ATLANTIS PRESS
Advances in Computer Science Research, volume 44
3rd International Conference on Wireless Communication and Sensor Network (WCSN 2016)

Arana: A Cross-domain Workflow Scheduling System

Jian-Hua Gu, Xue-Yuan Lan, Ying Hao, Yu-Tong Hu
School of Computer Science, Northwestern Polytechnical University, Xi'an, P.R.China
E-mail: gujh@nwpu.edu.cn, 18710832737@163.com, haoyying@mail.nwpu.edu.com, hytt502@gmail.com

***Abstract*-Single data center is difficult to meet current demands of data storage and computing resources, leading to data broadly stored in different data centers far from each other geographically. Cross-domain big data processing usually requires data from other data centers across different network. Since the network environment is extremely complex and large amount of data is required for computing, time spent in data transmission is unacceptable, resulting in slow scheduling. In this paper, we propose a cross-domain workflow scheduling system called Arana. By moving program close to data and integrating popular big data processing platform, this system enables user to complete computing with cross-domain data without transferring.**

Keywords-workflow; cross-domain; big data; close to data; schedule

I. INTRODUCTION

The continuous development and progress of computer technology has promoted data size growing in an incredibly quick speed in field of scientific research, computer simulation, internet applications, and e-commerce etc. [1] Storing such amount of data has far exceeded the capacity of single node, let alone single data center. A program may need to analyze data across different nodes to obtain correct results. When data centers are far from each other geographically, how to use remote data conveniently has become an urgent problem to be solved.

Since data are not locally available in cross-domain tasks, the traditional solution is to transmitting data to local and then processing task. There are two obvious shortcomings in this acquire-store solution. The first is that transmission time of huge amount of data may be too long to accept in practice. The second is that user may not have access to required data due to security issue or any other reasons.

To solve problems mentioned above, we propose Arana, a cross-domain workflow scheduling system, greatly shorten the scheduling time by moving program close to required data. Arana make program initiatively seek data that need to be processed, complete task calculation on the data node, finally receive and integrate results. Using Arana, a lot of time spent on scheduling data can be saved and user's computing tasks can be executed in time. Thus, the overall throughput can be significantly improved.

Arana also provides the function of workflow scheduling. In Arana we consider one workflow as a job. User can customize different task program for different data nodes, compose these programs as a job and then submit it to Arana. Arana will automatically schedule the job and ensure the reliability.

Moreover, Arana uses lightweight database to store workflow information, which makes Arana able to restore the runtime environment immediately after a performing error while meeting the demands of storage, thus the reliability of jobs can be guaranteed.

In section 2, we will briefly introduce related works, a brief description of the advantages and disadvantages of these system and Arana's advantages. Section 3 mainly introduces Arana and how Arana guarantee the reliability of the system. The evaluation of the system locates in section 4, where we can see Arana shorten job scheduling time greatly.

II. RELATED WORKS

The study of computing remote big data nowadays mainly focus on fast data acquisition from remote nodes and optimizing execution time. However, this two study aspects have been limited a lot. When devoting to decrease execution time, time spent on data transmission is also need to be considered, since the data transmission time usually accounts for a substantial part of overall time in big data calculation projects [2]. As well as the speed of data transferring is limited to network status. The crossing field of the two technologies has been repeatedly investigated by related researchers.

The Stork data placement scheduler [3] [4] aims at data intensive applications which access, create, and move large amounts of data over wide area networks. Stork provides solutions for many of the data placement problems encountered in the distributed computing environments with its data placement scheduler. Stork can locating data, moving data in a predicable way. HPGOSS [5] based on parallel file System is used for eliminating I/O performance bottleneck and deal with the data managing issues resulting from the close relevancy between geo-information and remote sending image data, which data are distributed on different nodes in different regions. Encountered with huge amount of data, either Stock or HPGOSS need considerable time to complete data transfer, which will cause low performance.

Close-to-Files (CF) job-placement algorithm tries to place job components on clusters with enough idle processors which are close to the sites where the input files reside [6]. Placing data as close as possible to computation is a common practice of data intensive systems, commonly referred to as the data locality problems [7]. DARE, a distributed adaptive data replication algorithm, aids the scheduler to achieve better data locality.

