as a resource. Therefore, we can dynamically select the most appropriate execution environment during the configuration stage, based on the user's and the application's requirements.

WEB OPERATING SYSTEM

The Web Operating System (WOS™) was developed to provide a user with the possibility to submit a service request without any prior knowledge about the service (where it is available, at what cost, under which constraints) and to have the service request fulfilled within the user's desired parameters (time, cost, quality of service, etc.). In other words, the WOS™ is designed to enable transparent usage of network-accessible resources, whenever a user requires a service, wherever the service is available. These services may be specialized hardware or software components, or a combination of both. A user needs only to understand the WOS™ interface, and does not need to understand how the service request is fulfilled. Therefore the WOS™ provides a computation model and the associated tools to enable seamless and ubiquitous sharing, and interactive use of software and hardware resources available on WOS™ compliant nodes of the Internet.

These features make the WOS™ a very attractive environment for metacomputing :

- The WOS™ environment can serve as a metacomputing brokerage service, by managing resource reservation requests and by launching the execution at the required time (Transparent Remote Execution).

- The WOS™ environment can be used to automatically discover all required resources, hardware and software, for the remote execution of a program (Transparent Remote Execution and Transparent Access Distributed File System).

The WOS™ computation model is based on eduction, where a query is only processed when needed and prior results are stored in a local database called a warehouse, where they can be accessed later on.

THE HIGH PERFORMANCE WOSP VERSION

Central to the WOS™ architecture are the communication protocols, which may be seen as the "glue" of the whole environment. Communication between nodes is realized through a simple discovery/location protocol (WOSRP) and a generic service protocol (WOSP). The WOSP is in fact a generic protocol definied through a generic grammar (Babin *et al.* 1998). A specific instance of this generic grammar, also referred to as a version of WOSP, provides the communication support for a service class of WOS™. The semantics of a version of WOSP depends directly on the service class it supports. In other words, knowing a specific version of WOSP is equivalent to understanding the semantics of the service class supported by that version. In the case of High Performance Computing, communication services are required (1) to locate potential compute nodes with the appropriate set of resources (hardware and software) and to reserve these resources, i.e., the configuration stage and (2) to launch the execution of the parallel program, i.e., the execution stage.

The Configuration Stage

This stage requires the most involvment from the WOS™. It is initiated at the user's request. As a starting point, the user specifies parameters and constraints to execute a parallel program, for example, date and time of execution, program to run, metacomputing environment to use (JAVA, PVM, MPI, etc.). From that point on, the control is passed to the WOS™.

The WOS™ performs the following tasks, if needed :

1. *Locate other WOS™ nodes*. This occurs only when an insufficiant number of WOS™ nodes that can perform parallel computing are locally known. In this case, the WOS™ will use a discovery/location protocol (WOSRP) to identify new nodes that understand the version of WOSP used for High Performance Computing (i.e., the version of WOSP we are currently describing).

2. *Locate nodes that can participate in the current request*. In this step, we want to identify nodes that can be used to answer the user's specific needs and requirements. We use the search approach developed for the WOS™ (Unger *et al.* 1998). The syntax associated to this search process corresponds to the "query command" construct in Fig. 1. Since one execution may require different resource sets, the search parameters will provide one set of parameters (**WOS_Params**) for each of these distinct resource sets.

   When it receives such a query, a WOS™ node will identify all the resource sets it can provide. The reply that the node builds, includes only those resource sets that it can provide (Fig. 1).

3. *Collect replies from all the nodes*. The requesting WOS™ node collects all replies. At this point, it will determine which nodes will be asked to participate in the execution. Note that it may be necessary to launch additional search steps (1 and 2 above) to complete the configuration.

4. *Reserve the resources*. Here, the WOS™ simply indicates to the selected nodes that it will use a certain set of resources, based on the information received. This corresponds to the "setup command" construct in Fig. 1. A node can still reject or accept a reservation request (Fig. 1).

The search results are preserved in the local warehouses. This way, subsequent executions with the same (or similar) parameters will reuse the results rather than perform the whole search again.

The Execution Stage

Once the configuration stage is completed, the WOS™ node can send a command to every node to start the execution (Fig. 1). The WOS™ just gives the starting signal and waits for the results.

