May-June 2020
ISSN: 0193-4120 Page No. 9618 – 9621

Data Transfer Scheduling for Maximizing Throughput of Big Data computing in multiple Applications

Dr G.Vijay Kumar, B.Ankitha, G.Ramya
Department of Electronics and Computer science
Koneru Lakshmaiah Education Foundation
Green Fields, Vaddeswaram, Guntur, Andhra Pradesh, India.

Article Info
Volume 83
Page Number: 9618 – 9621
Publication Issue:
May - June 2020

Article History
Article Received*: 19 November 2019*
Revised*: 27 January 2020*
Accepted*: 24 February 2020*
Publication*: 18 May 2020*

Abstract:
Now a days we can see that the big data computing applications have been to cloud platforms .The big data computing applications need concurrent data transfer among different computing nodes parallel processing among the nodes. We have to find best transfer scheduling technique that will lend to the least data retrieval time and the maximum throughput. The methods that already exist cannot achieve the maximum throughput due to link bandwidths ignorance and the assortment in replication data and paths. In this paper we develop a maximum throughput data transfer scheduling technique to reduce the data retrieval time in big data computing with data replication diversity.

Keywords*: Big data, cloud computing, Throughput, data retrieval.*

INTRODUCTION

Applications which are running in cloud environment will adopt Map reduce frame work for data computing in parallel nodes. Data can not be placed in one node as they are processed in the same node due to insufficient computing capacity and load balancing. Data blocks are maintained in triple format along with original data block in HDFS for robustness and redundancy[2][7]. Some times multiple paths are available for transfer of data from each node because of path redundancy in data center Networks[3]. Selection of the best node along with path is very much important for retrieval of non local data. So this is the problem of data retrieval. Selection of data retrieval leading is extremely valuable task because time completion will be more for long data retrieval task. The method proposed in [2] can not take less time for data retrieval when non local data is required and request is sent by the closest data node. Tasks will execute concurrently to retrieve data may result to serious congestions on the same links leads to long retrieval time for data because it pay no attention to bandwidth links and also overlaps of paths and selected nodes. Flow scheduling algorithms were proposed [6] to avoid path collisions. Although, they exploited path diversity but, not on data replication diversity. Maximum through data transfer scheduling is proposed [1] to reduce data retrieval time of various applications which are consisting concurrent tasks. Their simulated results are demonstrated for data retrieval regardless of path diversity. We suggest a max-throughput data transfer scheduling using diversities in the replica direction to reduce the data retrieval time. The problem is formulated in mixed integer programming & suggested approximation algorithm that can be used for multiple applications. The rest of this paper is organized as follows. Section 3 presents the motivation and overview the of the problem. Section 4 presents problem formulation for multiple applications. Section 5 presents an approximation of technique and the analyses on its approximation ratio. Section 6

May-June 2020
ISSN: 0193-4120 Page No. 9618 – 9621

presents the performance evaluation Section 7 concludes the paper.

Problem Overview and Motivation:

Although data is distributed among computer nodes in the cloud, it is not possible to get all the data locally, so some data from nodes can need to be retrieved from distant nodes. You can collect a data you need from one of the nodes where the replica is stored. If a node is selected for data recovery, a route for data transfer from it to the requesting node must be specified.

