Computer and Information Science
www.ccsenet.org/journal.html
Vol. 2, No. 3
August 2009

The Design and Implementation of the Distributed Computing Platform for Bioinformatics

Juan Huang
School of Information Science and Engineering, Central South University, Changsha 410075, China
Tel: 86-731-265-5236 E-mail: dealhelen@yahoo.com.cn

Yaoping Fei
Network Center, Central South University, Changsha 410075, China

Abstract

The processing of the huge amounts of information in the bioinformatics has been the bottleneck to restrict its development, and in this article, we used the distributed computation to solve this problem, and we described the structure, the design, the implementation of task decomposition, the distributed application program and the database management of the distributed computing platform. The distributed computing platform first decomposes one problem into many subtasks, and then the client sends the request of task computation for the server end, and the server end responses the request and takes out the information of the minimum subtask, and distribute the information to the client end. When the client end acquires the information of the subtask, it will transfer the operation module to compute, and when the task is completed, it will upload the result to the server end, and the server end repeats above process until all subtasks are completed, and according to the computation results of all subtasks, we can obtain the solution of the problem.

Keywords: Bioinformatics, Distributed computation, Task decomposition, RMI

1. Introduction

Along with the development of the human genome project in the world, huge amounts of genome information and the data about the structure of nucleic acid and protein issued present exponential growth every year, and the biological experiment data enter into the situation of “big bang”. In the data processing and analysis process about the huge amounts of information for bioinformatics, many processes such as the split joint of gene sequence, the sequence homology comparison, the establishment of the molecule cladogram, the structured information analysis and the forecast of the gene function all require the computation environment with higher performance (Shi, 2001, P.161-165). The distributed computation researches how to divide the problem which can only be solved by the super computer into many small parts, and distributing these parts to many computers, and integrating these computation results to obtain the result of the problem finally. The advantages that we adopt the distributed computation include that it can acquire same even better computation ability comparing with the super computer, and its expenses are much smaller than the expenses of the super computer.

2. Total structure and work flow of the distributed computing platform

The distributed computing platform needs to decompose the problem into many small tasks, and distribute these small tasks to the free computers in the network for computation, and manage these subtasks at the same time.

The bioinformatics distributed computing platform can realize the computation of the problem which needs the computer with high performance in the bioinformatics by the distributed mode in the Internet, and the solution of the problem about the bioinformatics is decomposed into the solutions of many subtasks. The platform is composed by three parts including the database, the server and the client end. The database stores relative information about the task, users and operation. The server realizes the task decomposition, the task management and distribution, the client management and the result processing. The client end realizes the task data download, the task computation and the task upload. The system structure of the platform is seen in Figure 1.

The personnel who participate in the distributed computation include the project mangers and clients, and the flows that the personnel participate in the distributed computation are different.

(1) The flow that the project mangers participate in the distributed computation.

First, describe the biological problem by the task tree, decompose the task into many subtasks and write the task information into the database.

Second, start the service of the server end, login the remote method for the users' transfer, and manage the data in the database.

(2) The flow that the clients participate in the distributed computation.

First, user interviews the website, enter into the login unit, input the information needed by the login, and complete the login procedure.

Second, download and install the application program in the download unit of the website.

Third, fill in user information in the client end, click "save", and the save user information at home, and the client end will automatically complete many operations such as the requesting the server, downloading task data, transferring the operation module and updating the task result.

3. Implementation of the task decomposition

The problem of the bioinformatics: give the set of DNA, $S = \{s_1, s_2, s_3, s_4 \ldots s_n\}$, and the length of each string is m, and find the string closing to other strings most (Ma, 2000, P.99-107).

If we adopt the distributed computation to solve this problem, we can decompose the set S into many small subsets. Every string has two states which are respectively denoted as 0 and 1, and the binary tree is used to store all combinations. In the original set S, the first layer is divided into the left child and the right child which are used to denote s_1, and the left child denotes the set S_10 has no the string s_1, and the right child denotes the set S_11 has the string s_1, and the second layer is used to denote s_2. In this way, the task tree with n layers can be generated, and the sub-trees which don't accord with the conditions will be deleted at the same time. If the amount of the string in the minimum subtask from the task decomposition is c, and the amount of the string in the decomposed set achieves c or the amount of the string which is removed in the set achieves n-c, we think this sub-tree doesn't according with the conditions and it will be deleted. For example, the amount of the factor in the set S, n is 6, and the amount of the factor in the minimum subtask set c is 3, and the established binary tree is seen in Figure 2.

According to the characters of the task tree, we code the task trees, and define the root node to denote the character "1", and the left branch to denote the character "0", and the right branch to denote the character "1", so we can take the character string composed by the branch characters from the root node to the leaf node as the code of the node.

