A Novel Cache-based Framework for Accelerating Real-time Flow Transmission

Xingyan Zhang*, Luokai Hu, Haijun Wang, Limin Liu

School of Computer Science, Hubei University of Education, Wuhan, Hubei.

*E-mail: zhangxingyan@hust.edu.cn

Keywords: online social networks; data center; flow transmission

Abstract: Real-time flow transmission is widely used in various online applications, such as video transmission, social networks, etc. Most of these tasks which require strict deadlines need to be handled by distributed systems, so the deadline guarantee of the whole system can be divided into each node. However, due to insufficient computing power or software errors, some nodes are prone to delay data processing, leading to the loss of computing deadline, resulting in the loss of the cut-off event of the whole system, and causing the data processing delay of the next link. In view of the performance loss caused by this situation, this paper proposes a cache-based processing framework to speed up the forwarding of real-time data flows. By estimating the data processing ability of nodes online, when the processing ability of nodes is out of order, some computing tasks can be mapped to other preparatory nodes online, so as to achieve fast task. Migration ensures the deadline of computing tasks on nodes. By adjusting reasonable system parameters, the processing capacity of nodes in distributed system can be accelerated, so that the processing capacity of the system can be increased dynamically and the data flow forwarding capability of the whole system can be accelerated. The results show that this method can increase the number of data forwarding by 23% and reduce the data loss rate by 31%.

1. Introduction

In recent years, with the maturity of cloud computing technology, some individuals and Internet companies no longer deploy independent servers to manage individual or corporate websites, but deploy applications on cloud platforms. The process of user accessing the server is actually the process of submitting computing tasks to the cloud platform. Such benefits can save labor costs for enterprises, and these service providers can also provide more reliable performance assurance. However, with the increasing number of users, especially with the development of applications such as social networks, there are more and more interactions between user applications. For example, the interaction of these applications will generate a lot of communication between tasks, similar like the Wechat public platform. In addition, the scale expansion of distributed systems makes communication between these tasks more complex, and the change of data traffic poses new challenges to task scheduling.

The communication mode generated by the application has changed. Most flow scheduling methods mainly deal with large data, such as MapReduce. The length of data flow generated by these applications is easy to predict and control. However, data flows generated by new applications include fixed length data and unpredictable data transmission, such as search services, social networks and so on. On the one hand, tasks include applications such as file transfer, such as chunk in distributed file systems. Throughput is an important performance to ensure these file transfers. On the other hand, burst data flows between online search services and users in social networks need to be completed within a certain deadline, and the transmission of these flows requires strict real-time performance. These mixed data flow distributions increase the difficulty of data flow processing, which makes the network flow scheduling method of data center more complex.

The expansion mode of data center scale has changed. The rapid growth of users has led to an increasing number of servers in data centers. When an enterprise establishes a data center, the number of supporting users tends to increase exponentially. When the scale of users is saturated, the number of users supported by data center users may show an incremental growth pattern. Such changes make the network structure of data center appear incomplete topology, such as Fat-Tree [1], BCube [2] and other classical network topology. These incomplete topologies easily lead to new network bottlenecks, resulting in the loss of performance of flow transmission.

The application of hybrid cloud [3] makes communication between tasks more complex. When the scale of enterprise data center can’t meet the user demands, some enterprises choose to rent some cloud platforms to meet the needs of user expansion in order to reduce their investment. Many tasks need to be deployed to public cloud platforms, and the communication between these tasks is different from that within the data center. The unreasonable migration may make communication between some tasks need to cross the public network, resulting in unnecessary communication overhead. How to select tasks that need to be migrated to public cloud platforms becomes the key to reduce performance loss.

In conclusion, the long flows are the main data flows in the data center network. The transmission of these data flows takes up the main bandwidth of the network. To avoid these long flows transmission through the network root nodes or across different data centers, the link congestion caused by long flows transmission can be avoided. In cloud computing data center, unreasonable task deployment will cause many long flows of communication between tasks, which will occupy most of the network bandwidth, and these problems are unavoidable when tasks are deployed. To avoid this situation, only relying on migration tasks to a reasonable location.

In this paper, we analysis the main problem of real-time flow transmission and point the lack of node computing ability is the main reason. And then, a novel cache mapping technology is proposed to accelerate the forwarding of real-time data flows, which from different dimensions to analysis the performance requirements of data flows. Our method can check the failure node or insufficient nodes and decide whether the computing task needs to be migration. We design an efficient cache-based task migration to transfer this task to the prepared node. The results show that this method can better increase the number of data forwarding and reduce the data loss rate. The background and motivation for our works is described in Section 2, and in the Section 3, we show the detail of our design. The next Section analysis the performance. At last, we conclude our works and direct the future works

2. Background and related works

In order to study the related technologies to improve system reliability, the requirements of flow processing applications and the characteristics of flow processing models must be considered. Broadly speaking, to meet the challenge of high reliability of data flow processing systems, solutions and related technologies must be designed to meet the efficiently, availability, adaptability and scalability, which requirement details are described as following.

