CaRA : Congestion-aware Routing Algorithm for Data Center Network

Xingyan Zhang[12, a], Runze Wan[3,b], Jinrong He[4,c]

[1] Huazhong University of Science & Technology, Wuhan, China

[2] College of arts & sciences of jianghan university, Wuhan, China

[3] Hubei University of Education, Wuhan, China

[4] College of information engineering, Northwest A&F University, Shanxi, China

[a]zhangxingyan@hust.edu.cn, [b] runzewan@126.com, [c] hejinrong@163.com

Keywords: data center networks, recursive routing, flow collision

Abstract. The data center networks (DCN) leverage redundant physical links to overcome the bottleneck of traditional tree topology and achieve high bisection bandwidth and goodput for cloud computing platform, so how to use these redundancy links efficiently becomes the key factor for routing designer. This paper analyzes the large flow collision in modular data center network and propose simple experiments to validate the performance decreasing in the aggregated throughput. And then, a recursive congestion-aware routing algorithm (called CaRA) is proposed for decreasing flow collision. We simulate the CaRA with NS-2 simulator, and the results show that CaRA can improve one-to-one packet forward throughput and all-to-all throughput most time. The peak throughput values in these two situations are 50% and 20% respectively.

I. Introduction

In recent years, data center has scaled from tens of thousands servers to hundreds of thousands servers[1], which supports various kinds of application. Modular data center leverages redundant paths to improve aggregate throughput, e. g. the BCube [2] employes BSR algorithm to split the file into small parts and simultaneously delivers all the parts to accelerate the file distribution. So the performance of BSR depends on how many backup parallel paths have been reserved and whether the path has enough bandwidth for new coming flow. For example, when several paths are simultaneously used to forward data from source node to destination node for file transmission, congestion of one path may result in single path completion time increasing and the aggregate transmission time increases.

We analyze performance decreasing in the aggregated throughput, and propose a congestion-aware routing algorithm for DCN (CaRA) to improve aggregated throughput of BCube. We simulate the CaRA in ns-2 simulator, in which the results show that CaRA can respond congestion and get better performance on throughput and bandwidth in BCube network.

The organization of this paper is as following. Section II describes the problem of current routing algorithm in BCube architecture. Section III continues with the description of methodology, algorithm and details design. Section IV describes the evaluation and analysis for CaRA algorithm. The last section concludes and gives some future directions.

II. Motivation

BCube is a good design for improving network throughput and bandwidth. All switches are linked with the servers in BCube architecture, and there is no direct link between switches. The BSR is based on recursive routing which is able to fully utilize the high capacity of BCube and automatically load-balance the traffic. BCube essentially increases bandwidth between source node and destination node with multipath.

When a flow comes, source node probes the k+1 parallel path set and chooses best one for the flow, in which k is the number of BCube level. But in data-center traffic, more than 90% of bytes are in

flows between 100MB and 1 GB [3]. That means many hosts keep more than 10 large flows, which take most of the time. On the other hand, the number of parallel path of one host is less 5 [2] from the current data center scale perspective. As a result, there must be flow collision between source node and destination node in default BCube intuitively. We simulate a 16-nodes BCube network to estimate throughput with the large flow collision. As shown in figure 1, the BCube network consists of 8 switch nodes and 16 host nodes, in which the level k is 1. So every node keeps 2 parallel paths according to the BSR protocol.

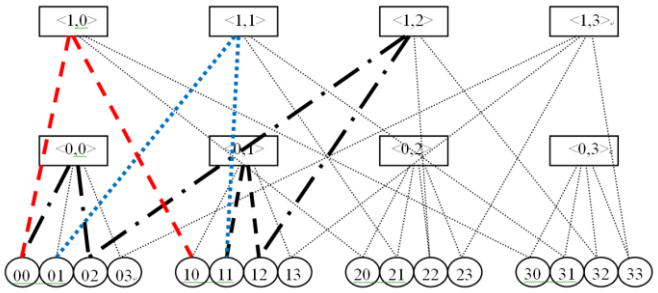

Figure 1: BSR keeps many parallel paths for data transmission in Bcube

We assume that there is a large flow from node 00 to node 10 and another flow is through node 01 to node 11 at the same time. If a new large flow which is from node 00 to 11 is coming, BSR routing algorithm will choose a better path from 00-01-11 or 00-10-11. So the problem is that the new large flows must conflict with the existed “busy” path, whichever path is selected. As the result, the flow conflict decreases approximate half aggregated throughput based on simple simulation. But if new flow is forwarded as 00-02-12-11, intuitionally that large long-lived flow confliction may disappear. And then, we implement these situations for validating above assumptions.

