Ultra Long Integer Multiplication on GDPS

Y K Viswanadham
Dept of CSE
S V University
Tirupati – 517 502, India

Dr Ch D V Subba Rao
Dept of Computer Sci Engg
S V University
Tirupati – 517 502, India

T V Subrahmanyam
Dept of IT, Gudlavalleru
Engineering College,
Gudlavalleru-521 356, India

ABSTRACT

Many Internet applications require intensive cryptographic calculation such as public-key encryptions and digital signatures. These schemes require a computation of large integer multiplications. Those cryptographic schemes are vulnerable to a brute-force attack, and the large key is the countermeasure. In practice, the key size that makes brute-force attack impractical will slows down the speed of encryption and decryption. Multiplication of two very long integers usually takes time to compute. Distributed Karatsuba algorithm is proposed to reduce the time of multiplication of two very long digits. The proposed architecture that makes use of Karatsuba algorithm achieves faster multiplication.

Keywords

GDPS, IMDP, Karatsuba Algorithm.

1. INTRODUCTION

Distributed computing is the process of aggregating the power of several computers to collaboratively run a single computational task in a transparent and coherent way. In other words it enables usage of the idle time of large numbers of networked computers to work on projects too large for any single group[7]. Recent distributed computing projects have been designed to use the computers of hundreds of thousands of volunteers all over the world, via the Internet, to look for extra-terrestrial radio signals [8], investigate and reduce uncertainties in climate modeling [9], detect earthquakes [10], prime numbers so large that they have more than ten million digits [11] etc. These projects require so much computing power to solve and they would be impossible for any single computer to solve in a reasonable amount of time.

1.1 Cryptography

During this time when the Internet provides essential communication between tens of millions of people and is being increasingly used as a tool for commerce, security becomes a tremendously important issue to deal with. There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords. One essential aspect for secure communications is that of cryptography. But it is important to note that while cryptography is necessary for secure communications, it is not by itself sufficient. This field required very fast large integer multiplication. Multiplication of large integers is a fundamental requirement in polynomial multiplication for signal processing, coding theory [1].

2. PROBLEM DEFINITION

Multiplication is one of the most frequently used arithmetic operations in public key cryptography and the performance of a cryptosystem often depends mostly on the efficiency of a multiplication operation. The efficiency of cryptosystem is often determined based on how fast an encryption/decryption operation can be done. Modular exponentiation is hard operation used in most cryptosystems. It consists of performing modular multiplication repeatedly. Modular multiplication can be performed by first multiplying then reducing or by interleaving the multiplication and reduction steps. The former way is preferred as there are very fast multiplication algorithms as they were over studied while the latter is used when there is only a limited storage in the case of smartcards.
Multiplication of long integers is cornerstone primitive in most public-key cryptosystems. These kinds of multiplications are also required in mathematics, signal processing and astrophysics [1]. Multiplication of large numbers requires huge resource of time and space. The study of fast multiplication algorithms is always an important task pursued by mathematicians and computer scientists.

2.1 Existing Systems

The multiplication of ultra long integer can not be handled by a single computer in reasonable amount of time, but such kind of problems can be solved by Supercomputer.

2.2 Limitations of Existing System

The super computers may be fast, however those are not widely available because of its huge cost and maintenance. The simple client-server system may not achieve the speed of the super computer.

3. PROPOSED SOLUTION

The main objective of this proposal is multiplying the two ultra-large numbers (12000 digits) using low cost computing resources and getting faster, accurate result. Generalized Distributed Processing System (GDPS), which aims at providing a simple, generic, reusable and extensible platform for the purpose of Internet, based Massively Distributed Processing (IMDP). These systems are used to solve scientific problems which are too large to be solved by even the most powerful Supercomputers. In the Internet world there are so many computers having idle processing. So we can use such kind of idle CPU processing time by activate our own program on the idle client. For example

SETI@Home[8], Predictor @ Home [6]...etc, provide for solving huge data analysis problems in a Distributed Environment.

3.1 Methodology

We propose GDPS for solving ultra long integer multiplication using Karatsuba algorithm. With this we can achieve faster result. It is an economical solution, because large volumes of computing resources are available on the internet. We can form those resources as a community and use them in their idle time as background process of our screensaver. Any Solution to Distributed Processing must provide the following:

- The Breakdown of Large tasks or Jobs into smaller independently possible portions or subtasks.
- The Allocation of these subtasks to the Participant Computers.
- The Physical Distribution of these tasks to the Participant Systems.
- The Processing of these subtasks by the relevant systems.
- The Physical Collection of partial results from the Participant Computers.
- The Combination of the partial solutions thus received into a Complete Solution.