Thus, placing computation close to data can save a lot of work and make great improvement on I/O performance in remote data computing environment. In traditional ways, Telnet protocol can be used to login remote node to execute

tasks, analysis data, debug program etc. Except for running programs background, using Telnet requires enough speed to meet the need of interactive processing.

Using Telnet command, local node needs to have the privilege to login remote node. When computing on remote node, user firstly needs to manually copy program from local node to remote node, then debug and run program on remote node, wait till the end of running, and finally copy result from remote node to local node manually. User needs to participate the whole procedure. Except for coding computing program, Arana can complete the whole procedure automatically.

To some extent, Arana is a lightweight job scheduling system. It sends programs to nodes which holds data instead of execution after acquire data from the remote, which saves a lot of time spent on data transmission. It realizes high performance in scheduling and quick recovery, scalability, and reliability. Moreover, by our predefined interface, Arana is applicable to execute mapreduce, java programs etc.

III. SYSTEM DESIGN

A. Architecture

Arana can be structurally divided into two modules, job scheduling engine and task execution engine. Job scheduling engine module, which is deployed in local node, consists of two layers: job monitoring layer and job scheduling layer. It is responsible for parsing the job that user submit. Job, as well as the tasks in the job is descripted by the process description language that we define. The job scheduling engine is also responsible for receiving the state update of the task, so as to decide the next task that should be scheduled.

The task execution engine module utilizes the Hadoop distributed system at the bottom layer to achieve high scalability and big data distributed storage. The upper layer schedules the task that send to this computing node. At the same time, by real-time monitoring Hadoop, this layer acquires the state of the task, and update them to the job scheduling engine. Fig. 1 illustrates the Arana architecture.

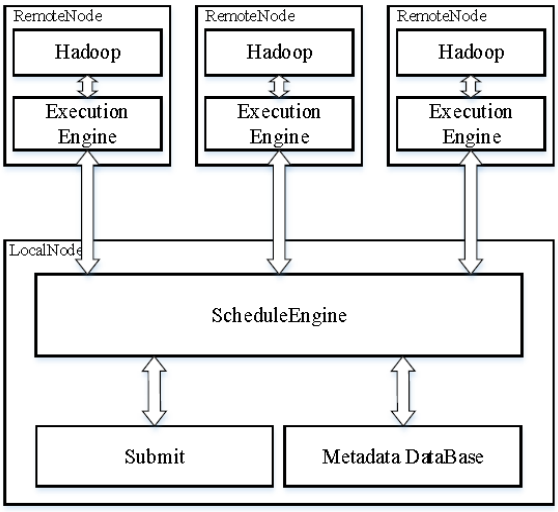

Figure 1. Arana architecture

B. Job Scheduling Engine

As the core of Arana, scheduling Engine receives jobs that user submitted, schedules executable tasks, dispatches tasks to their nodes and monitors jobs' life cycle. Finally, it aggregates the results from each node into the final result.

In Arana we see a user defined process as a job. In general, JOB= (ID, Name, $Task_1$, $Task_2$, $Task_3$…), TASK= (Sequence Number, Type, Target Host Address, Program, Parameter, Subsequent Task). Usually, job defined by user is a DAG (Directed Acyclic Graph). According to the above definition, we can split a DAG into several vectors. Job scheduling engine only need to pay attention to the JOB vector.

As shown in Fig. 2, the job scheduling engine allocates resources for each separate job. Each job schedules its own tasks respectively. Through pre-allocating resources, after user submit a new job, scheduling engine can quickly set the job ready, involve the job in scheduling immediately. After picking out the tasks that can be executed in the next step, scheduling engine will sent out the tasks to release the resources. By storing job and its metadata in database, virtually every job occupies few resource in memory. Scheduling engine only needs to know job ID. A large number of other information are stored in the database. So job scheduling engine can maintain a high throughput under a large number of jobs, and job queuing rarely appears.

Fig. 3 explains the complete scheduling process for a typical job:

a) When a new job is submitted to the job scheduling engine, it will first be decomposed into a few task vectors by Analyzer.

b) Next, these vectors will be stored in metadata database. Job will be identified by its ID, and task will be identified by its job ID and sequence number.

c) Scheduler will retrieve the database table, pick out the executable tasks, and send them to Dispatcher.

d) On the basis of the information in the task vector, dispatcher will sent task to the corresponding remote node for execution.

e) According to the information returned by remote node, (3) (4) may be executed several times to complete the entire job scheduling.

f) Finally, scheduling engine will integrate different tasks' result. Usually, the integration program is also written by user.