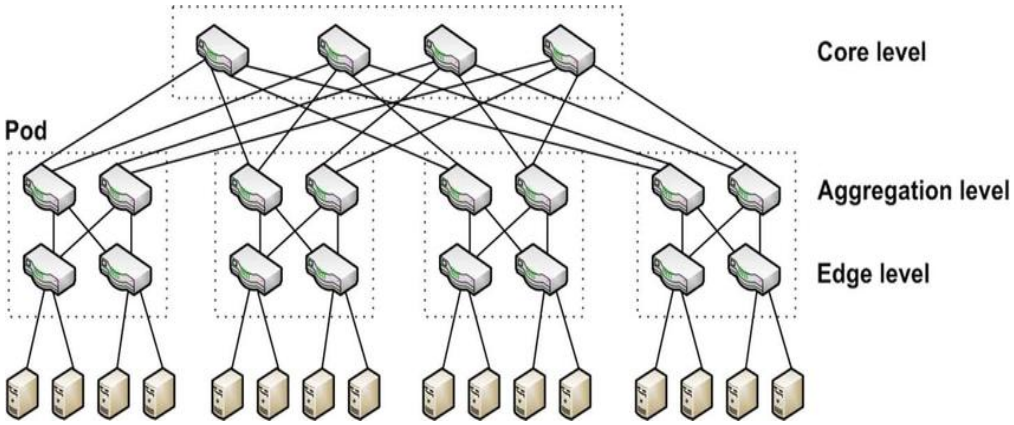


Fig 1: Topology of data center network with 4 port switches

A naive approach is to pick nodes and paths randomly, but it can lead to heavy congestion on certain connections, resulting in long data recovery time as it ignores bandwidths and selected paths and nodes overlapping. The naive approach also falls short when several applications are running in the cloud where different applications that have different criteria, i.e. the upper limit of data retrieval time. Because all requirements can not be met, we're minimizing the applications penalties. Therefore our job is to pick nodes and paths in order to minimise the penalties for applications for each data needed.

Problem Formulation for multiple Applications:

In this section, data recovery problem is formulated. In the first section we explained about single application and next deal with multiple applications. In a graph nodes are denoted with V and edges are denoted with E. Graph is represented as G = (V,E), where every edge is a bi-directional link and this is a classical network configuration in data center network. A data center network is represented as a combination of computing nodes and switches as demonstrated in the graph [1].Every node has a relation to path. The nodes are classified in to two types one is computing nodes V_C, second one is switch nodes V_X. That is V can be represented as V_X U V_C. Assume the application processes a set of data objects and all data objects are of same size S for simplicity and it is represented as D={ od_1,od_2,od_3,.....od_n} , which are placed in computing nodes. Data Objects are replicated in multiple nodes for data redundancy. The set of nodes which have the data object od_k is represented as V_{ck} is a subset of V_c[8]. A_{jk} represents whether Vj requires data object od_k or not to run the assigned task for this. Suppose Vj needs od_k then we have to select a node vi ∈ Vck from this od_k can be retrieved. The group of possible flows of transferring data od_k to node V_j is represented as Fjk where possible flow specifies the transfer of data from od_k to v_j. Group of all flows is represented as

$$F = U \quad \forall_{(j,k) \text{ where } Ajk= power(1,Fjk)}$$

In cloud platform, resources are shared by multiple applications to run simultaneously to use available resources more efficiently. As requesting from every application it is essential to select nodes and paths, it also treats all applications as a single one to solve the problem using the above model. This is the

May-June 2020
ISSN: 0193-4120 Page No. 9618 – 9621

same to diminish the maximum time for data retrieval among all applications. Nevertheless, all applications may not have same requirements on their data retrieval time, the naive approach ignores the difference of requirements. Thus, instead of minimizing data retrieval time, better reduce penalty. Assume group of applications are given U and application u ∈ U has an upper bound $t`_u$ on its data retrieval time t_u, penalty c_u as referred in [1]. When a source node is determined for each request then the data retrieval problems for both cases are NP-hard, a set of commodities are formed. Here we call a triple consisting of a source, a destination, and a demand (i.e., the amount of data to be routed) a commodity. For the case of a single application, given the set of commodities and a network, then our problem is to compute the maximum value 1=t for which there is a feasible multi commodity flow in the network with all demands multiplied by 1=t, which is a concurrent flow problem [5]. Since we have the additional restriction that each commodity must be routed on one path, the problem is an unsplittable concurrent flow problem, which is NPhard [5]. It is the same for the case of multiple applications.

Max-throughput Approximation Algorithm:

An approximation algorithm is prepared for solving the data retrieval problem.

Given a data retrieval problem and its MIP formulation, our algorithm has three major steps:

Step 1: solve its LP relaxation

Step 2: construct an integral solution from the relaxation solution using a rounding procedure

Step 3: analytically compute the data sending rate of each flow for scheduling.

we can obtain that the approximation algorithm has the same approximation ratio of RL for the case of multiple applications, but now the approximation ratio affects penalty plus 1, i.e., c þ 1, rather than c, as follows,

$$C^A + 1 \leq RL(opt(MIP)+1)$$

where C^A is the objective value of an approximation solution, and opt(MIP) is the optimal value for an MIP instance, means that the worst time ratio obtained by the approximation algorithm is at most RL times greater than the optimal value. As analyzed previously, approximation results are upper bounded by approximation ratio RL and the optimal value of an MIP instance. The lower the upper bound is, the better approximation results may be. Thus, we may obtain better approximation results by reducing the upper bound

Performance Evaluation

Throughout this section, we analyze the performance extensively and demonstrate that our algorithm can obtain near-optimal solutions with the availability of additional data replicas and abundant paths. It show that even an additional replica in many cases can greatly improve the performance.

