

A Model for Distributed Computation over the Internet

Jing Zhu, Yongwei Wu, Fei Xie, Guangwen Yang, Qing Wang, Jiayin Mao, Meiming Shen

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Abstract

In this paper, we put forward a model for distributed computation over Internet, which enables diversified and geographically located computational resources to be aggregated into a powerful computation entity to provide computational services. It can also integrate software resources on computers to run user's applications. The overall model architecture, our efforts in key issues in building such a distributed environment (including resource management, resource discovery and selection as well as programming model) are presented. A prototype has been constructed and two representative applications are also developed.

1. Introduction

The availability of powerful computers and high-speed networks as low-cost commodities over Internet have changed the traditional way we do large-scale parallel and distributed computation. Great interest is already growing in coupling large numbers of geographically distributed resources to achieve gigantic potential [1,2,3,4]. In this paper, we present a model for distributed computation over Internet, in which heterogeneous computers distributed over the globe are seamlessly integrated into a distributed computational environment to provide computational services and individual resources are so completely transparent to consumers that they can use them without having to be aware of their existence.

In our distributed computational model, all participants are roughly divided into two categories: one is referred to as resource contributor, who offers his own resource for public use. The other group is denoted as resource consumer, who submits his application to be executed and therefore consumes others' resources.

Among key issues in building such a flexible, scalable and robust distributed computational environment, resource management, resource discovery and selection, programming model for user's application are of greatest importance. Main contributions of this paper lie in designs and implementations of such three aspects and corresponding applications. We bring forward a distributed uniform resource management model (DURM), a distributed resource discovery and selection model (DRDS) and a thin kernel parallel programming model (TKPM).

The rest of our paper is organized as follows: Section 2 gives the framework of our model for distributed computation over Internet; Section 3 to 5 explains in detail several key components and issues in our design. Section 6 introduces two representative applications based on our model. Finally is the conclusion.

2. Model Framework

Logical framework of our distributed computation environment is shown in the following

figure (Figure. 1).

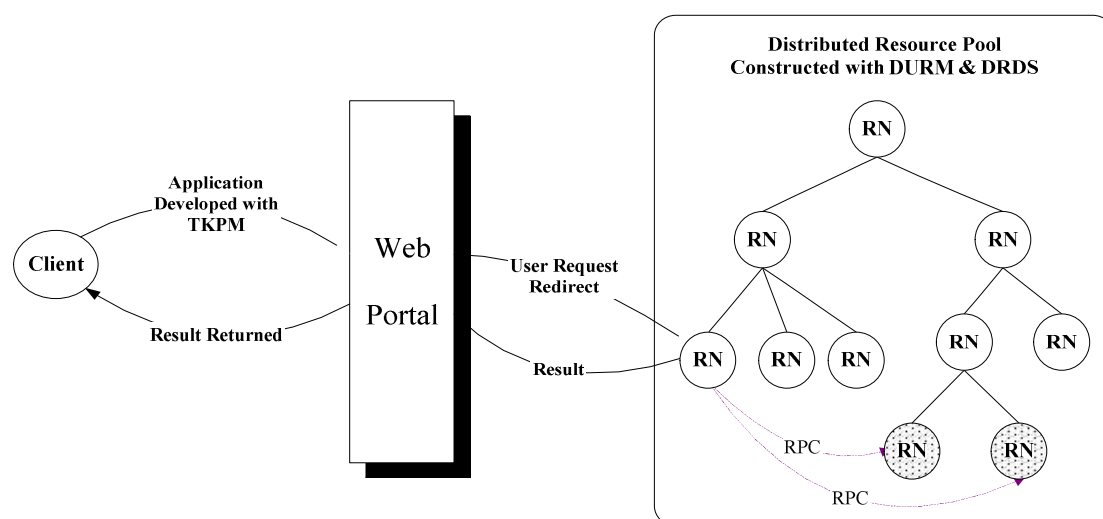


Figure. 1 Framework for Distributed Computation over Internet

To construct such a distributed computation system, there are several aims to achieve.: scalability, usability, robustness and etc. First, resource contributors should be organized in a scalable and efficient way. New resource contributors should be easily added into the system without affecting the already existing ones. Second, a friendly user interface should be provided for resource consumer to utilize services and resources in the system.

