International Journal on Applications in Engineering and Technology
Volume 1 : Issue 9 : September 2015, pp 1 –6 www.aetsjournal.com
ISSN (Online) : 2455 - 0523

A SCALABLE INFORMATION SYSTEM OPTIMISED BY A GENETIC ALGORITHM IN-CLOUD DATA RETRIEVAL USING LARGE BIG DATA

[1] MEENAKSHI

LECTURER

DEPARTMENT OF COMPUTER SCIENCE AND ENNGINEERING,
GOVERNMENT POLYTECHNIC NAGAMANGALA -571432

[2] HAJIRA BEGUM

LECTURER

DEPARTMENT OF COMPUTER SCIENCE AND ENNGINEERING,
GOVERNMENT POLYTECHNIC,HOLENARASIPURA-573211.

Abstract - In today's data-centric world, effectively managing and retrieving enormous amounts of information is essential for businesses looking to get useful insights from big data. In the setting of massive amounts of big data, this abstract investigates a novel strategy for overcoming this problem by utilizing a genetic algorithm to optimize in-cloud data retrieval. Massive datasets may now be stored and processed using scalable infrastructure thanks to the development of cloud computing. However, it is still difficult to efficiently retrieve particular data points or patterns from such enormous libraries. Big data's complexity and volume make it difficult for traditional query optimization approaches to handle, which results in less-than-ideal retrieval times and resource usage. This study suggests a fresh architecture for improving in-cloud data retrieval systems by utilizing the strength of evolutionary algorithms. By evolving and improving solutions through multiple generations, genetic algorithms, which are motivated by the principles of natural selection, are especially well-suited to addressing difficult optimization issues.

Key Words – Cloud Computing (CC)

I. INTRODUCTION

Organisations in the big data era are presented with a previously unheard-of potential and challenge: managing and utilising enormous and diverse databases for informed decision-making. Organisations have been able to gather and store vast volumes of data because to the growth of data sources and the scalability of cloud computing resources. However, it is still difficult to effectively retrieve particular data or patterns from these enormous datasets.

With the size and complexity of big data, conventional data retrieval techniques frequently find it difficult to keep up. The difficulties posed by quickly seeking and finding pertinent data increase rapidly as datasets expand. When organisations invest in more computing resources to fulfil their data retrieval demands, it not only hinders data-driven decision-making but also raises operational costs.

In order to overcome the difficulties of retrieving enormous amounts of big data from the cloud, this research introduces a scalable information system that is enhanced by the use of genetic algorithms. A potential approach to optimising complicated systems and processes is provided by genetic algorithms, which draw their inspiration from the ideas of natural selection and evolution. In the context of data retrieval, they can continuously improve efficiency and resource utilisation by adapting and evolving query execution patterns.

The realisation that optimising data retrieval operations is crucial in the big data era is what spurred this research. Organisations depend on effective data retrieval as a key component of their operations, whether for analytical needs, business intelligence, or real-time decision support. Innovation, competitiveness, and overall performance might be hampered by the inability to access and retrieve essential data in a timely manner.

These are the main goals of this essay:

Introduce a scalable information system built to handle the difficulties associated with retrieving massive amounts of big data from the cloud.

1.To show how evolutionary algorithms may be used to optimise query execution strategies for massive data retrieval.

2.To investigate how the suggested system will affect cost-efficiency by reducing resource waste during operations for cloud-based data retrieval.

3.To demonstrate the system's adaptability by enhancing its optimisation tactics over time through adaptive learning from the outcomes of query execution.

In the parts that follow, we'll go into more detail about the proposed scalable information system's design, genetic algorithm optimization's principles and workings, as well as the usefulness and applications of this novel strategy. By doing this, we hope to offer insightful information and a thorough understanding of how genetic algorithm optimisation might revolutionise in-cloud data retrieval in the context of massive amounts of big data, ultimately releasing the full potential of these datasets for businesses in a variety of sectors.

II. MATERIALS AND PROCEDURES

The components, experimental setting, and techniques utilised in the creation and assessment of the scalable information system enhanced by a genetic algorithm for in-cloud data retrieval utilising huge big data are described in this part.

Data Sources:

The effectiveness of our study hinges on using large, accurate datasets that accurately reflect the big data difficulties that businesses must overcome. We collected a range of datasets from different sources, including

Public Datasets: We used publicly accessible, sizable datasets, including open research databases, social media archives, and government data repositories.