3.1.1 Server Side

The server side architecture is as illustrated in the Figure 1.

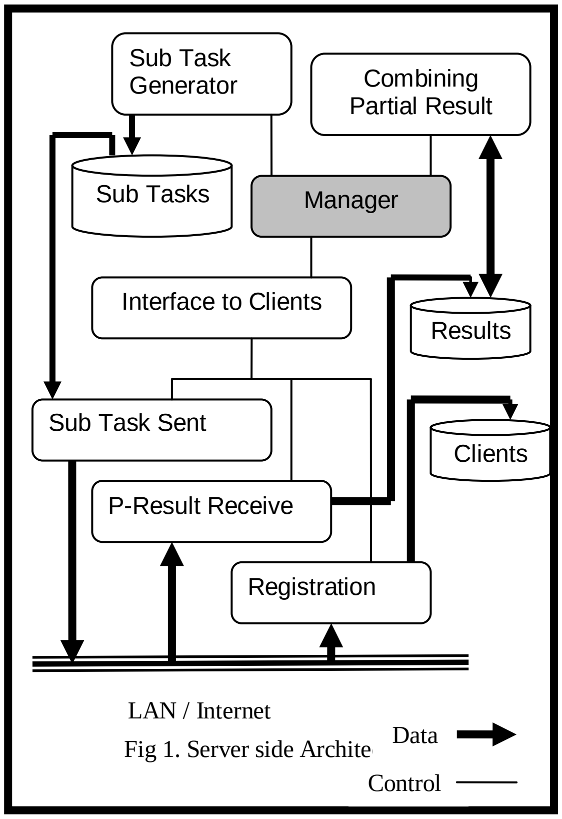

Fig 1. Server side Architecture

Manager

This is the Most Important Component of the Server End Architecture. It accesses the Sub Task, Partial Result and Client Databases, and interfaces with the Internet Interface, The Sub task Generator and the Combining partial result Algorithms.

The partial result combination algorithm interface

This involves combining the partial result into final result and it is imitated by manager after receiving all partial results.

The subtask generator algorithm interface

This involves provision of a program call to the end user whereby he may create and store a sub task in the subtask database.

The internet interface

The Internet Interface has modules to handle the server end databases.The Server end of GDPS bears 3 databases which may be implemented using any standard database package or as files. They are

- A Database of the Various Subtasks or Work Units yet to be assigned to volunteers. This is a file handled by the subtask generator and updated or flagged when work units are assigned.
- A Database of the partial result for each subtask. All received result fragments are added here. Once all fragments for a subtask are received, the combination algorithm is initiated and these results are passed to it.
- A Database of clients who involved in this evaluation.

3.1.2 Client Side

The client side architecture is illustrated in the figure 2

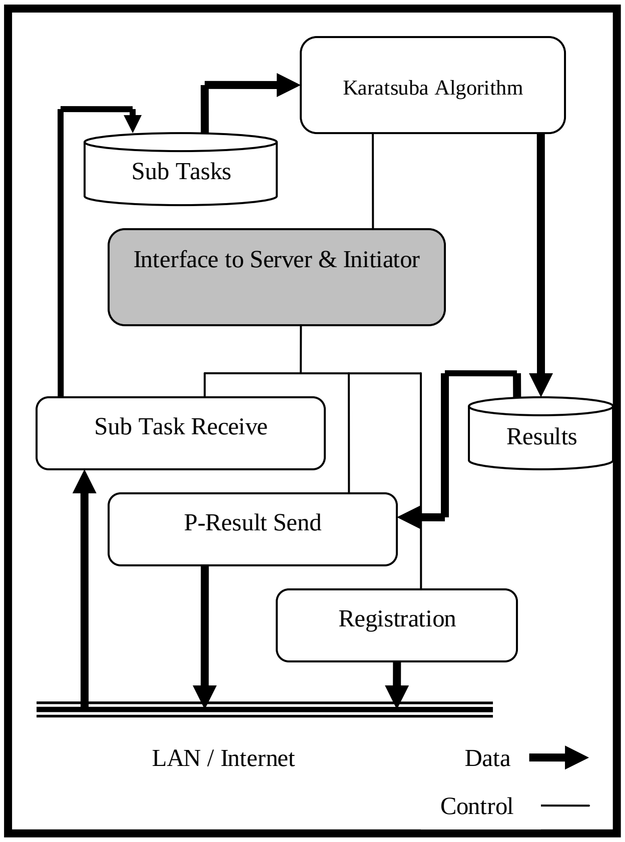

Fig 2. Client side Architecture

The Client End Algorithm Initiator

This component detects the activation of the Screensaver in Windows based PCs or waits for an explicit start up on other systems. It then initiates the relevant client end algorithms for subtask on the client PC.

