MM Research Preprints,165–172
No. 19, Dec. 2000. Beijing

On the Parallel/Distributed Network Computing Environment for Wu's Method

D. Lin, Y. Wu
Institute of Systems Science, Academia Sinica
Beijing, 100080, China

Abstract. In order to get better performance for the implementation of Wu's method, A network environment which could provide parallel/distributed processing is discussed. Feasibility, requirement, message transmission and architecture design with ELIMINO are presented.

1. Introduction

Mathematics mechanization[1] is a new mathematical research subject introduced by Wu Wentsün, a chinese famous mathematician, in the late of 1970's. Mechanization of a mathematical problem means "making inflexible" or "standardizing" in the process of proving or calculation, that is, after each step, we should have a fixed next step to choose. The essence of mathematics mechanization is to convert the qualitative difficulties inherited in usual mathematical proofs into quantitative complexities of calculations on standardizing the proof procedure in an algebraic manner. It's clear that mathematics mechanization is a new thought suitable for the era of computer.

In the past two decades, Wu's method has been developed very fast. Now it can be used not only in mechanically geometric theorem proving, but also in the field of multivariate polynomial equations solving, theoretical physics, CAGD, robotics and so on.

At present, several implementations of Wu's method have been developed by using mathematical softwares, such as ELIMINO[2], MAPLE, SACLIB. But as Wu's method has found applications in more and more fields, it is challenged by more complicated calculation problems, some of them can be calculated for several days or even failed because of memory limitation of a single computer. In order to get better performance for the implementation of Wu's method, it is reasonable and necessary to consider parallel/distributed implementation under the network environment. In another way, "the network is the computer" have been a concept with global impact and the network make many kinds of services easily accessible. Why do not we use it for the implementation of Wu's method to improve the performance? So, the establishment of Parallel/distributed Network Computing Environment(P/dNCE) for Wu's method become reasonable.

The parallel/distributed computation is built based on remote execution. That means the smallest execution unit is process. In the P/dNCE, One task is divided into two or more processes which contain same program set and can be executed on the different host at the same time. See Fig. 1,

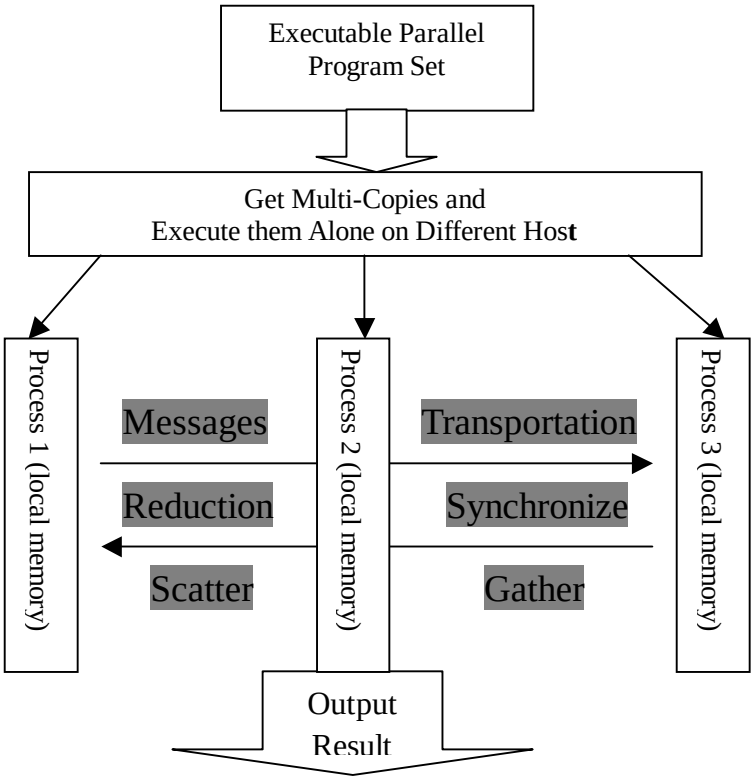

Fig. 1. Flow Chart of Parallel/Distributed Computation

From Fig. 1, we can see that more hosts (processes) might make computation more quickly; Each process could use the local memory; NFS (Network File System) services can help us to keep the storage of program set on only one host.

In this paper, we will show our thought about the construction of parallel/distributed network computing environment for Wu's method. A simple model for the environment is presented.

2. Feasibility and Requirements

In Wu's method, the concept of *characteristic set* is very important. The characteristic set of a polynomial set contains information about the zero structure of the related polynomial set. In fact, the most essential part of Wu's method is to calculate the characteristic set. After getting the characteristic set of a polynomial set, we can easily get or analyse the zeros of the polynomial set.