In our model, resources contributed by resource contributors can be divided into two categories: one is pure computation power of the computer node, also referred to as hardware resource; the other is software installed in the computer node. For the first category, user can deliver his application totally developed by himself for several nodes in the system to execute computation; for the second, user's application is based on the software provided by certain nodes in the system, and he may only need to submit certain inputs for his problem.

Following is detailed explanation of key components and research issues in the framework.

3. Resource Management

3.1 Distributed Uniform Resource Management Model (DURM)

Resource management in distributed computing is an important research issue. It aims to make effective description and organization of diverse resources in the system so that user can utilize them conveniently to compute their applications. We present a distributed uniform resource management model (DURM) in this paper. In DURM, resource management consists of both resource abstraction and resource organization.

Computer resources distributed over the Internet are various, ranging from supercomputer, personal PC to cluster and etc. They are also heterogeneous in hardware structure, operating systems as well as software installed for shared use. However, it is not difficult to find that some characteristics or attributes irrelevant to system platform can be abstracted for a uniform description of these resources. In DURM, the parameters abstracted for resource description contain two parts: one is static, which includes relatively fixed information such as network IP address, geographical location, total CPU number and frequency and so on; the other part may

vary dynamically, which includes pure computation power information such as current CPU load of the computer resource, currently available CPU number, available memory size and so on, and information about software shared in the computer node (such as software category, software name, software executable path and etc). By this means, each computer resource can be represented by a combinational set of static and dynamic parameters.

The benefits of such a uniform resource representation are dual. It not only hides diversity and heterogeneity among resource nodes and therefore simplifies resource management and task scheduling, but also enables any computer node in the distributed environment to provide computational services in the same way.

As to resource organization, DURM organizes resources into a hierarchical tree-like structure, in which each node (referred to as RN in Figure.2) has a father (root node excluded), several siblings and children (leaf nodes excluded). A resource node in the resource tree along with all its offspring nodes constitutes a sub resource tree with the node as leader. And the whole resource tree has a single root node. Construction of resource tree relies in the information center in the whole resource system. The function of information center is analogous to MDS [5,6] in Globus [4]. It periodically retrieves newest resource tree from the root node. When a new computer node intends to join the system, it first connects to the information center, and the latter will designate a father node for this new node according to closeness of their IP addresses. Then the new node communicates with its father and both of them get aware of each other. The resource tree is a logical structure, and it does not represent actual network connection topology among resource nodes. We can easily find that in our resource tree, nodes with near IP addresses, which imply low network communication cost among them, are likely to reside in the same sub tree. Besides, the leader of a sub-tree is the one from its children node in the tree. So, all the leaf nodes in the resource tree is actual resource nodes which perform the computation of user's tasks; while other non leaf nodes are logical ones which do not participate in actual computation. As stated in the Section 3.2, we will find such a tree structure is able to provide useful and effective guidance to resource discovery and selection as well as task scheduling.

