Parallel String Matching Algorithm Using Grid

K.M.M Rajashekharaiah[1], Ch MadhuBabu[2] and Dr. S. Viswanadha Raju[3]

[1]Asso Prof, Dept of CSE, BVRIT, Research Scholar Rayalamaseema University, Kurnool
rajashekharaiah.kmm@bvrit.ac.in
[2]Professor Ch MadhuBabu
madhubabu.chunduri@bvrit.ac.in
[3]Professor, School of IT, JNTUH
viswanadha_raju2004@yahoo.co.in

Abstract

Grid computing provides solutions for various complex problems. Implementing computing intensive applications such as string matching problem on grid is inevitable. In this paper we present a new method for exact string matching based on grid computing. The function of the grid is to parallelize the string matching problem using grid MPI parallel programming method or loosely coupled parallel services on Grid. There can be dedicated grid, other way it can be resource sharing among the existing network of computing resources, we propose dedicated grid. Parallel applications fall under three categories namely Perfect parallelism, Data parallelism and Functional parallelism, we use data parallelism, and it is also called Single Program Multiple Data (SPMD) method, where given data is divided into several parts and working on the part simultaneously. This improves the executing time of the string matching algorithms on grid. The simulation results show significant improvement in executing time and speed up.

Keywords: *Grid Computing, Loosely coupled, SPMD, Parallelism.*

1. Introduction:

String matching is very important subject in the domain of text processing and it has been one of the most extensive problems in computer technologies during past two decades. It has applications such as DNA analysis, information retrieval systems and several other fields. String matching is the process of finding the occurrence of a pattern P in a text T, where T is longer than P. This occurrence is either exactly matched or partially matched with the pattern, based on this; string matching algorithms are classified as Exact and Approximate string matching algorithms. Exact string matching algorithms are concerned with the number of occurrences of the pattern into a given text, while approximate string matching algorithms are concerned with the similarity percentage between the pattern and the text or any part of the text [2][3]. This paper concentrates on exact string matching algorithms. Data dependence is minimal in string matching operations and hence it is ideal for parallelization. Parallel applications fall under three categories namely Perfect parallelism, Data parallelism and Functional parallelism [10][11].

1.1 Perfect parallelism: Also known as embarrassingly parallel. An application can be divided into sets of processes that require little or no communication.

1.2 Data parallelism: The same operation is performed on many data elements simultaneously. An example would be using multiple processes to search different parts of a database for one specific query.
A Single Program Multiple Data streams (SPMD [18]) is a parallel processing technique where the same program (or task, in workflow terminology) is applied to multiple data elements. All elements are processed in parallel, if – as before – no dependencies exist among them. The *Data Parallelism* pattern applies this idea to speed up the execution of task that are run over large input datasets. Instead of feeding the entire dataset to a task, the data is partitioned and a copy of the same task is applied in parallel over each (independent) partition. Once all data partitions have been scheduled for execution, there can be different synchronization semantics to proceed with the execution (wait for all, wait for one, n-out-of-m).

1.3 Functional parallelism: Often called control parallelism. Multiple operations are performed simultaneously, with each operation addressing a particular part of the problem.

1.4 Single Program Multiple Data (SPMD)
These are the applications in which the input data can be partitioned and processed concurrently using the same program. This type of application comprises the majority of applications that utilize the Grid today and covers a wide range of domains. The primary motivation for running this class of applications on a grid is to significantly improve performance and/or scope by scaling the application out to as many resources on the grid as possible.

SPMD usually refers to message passing programming on distributed memory computer architectures. A distributed memory computer consists of a collection of independent computers, called nodes. Each node starts its own program and communicates with other nodes by sending and receiving messages, calling send/receive routines for that purpose. Barrier synchronization may also be implemented by messages. The messages can be sent by a number of communication mechanisms, such as TCP/IP over Ethernet, or specialized high-speed interconnects such as Myrinet and Supercomputer Interconnect. Serial sections of the program are implemented by identical computation on all nodes rather than computing the result on one node and sending it to the others.

The remainder of this paper structured as follows, section 2 introduces Grid computing model. Next in section 3 we discuss the related carried out in parallel string matching. In section 4 we describe text partition, which is used for distribution of divided given text to different nodes. In section 5 we discuss experiment environment and results and concluding in section 6.

