ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
December 2014,
Vol. 7, No. (3):
Pgs. 411-415

Implementing Divide and Conquer Technique for a Big-Data Traffic

ISHWAR BAIDARI, S P SAJJAN, VIJAYKUMAR G and AJITH H

CSDEPT, KU, Dharwad, India.

(Received: November 01, 2014; Accepted: December 20, 2014)

ABSTRACT

Real Time Analytic Processing (RTAP) in this modern world is inducing huge data traffic by everyone knowingly or unknowingly compared to few years back data traffic where only few companies was source of data traffic. This is really a challenge to the technology and needs a solution which can be implemented at the earliest and with an ease. This paper tries to discuss about the solution for handling the Big-Data traffic without downgrading the processing time. However, the analysis of big data can be troublesome because of its heterogeneous nature i.e. Big-data often involves the collection and storage of mixed data based on different patterns or rules. This has made the heterogeneous mixture property of data a very important issue. This paper focuses on applying "Divide and Conquer Technique" to handle the Big-data traffic using parallel processing in Network"

Key words: Big-Data, Divide and Conquer Technique, Parallel Computer and Networks.

INTRODUCTION

Recent business trends suggest that, big data analysis is becoming indispensable for automatic discovering of intelligence that is involved in the frequently-occurring patterns and hidden rules but it takes more time to process, manipulate and update the data in the database. So in computer science, Divide and Conquer (D & C) is an important algorithm design paradigm based on multibranched recursion. The basic idea of divide and conquer algorithm is to recursively break down a problem into two or more sub-problems of the same (or related) type, until the problem reaches the state where it can be solved directly. The solutions of these sub-problems are then combined to find the solution to the original problem. Divide and conquer is well known technique, but in this paper its main focus is on Big-Data traffic and steps to handle the Big-data using parallel processing in Network.

About Big-Data

(a) "*Big Data*" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it.

(b) Characteristics of Big Data:

1-Scale (Volume): Data volume is increasing exponentially

Terabyte	Petabyte	Exabyte	Zettabyte

The amount of data stored by the average company today

(c). 2-Complexity (Varity):

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
- Static data vs. streaming data
- A single application can be generating/ collecting many types of data

Speed (Velocity)

- Data is being generated fast and need to be processed faster
- Online Data Analytics
- Late decisions missing opportunities

Working with these 3V's (Volume, Varity and Velocity) is certainly a very difficult task to handle the Big-data as it is, so we have two options to handle the Big-data i.e. to Redesign existing Algorithms or redesign the hardware architecture with high-end processor and memory but the hardware change may not suffice solution to Big-Data complexity.

Importance of parallel computing:-

1. Very large application domains, and Physical limitations of VLSI circuits
2. Computers are becoming faster and faster, user demands for solving very large problems are growing at a still faster rate. Some examples include weather forecasting, simulation of protein folding, computational physics etc
3. <add one more point>

Proposed System

In this technique it is applicable for all divide and conquer algorithms, usually if N (define N) is very large then we will go for divide and conquer technique, but now a day's database is very large so we are applying same technique but assigning the problem to multi-processor using network, the data traffic is divided into 2 parts where each traffic is diverted to different processor in parallel then the solution of each client processor is returned back to server machine.

A vision for next-gen big data analytics

- Data captured & processed in a (semi) streaming/online fashion
- Real-time & history data combined and mined interactively and/or iteratively
  – Complex OLAP / BI in interactive fashion
  – Iterative, complex machine learning & graph analysis
- Predominantly memory-based computation

Here one more Interesting characteristic of Big-Data is that, not only Simple (SQL) – descriptive analytics is also Complex (non-SQL) – predictive analytics. So that while dividing the data we need to analyze in descriptive and predictive manner,