For a given polynomial set PS, the algorithm for computing its characteristic set can be stated as follows (see in [3]):

Input: a polynomial set PS

Output: the characteristic set of PS

Step 1 Let $BS = BasicSet(PS)$, if BS is contradictory then return .

Step 2 Let $RS = RemainderSet(PS, BS)$, if $RS = \emptyset$ then return(BS).

Step 3 Let $PS = PS \bigcup RS$, go to Step 1.

There are theorems to assert that the above procedure will be terminated after finite many steps, that is, we can certainly come to a point that $RS = \emptyset$. The final basic set BS (let it be CS) is the characteristic set to compute.

From the algorithm we can see that the basic operation in the CS method is the pseudo-division of polynomials. Analysis and experiments have shown that many costly pseudo-division are needed to form RS, and so the computations of set RS is the most time consuming task, especially when the intermediate polynomials become large. Thus the computation of the remainder set RS is a prime candidate for parallelization. In fact, it is the right point suited for parallelization. In any step of above algorithm, in order to form RS, every polynomials of the set PS will be picked and a pseudo reduction with respect to BS will carried out independently. During the computation of RS, PS and RS remain unchanged, so we can divide PS into several disjoint subsets of polynomials, and then pseudo-reduce these subset in parallel. We believe that the power of the Wu's method could be enhanced by use of parallel/distributed processing.

Considering that Wu's method is a symbolic method that manipulates polynomials, the parallel/distributed network computing environment of Wu's method should include following parts at least:

1. Basic symbolic computing functions libraries.

   Symbolic computation is expected to manipulate mathematical expressions symbolically and handle numbers exactly so that no error arises in their calculations. Since Wu's method, as a symbolic method, mainly handle polynomials, so the big integer (which can be as large as desired with the only limitation of total storage of the system) arithmetic[4] and polynomial operations[5] (such as polynomial factorization, GCD, pseudo-division of polynomial and so on) should be essential part of this library. Of course, this library should also contain some implementing functions of Wu's method.

2. Parallel and distributed computing libraries.

   Parallel and distributed computation let many different computers do same things at the same time. Clearly, these computers must communicate with each other for getting tasks , sending results, telling the progress and so on. So the communication between these computers is essential. At present, there are several libraries which can provide these communication, such as MPI (Message-Passing Interface) and PVM (Parallel Virtual Machine). Both MPI and PVM can deal with processes management, basic-message transmission and so on. See more in [6, 7, 8].

3. Message transmission for complex datum.

   Since Wu's method is a method specially for manipulating polynomials symbolically, so that only the simple data types (such as INTEGER, REAL, CHARACTER, LOGICAL DOUBLE and so on) provided by general parallel/distributed computing library (such as MPI, PVM)are not enough for the implementation of P/dNCE of Wu's method.

So new and complex data structures (such as big integer and polynomial) must be introduced.

4. Parallel algorithm of Wu's method.

   At present, most of implementation of Wu's method are based on the sequential algorithms. In order to construct a P/dNCE, it is necessary and important to develop high effective parallel algorithms. Discussions in Section 2 has given a natural parallelism of Wu's method.

3. Message Transmission

In any parallel/distributed computing environment, Sending and Receiving of messages are the basic communication mechanisms. Normally, there are two types of communication mechanism model. one is point-to-point communication and another is collective communication.

1. Point-to-point communication.
   Point-to-point communication is defined as communication that involves two processes: one sends data and another receives data. Following point-to-point operations SEND and RECV are most elementary and important, which is also the basis of any other communication operations.

   - **SEND**(*buf, count, datatype, dest, tag, comm*)

     Input Parameters

     buf: initial address of send buffer (choice)
     count: number of elements in send buffer (nonnegative integer)
     datatype: datatype of each send buffer element (handle)
     dest: rank of destination (integer)
     tag: message tag (integer)
     comm: communicator (handle)

     SEND does not return until the message data and envelope have been safely stored into the send buffer so that the sender is free to access and overwrite the send buffer. The message might be copied directly into the matching receive buffer, or it might be copied into a temporary system buffer.
     Message buffering decouples the send and receive operations. A blocking send can complete as soon as the message was buffered, even if no matching receive has been executed by the receiver. On the other hand, message buffering can be expensive, as it entails additional memory-to-memory copying, and it requires the allocation of memory for buffering.

   - **RECV**(*buf, count, datatype, source, tag, comm, status*)

     Input Parameters

     count: number of elements in receive buffer (integer)
     datatype: datatype of each receive buffer element (handle)

source: rank of source (integer)
tag: message tag (integer)
comm: communicator (handle)

Output Parameters

buf: initial address of receive buffer (choice)
status: status object (Status)

The receive buffer consists of the storage containing *count* consecutive elements of the type specified by *datatype*, starting at address *buf*. The length of the received message must be less than or equal to the length of the receive buffer. An overflow error occurs if all incoming data does not fit, without truncation, into the receive buffer. If a message that is shorter than the receive buffer arrives, then only those locations corresponding to the (shorter) message are modified.