2. Grid Computing Model

Grid computing provides new solutions for numerous complex problems. It is inevitable to implement the distributed parallel computing of large-scale problems with the grid. There are two types of parallelization possible in grid environment: Loosely coupled parallel services and tightly coupled parallel (Grid MPI parallel program) services [1], we use loosely coupled technique in our simulation.

2.1 Loosely coupled parallel Services:

The architecture of the implementation framework of loosely coupled parallel services on grid is shown in the figure 1. The input data can be partitioned and processed concurrently using the same program, alternatively it could be assumed that the database of an information retrieval system contains independent documents. Therefore in both the cases all the above partitions yield a number of independent tasks (threads), each comprising some data (i.e. a string and a large subtext) and a sequential string matching procedure that operates on that data. Further each task completes its string matching operation on its local data and returns the number of occurrences [1].

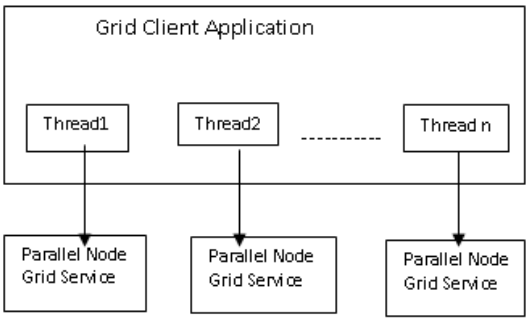

Figure 1: Loosely coupled services on grid.

2.2 Model Advantages

1. Much more efficient use of idle resources. Jobs can be farmed out to idle servers or even idle desktops. Many of these resources sit idle especially during off business hours. Policies can be in place that allows jobs to only go to servers that are lightly loaded or have the appropriate amount of memory/cpu characteristics for the particular application. **2.** Grid environments are much more modular and don't have single points of failure. If one of the servers/desktops within the grid fails there are plenty of other resources able to pick the load. Jobs can automatically restart if a failure occurs**. 3.** This model scales very well. Need more compute resources? Just plug them in by installing grid client on additional desktops or servers. They can be removed just as easily on the fly. This modular environment really scales well. **4.** Jobs can be executed in parallel speeding performance. Grid environments are extremely well suited to run jobs that can be split into smaller chunks and run concurrently on many nodes.

3. Related Work

Several computational models have been considered for parallel string matching algorithms the PRAM model, mesh structure [3] and S.V Raju and A. Vinay Babu proposed efficient parallel pattern matching using partition method [2] and obtained deterministic one and two dimensional arrays which are probably the best in both preprocessing, text search and reduced many calls across back end interface [3].

Data dependence is minimal in string matching operations and hence it is ideal for parallelization. Parallel computing models helps to improve executing time. In parallel computing a problem is divided into smaller problems which are then processed

simultaneously. The parallelism degree can be controlled by changing Several computational models have been considered for parallel string matching algorithms the PRAM model, mesh structure [14] and etc. The parallel string matching algorithm is often said to be optimal if its cost is O (nm).

A Single Program Multiple Data streams (SPMD [15]) is a parallel processing technique where the same program (or task, in workflow terminology) is applied to multiple data elements. All elements are processed in parallel, if – as before – no dependencies exist among them.

Park and George [10] presented a dataflow schemes string matching algorithms parallelization. In their work, they covered exact matching and k mismatched problems. The time complexity of the proposed parallel algorithm was O ((n/d) +α), 0<=α=m, where n and m are the length of the text and pattern with (n>>m) and the value of the variable d, which is present in the input stream. Due to the one pass dataflow algorithms, there was no preprocessing and memory space used for this schema.

S.V Raju and Vinay Babu [4] proposed a parallel technique for string matching algorithm. They considered the linear array with reconfigurable pipelined bus system (LARPBS) and 2D LARPBS for string matching in their work, which has many existing applications such as cellular automata, computational biology and string database. The proposed method increases the speed of the string matching process. They obtained the time complexity O (1) for the string matching on 2D LARPBS where no preprocessing is done to the text and pattern.

Recently Mosleh M. Abu-Alhaj and M. Halaiyqah [9] proposed a general platform called the PXSMAlg platform, in order to improve the Exact-Strings-Matching algorithms performance. In this platform they used MPI model over the Master/Slave paradigm to improve the performance in terms of speeding up of executing time.