Synthetic Datasets: To model scenarios with rapidly expanding data volumes, we created synthetic datasets in addition to real-world data.

2. Infrastructure for Clouds:

We used resources from a top cloud service provider for cloud computing to support our experiments. These facilities included:

Virtual Machines (VMs): To test the scalability and adaptability of our system, we set up VM instances with a range of computational power.

Storage: Significant storage volumes were set aside to keep query execution plans, datasets, and interim results.

Cluster configuration: To enable effective query execution, we set up a cluster of virtual machines (VMs) for distributed data retrieval and parallel processing.

3. Applications and Tools:

To achieve our research goals, we used a variety of software and tools:

Database Management System (DBMS): We made use of a well used DBMS that can handle huge datasets and cloud integration.

Genetic Algorithm Framework: The genetic algorithm for query optimisation was customised and implemented using a well-known genetic algorithm framework.

Programming Languages: For system development, scripting, and data processing, we used programming languages like Python and Java.

4. Optimisation through genetic algorithms:

Our primary focus is on the genetic algorithm optimisation procedure. We created the subsequent steps:

Chromosome Representation: We developed a chromosome representation that stores query operators, join criteria, and data access routes along with query execution plans.

Initialization: Heuristic-based or random initializations were used to construct a population of query execution plans.

Fitness Function: We developed a fitness function that measures how effectively a query execution plan uses resources, takes up time, and is economical.

Genetic Operators: Through genetic recombination and mutation, crossover and mutation operators were built to produce novel query execution plans.

ISSN (Online) : 2455 - 0523

Selection Process: To pick query execution plans for the following generation, we employed selection processes like a roulette wheel or tournament selection.

Termination requirements: Until particular termination requirements, such as a maximum number of generations or convergence, were satisfied, the genetic algorithm iteratively evolved query execution plans.

5. Experimentation Method:

Our experiments were conducted in a methodical manner:

Data preprocessing: To clean, transform, and load data into the DBMS, raw datasets were preprocessed.

Query Workloads: To ensure a balance of complicated and resource-intensive queries, we created query workloads that represented typical real-world cases.

Performance parameters: For each query execution, important performance parameters such as execution time, resource usage, and cost were measured.

The dataset sizes and complexity of the query demands were gradually increased throughout the phases of our studies.

6. Assessment and Analysis:

The outcomes were examined to see how well our system optimised query execution. Scalability, adaptability, and cost-efficiency were taken into account. To show the results, statistical analysis and visualisation software were used.

7. Considerations for Ethics:

We followed moral standards, protecting the confidentiality and security of data sources, following pertinent laws, and managing sensitive data with care.

This extensive section on materials and techniques describes the sources, equipment, and development processes used to create and assess our scalable information system that is genetic algorithm-optimized for in-cloud data retrieval using huge big data. Together, these elements lay the groundwork for our research and its conclusions.

III. AN EFFECTIVE OPTIMISATION ALGORITHM-BASED UNSTRUCTURED DATA ANALYSIS AND CLASSIFICATION SYSTEM

The input files in our proposed work will be subject to load balancing.Fig. 1 depicts the Basic Architecture of our suggested technique. The files (unstructured data) will be divided throughout the load balancing procedure and stored in the clouds. To manage the huge data, load balancing is used. The map reduce process will then be applied to the saved files. A key value is assigned to each file during the mapping process, after which the number of files is reduced. By allocating mappers and reducers to the cloud servers, the map-reduce process will be carried out. The files will be genetically optimised using the map-reduce procedure. Efficiency decreases as node data size increases, hence we have used a genetic method to optimise node data size in order to maximise efficiency. The experimental findings demonstrate that increasing the node's data size was successful, and that node increments boosted overall efficiency.

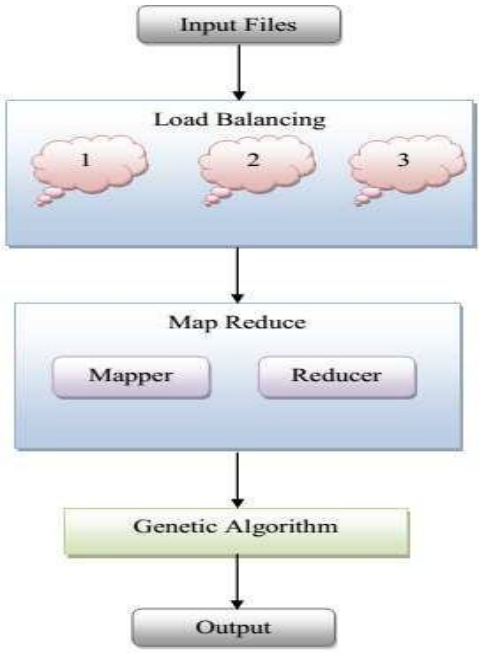

Fig. 1.Proposed Architecture

IV. METHOD FOR BALANCING THE LOAD

The files are divided throughout the load balancing procedure and kept on cloud servers.