2. Collective communication.
   The collective communication is defined as communication that involves a group processes which can complete following functions:

   - Broadcast from one member to all members of a group.
   - Gather data from all group members to one member.
   - Scatter data from one member to all members of a group.
   - Scatter/Gather data from all members to all members of a group.
   - Barrier synchronization across all group members.
   - Global reduction operations such as sum, max, min, or user-defined functions, where the result is returned to all group members and a variation where the result is returned to only one member.

In the above send-receive operations, a *datatype* object must be committed before it can be used in a communication, so that only the datatypes provided by parallel/distributed computing libraries can be transported. But Wu's method is a method specially for manipulating polynomials symbolically, so that only the simple data type, such as INTEGER, REAL, CHARACTER, COMPLEX, LOGICAL and so on, are not enough. Function **VECTOR** has been introduced for transmission of the polynomials and big integers in the P/dNCE of Wu's method.

VECTOR(*count, blocklength, stride, oldtype, newtype*)

Input Parameters

count: number of blocks (nonnegative integer)
blocklength: number of elements in each block (nonnegative integer)
stride: number of elements between start of each block (integer)
oldtype: old datatype (handle)

Output Parameter

newtype: new datatype (handle)

VECTOR allows replication of a specific datatype into locations that consist of equally spaced blocks. Each block is obtained by concatenating the same number of copies of the old datatype. The spacing between blocks is a multiple of the extent of the old datatype. For example, assume that we have type *oldtype* with extent 16. the function call of

$$VECTOR(2, 3, 4, oldtype, newtype)$$

will create the type *newtype* with type *oldtype*:

$$\{(double, 0), (char, 8), (double, 16), (char, 24), (double, 32), (char, 40),$$
$$(double, 64), (char, 72), (double, 80), (char, 88), (double, 96), (char, 104)\}$$

That is, two blocks with three copies of the *oldtype*, with a stride of 4 elements (bytes) between the blocks.

After giving the definition of new datatype, the followed three steps is essential for the transmission:

1. Commit a new datatype with command **COMMIT**(newdatatype);
2. Send/receive a new datatype as normal datatype with command **SEND/RECV**(buf, count, newdatatype, dest, tag, comm);
3. Free a newdatatype defined by user with command **FREE**(newdatatype).

Now, let's see how to transmit big numbers and polynomials by using the above function VECTOR. From [4], we can see that a big number can be expressed in a list of integers, so it is easy to transmit big integers by using VECTOR. Now let us consider polynomials. In fact, any polynomial can be expressed with two list: One is variable sequence list $l1 = [x_1, x_2, x_3, \cdots]$ and another is a list of list: $l_2 = [l_{2,1}, l_{2,2}, l_{2,3}, \cdots]$, where the first item of the list $l_{2,i}$ is the basic coefficient of ith term of the polynomial and other items of the list $l_{2,i}$ are the exponents of corresponding variables in the ith term of the polynomial. So transmitting polynomial can be converted to transmitting above 2 lists by using VECTOR. Of course, this method of transportation maybe not the best when parallel/distributed efficiency is considered.

4. Architecture Design with ELIMINO

As we discussed in section 2, Wu's method contains natural sources of parallelism. for example, it produces a lot of mutually independent subproblems (such as puedo-remainders of polynomials) that may be treated in parallel by process at different nodes. We believe that the power of the Wu's method could be enhanced by use of parallel/distributed processing. In this section, we will give a design of P/dNCE for Wu's method with ELIMINO.

ELIMINO is a open and flexible symbolic computation software platform developed in MMRC, Institute of System Science, the Chinese Academy of Sciences. It provides various kinds of functions for multiprecision numbers and multivariate polynomials, as well as a complete implementation of Wu's method. so we can easily implement parallel algorithm of Wu's method by using the library provided by ELIMINO and parallel/distributed libraries discussed in section 2.

There are several (star, tree and circle) topologies and each topology include several models (master-slave and parallel) that can be used to accomplish our goal. In our P/dNCE of Wu's method, a master-slave model of star topology (shown in Fig. 2) is selected, that is one master process and several slave processes in the parallel computation scheme. Master process performs special task such as initialization, coordinating the work of slave processes, collecting data and printing out the computation results, the slave processes perform the actual computations such as pseudo-reduction.