From the coding rule, the code of the left child of the node is the code of the node adding "0", and the code of the right child of the node is the code of the node adding "1", so to interview the left node is to interview the node with the code adding "0", and to interview the right node is to interview the node with the code adding "1", and to interview the father node is to interview the node eliminating the last bit. The condition to achieve the minimum subtask is that the amount of "1" has achieved the maximum or the amount of "0" has been less than the amount of "the total – the amount of the maximum "1"".

Based on the recursion definition of the binary tree, the process of the task tree traversal includes four approaches.

(1) Judge whether the minimum subtask is achieved, if it is, turn to (4).

(2) Traverse the left sub-tree post-orderly.

(3) Traverse the right sub-tree post-orderly.

(4) Interview the root node, and write the task information into the database.

4. Implementation of the task management

The project managers need to grasp the task distribution and the development of the project, and manage the tasks in the database, and inquire about the task information table, the client information table and the operation information table. And the information of the inquiry possesses the comparability, and they are in same module, so the database management can be divided into two models, i.e. the task management and the information inquiry.

4.1 Management of the task tree

There are two sorts of representations for the task tree, and one sort is to list all minimum subtasks, which is similar with the list box, and all minimum subtasks can be inquired in the database, and they are taken as the roots of the tree structure to be inserted into the box. The other sort is to represent the task information by the tree form, and it's the hierarchy can be presented very intuitively. Because the records in the task information table are numerous, the time to generate the tree once needs long time, so we adopt the mode of classification generation, i.e. when the user clicks, its next level task tree can be generated. To process the event of "constructor" and the event of "itempopulate" in the control of "TreeeView", and in the event of "constructor", if the task tree is presented by the tree form, so we only generate the tree roots, and judge whether the root has children to set up the attribute of the root node. In the event of "itempopulate", add the child which clicks the node in the tree, and judge when its child has children to set up the attribute of the child.

4.2 Information inquiry

Because of the comparability among the inquiry of task information, the inquiry of client information, the inquiry of project information and the inquiry of operation, the system can use the general interface to offer the user selection table information, and generate the list option of the inquiry table which user wants to inquire according to the selected table, and when the user select the list selection item, the comparison conditions will be generated according to the data types (including the numerical value, the character and the time). According to user's selection and the query of input, the information needed by the project manager will be picked up to the data window. To achieve this effect, two more tables storing table information and list information should be established, the information can be obtained from the table. Different information will be showed in same data window, and we can use different data windows to set the proper data into the data window according to different table information.

5. Design and implementation of the distributed application program

The distributed application programs include project processing, the communication between the server and the clients, and the task management. The multithreading mechanism transferring RMI and Java by the remote method based on Java language offers a sort of application oriented and simple solution to realize the software implementation of the distributed computation. RMI encapsulates the distributed processing details on the bottom, eliminates the complexity brought by the network distributing and the diversity of the agreements, and offers transparent remote object method transfer mechanism, and the multithreading character of Java make the parallel computation and control be realized easily in the distributed computation environment (Zhu, 2003, P.60-62). The connection of the server and client based on RMI is seen in Figure 3.

The multilayer model distributed application based on RMI generally includes the remote interface, the remote object, the service program and the client program.

5.1 Implementation of the remote interface

The remote interface regulates the interactive interface between the client program and the service program, and to create a RMI program, we first should create the interface to extend the interface. In this interface, we can add the method which can be transferred in the remote computer. With this interface, we can transfer the object in the remote computer like in the local computer. The task information about the user distribution and the method uploading task result should be included in the interface (Zhang, 2002, P.415-451).

5.2 Implementation of the remote object

The remote object can offer real implementation for every method regulated by the remote interface. The functions of the remote method include the identification of user' ID and the judgment of user's operation type. If the user asks for participating in the task computation, modify the state of the minimum subtask which has not returned the result for a long time, inquire the minimum subtask which task state is un-computed, modify the task information table, the operation information table and the user information table, and return the task name of the minimum subtask which was computed. For the computation that the user has participated in the subtask, obtain the result of the task, upload the task result, write the task result into the task information table, and correspondingly modify the data in the operation information table and the user information table. Compile the written remote method class by the RMIC compiler, and generate the peg and framework.

5.3 Implementation of the service program

The server charges for establishing the service example and logging in the remote method logging place which includes the position of the logging place, the selective end number and the server name. The static state method can be referenced by the remote object placed in the logging place which charge for the index through the class of "java.rmi.Naming". After logging in the service, client can interview the service. To make the client go to special service, the banding process establishes a logging item in the logging place. And if the service stops, it will cancel the banding by itself. If the banding is correct, the remote method transfer can be used (David, 2003, P.19-37).