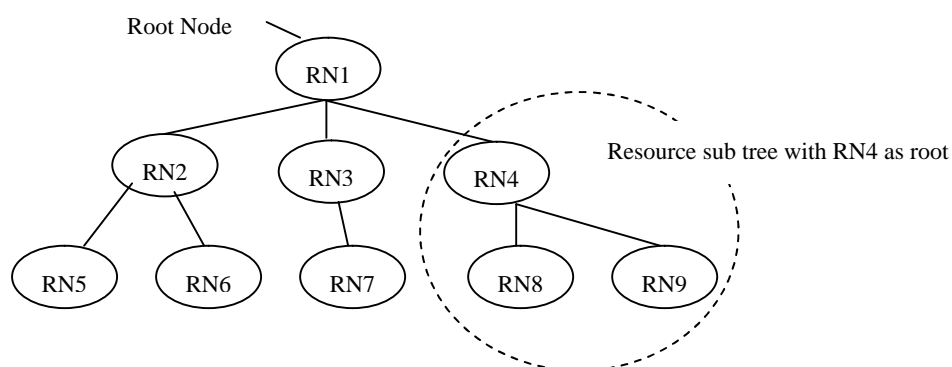


Figure.2 Resource Tree Structure

In our distributed computation environment, computation power as well as the software information are the most important resource parameters. Each node in resource tree periodically report the status (including up-to-date computing power, software information) of the sub-tree leaded by it to its father, which then collects all information from its children, generates an overall

resource status description of its sub-tree and report to its upper-level father.

For measurement of computation power of a resource node, we adopt SciMark2 [7], which is a set of benchmark designed for scientific computation. It provides two versions both in Java and C language, the former of which can be easily integrated into our system implemented in pure Java. When there is no computational task running in the resource node or the current load is relatively low, the SciMark2 benchmark program will be executed on the node periodically, and the result is a value which takes MFlops as unit and represents the current computation power of the node. The computation power of a resource sub tree is the sum of that of all the nodes in this sub tree. The same is with the software information.

3.2 Distributed Resource Discovery and Selection Model (DRDS)

In this section, we will introduce the key issue in task scheduling: resource discovery and selection, i.e how to find appropriate resource nodes for user's applications. The task scheduling strategy is critical to the performance of user's application as well as the performance of the whole system.

Before explaining our distributed resource discovery and selection model (DRDS), it is necessary to introduce the applications supported by our system. User's applications supported by our system are common in that they each have a divisible workload, namely the application consists of several independent atom tasks. Here we use the notation "independent" to indicate that no communication exists between any two tasks.

DRDS accomplishes resource discovery and selection in two stages. First, to find a sub resource tree that can meet the requirements of user's whole application. Second, to find qualified nodes in the sub tree for each atom task contained in user's requests. When user's application is submitted and directed to a certain resource node, the node first checks whether the resource sub-tree it leads is able to satisfy the requirements of user's whole application. If so, it will select one qualified child node with minimum superfluous computation power, and redirect user's request to this child. If not, user's request will be redirected to its father node, which performs similar operations as above. All process is enforced iteratively. Once user's request is accepted by a certain resource node, the first stage in resource selection is finished. Then the node will further partition atom tasks in user's request into several groups according to the computation power and software information of its children nodes and assign each group to corresponding child. These children nodes perform similar operations as above. Finally, each atom task will find the appropriate resource node for itself.

After each node decides that it will accept specific task of the whole application, it will notify the original node where the whole task is submitted. Eventually, the original node knows of the information about where each atom task of the user's application is to be executed (such as two RNs with dot shaded in Figure.1), it then transfers necessary code and data to these nodes and remotely invokes certain computation services in these nodes to execute actual atom tasks, receive and collect results of each atom task into final result for the user's application.

From above description, we can see that resource discovery and selection for user's application is accomplished by several nodes in the system in a distributed way. Such an approach can achieve load balance of the whole resource tree to a certain degree, eliminate the single-point failure and performance bottleneck encountered in traditional centralized method.

4. Thin Kernel Parallel Programming Model for Application (TKPM)

As stated in Section 3.2, user's applications supported by our system consist of several independent atom tasks. Such a characteristic enables the application to be dispatched and executed in parallel in several computer nodes over the Internet. Unaffected by the complexity and uncertainty in communication on wide-area Internet, this kind of application can make full use of resources over Internet and gain high efficiency.

In traditional parallel programming model (such as PVM, MPI), the partition of user's parallel tasks is hard-coded into programs. So when programmer intends to make adaptation on tasks partition, he has to modify the source code, re-compile and re-link into new executables. This may be feasible in traditional local area network, but is not appropriate for large-scale parallel computation over wide-area Internet.