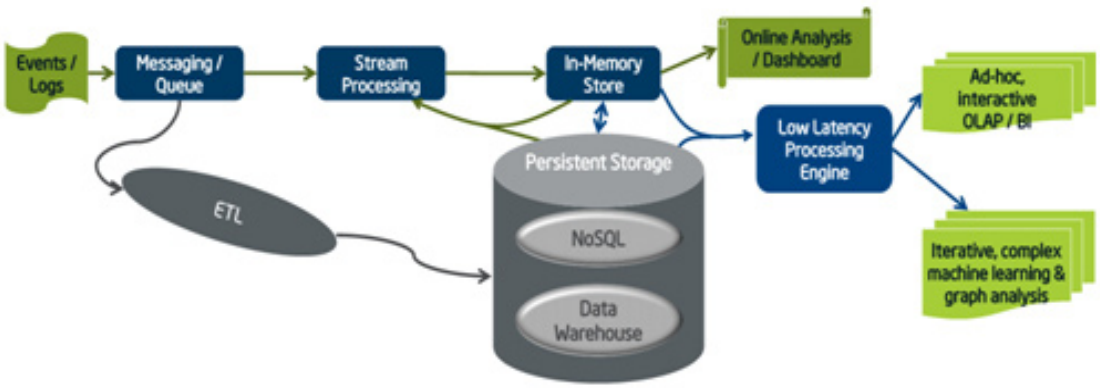

Fig .1: Vision of Leaner Processing Architecture of BigData,

defends on nature of the data assign work for different computer.

Flow Chart

If problem size (N) is very large then divide the problem into sub problems then assigning the task to different Computers using Computer Networks (Using different IP address), while dividing the data traffic it should be ensured that depending on type of data the traffic should be diverted to appropriate process which is capable to handle that particular type to data.

Algorithm DACPA(p)

Purpose: Solve the problem of a given instance (p) by dividing into various smaller instances such as $p_1,p_2,p_3,$——,p_n using divide and conquer and assign $p_1,p_2,p_3,$——,p_n task to assign each sub-problem to different computer.

Input: An instance of a Big-Data problem (p)