5.4 Implementation of the client end

The client program first acquires the user name and password from the local computer, reads the information in the task configuration files, and judge the task state of the computer, and if the computer has no the task information or the task result has been uploaded, it will send the request to the server program end (i.e. transferring the remote method). The client program first acquires the citation of the remote method in the register of service program, and once the citation is acquired, the client program will use the remote transfer method to obtain the task information. When the client obtains the task information, the client will take out the data needed by the task computation in the database, write the data into the local file which takes the task name as the file name, and transfer the operation model of the client end, and circularly test whether the result file has been generated. When the result file is generated, transfer the remote method once again, upload the task result to the server, and write the result into the database. The key technologies that the client end needs to realize include inquiring and transferring the remote method, read-out and write-in the text, writing the local last user information into the log file convenient for user operation, writing the task data in the database into the file of the local computer which is used for the data file for the operation of the module computation, and testing the result (utilizing the thread to test whether the result file is generated in every period of time).

The distributed application is the core of the whole distributed computation platform, and it connects the server with hundred thousands of client ends and harmonizes the task assignment and manages many aspects, and its design will directly influence the performance of the distributed computation. It is especially important to adopt the distributed computation technology based on RMI for the development of large-sized system, because it will make the idea which distributes the resources and processes loads on multiple computers possible (Zhang, 2002, P.415-451).

6. Example and conclusions

For the connectivity of the Mesh network, the connectivity is denoted by the form of probability. If the amount of total node and the amount of the dead node in the Mesh network are known, the probability of the connection can be denoted by the following formula.

The connection probability of Mesh network = the connection times/ the total times

Here, the connection times equals to the connection times of Mesh network in each error state.

By the data obtained by the computation, we can analyze the connectivity of Mesh network. Though the problem is simple, but because the amount of the node in the Mesh network is always very numerous, so the computation amount of the connection probability is very large, and it can achieve hundred thousands even more.

Take the connectivity of the Mesh network as the running example, this platform can not only solve this complex problem, but can test the running status of the platform. The process using the platform to solve the connectivity of the Mesh network includes following approaches.

(1) Establish the connectivity problem of Mesh network in the database, i.e. GTT.

(2) Add the program code about the Mesh network connectivity in the client end program.

(3) Start the server end program which connects the background database, and start the tcp monitoring thread and the http monitoring thread.

(4) Install and implement the client end program in various client computers, various client end programs ask for data to the server end program, and return the connection times of the subtask after processing.

(5) The server end program processes the return data from the client end, accumulate the connection times of various subtasks, and correspondingly modify the background database.

(6) When the task is finished, the server end program gives the finally total connection times and the connection probability, and the work stops.

For the connectivity problem of Mesh network tested by us, the total amount of the network node is 120, and the total amount of the dead node is 5. The time to solve this problem in the single computer averagely is 72.554900 hours. The platform used 12 computers to solve it, and the solving time reduced to 6.627607 hours. The test result is seen in Table 1, and the result indicated the platform works normally. Because the input spaces of many biological problems are finite state sequence, the platform can be further applied to solve the problem of biological computation.

The test indicated that for the bioinformatics problems with huge amounts of computation and better distributed computation characters, we could utilize the idea of distributed computation to divide the original problem into many sub-questions, and concurrently exert the subprograms solving the sub-questions by the form of remote method on multiple computers in the network by virtue of Java RMI mechanism and its multithreading mechanism, which could effectively reduce the computation time, and this distributed computation mode required little for the hardware and software environment, and it was very simple and quite effective.

References

David Reilly & Michael Reilly, interpreted by Shenfeng. (2003). *Java Network Programming and Distributed Computation.* Beijing: China Machine Press. P.19-37.

Hu, Kai, Hu, Jianping & Wang, Qiang. (2000). Research of Distributed and Parallel Computing Network Architecture. *Mini-Micro Systems.* No.21(2). P. 113-125.

Ma, Bin. (2000). A polynomial time approximation scheme for the closest substring problem. *11th Annual Symposium* *on Combinatorial Pattern Matching.* P. 99-107.

Shi, Xiaoqiu & Kong, Fansheng. (2001). Application of Computer Science in Bioinformatics. *Journal of Zhejiang University of Technology,* No. 29(2). P.161-165.

Wang, Jianxin, Huangmin & Li, Shaohua. (2006). Design and Implementation of a Distributed Computing Platform Based on Task Tree. *Mini-Micro Systems.* No. 27(5). P. 940-944.

Zhang, Yao, Zhang, Qing & Guo, Lishan. (2002). *Tutorial of Java Program Design.* Beijing: Metallurgical Industry Press. P. 415-451.

Zhu, Jianzhong. (2003). Implementation of Distributed Computing Based on RMI. *Computer Engineering.* No. 29(10). P. 60-62.