Efficiently. The essence of the real-time characteristics of flow processing is that related technologies are required to maintain low processing latency. In other words, these technologies must be able to provide fast recovery and fail-free cycles without interrupting conventional processing.

Availability. Data flows are fast and variable in speed. For this reason, many problems of flow processing focus on the efficient utilization of system resources. Usually, the improvement of system reliability is achieved by backing up related resources or data and utilizing additional resources of the system. Therefore, the economic use of resources has become a key issue.

Adaptability. One of the main characteristics of data flow is that its rate varies with time. This change will affect the processing load of the operating box and the CPU cycle that can be used to achieve high reliability. In addition, as the operating box processes more input data, the state of the operating box and the overhead of checkpoint technology will increase accordingly. Therefore, the relevant technologies adopted must be able to adapt to this change.

Scalability. A data flow cluster processing system consists of a large number of components, so the related technology must be designed to have good scalability.

Compared with traditional Internet networks, data center networks use multi-path to expand bandwidth, so data flow scheduling strategy should take full account of the impact of multi-path. According to different application performance requirements, flow scheduling algorithms can be divided into high throughput-oriented flow scheduling and real-time flow-oriented flow scheduling methods.

(a) The high-throughput flow scheduling

High throughput has always been the goal of data center network, which means that more data flows can be accommodated. Existing data centers mainly support applications such as MapReduce [4], Hadoop [5], which require a lot of data transfer and more throughput. Raiciu et al. [6] used multipath TCP protocol to increase network performance and robustness. Multipath TCP selected multiple links for different data flows to transmit data, and used the rate control on these paths to meet the high-speed transmission of multiple data flows. From the experimental results, multi-path TCP is more suitable for the data center network than traditional TCP. However, to play a greater role, it needs the support of transport layer protocol. Experiments also show that multipath has advantages in failure recovery, but large-scale network topology determines that there are many problems to be solved to achieve high throughput. Al-Fares et al. [7] studied data flow conflicts in data center networks. Long flows in data center networks are prone to collisions on core switches, resulting in a decline in network throughput. The authors use global communication matrix to detect long flow conflicts, and use simulated annealing algorithm to find lighter load paths for conflicting data flows to expand network throughput. Popa et al. [8] raised the issue of how to fairly and reasonably allocate network bandwidth in data centers. The main objectives include minimizing bandwidth, allocating bandwidth proportionally and high utilization, which the best resolution is based on the tradeoff of these three performances. The author puts forward three compromise schemes to solve this problem.

(b) The real-time flow scheduling

Applications like data lookup and social networks need to return user results in real time. However, the unique network structure of data centers can affect the transmission of these data flows, such as Incast [9]. Because overhead switches need to connect to multiple servers, the communication between these applications and these servers will lead to the congestion of switches, thus delaying the transmission of data flows. Explicit Congestion Notification (ECN) [10] is used to notify senders to reduce the transmission of data flows, which can avoid sending congestion due to the exhaustion of buffer space in switches in advance, thus reducing the contention time of data flows.

3. The methodology and design

In this chapter, we describe how to use Cache-based to quickly transfer node tasks that are already under-resourced. Firstly, we describe how to use existing methods to test whether tasks need to be migrated based on detection points, and then describe the design essentials of cache-based systems.

Data flow processing system expands its scale and computing power by clustering, but at the same time, it also brings the problem of reliability. The integrity of servers in the runtime system may be damaged due to power outages, system failures, network partitioning, and so on. Therefore, the system must solve the problem of reliability. When a server fails, other servers take over the work of the failed servers to ensure the continuous and stable operation of the system.

The premise of implementing parallel recovery scheme is that backups can be distributed on multiple servers. On the basis of passive standby mode, this paper combines operation boxes to form checkpoint units. The system takes checkpoint units as backup granularity. In order to achieve optimal global backup allocation, backup reallocation algorithm is adopted to backup checkpoint units equally on multiple servers. When one server in the cluster system fails, multiple backup servers take over the work of the failed servers and cooperate to achieve rapid and accurate recovery.

The system uses the form of detection points to detect the fault of nodes. Checkpoint technology is the key support for operation box migration and system backup recovery. Recovery after failure must be built on the condition that some of the crash data and environment information are known.