For simplicity, we use default dynamic protocol instead of BSR. We run 3 flows on link 00-10 and 3 flows on links 01-11 in 280s, both using long-lived flows of NewReno TCP[4]. We add 3 new flows on path 3: 00-10-11 at the point of 80 seconds. The path collision may happen on the path 1: 00-10 or path 2: 01-11. We evaluate the throughput and packet loss rate of these paths respectively, while these metrics are closely related with the congestion [5]. And we should find out the key metric which may mainly effect the network performance.

Observation from the trace of simulator, we find that the new flows are through the path 00-10-11 and the path 2 is always keeping the same performance as the path 1 in the first minute. So we ignore the path 2 and just calculate the throughput and packet loss rate of path 1 and path 3 respectively.

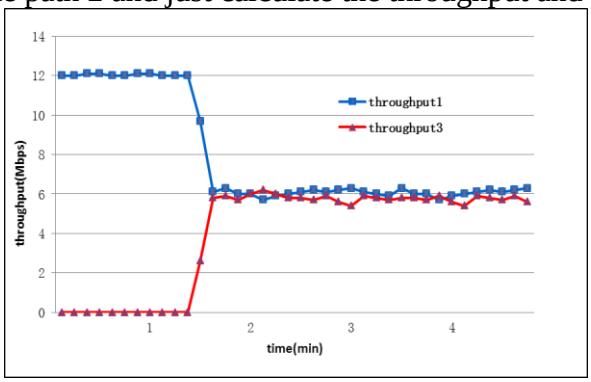

Fig. 2: throughput of path 1 and path 3

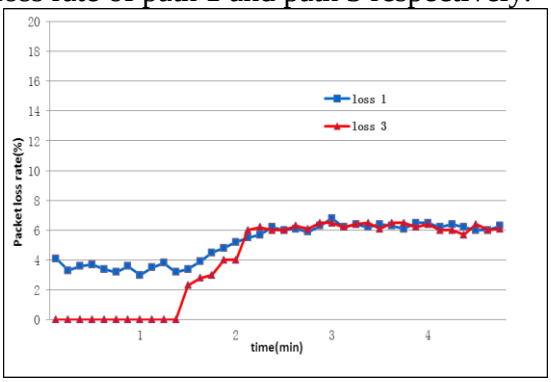

Fig. 3: packet loss rate of path 1 and path 3

From the figure 2, we find that the throughput 1 is constantly keeping approximately 12 Mbps in the first 80 seconds. When the three new flows come, the throughput of path 1 degrades to the path 3 level. The throughput of two paths is always keeping the approximately 6 Mbps. This reason is that the large flows conflict on path 00-10 and the aggregated throughput is divided two half as the same flows on each path. From the figure 3, we find the packet loss rate of path 1 increases while the new

flow comes. And the packet loss rate of another path exhibits the same phenomenon, which the value is increasing with the more flows coming and keeping constant level without flows changing.

From the above experiments, we conclude that the key metric of path collision is throughput, and the packet loss rate is at least not intuitionistic affection. We will give the quantificational analysis of the BSR algorithm in the following part. Existing BSR algorithm tries its best to keep k+1 parallel paths. But the results of above experiments show that these parallel paths may suffer from large flow collision. This paper proposes a new algorithm, named congestion-aware routing algorithm, which adds congestion status information to keep k+1 parallel paths to increase the aggregated throughput.

III. Methodology and Algorithm

The main idea of CaRA is choosing the lightly traffic path for new coming flow. If the probing path based on the default BSR path set is successful, the flow will start to be forwarded immediately, or else we put the neighborhood node into the path sets and search possible available path from the neighborhood node recursively.

Figure 4 shows that the CaRA algorithm finds the best transmission path based on the existing k+1 parallel paths set. We need default BSR keep k+1 parallel paths, and we probe each path from source node A to source node B one by one. If there is no any available paths back, we need to probe new path for node B from one of the new available neighbor N of node A. If available path returns, then we can append the available neighbor of source nodes and this is the path we want. Or else, we need the other neighbor nodes instead of the node N, and probe path again till new available path returns.

```
CaRA description:
FindPathforFlow(A, B):
   PathforFlow(A, B) = {};
   pathSet = BuildPathSet(A,B);
            //building k+1 parallel path
   for ( i=0; i<k+1 ; i++){
      if (probe pathSet(i) == true)
         PathforFlow() = pathSet(i);
         Break;
      else
         N = a neighbor of node A;
         PathforFlow() = { N };
         PathforFlow() += FindPathforFlow(N, B);
   }
```

Fig. 4: The algorithm to find best path based on BSR algorithm.