The Internet Interface

This detects the initiation of an Internet Connection and proceeds to upload partial result and download more subtasks. The Client end PCs may not have standard database systems available and hence all data would be stored in files on the Clients.

3.2 Karatsuba Algorithm

In 1962 Karatsuba developed an algorithm for multiplying two numbers by using a divide-and-conquer method [2].Suppose two n-digit radix b numbers X and Y are expressed as $x_1 b^{n/2}+x_0$ and $y_1 b^{n/2}+y_0$ respectively.

The grade-school method computes the product Z of X and Y as follows:

$$\begin{aligned} Z &= X.Y \\ &= (x_1 b^{n/2}+x_0)\,(y_1 b^{n/2}+y_0) \\ &= x_1 y_1 b^{n} + (x_1 y_0+x_0 y_1)\, b^{n/2}+x_0 y_0 \end{aligned}$$

Karatsuba proposed a multiplication algorithm that computes the product Z of X and Y as follows:

$$\begin{aligned} Z &= X \;.\; Y \\ &= (x_1 b^{n/2}+x_0)(y_1 b^{n/2}+y_0) \\ &= x_1 y_1\, b^{n} \\ &\quad + [(x_1+x_0)(y_1+y_0) - x_1 y_1 - x_0 y_0] b^{n/2} \\ &\quad + x_0 y_0 \end{aligned}$$

To multiply two n digit numbers, the grade-school method has four n/2 digit multiplications and Karatsuba Algorithm has three n/2 digit multiplications [3].

$$\begin{aligned} T(n) &= 4\,T(n/2)+cn \\ &= n^{\log_2 4} \\ &= n^{2} \end{aligned}$$

For Karatsuba Algorithm

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 3\,T(n/2)+cn \\ &= n^{\log_2 3} \\ &= n^{1.58} \end{aligned}$$

4. EXPERIMENTAL RESULTS

The resulting program is implemented on *Intel Core2Duo* 2.0 GHz Processor with Jdk1.6.0 programming language. A graphical user interface is designed with swings to facilitate the user give the input in file mode as well as manually. The given input is divided into no. of segments which is a fixed length of digits. Those segments will be given as subtasks to clients. Those clients will send the result back to the server and stored in file. Finally combining algorithm will conquer the partial result and gives the final result.

It is tested for various combinations of size and no. of clients. The experimental results are analyzed in Fig-3 & 4 and noticed that best performance is only possible with no. of clients with respect to size of the problem. The increase of no. of clients for small problem degrades the performance of the system due to overhead. So the no. of clients should directly proportional to the Length of the problem up to threshold value.

4.1 Performance Analysis

The experiments have conducted on LAN in idle condition; all clients have the same specification Intel *Core2Duo* 2.0 GHz Processor 1 GB RAM.

Table 1: Experimental Results(Time ms)

For school book method

T(1) = 1

No.of Clients / Length	**1**	**2**	**3**	**4**	**5**	**6**	**7**
100	235	63	31	62	78	94	104
200	203	141	140	78	79	102	124
300	438	313	234	157	188	218	252
400	656	594	531	391	203	297	341
500	812	766	656	437	302	406	512
600	1250	1047	1079	1046	1126	859	902
700	1765	1594	1563	1078	1124	844	983
800	1984	1687	1749	1797	1735	1579	1592
900	3374	3250	3172	3156	3124	3117	3109
1000	5265	4937	4906	4829	4719	4680	4562
1500	10266	10204	10141	9859	9625	9214	8609