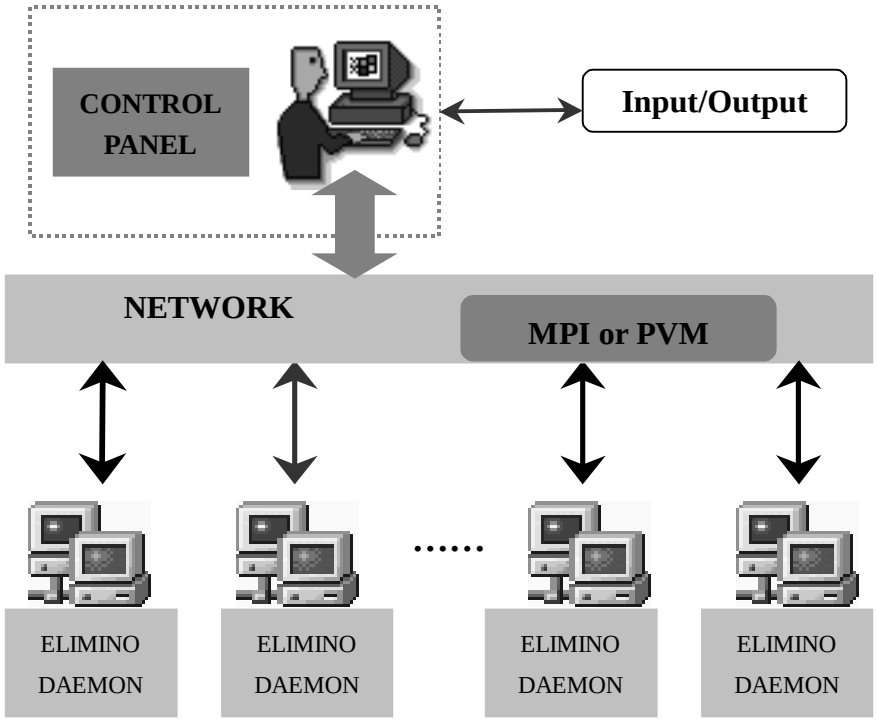

Fig. 2. Architecture of P/dNCE for Wu's Method

In the master-slave model, one of the major concern is balancing the loads of slave process. Ideally, all slaves are kept busy doing useful work all the time till the entire computation is finished. Perfection is hard to achieve, a major reason for the load imbalance is that different process take different amount of time on the different nodes, as a result, one or several slaves can be idle while other slaves are still busy although they all execute the same program set.

A flexible load balancing scheme is required in order to achieve good performance. there are the basic scheduling schemes: dynamic scheduling scheme and static scheduling scheme. dynamic scheduling scheme would send polynomials from master to each slave when it is ready. This can reduce idle time but maybe incur too much communication overhead and may cause a bottleneck specially when many slave processes are used. static scheduling scheme would partition the total polynomial system into equal-sized sub-systems of polynomials and send each sub-system to a different slave. this approach reduces communication time but may result in unacceptably high idle time since it is hard to predict each sub-system is harder to deal with. base on analysis, a hybrid approach of above two scheduling schemes has

been adopted for our implementation. In the hybrid scheme, we would farm out *several(s)* polynomials at once using static scheduling scheme and hold the *remaining (r)* polynomials to dish out using dynamic scheduling scheme. The ratio *s/r* to use depends on the number of slave precesses *(n)* and the total number *(t)* of polynomials to be treated with. the value of *(t)* depends on how many intermediate polynomials are generated in the algorithm and is not easily predicted. more experimentation can provide some basis for heuristic value of *s/r*.

References

[1] W. T. Wu, On the Decision Problem and the Mechanization of Theorem in Elementary Geometry, Scientia Sinica 21(1978), 159-172; Also in Automated Theorem Proving: After 25 years, A. M. S., Contemporary Mathematics, 29(1984), 213-234.

[2] D. Lin, J. Liu & Z.Liu, Mathematical Research Software: ELIMINO, ascm'98, pp.107-114.

[3] D. Wang, Master Degree Thesis: Polynomial equations Solving & mechanical geometric Theorems Proving, 1993.

[4] H. Yang, Z. Liu, D. Lin, Development of An Object-Oriented number system, mm-research Preprint, 1999(18), pp.212-219.

[5] Y. Wu, Master Degree Thesis: Polynomial Factorization Algorithm and its implementation in ELIMINO System, 1999.

[6] S. Gray, A guide to Programming with MP Version 1.1.3, Department of Mathematics and Computer Science, Kent State University, 1997.

[7] W. Gropp & E. Lusk, User's Guide for mpich, a Portable implementation of mpI, Argonne National Labrotary, University of Chicago,1996.

[8] W. Gropp & N. Doss, MPICH Model MPI Implementation reference manual, Argonne National Labrotary, University of Chicago, 1999.

[9] I. A. Ajwa, P.S. Wang, D.Lin, An Attempt for Parallel computation of characteristic Set, 2000.

[10] P. S. Wang, IAMC: Internet Accessible Mathematical Computation, ascm'98, pp.1-14.

[11] W. T. Wu, Mathematics Mechanization, Kluwer Academic Publishers, 2000.