ISSN (Online) : 2455 - 0523

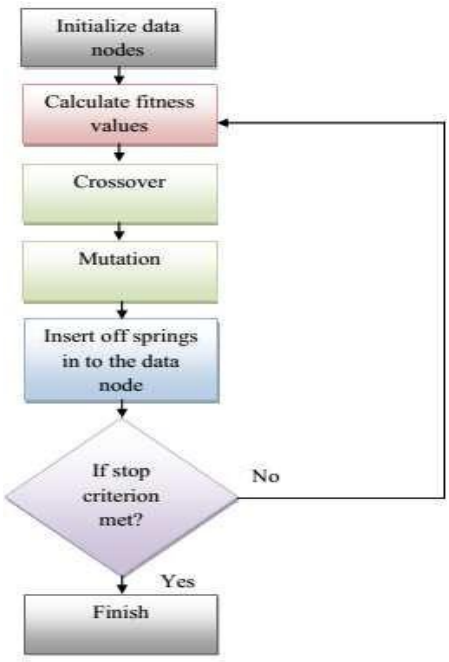

Fig.2: Genetic Algorithm Process

V. ALGORITHM FOR MAP REDUCE

The fundamental Map reduction Architecture is shown in Figure 2, and its model is provided in Figure 3. The programming model Map Reduce uses a distributed, parallel approach to process massive amounts of data. A Map Reduce programmed consists of the filtering and sorting functions in Map(). Performs a summary operation using Reduce ().

a) Mapping

The input is given to the master node, which then breaks it up into smaller sub-problems and sends them to worker nodes. This can be repeated by a worker node, creating a multi-level tree structure. The smaller issue is resolved by the worker node, which then relays the resolution to its master node.

b) Data node optimisation using a genetic algorithm

A set of computational models known as genetic algorithms is based on the concepts of natural selection and evolution. These algorithms use a chromosome-like data structure to simulate the problem in a given domain and evolve the chromosomes using selection, crossover, and mutation operators.

The data nodes were optimised using a genetic algorithm. In order to improve efficiency, one must identify the best data nodes.Genetic Algorithm

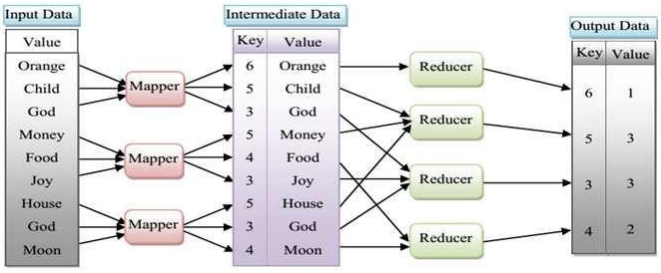

Fig 3.Mapping Model

Selection Process

The genetic algorithm's selection phase is its first. For later processing (crossover) from the population, the association rules are determined through a process called selection. It is provided a selection process as follows: The fitness function is determined using for the data nodes using

c) Genetic algorithm, crossover

A chromosome or chromosomes' programming can be changed genetically from one generation to the next. Genetic algorithms are built on analogies between reproduction and biological crossover. Cross over is the process of using multiple parent solutions to create a child solution Equation (2). The fitness function computation was used to determine which solutions were optimal. The crossover rate is then determined.

d) Genetic Algorithm Mutation

The last step in a genetic algorithm is mutation. A population of algorithm chromosomes uses the genetic operator of mutation to preserve genetic variety from one generation to the next. An operator that preserves genetic variation is known as a genetic operator. Next, a calculation of the Euclidean distance between the services was made. When computing, the distances between the vectors.