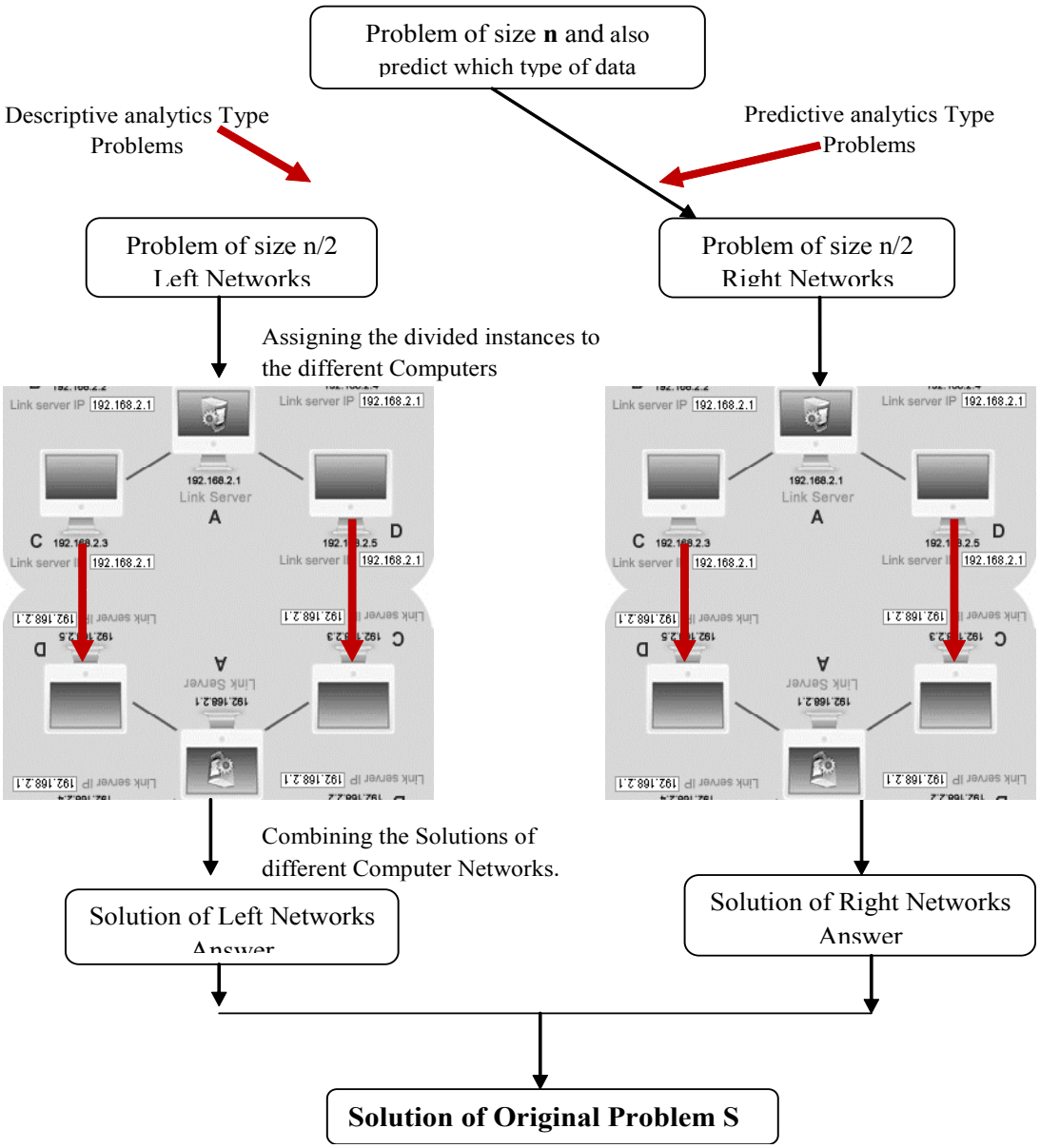

Fig. 2: Flow Chart of Divide and Conquer Technique in Parallel Computer

Output: Solution S to the input instances
If small(p)
Return g(p);
Else
If(n is very large and also predict which type of data (p))
If(data == Descriptive)
{
p is divided into p_1,p_2,p_3,——,p_n
Assign instance p_1 to computer 1
Assign instance p_2 to computer 2
Assign instance p_3 to computer 3
——
——
——
Assign instance p_n to computer n
Sß DACPA(p_1)+ DACPA(p_2)+ _ _ _ + DACPA(p_n)
Return
}
Else(Predictive Type)
{
p is divided into p_1,p_2,p_3,——,p_n
Assign instance p_1 to computer 1
Assign instance p_2 to computer 2
Assign instance p_3 to computer 3
——
——
——
Assign instance p_n to computer n
Sß DACPA(p_1)+ DACPA(p_2)+ _ _ _ + DACPA(p_n)
Return
}
End if

Analysis

Simple parallel algorithm analysis

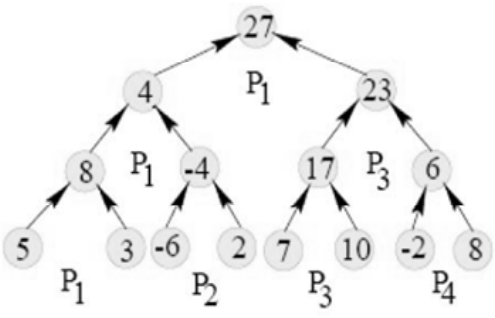

Fig. 3: Processing Tree

Adding n numbers in parallel
A simple parallel algorithm analysis

- Example for 8 numbers: We start with 4 processors and each of them adds 2 items in the first step.
- The number of items is halved at every subsequent step.