4. Text Partitioning

The exact string-matching problem can achieve data parallelism with data partitioning technique. We decompose the text into r subtexts, where each subtext contains (T/p)+m-1 successive characters of the complete text. There is an overlap of m-1 string characters between successive subtexts, i.e, a redundancy of r(m-1) characters. Alternatively it could be assumed that the database of an information retrieval system contains r independent documents. Therefore, in both the cases all the above partitions yield a number of independent tasks each comprising some data (i.e. a string and a large subtext) and a sequential string matching procedure that operates on that data. Further, each task completes its string matching operation on its local data and returns the number of occurrences. Finally, we can observe that there are no communication requirements among the tasks but only global (or collective) communication is required. The main issue to be addressed is how the several tasks (or r subtexts) can be mapped or distributed to multiple processors for concurrent execution. In [5] different ways of distributing the database across a multi computer network were discussed. Let p be the number of processors in network and r be the number of subtext in the whole collection then the text partition is defined as, if r=p then each subtext contains T/p+m-1 characters. This is called static allocation of subtext as shown in Fig2.

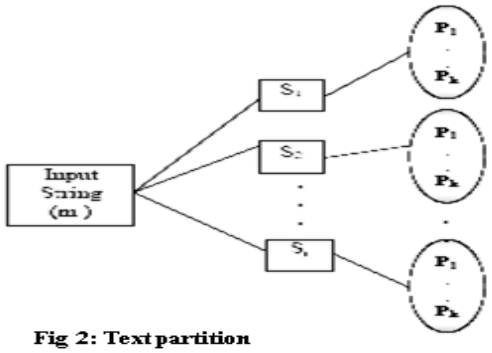

Fig 2: Text partition

4.1 Implementation Procedure

4.1.1 String Matching Algorithms

Researchers have developed several exact pattern matching algorithms with the view to enhance the searching processes by minimizing the number of characters comparisons and maximize the length of the shifts [16]. Boyer-Moore (BM) [17] algorithm compared character from right to left of pattern and did not require the whole pattern to
be searched in case of mismatch. In case of a mismatch or complete match, it used two shifting rules; bad character and good suffix rules to shift the pattern toward right. The time complexity of preprocessing phase is O(m). Worst case running time of searching phase is $O(nm + |\sum|)$. The best case of searching phase is $O(n/m)$. Boyer-Moore Horspool (BMH) [18] is improved version of the Boyer-Moore algorithm. It used only the bad character rule of the Boyer-Moor algorithm to improve the length of the shifts. In preprocessing phase, it scans pattern for right most character of partial text window from right to left. As occurrence of the character is found it aligned the found character with the rightmost character of the text window. If the character is not found in the pattern then take maximum shift of size m. It's preprocessing time complexity is $O(m)$ and searching time complexity is $O(mn)$. Boyer-Moore Smith (MBS) [19] noticed that computing shift by BMH sometimes maximize the shifts than QS. It uses the bad character shifting rule of BMH and QS bad character rule to shift the pattern. It's preprocessing time complexity is $O(m+|\sum|)$ and searching time complexity is $O(mn)$. The preprocessing phase of Quick Search (QS) [20] algorithm scans pattern from right to left for one character right to the partial text window to identify the shifts by applying bad character shifting rule. It performs character comparison from left to right of the pattern with selected text window. The worst case time complexity of QS algorithm is same as BMH algorithm but it can take more steps in practice.

According to the implementation of loosely coupled parallel services, parallel string matching algorithm on Grid computing environment is developed as follows [1].The input data can be partitioned and processed concurrently using the same program.

1. Write a program which operates on different data streams as Grid service.
2. Create a distributed thread by either by extending the thread or implementing Runnable interface in java.
3. Create a grid service instance inside the run method of the distributed thread.
4. Create a grid client application program that can manage and run distributed threads.
5. The Grid client application collects results of Grid service instances on Grid nodes.

5. Results and Discussions.

5.1 Test Environment: We considered a heterogeneous environment, in which we used 10 computers, 8 of which had the configuration of Intel 2.2 GHz CPU, 512 MB RAM, windows XP OS, Globus toolkit 4.0. And 2 computers had the configuration of Intel 2.8vGHz 1 GB RAM UBONTO Linux OS and Globus 4.0 toolkit. Each node is interconnected through 100MBits/Sec Fast Ethernet card. We have carried out 10 different experiments to search patterns of fixed length in a file of size 3 MB. We have used Quick search algorithm, the result showed improvement in the executing time and speedup. The fig 3 shows improve in executing time and fig 4 shows improvement in speedup.