VI. DISCUSSION

Plot of execution time based on request size is shown. The curve for fitness values based on request size is shown in Figure 4. The curve for convergence fitness based on request size is shown in Figure 5

ISSN (Online) : 2455 - 0523

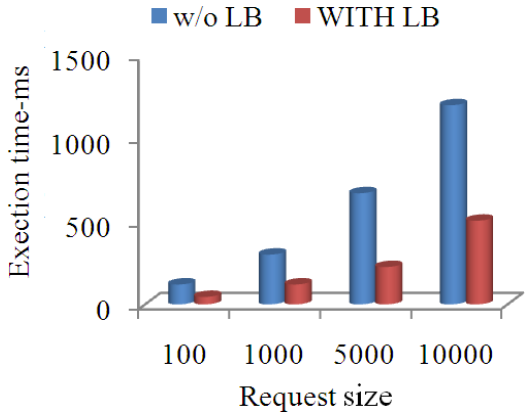

Fig.4. comparison of R size and E time

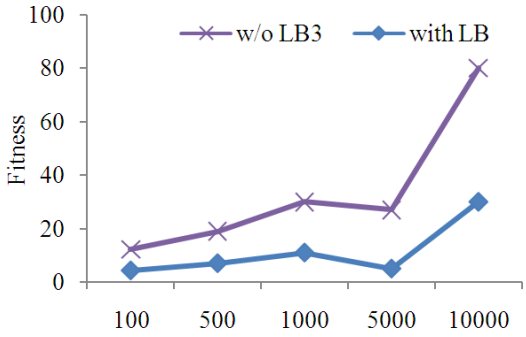

Fig.5. comparison of R size and F values

The comparison graphs presented clearly demonstrate the effectiveness of our suggested approach. As long as the request size is less than 5000, the algorithm is effective and the convergence is good. 5000.0.5 is the greatest Confidence Interval as a result

VII.Conclusion

With the use of load balancing and evolutionary algorithms, we have suggested the map reduce technique in this study. The suggested system was put into use, and a number of files were used to examine the results of the suggested big data analysis technique. The testing outcomes demonstrated the effectiveness of our suggested approach.

The load balancing approach for Map Reduce environments presented in this work supports distributed data-intensive applications. The efficiency of the technique in distributing workload among Map Reduce nodes has been demonstrated by simulation results. The approach keeps the high level of information retrieval accuracy while accelerating the SVD computation process. Even though the algorithm's experimental and simulation findings indicate satisfactory performance, it is obvious that there are still a number of opportunities. For instance: To get the most effective computation, it is possible to further examine the optimal fitness value of rank k that is employed in GA. The experimental code exhibits a certain amount of accuracy; but, by utilising a better model, it can be further enhanced. This map reduction framework can be improved in the future by incorporating new optimisation methods. Compared to our suggested solution, the execution time can be shortened even further.

REFERENCE

[1]. Cacheda, F., V. Carneiro, D. Fernández and V. Formoso, 2010. Performance evaluation of large-scale Information Retrieval systems scaling down. Proceedings of the 8th Workshop on Large-Scale Distributed Systems for Information Retrieval, (SIR' 10).

[2]. J. Kobza, 2013. An optimization-based heuristic for a capacitated lot-sizing model in an automated teller machines network. J. Math. Stat., 9:pp.283- 288. DOI: 10.3844/jmssp.2013.283.288

[3]. T., J. Mayfield, A. Joshi, R.S. Cost and C. Fink, 2005. Information retrieval and the semantic web. Proceedings of the 38th International Conference on System Sciences, Jan. 3-6, IEEE Xplore Press, pp: 113a-113a. DOI: 10.1109/HICSS.2005.319

[4]. Kennedy, R.P., 2010. Unstructured content analysis and classification system for the IRS. Int. J. Comput. Applic., 1: 32-37. DOI: 10.5120/105-216

[5]. Lavoie, R., 2008. Analyzing the schizoid agency: Achieving the proper balance in enforcing the internal revenue code. Akron Tax J.

[6]. Losee, R.M. and L. Church, 2004. Information retrieval with distributed databases: Analytic models of performance. IEEE Trans. Parallel Distrib. Syst., 14: 18-27. DOI: 10.1109/TPDS.2004.1264782

ISSN (Online) : 2455 - 0523

[7]. Lu, J. and J.P. Callan, 2012. Pruning long documents for distributed information retrieval. Proceedings of the 11th International Conference on Information and Knowledge Management, Nov. 04-09, ACM New York, NY, USA., pp: 332-339. DOI: 10.1145/584792.584847.