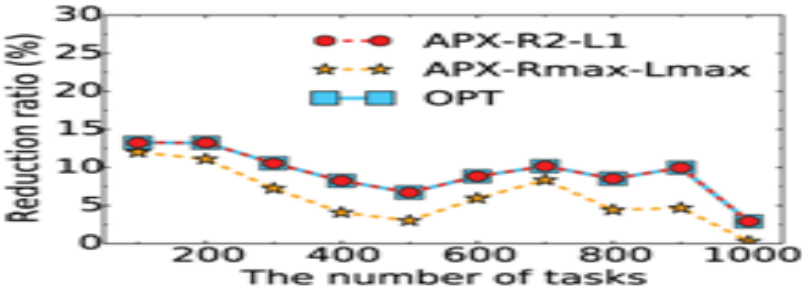
Simulation Setup

Network topologies and parameters, and then discuss data retrieval setups. Our simulation test bed has three types of Data Center topologies: 1) a fat-tree topology built from 8-port switches 2) a three-tier topology as shown in Fig. 2 in which every four neighboring hosts are connected to a ToR switch, and every four neighboring ToR switches are connected to an aggregation switch, and all eight aggregation switches are connected to each core switch 3) a VL2 topology [7] with 8-port aggregation and 4-port core switches, as shown in Fig. 5. Both the fat-tree topology and the three-tier topology have 128 computing nodes, while the VL2 topology has 160 instead[2].

we evaluate the performance of multiple applications. Consider OPT and RND for comparison. The worst penalty among all applications is used for evaluation. It is directly obtained from objective value for OPT, while for

May-June 2020
ISSN: 0193-4120 Page No. 9618 – 9621

APX and RND it can be computed as in[1]. We also start with running simulations to find optimal R and L where APX performs best. Nine pairs of R and L are tried, where R is chosen from {1, 2, 3} and L is chosen from {1, 4, 16}. We simulate a scenario of two applications each having 500 tasks and 100 data objects with many-to-one access relationship under full bandwidth. The performance of APX in nine settings where the results of RND and OPT are represented by red line and black line respectively. It is observed that the APX with R ¼ 2 and L ¼ 1 performs closest to OPT. Thus we choose to evaluate APX with this setting (i.e., APX-R2-L1), besides the one without the preprocessing referred in [1].



(a)

The results are shown in above and It is observed that APXR2-L1 performs almost as good as OPT also better than RND, and a bit better than APX-Rmax-Lmax. In the topology with full bandwidth, the reduction ratio is around 3% 13% for one-to-one setting; while for many-to-one setting it increases significantly as more tasks access each data at the same time.

Conclusion

In this paper, we discuss the question of DCN data recovery, which is to pick data replicas and paths for simultaneous data transfer together in order to minimize data recovery time. The proposed approximation algorithm is used to solve the problem with an RL approximation ratio, where R is the factor for data replication and L is the largest number of candidates. It is helpful to solve the data recovery problem in multiple applications, retaining equity among them. The simulations show that our algorithm can achieve near-optimal efficiency, with the best R and L.

References

1. Ruitao Xie and Xiaohua Jia "Data Transfer Scheduling for Maximizing Throughput of Big-Data Computing in Cloud Systems" IEEE Transactions on Cloud computing, volume 6 No 1 January-2018, pp. 27-38.
2. M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in Proc. SIGCOMM, 2011, pp. 98–109.
3. J. L. Gross, J. Yellen, and P. Zhang, Handbook of Graph Theory, 2nd ed. London, U.K.: Chapman & Hall, 2013.
4. M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation, 2010, pp. 19–19.
5. J. L. Gross, J. Yellen, and P. Zhang, Handbook of Graph Theory, 2nd ed. London, U.K.: Chapman & Hall, 2013
6. Vijay Kumar G., Krishna Chaitanya T., Pratap M. "Mining popular patterns from multidimensional database" in Indian Journal of Science and Technology vol 9 Issue 4.
7. A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in Proc. SIGCOMM, 2009, pp. 51–62.
8. Praneeth, J.N, M.Sreedevi "Detecting and analyzing the Maliciouswindows events using Winlogbeat and ELK stack" 2019, vol 7 Issue 6, pp 156-160.