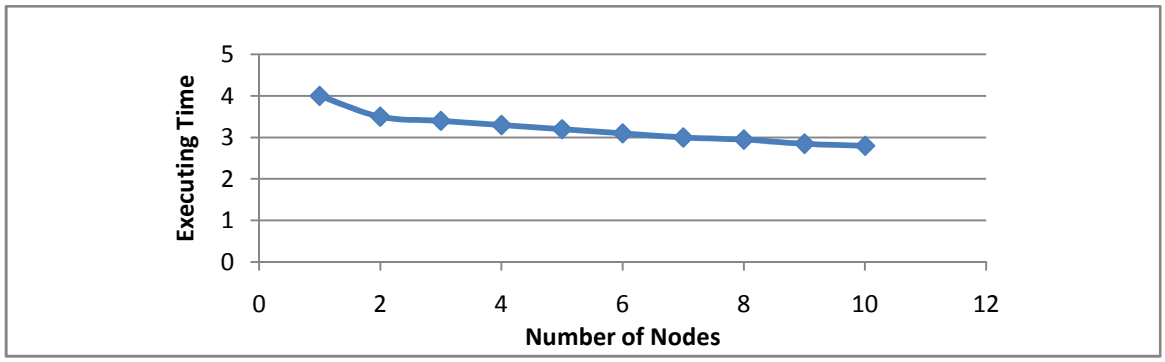

Fig 3. Executing Time Vs Number of Nodes.

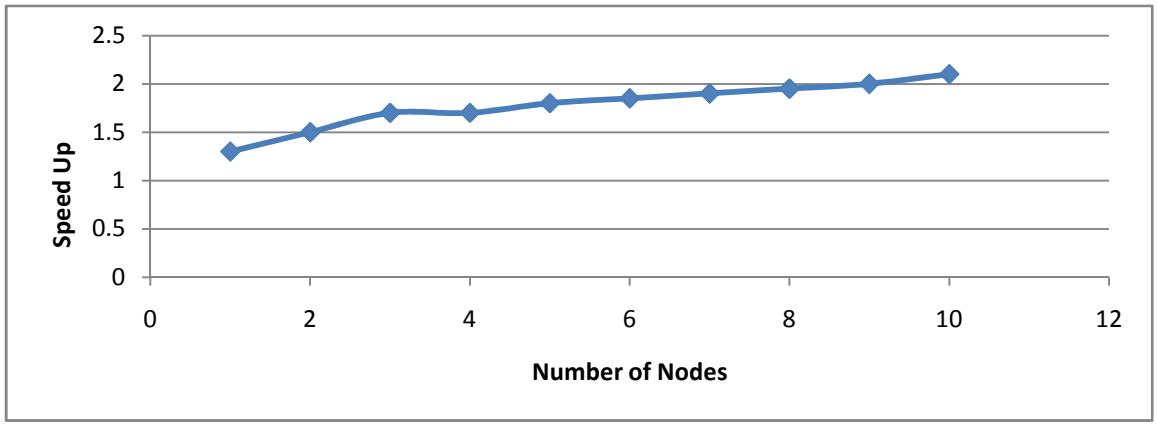

Fig 4. Speed Up Vs Number of Nodes.

6. Conclusions and Feature work

In this paper we proposed generalized parallel grid computing environment for executing exact-string matching algorithm, and used SPMD method. The simulation results show that the performance of string matching algorithms namely execution-time and speedup improved

compared to sequential execution. Our feature work includes evaluating reliability of Grid and memory optimization.

Acknowledgements

We sincerely thank our colleagues in the Department of Computer Science and Engineering for their advice and helpful discussions.

7. References

[1] Weiwei Lin, Changgeng Guo, Deyu Qi, Yuehong Chen and Zhang Zhili, (2006) "Implementations of Grid-Based Distribute Parallel Computing", IMSCCS'06.

[2] S.Viswanadha Raju and A.Vinaya Babu, "Efficient Parallel Pattern Matching Using Partition Method", Proc. of PDCAT 2006, IEEE Computer society, 2006.