The bandwidth is the effective metric for judging whether the path is available. For simplicity, we assume that every large flow needs the same bandwidth *bw*, so we calculate the bandwidth as *bw* multiply by N, a number of flow at this one path. Then, we need to definite the bandwidth threshold *bw_threshold* as critical metric for judging whether the path is suitable a coming flow. If N**bw* is less than *bw_threshold*, that means this path has enough available bandwidth for a new flow, and probe operation is done. Or lese, we need keep looking for a new available path for a coming flow.

Although the process of finding available path of CaRA is a recursive, the completion time is not too long. If we probe from the neighborhood one-by-one, the time complexity of CaRA is $O(n^k)$. It is a very large end-to-end delay for application. So we choose multicast manner to probing paths for efficiently researching. The maximum probing rounds do not exceed *n*, which is the number of the switch ports.

IV. Evaluation and Analysis

We simulate CaRA in NS-2, which is a discrete event simulator targeted at networking research. We construct BCube networks and implement CaRA source routing instead of BSR routing. We evaluate the one-to-one throughput and the all-to-all packet forwarding throughput of the two algorithms. In the first experiment, we show the one-to-one throughput improvement of CaRA. We

choose node 001 and node 123, so the parallel path set is 001-101-121-123, 001-021-023-123 and 001-003-103-123. The figure 5 show CaRA can achieve more than 22Mb/s throughput and the highest one-to-one throughput improves by 50%. It implies that the one-to-one throughput of CaRA is actually higher than BSR routing.

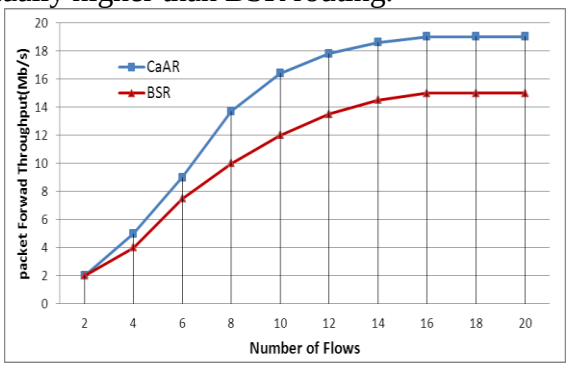

Fig. 5: One-to-one packet forwarding throughput

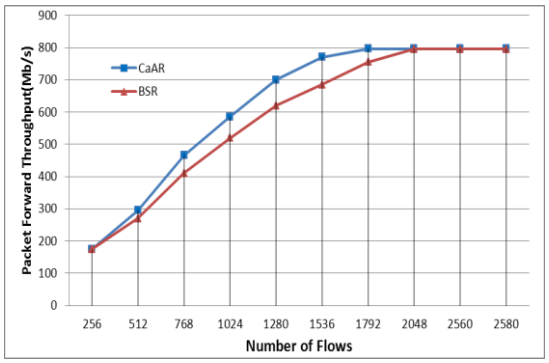

Fig. 6: All-to-all packet forwarding throughput

Figure 6 show us the all-to-all packet forwarding of CaRA and BSR respectively. While we increase the number of flows, the packet forwarding throughputs both increase while they do not exceed aggregated throughput. The reason is that all the flows saturate all the paths. Based on observation from the figure 6, the throughput of CaRA is better than that of BSR and the peak value is even 20% more than the BSR routing. But at last, the both have the same throughputs, which means all the paths are in congestion and cannot carry more flows.

Conclusion

This paper analyzes the problem of large flow collision in the BCube networks and proposes CaRA to detect congestion and finding light-weight path in modeler DCN. We evaluate the CaRA algorithm in NS-2 simulator. The results show that CaRA has good performed in one-to-one packet forwarding and all-to-all aggregated throughput. We will leverage the bloom filter to check the flow collision in data center networks.

Acknowledgment

This work is supported by the Fundamental Research Funds for the Central Universities (No.2452015197). This work is also partly supported by Hubei Provincial Department of Education scientific research programs for youth project (No. Q20153003), and Cooperation fund of enterprises and universities of Higher Education of Wuhan (No. 2013CXY20).

References

[1] Al-Fares M, Loukissas A, Vahdat A,. A scalable, commodity data center network architecture. ACM SIGCOMM Computer Communication Review(2008) 38 (4):63-74

[2] Guo C, Lu G, Li D, Wu H, Zhang X, Shi Y, Tian C, Zhang Y, Lu S,. BCube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Computer Communication Review (2009) 39 (4):63-74

[3] Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S VL2: a scalable and flexible data center network. In: ACM SIGCOMM computer communication review, 2009. 4:51-62

[4] Parvez N, Mahanti A, Williamson C,. An Analytic Throughput Model for TCP NewReno. Ieee Acm T Network (2010) 18 (2):448-461.

[5] Caceres, R. Duffield, N. Measurement and analysis of IP network usage and behaviour, J. Sci. Commun (2000) 38(5) :144-151