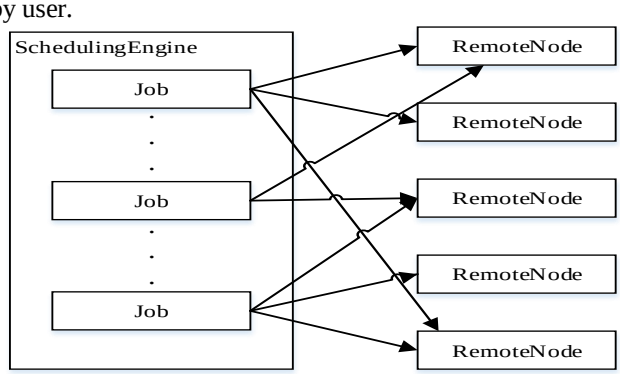

Figure 2. Job scheduling engine

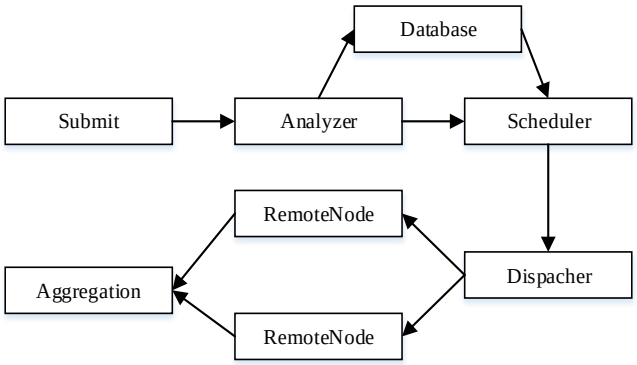

Figure 3. Job scheduling

C. Task Execution Engine

The task execution engine receives the task vector from job scheduling engine. Because the main analytical work has been completed by job scheduling engine, the task execution engine only needs to complete the preparatory work and submits the task to the specified environment or execution platform. Then begin the task execution phase.

During task execution, the execution engine also needs to complete some operations specified by user, such as creating result file, cleaning temporary data etc. Task monitoring as a core function is essential for task execution engine. Now we have realized the monitoring of the Hadoop platform and Java program, which can monitor the state of submitted job in real-time, pull the execution information from Hadoop and send them back to user.

And through the interface that we define, Arana can easily be extended to other system. The monitoring module returns the task execution state and real-time running information to scheduling engine. Scheduler combines these information to complete job scheduling. Scheduling engine can also be aware of task execution error through these information, and feedback error information to user timely.

D. Metadata Database & Reliability

Metadata database is a powerful means for Arana to ensure reliability. Metadata database is an independent database system, so it can be configured to satisfy the needs of different levels of reliability. At the same time, database keeps all the information jobs and tasks need. The reliability of the scheduling engine and execution engine both depend on metadata database.

The state of job and task are defined in detail. In example of task state migration, a simplified task state migration can be roughly expressed as follows: RECEIVE—RUNNING—SUCCESS (or FAIL)—FINISH. Of course the actual state migration is more complicated, but the main state transfer can be divided into the above steps. Through real-time monitoring, the task state changes will update to the database.

When system or process fails, first step after the resumption is to scan the database overall. Since the state information has been saved, Arana can quickly locate the task execution position and the next scheduling will soon start again. When crash results in inconsistent information of scheduling engine and execution engine, as state transfer cannot fallback, through one update information system can quickly recover. Based on its preserved information, execution engine can also avoid unnecessary task re execution errors.

IV. EVALUATION

In this section, we mainly test the performance of Arana under multi nodes cluster. Whether start scheduling time of Arana will be significantly affected when a large number of jobs is submitted to system is evaluated. And the resources Arana uses is evaluated when scheduling a large number of jobs.

A. Environment Preparation

20 nodes with Intel Xeon E5-2670, 64G memory, 8 SATA hard disks of 16T (single disk capacity 2T) and 8 processor (2.60GHz); 4 Hadoop clusters is simulated, each including 5 nodes.

Test data sets are mainly collected from Internet, including the latitude and longitude coordinates of geographic names. But there is a lot loss in the latitude and longitude information. The test program is to calculate the latitude and longitude of geographic names according to the existing latitude and longitude information.