Table 1. The running time to solve the connectivity of the Mesh network with 120 crunodes and 5 dead dots

No.	Running time of single computer	Running time of double computers	Running time of four computers	Running time of twelve computers
1	72.566700	37.089070	18.995645	6.740065
2	72.670900	37.485003	19.243402	6.788951
3	72.589133	37.198142	18.989061	6.718954
4	72.600190	37.896455	19.358326	6.852362
5	72.576899	37.574122	19.177826	6.770057
6	72.498750	36.599087	18.779556	6.593502
7	72.709003	37.990450	19.495235	6.865860
8	72.598706	37.434912	19.137332	6.745364
9	72.602004	37.377904	19.150420	6.739288
10	72.554900	36.608720	18.758371	6.627607
Average	72.596719	37.325387	19.108517	6.744201

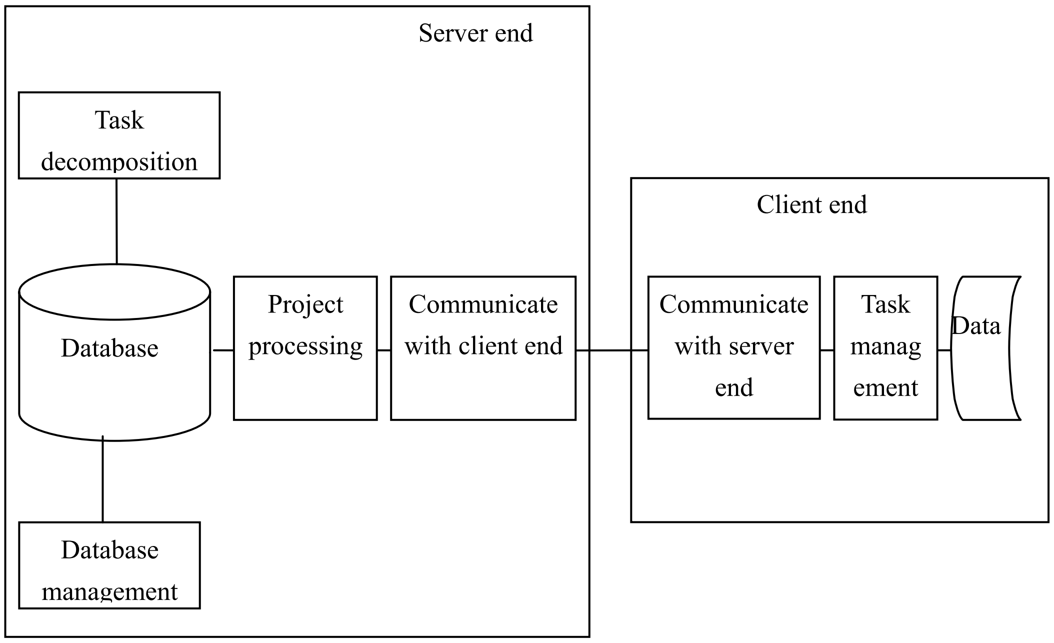

Figure 1. Platform System Structure

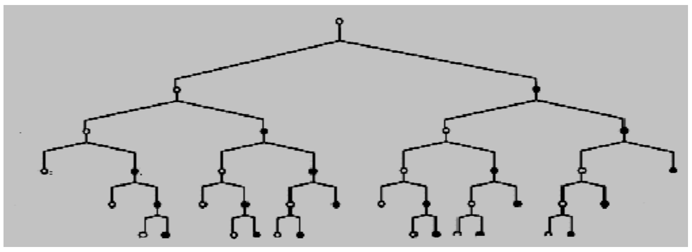

Figure 2. The Task Tree (n=6, c=3)

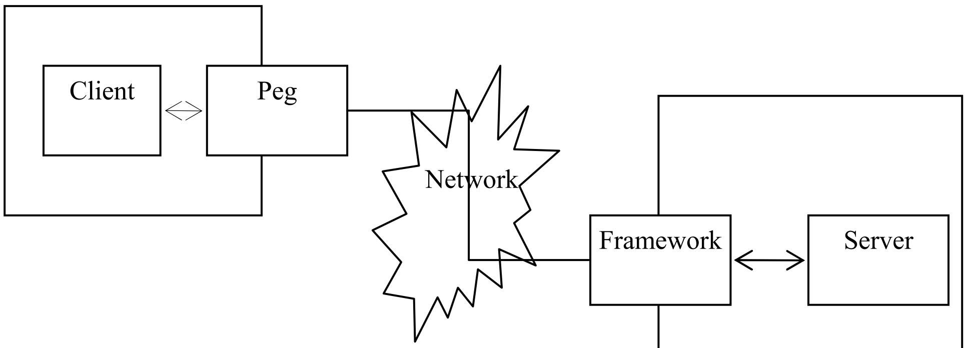

Figure 3. The Client End and Server End Connected by Peg and Framework