[3] S.Viswanadha Raju and A.Vinaya Babu," Backend Engine For Parallel String Matching Using Boolean Matrix", Proc.of PARALEC 2006, IEEE Computer society, 2006.

[4] A.Vinaya Babu and S.Viswanadha Raju, "Parallel Algorithms for String Matching on Single and Two Dimensional LARPBS", Journal of Computer Science, 2007.

[5] Park, J.H., And George, K.M., "Parallel string matching algorithms based on Dataflow", In proceedings of the 32nd Hawaii International on System Sciences, 1999.

[6] Alimohammad Saghiri and Alireza Bagheri, "An adaptive architecture for personalized search engine in Ubiquitous environment with peer to peer systems", International conference on Information and Multimedia Technology, 2009.

[7] Sikan Chen, Minglu Li, Feng He (2006) "GridPPI: A Lightweight Grid-Enabled Parallel Programming Framework", APSCC'06.

[8] S.Viswanadha Raju and A. Vinaya Babu, "Performance in the Design of Parallel Programming", Obcom 2004, Vellore Institute of Technology, Vellore, 2004.

[9] Mosleh M. Abu-Alhaj and M. Halaiyqah, "An Innovative platform to improve the performance of Exact-String-Matching Algorithms", International journal of computer and information security, Vol 7, No. 1, 2010.

[10] Joshy Joseph and Craig Fellenstein "Grid Computing" Pearson Education, IBM Press.

[11] Ahmar Abbas, "Grid Computing A practical Guide to Technology and Applications", FIREWALL MEDIA

[12] Michailids, P.K. Margaritis (2001), "String Matching problem on Cluster of Personal Computers: Experimental Results" Proc. 15th International Conference Systems for Automation of Engineering and Research.

[13] Park, J.H., And George, K.M., "Parallel string matching algorithms based on Dataflow", In proceedings of the 32nd Hawaii International on System Sciences, 1999.

[14] A.Vinaya Babu and S.Viswanadha Raju, "Optimal Parallel algorithm for String Matching Mesh Network Struture", International Journal of Applied Mathematical Sciences, 2007.

[15] K. Hwang and F. A. Briggs. Computer Architecture and Parallel Processing. McGraw-Hill, 1984.

[16] Thierry Lecroq, **"**Fast exact string matching algorithms," *Information Processing Letters,* Volume 102 , no. 6, Year of Publication: 2007, Pages 229-235.

[17] R.S. Boyer, J.S. Moore, "A fast string searching algorithm," *Communication of the ACM*, Vol. 20, No. 10, 1977, pp.762–772.

[18] R. N. Horspool, "Practical fast searching in strings," *Software—Practice and Experience,* Vol. 10, No. 3, 1980, 501–506.
[19] Smith, P.D., "Experiments with a very fast substring search algorithm," *Software-Practice and Experience*, Vol. 21, No. 10, pp.1065-1074.
[20] Sunday, D.M., "A very fast substring search algorithm," *Communications of the ACM*, Vol. 33, No. 8, 1990, pp. 132-142.

8. About Authors

K.M.M Rajashekharaiah obtained M.Tech (CSE) from VTU, pursuing PhD in computer Science discipline from Rayalaseema University, Kurnool, AP. Presently working as Assoc Prof in the department of CSE, BVRIT, Narsapur, Medak Dist, AP, India.

Prof Ch. MadhuBabu obtained M.Tech (CSE) from JNTUH, Pursuing PhD in Computer Science discipline from ANU, AP, India. Presently working as professor in the department of CSE, BVRIT, Narsapur, Medak Dsit, AP, India. His area of interests are programming languages, software engineering.

Dr. S. Viswanadha Raju obtained his Ph.D in Computer Science &Engineering from ANU. He obtained his M.Tech in CSE from JNTUniversity. He has a good academic background with a very sound and academic research experience. At present he is working as a professor in School of Information Technology in JNTUniversity, Hyderabad. He is guiding 10 research scholors for Ph.D and also conducted several conferences/workshops/seminars with sponsored agencies such as AICTE, DST, TCS, IEEE and CST. His research includes Information Retrieval, Databases, Image Retrieval, Data Mining and related areas. He published 25 research papers in reputed International Journals/Conferences proceedings in his research area. He is active member in different professional bodies with life membership like IETE, ISTE and CSI.