In order to solve such a problem, we present a thin kernel parallel programming model (TKPM). The notation "thin kernel" means the separation between actual computation code (referred to as kernel) and task partitioning. Such a separation enables user to focus on implementation of code for his actual application. And after kernel code is developed, he can adjust the running of the whole application by changing the task partition part only without boring to modify the kernel code.

Furthermore, for applications supported by our system, according to characteristics of atom tasks, they fall into two categories as follows: (1) atom task code is written in Java by user himself, he wants to make use of the powerful computation to compute and execute his program. (2) user's application is executed by specific software provided by certain nodes in the system, then the user only needs to prepare input data for each atom task. No matter which category user's application belongs to, both need a metadata description of applications which is essential for resource selection and task scheduling. In our TKPM, we provide an application description file in XML format for this use. And the description file is composed of two parts. One is used to designate the location of task executables for user's application (in category 1) or the input file location for user's applications (in category 2). The locations are denoted by one or several URIs (Uniform Resource Identifier). The other part include partition of application into atom tasks, as well as resource requirements of tasks. Here requirements for resources mainly include software needed and computation power of intended executing node.

Besides, for the first category of application where atom tasks are coded in Java, our TKPM model provides a concise programming interface similar to Java Applet for programmer. Please refer to figure.3 for detailed information.

<i>TKPM Programming Interface</i>
<pre>public interface AtomTask { public void init (ParamList params); public void kernel(); public void finalizeAtom(); }</pre>

Figure.3 Programming Interface of TKPM

init() method is responsible for initialization of atom tasks, including preparing and parsing necessary input parameters; kernel() method contains material computation code; finalizeAtom() method is responsible for sending completion notification or results. When an atom task is assigned to a resource node, the node will startup a new thread for this task and invoke the three methods in order.

Advantages of TKPM lie in that it separates partition of parallel tasks from computational kernel of the problem, which is more flexible than traditional programming model and enables a more efficient task scheduling strategy.

Besides, due to the separation between application description file and actual application codes, during the resource discovery and selection process, only task partition part in the application description file needs to be transferred between nodes. And only after the appropriate nodes have been selected out for user's application, actual codes and data will then be transferred to these nodes for execution. Such a method can efficiently reduce traffic amount and save network bandwidth.

5. User Interface

Here user interface includes two parts. One is for resource contributor, how they can conveniently join the system and share their resources; the other is for resource consumer, how they can easily utilize the resources distributed over the Internet to solve their problems.

For resource contributor, we developed an easy-to-use client. The computer node only needs to run the client for joining the system and sharing its resource. The client program is written in pure Java, so theoretically speaking, each computer node installed with JRE (java runtime environment, version 1.3 or later) can be integrated into our system, no matter which hardware architecture and OS platform the node has. Besides, considering the difference among diverse platforms, a graphical client is provided for nodes with Windows platform; for other platforms such as UNIX, Linux and etc, a text-based client is provided so that it can be easily invoked and executed in command line.

With usability taken into account, we provide a Web interface as a portal between user (i.e resource consumer) and the back-end resource contributors. The Web portal provides functions that mainly include user registration, application submission, result retrieval and etc. User who wants to utilize the back-end computation power and software service to solve his problems does not need to know about details of back-end resource nodes, he only needs to interact with the Web browser already familiar to him. After web server receives user's request, it will consult the information server. And the information server will select a node randomly and the user's request is then redirected to this node. Then the subsequent process is accomplished by several nodes in the system cooperatively as described above.

6. Applications

Up to now, we have constructed a prototype of the distributed computation environment put forward in this paper. A Web portal is employed for user to submit his application and two applications are developed, with each representative of an application type described in section 4. One is decryption against DES (Data Encryption Standard). Another is computation based on

ELIMINO, a new computer-mathematics research system.

6.1 Key Search Against DES

DES is abbreviation for Data Encryption Standard [8]. It is a widely-used method of data encryption. The encryption is controlled by a key of 56 bits. Up to now, exhaustive key search is the most practical and efficient attack to DES. We developed an application for evaluating the resistance of DES to a known-plaintext key search.