CONCLUSION

This article has discussed how the WOS™ environment could be adapted to support High Performance

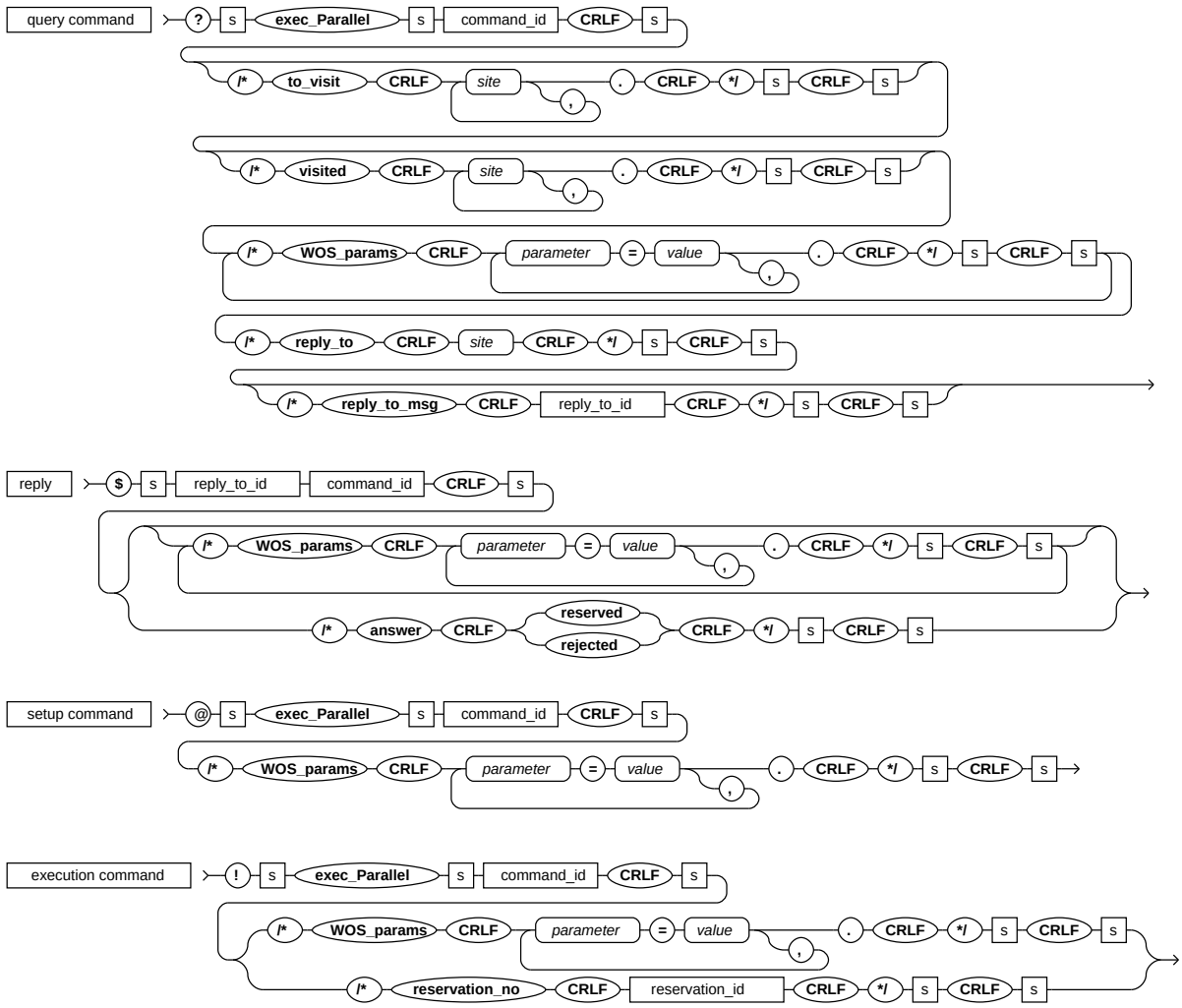

Figure 1: WOSP for High Performance Computing

applications. More specifically, a detailed description of how WOS™ nodes should interact was presented. It was also discussed how the proposed version of WOS™ can dynamically select the most appropriate execution environment for the parallel program to be run. This new service class enables the support of Wide-Area Parallel Processing within the WOS™ environment. The search method adopted can satisfy the constraints imposed by the user and is general enough to take into consideration all kinds of constraints. Finally, it is important to point out that, altough the WOS™ allows for the dynamic selection of the execution environment most suitable for a specific application, within specific constraints, it is not involved in the execution stage *per se*. As a consequence, an application could take advantage of a new, possibly more efficient execution environment without any modification of the WOS™itself. We plan to implement this version of WOSP, integrate it into the WOS™ environment, and conduct a significantly large series of experimentations in the near future.

References

Babin, G.; P. Kropf; and H. Unger. 1998. "A two-level communication protocol for a Web Operating System (WOS™) (Västerås, Sweden, Aug.)." In *IEEE Euromicro Workshop on Network Computing*, 939–944.

Buyya, R. 1999. *High performance cluster computing : Architectures and systems*, vol. 1. Prentice Hall PTR, Upper Saddle River, N.J., USA.