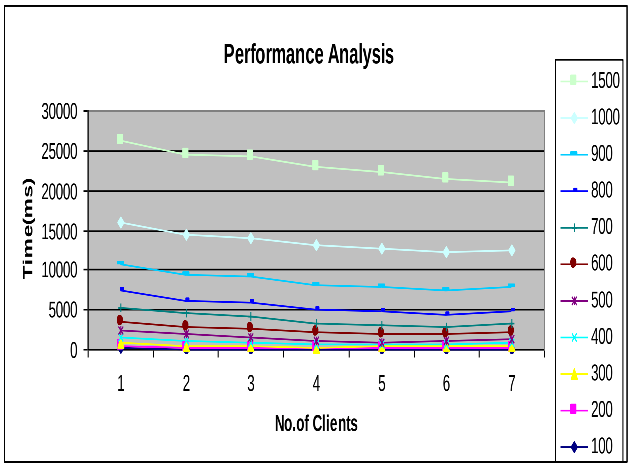

Fig 3.No.of Clients Vs Time for Size of the Problem

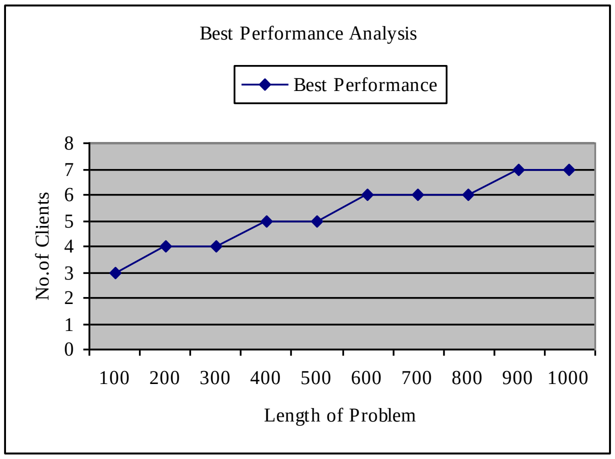

Fig.4: Length of the problem Vs No. of Clients

5. CONCLUSION

It has been showed that distributed Karatsuba algorithm running on network of nodes/ systems is fast and economical. Distributed Computing is a new methodology of solving problems when it was first introduced. With its advancements, many small computers could be linked together in order to achieve tremendous processing power. This computing power will be an alternative solution for costly supercomputers.

We hope that the generic platform and protocols we have conceived will fuel further growth in this emerging field, with newer and better applications being developed every day. The Internet is fast changing, with round the clock, high-speed connections replacing intermittent and slow dial up lines. Thus, looking towards the future, one may say that a shift from the Client-Server models of IMDP that we see today; to an even more generic Peer-to-Peer model is in the offing.

In this case, the distributed computing concept may be applied by all users connected to the Internet to solve their specific problems, each with the power of several thousand PCs at his disposal. Such a scenario would surely require an agreed set of standards for data communication, subtask assignment and result correlation, with support for multiple algorithms at all points in the network. It is our contention that GDPS, with its protocols, predefined structures and support for multiple algorithms could well prove to be a precursor to these.

6. FUTURE SCOPE

We are working on a new parallel implementation of the Karatsuba algorithm for handling division operation. We hope that will give us notable results. The fast multiplication approach can be used in prime number generation.

7. REFERENCES

[1] Andre Weimerskirch, "Generalizations of the Karatsuba Algorithm for Efficient Implementation",http://www. Crypto.ruhr-uni-bochum.de/imperia/md/content/textekaweb.pdf, 2003

[2] A.Karatsuba and Y.Offman, "Multiplication of multi digit numbers on automata", Soviet Physics-Doklady,1962.

[3] Chin-Bou Liu, "Design and Implementation of Long-Digit Karatsuba's Multiplication Algorithm Using Tensor Product Formulation", In The Ninth Workshop on Compiler Techniques for High-Performance Computing, 2003.

[4] Dan Zuras, "More On Squaring and Multiplying Large Integers", IEEE Transactions on Computers, AUGUST 1994

[5] J. Tudor, "Using the Parallel Karotsuba Algorithm For Long Integer Multiplication", European Conference on Parallel Processing,, 1997.

[6] M. Taufer, "A Protein Structure Prediction Supercomputer Based on Public-Resource Computing". IEEE Transactions on Parallel and Distributed Systems, Aug 2006.

[7] en.Wiktionary . org / wiki / distributed _ computing

[8] SETI@Home - setiathome.ssl.berkeley.edu

[9] Climate prediction - Climateprediction.net

[10 Quake-Catcher Network - qcn.stanford.edu

[11] PrimeGrid-http://distributedcomputing.info/ap-math.html#primegrid