B. Performance Test

First, we analyze the start scheduling time. The start scheduling time of a job is refer to the time spent from the job is submitted to its first task is scheduled. Task execution time is related to specific task, which we don't analyze here. Fig. 4 shows that Arana has a good scheduling efficiency for submitted jobs. Under the number of submitted jobs of 1, 5, 15, the start scheduling time slightly fluctuates. When the number of submitted jobs increase up to 800 or 1000, the average start scheduling time of each job is gradually approaching to 600ms, and the fluctuation is within reasonable range. Fig. 4 indicates scheduling time of a job can be well controlled in Arana and job can be processed immediately after submitted to Arana.

Figure 4. Start scheduling time

Fig. 5 shows the memory usage of Arana while job continues to be submitted to the system. When system starts,

Arana takes less memory, only 57M. After the first job is submitted, the memory usage begins to increase rapidly, which is because some resources are only allocated when job is being scheduled. The growth rate slows down in the second half of the curve, which is because resources used by previous complete jobs has released while there are still new jobs coming. As described in the previous section, Arana store workflow information in database, therefore, the cost of the new jobs is reduced to minimum. There is a small amplitude increase in the latter half of the curve, because Arana will cache some key information to speed up the scheduling of tasks. Finally, the memory usage of Arana floats around 300M.

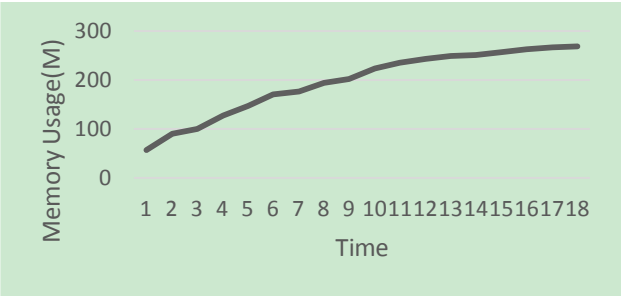

Figure 5. Memory usage

V. CONCLUSION

This paper firstly introduces some big data processing platform aims to explain the differences between them and Arana. The difference is that their optimization objective is to make data more close to calculation. Arana, on the contrary, based on the idea of make calculation more close to data, reduces unnecessary data movement to obtain better performance, which is much more suitable for cross domain job scheduling. In addition, using Hadoop platform in the bottom layer, scalability and distributed storage are significantly improved.

We use database to obtain higher reliability, but with the number of tasks grows, communication with database has become the bottleneck of system performance. And Arana currently only supports Hadoop, but using the interface that we define, Arana has the ability to access more big data processing platform, by integrating more big data processing platform, we can achieve a more unified scheduling system. These are what we should devote to in the future.

ACKNOWLEDGMENT

Sponsored by the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University.

This work is supported by the Science & Technology Innovation Engineering Program of Shaanxi Province under Grant No.2015KTZDGY08-03-01.

REFERENCES

[1] Zhou AY. Data intensive computing-challenges of data management techniques. Communications of CCF, 2009,5(7):50-53.

[2] Thakkar, Shraddha; Patel, and Sanjay, Scheduling in big data heterogeneous distributed system using Hadoop. Advances in Intelligent Systems and Computing, v 409, p 119-131, 2016.

[3] Kosar, Tevfik, Data intensive computing-challenges of data management techniques. Proceedings - Challenges of Large Applications in Distributed Environments, CLADE 2006, v 2006, p 5-12, 2006.

[4] Kosar, Tevfik, Livny, Miron, A framework for reliable and efficient data placement in distributed computing systems. Journal of Parallel and Distributed Computing, v 65, n 10, p 1146-1157, October 2005.

[5] Ma Yan, Liu Dingsheng; Li Jingshan, A new framework of cluster-based parallel processing system for high-performance geo-computting. International Geoscience and Remote Sensing Symposium (IGARSS), v 4, p IV49-IV52, 2009.

[6] Mohamed H.H., Epema D.H.J., An evaluation of the close-to-files processor and data co-allocation policy in multiclusters. IEEE International Conference on Cluster Computing, ICCC, p 287-298, 2004

[7] Abad Cristina L., Lu Yi, Campbell Roy H., DARE: Adaptive data replication for efficient cluster scheduling. IEEE International Conference on Cluster Computing, ICCC, p 159-168, 2011.