Casanova, H. and J. Dongarra. 1997. "NetSolve: A network server for solving computational science problems." *International Journal of Supercomputer*

Applications and High Performance Computing 3, no. 11: 212–223.

Foster, I. and C. Kesselman. 1997. "Globus: A metacomputing infrastructure toolkit." *International Journal of Supercomputer Applications*.

Foster, I. and C. Kesselman, eds. 1999. *The Grid : Blueprint for a new computing infrastructure*. Morgan Kaufmann, San Francisco, CA, USA.

Grimshaw, A.; W. Wulf; J. French; A. Weaver; and P. Reynolds. 1997. "The Legion vision of a Worldwide Virtual Computer." *CACM 40*, no. 1 (Jan.).

Kropf, P. 1999. "Overview of the WOS project (San Diego, CA, USA, Apr.)." In *1999 Advanced Simulation Technologies Conferences (ASTC 1999)*.

Kuonen, P. and R. Gruber. 1999. "Parallel computer architectures for commodity computing and the Swiss-T1 machine." *EPFL-Supercomputing Review*, no. 11 (Nov.): 3–11.

Levy, E. and A. Silberschatz. 1990. "Distributed file systems: Concepts and examples." *ACM Computing Surveys 22*, no. 4 (Dec.): 321–374.

Lindahl, G.; A. Grimshaw; A. Ferrari; and K. Holcomb. 1998. *Metacomputing — what's in it for me ?* http://legion.virginia.edu/papers/why.pdf, last visited on Jan. 20, 2000.

Plaice, J. and P. Kropf. 1999. "WOS communities — interactions and relations between entities in distributed systems (Rostock, Germany, June)." In *Distributed computing on the Web (DCW'99)*.

The Globus project. http://www.globus.org, last visited on Jan. 20, 2000.

Unger, H.; P. Kropf; G. Babin; and T. Böhme. 1998. "Simulation of search and distribution methods for jobs in a Web operating system (WOS™) (Boston, MA, USA, Apr.)." In *High Performance Computing Symposium '98*, Tentner, A., ed., SCS International.

BIOGRAPHIES

Nabil Abdennadher received the engineering degree from the Ecole Nationale des Sciences de l'Informatique (Tunis, Tunisia), and the Ph.D. degree from University of Valenciennes (France). From 1992 to 1995, he was Assistant Professor at Université de Tunis II and from 1995 to 1998, he was Associate Professor at the same Univesity. Since 1999, he is Research Scientist the Swiss Federal Institute of Technology, Lausanne (EPFL; Switzerland). His major research interests include Metacomputing, Parallel Programming, Object-Oriented Technology, High Performance Computing (HPC), and Web-Supercomputing. He worked on the parallelization of several specific applications : Text-To-Speech synthesis, OCR for Arabic printed characters, DataBase Queries, Pattern Recognition. He worked also on the problem of mapping and load balancing.

Gilbert Babin received the B.Sc and M.Sc. from Université de Montréal (Canada), and the Ph.D. from Rensselaer Polytechnic Institute (USA). Since graduating in 1993, he has been a faculty member at Université Laval (Canada), where he currently is Associate Professor. He has been working on the integration of heterogeneous, distributed databases using reactive agents and a central knowledge repository. Some of his work has been published in IEEE Transactions. His current interests include the Web Operating System (WOS™), trust management and e-commerce.

Peter Kropf received the M.Sc. and Ph.D. degrees from University of Bern (Switzerland). Subsequently, he was a research scientist at the University of Bern for several years. From 1994 to 1999, he has been Assistant and Associate Professor at Université Laval (Canada). In 2000, he joined Université de Montréal (Canada) as an Associate Professor. He has been carrying out research in parallel computing for over ten years, particularly in the field of mapping and load balancing. Current projects especially include Internet computing and e-commerce, and the efficient use of parallel computing in real world applications. His research interests include Internet computing, parallel/distributed computing tools, networking and simulation.

Pierre Kuonen obtained the diploma of electrical engineering from the Swiss Federal Institute of Technology, Lausanne (EPFL; Switzerland) in January 1982. After having worked during five years in industry (petroleum and CAD development) he joined, in 1988, the Computer Science Theory Laboratory of the Computer Science Department of EPFL. He was involved in teaching and research on parallel programming and Computer Science Theory. He obtained the Ph.D. degree in 1993. Since 1994 he has been a scientific collaborator, heading the Parallel Computing Research Group (GRIP) of the Computer Science Department of EPFL. He is senior lecturer in parallel computation courses and manages several European and national research projects. Pierre Kuonen is author or co-author of more than 25 scientific publications.