Hence log n steps are required for adding numbers.
The processor requirement is O(n) for single computer, this one same as n computers.
We have omitted many details from our description of the algorithm.

- How do we allocate tasks to processors?
- Where is the input stored?
- How do the processors access the input as well as intermediate results?

We do not ask these questions while designing sequential algorithms.
How do we analyze a parallel algorithm?
A parallel algorithm is analyzed mainly in terms of its time, processor and work complexities.

- Time complexity T(n): How many time steps are needed?
- Processor complexity P(n) : How many processors are used?
- Work complexity W(n):What is the total work done by all the processors? Hence,

For our example for single computer time complexity similarly extend this for n computers
T(n) = O(logn)
P(n) = O(n)
W(n) = O(nlog n)
How do we judge efficiency?

- We say A1 is more efficient than A2

if W1(n) = O(W2(n))
Regardless of their time complexities.
For example, W1(n) = O(n) and W2(n) = O(nlog n)

- Consider two parallel algorithms A1 and A2 for the same problem.

A1: W1(n) work in T1(n) time.
A2: W2(n) work in T2(n) time.
If W1(n) and W2(n) are asymptotically the same then A1 is more efficient than A2
if T1(n) = O(T2(n)).
For example, W1(n) = W2(n) = O(n), but T1(n) = O(logn), T2(n) = O(log2n)
Optimal parallel algorithms

- Consider a problem, and let T(n) be the worst-case time upper bound on a serial algorithm for an input of length n.
- Assume also that T(n) is the lower bound for solving the problem.
- Hence, we cannot have a better upper bound.
- Consider a parallel algorithm for the same problem that does W(n) work in T_{par}(n)time.

The parallel algorithm is work optimal, if W(n)= O(T(n))

It is work-time-optimal, if T_{par}(n) cannot be improved.

CONCLUSION

We are representing the Divide and Conquer Technique for BigData in parallel computer using Network, This is very simple to understand and implement. This approach executes parallel in network so it's saves the time, because we are implementing using divide and conquer technique in many computer so this algorithm takes half of the time comparing with existing algorithm.

REFERENCES

1. S P Sajjan, Jitendra Rohilla, Vijakumar Badiger, Girish Badiger, Shivaraj Angadi. "Implementing Divide and Conquer Technique in Parallel Computer Using Network." *international journal of electronics & communication technology* **5**.1 (2014): 1.
2. Real-Time Analytical Processing (RTAP) Using the Spark Stack Jason Dai Intel Software and Services Group
3. Real Time Analytic Processing (RTAP) with InfoSphereStreams by © 2009 IBM Corporation.
4. Dijkstra, Edsger W. "Recursive Programming". *Numerische Mathematik* **2** (1): 312–318 (1960). doi:10.1007/BF01386232 .
5. "An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins" (Knuth 1973:5).
6. Moschovakis, Yiannis N. "What is an algorithm?". In Engquist, B.; Schmid, W. *Mathematics Unlimited — 2001 and beyond*. Springer. pp. 919–936 (Part II). ISBN 9783540669135.
7. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms* (MIT Press, 2000)..
8. Jump up Brassard, G. and Bratley, P. Fundamental of Algorithmics, Prentice-Hall, 1996.
9. Parallel Algorithms – Introduction Advanced Algorithms & Data Structures Lecture Theme 11 by Prof. Dr. Th. Ottmann Summer Semester 2006.
10. CS170 { Spring 2007 { Lecture Notes - Feb 1}}
11. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran: Fundamental of computer algorithms, 2nd Edition, University press, 2007.
12. A.M. Padma reddy Desgin and analysis of algorithms 6th Edition, 2012.
13. Radu Rugina and Martin Rinard, "Recursion unrolling for divide and conquer programs," in *Languages and Compilers for Parallel Computing*, chapter 3, pp. 34–48. *Lecture Notes in Computer Science* vol. 2017 (Berlin: Springer, 2001).