Therefore, using checkpoints to save these necessary information is conducive to shorten the recovery time and improve system performance.

The working process of checkpoint setting is divided into two stages: generating checkpoint messages at checkpoint time and sending them to backup server; backup server obtains the contents of checkpoint messages and replicates them into corresponding backup images. These two phases are called capture and paste tasks.

Register a continuous query request in the system and form a query plan after optimization and compilation. In order to make more effective use of system resources, the newly formed query plan will be combined with the existing query plan in the system and optimized. All query plans running in data flow system can be regarded as a directed acyclic graph composed of "points" (data query operations) and "edges" (data flow direction). We call it query network vividly.

In the data flow cluster processing system, the system backup is completed by distributed backup checkpoint unit. Here the backup granularity is neither a server nor an operating box, but a checkpoint unit. This is because if the server is used as the backup unit, the backup granularity is too large, which brings significant overhead to the capture task and paste task, and the backup scheme cannot achieve the optimal. If the operating box is used as the backup unit, the backup granularity is too small, which leads to the system having too many backup units to manage and allocate, the system cannot optimize the rational use of resources, and the backup scheme cannot achieve the optimal. Therefore, the choice of backup granularity is very important, and it is an important factor to reduce backup overhead. Therefore, this paper combines operation boxes to form checkpoint units, and redistributes checkpoint units from the perspective of global optimum of the system through backup allocation algorithm.

Figure 1(a) shows the framework of job acceleration system based on cache. The system can be logically divided into three levels. The bottom level uses detection point method to detect the effectiveness of nodes, the middle level uses cache system to transfer tasks quickly, and the top level is various backup systems, which is the resource guarantee for task migration. The implementation process of the algorithm is shown as figure 1(b): when a message arrives, the flow identifier of the header of the message (< SRC, DST > binary group) is used as the input of the mapping function, and the K outputs are mapped to the array bits of cache 1 and each bit of the array is added 1 (1-3 rows), then the minimum value of the K array bits mapped (4-7 rows) is found. Then, it is judged whether the minimum value is greater than or equal to the pre-set threshold len_tsh. If the condition holds, the array bits corresponding to Cache1 are directly subtracted from len_tsh. And map to the Cachenext level of Cache 2, so that the K corresponding array bits in Cache 2 add 1 (8 ~ 13 rows). By analogy, if the number of bits in Cache 2 is greater than len_tsh, it maps to the next level of Cache.

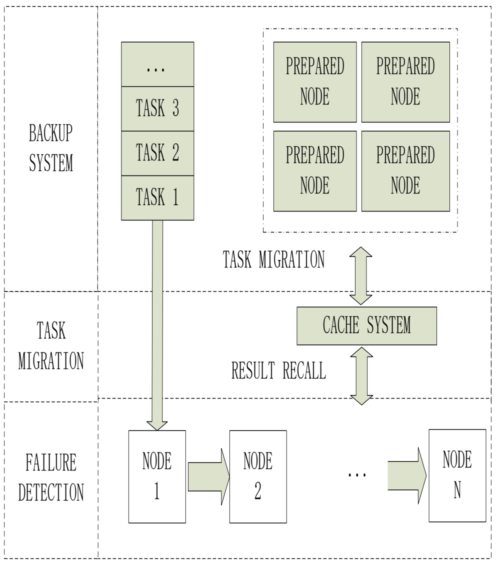

Fig 1 (a). The cache-based accelerated framework system

```
input: <SRC, DST>: the two-triple of task flow;
       $C_{1[m]}$: the value of position m;
       len_tsh: the threshold of mapping to next level;
output: the order of task which should be migrated;
  step:
  1.  for ( i in k ) do
  2.      $C1_{[j]}$++;
  3.      counter = $C1_{[j]}$;
  4.      if ( $C1_{[j]}$ > counter ) do
  5.          counter= $C1_{[j]}$;
  6.        done
  7.      end
  8.      if (counter >= len_tsh ) then
  9.        for ( j in m) do
  10.           $C1_{[j]} = C1_{[j]}$ - len_tsh;
  11.           $C2_{[j]}$++;
  12.       end
  13.  end
```

Fig 1(b) Description of cache-based job resoure allocation

4. Analysis

The migration process of tasks is represented by the transmission of long flows, whose length is equal to the amount of data transferred by tasks. This experiment does not pay attention to the capacity of the target location server of the migration task, but verifies the algorithm's effect on the performance of the whole network when using the task migration. In the simulation environment, multi-level cache is used to detect the tasks that need to be migrated in the system and migrate to the backup server quickly. Task migration performance is mainly reflected in large data flows. When these tasks are migrated, the original task server will be transferred to the target location under the switch.