Because the key space can be easily partitioned into independent subsets, the whole application correspondingly consists of several independent atom tasks, and it is fit for execution on multiple resource nodes in parallel. The kernel code is written in Java according to programming interface stated in section 4. And in application description file, user can divide the whole key search space into several different subsets which is input parameter for each atom task. Due to the separation between task partitioning and kernel application code, our system can flexibly distribute atom tasks to multiple appropriate resource nodes and utilize their powerful computation capabilities to perform exhaustive key search against DES.

6.2 Computation with ELIMINO

ELIMINO [9] is a computer-mathematics research system as well as a mathematical software. It is designed to focus on the implementation of Wu's method and to be used by researchers to perform sophisticated mathematical computations.

Now assume that in our system, several nodes have been installed with ELIMINO. And user can utilize them to do mathematical computation with Wu's method. In our distributed computation environment, user's application is composed of several independent atom tasks, each one of which is a computation with ELIMINO on certain input data. For this kind of application, programming interface introduced in section 4 is of no use. User only needs to focus on application description file and prepare input data for each atom task. When user submits his application through our Web portal, qualified nodes with ELIMINO installed will be selected out from all the resource nodes, and ELIMINO will be remotely invoked to perform actual computation tasks. Finally, the computational results will be returned and collected. All these back-end operations are accomplished among resource nodes themselves distributedly and are totally transparent to the user.

7. Conclusion

Numerous computers distributed over Internet and high-speed networks increase people's interest in coupling large numbers of geographically distributed resources to achieve gigantic potential. In this paper, we present a model for distributed computation over Internet. It aims to aggregate diversified resources scattered over Internet into a powerful computation entity and help user solve their computational problems. All participants in our computation model are either resource contributor or resource consumer. Key research issues in building such a distributed computation model include resource organization and management, resource discovery and selection, programming model for user's applications. Main contributions of our paper rest with such three aspects. DURM model enables diversified and heterogeneous resources to be represented in a uniform approach and organized hierarchically. DRDS model enables resource discovery and selection operation to be performed by several resource nodes cooperatively, thus

eliminating performance bottleneck in traditional centralized approach. With TKPM, task partitioning and the kernel of application codes are separated, and a concise programming interface is provided for user to implement kernel code for his applications. Up to now, we have built a prototype of the distributed computation environment put forward in this paper and developed two representative applications. Experiments show that it can effectively harness heterogeneous resource nodes distributed widely and provide enormous computation power to solve user's actual computational problems.

References

- [1] M. Migliardi, V. Sunderam. Heterogeneous Distributed Virtual Machines in the Harness Metacomputing Framework, Proc. of Heterogeneous Computing Workshop of IPPS/SPDP99, pp. 60-72, S. Juan, Puerto Rico, April, 1999.
- [2] M. Romberg. The UNICORE architecture: Seamless access to distributed resources. Proceedings of the 8th International Symposium on High Performance Distributed Computing (HPDC-8), pp. 287--293, August 1999.
- [3] I.Foster and C.Kesselman. Globus: a metacomputing infrastructure toolkit, International Journal on Supercomputing. 1997, 11.
- [4] G. Allen, T. Dramlitsch, I. Foster, N.T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen. Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Catus and Globus, Proc. SC'01, USA, 2001.
- [5] A. Iamnitchi and I. Foster. On Fully Decentralized Resource Discovery in Grid Environments. International Workshop on Grid Computing, Denver, CO, November 2001.
- [6] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Grid Information Services for Distributed Resource Sharing. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
- [7] SciMark2 benchmark, <http://gams.nist.gov/scimark2/>
- [8] "Data Encryption Standard", National Bureau of Standards (U.S.), Federal Information Processing Standards. Publication (FIPS PUB) 46, National Technical Information Service, Springfield VA, 1977.
- [9] Dongdai Lin, Jun Liu and Zhuojun Liu. Mathematical Research Software: ELIMINO. Proceedings of Asian Symposium on Computer Mathematics, Lanzhou University Press, pp 107-114, China, 1998.