The data source used in the experiment is the experimental data published by the Distributed Laboratory of Hebrew University in Israel. The eigenvalues of these data are improved to make the completion time of these tasks meet the distribution of network flow in the data center. The main reason for choosing these data as test data is to test the effect of user experience. When the data flow cannot complete the work within the specified time, it will seriously affect the user experience.

In order to test the performance of our algorithm, we evaluate the performance of the network system itself and user experience. This experiment tested the proportion of tasks that lost deadlines. For this reason, the congestion condition of links is defined firstly, and links that occupy more than 70% of the bandwidth are set as hot links. The influence of Cache-based algorithm on the performance of the whole network is determined by the number of hot links in the process of flow transmission. Through empirical data, the number of hot links can be obtained by monitoring the status of network links. The distribution of hotspot links can reflect the performance of the system, and fewer hotspot links show that the network is more usable. In addition, the throughput of the system can be determined by all the flows successfully accessed to the network. Especially in the case of heavy system load, the throughput can well reflect the ability of the system to process jobs. This performance is the most concerned indicator of cloud computing investors, reflecting the number of users the system can accommodate. Throughput calculation method is the data generated by job communication between multiple tasks. It is worth noting that the data generated during task migration is not considered as throughput, because this part of data is the additional load of system scheduling.

Table 1 shows how jobs are completed by different scheduling algorithms with the same number of jobs. The three graphs represent the proportion of tasks with lost deadlines in FAT-TREE and BCUBE structures respectively. The number of jobs completed by Fat-Tree, BCUBE and our Cache-based algorithms is calculated. Ideal state refers to the number of jobs that need to be completed every day. This part is obtained directly from the files in the network log. The number of assignments to be completed each day is roughly the same, but the number of assignments to be completed each day is different. Both Fat-Tree algorithm and Cache-based algorithm have fewer jobs completed on time than ideal state. This shows that using Cache-based algorithm can effectively reduce network congestion and reduce the delay of jobs caused by these congestions.

Table 1. The missed deadline with different parameters

	Q=0	Q=0.2	Q=0.4	Q=0.8	Q=1
Fat-Tree	0.22	0.18	0.15	0.21	0.28
BCube	0.24	0.16	0.14	0.24	0.22
Cache-based	0.18	0.12	0.11	0.18	0.13

5. Conclusion

Tasks in distributed systems may not be completed in time. The long flow of these tasks can easily lead to system performance. A large number of tasks migration may cause link congestion in the network. If the location of these tasks can be reasonably managed, the resources consumed by these tasks can be reduced. Various new application forms make it difficult to predict the distribution of data flow between tasks, so that it is impossible to avoid the congestion caused by this long-flow communication in application deployment.

Acknowledgment

This work has been supported by the project funded by Hubei Provincial Department of Education (No. B2018203).

References

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

[2] C. Guo, G. Lu, D. Li, et al. BCube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Computer Communication Review, 2009, 39 (4): 63-74.

[3] N. Chopra, S. Singh. Survey on scheduling in hybrid clouds. In: Processings of 5th International Conference on Computing, Communication and Networking Technologies (ICCCNT'14), 2014, Hefei, China, July 11-13, 2014. Washington, DC, USA: IEEE Computer Society, 1-6.

[4] J. Dean, S. Ghemawat. MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008, 51 (1): 107-113.

[5] K. Shvachko, H. Kuang, S. Radia, et al. The hadoop distributed file system. In: Processings of IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST'10), 2010 Incline Village, NV, USA, May 3-7, 2010. Washington, DC, USA: IEEE Computer Society, 1-10.

[6] C. Raiciu, S. Barre, C. Pluntke, et al. Improving datacenter performance and robustness with multipath tcp. ACM SIGCOMM Computer Communication Review, 2011, 41 (4): 266-277.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, et al. Hedera: Dynamic Flow Scheduling for Data Center Networks. In: Processings of 7th USENIX symposium on Networked Systems Design and Implementation (NSDI'10), 2010, San Jose, California, USA, April 28-30, 2010. Berkeley, CA, USA: USENIX Association, 19-19.

[8] L. Popa, G. Kumar, M. Chowdhury, et al. FairCloud: sharing the network in cloud computing. In: Processings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM '12), 2012, Helsinki, Finland, August 13-17, 2012. New York, NY, USA: ACM Press, 187-198.

[9] H. Wu, Z. Feng, C. Guo, et al. ICTCP: Incast congestion control for TCP in data-center networks. IEEE/ACM Transactions on Networking (TON), 2013, 21 (2): 345-358.