Advances in Engineering Research, volume 113
International Forum on Mechanical, Control and Automation (IFMCA 2016)

A Study on Load Balancing Techniques for Task Allocation in Big Data Processing*

Jin Xiaohong[1,a], Li Hui[1, b] , Liu Yanjun[1, c], Fan Yanfang[1, d]

[1] Beijing Institute of Science and Technology Information, Beijing 100048, China;

[a]jinxh@bjstinfo.com.cn, [b]lisa-lh@126.com, [c]liuyj@bjstinfo.com.cn, [d]34678883@qq.com

Keywords: Big Data; Job Schedule; Distributed Computing; Clustering; Load Balancing

Abstract. This paper introduces the task allocation techniques with clustering and load balancing in the field of Internet to the field of image processing job allocation of alternative big data. It designs and realizes a load balancing cluster architecture for the alternative big data, and an improved load balancing algorithm applicable to large-scale image processing. The experimental results show that the cluster architecture can execute task allocation and data processing continuously and stably, and the improved load balancing algorithm could improve the processing efficiency about 10% and more .

Introduction

With the increasing expansion of remote sensing image application market, the means and capability to obtain remote sensing images are greatly improved, and the data scale of remote sensing images acquired annually grows rapidly. Facing such a large number of image data, the traditional data processing techniques lag far behind meeting the data processing requirements in the age of big data. The automation of data processing and the intelligence of task allocation will certainly become the inevitable development trend in the future.

The production organization and task allocation of data processing still stay in the mode of traditional manual allocation although automated processing without manual intervention or with little manual intervention is basically realized in all data preprocessing links of remote sensing images. Different from traditional Internet big data, the single remote sensing image is large in size and the single processing task in each link is time-consuming. The data processing period of each image vary from tens of seconds to tens of minutes (or even longer) based on the data size and processing link. Although the single batch of data processing tasks do not reach the massive scale, the expected data processing efficiency is difficult to achieve if hundreds or even thousands of data processing jobs are fully allocated in a manual way.

In addition, limited by the data processing features of remote sensing image as well as the processing capacity of a single graphic workstation, it is more inclined to use a multi-device cluster system featured by high performance, low price and strong scalability in practical production [1].Therefore, the problem that needs to be addressed urgently is how to automatically allocate a large number of data preprocessing tasks of remote sensing images to several graphics processing units with equivalent capacities, and how to keep all processing devices of the cluster system efficient without allocating unbalanced loads among such processing devices.

This paper uses the Web cluster technology to build a clustered/distributed processing system for the data preprocessing of remote sensing images, the round robin and quota management mechanism to allocate processing jobs for each processing node, and the improved load balancing algorithm to balance the task loading amount of each graphic workstation in the cluster.

* Fund project: This is one of the research results of Beijing municipal finance projects "Information Processing and Analysis Abilit y Building Against Text Information " (PXM2016-178214-000006) and "Design and Implementation of Specific Entity Relation Extr action and Data Mining Tools"（PXM2016_178214_000007）in 2016.

Web cluster architecture

The server cluster is to organize several servers in the intranet. From the view of outside users, the cluster is just a virtual server; looking from the inside, the task requests generated by the users are distributed to different processing server nodes by such devices or programs as proxy server or load balancer through some allocation mechanism. The clustered servers can achieve the high performance equal to that of a giant server, with high cost performance. The single server in the server cluster is called node which can independently process a request from the user[2].

Load balancing cluster architecture

The Web cluster for remote sensing image preprocessing consists of a task manager (front-end dispatcher) and several graphic workstations (back-end server) deployed in LAN. After an operator uploads the processing tasks to the task pool through the front-end task manager, the system will distribute the data processing requests in the task pool to the back-end graphic workstation nodes by using round robin algorithm[3].

The front-end node is the core of the whole cluster system. It needs to respond to the http request in the operator task submission interface, and also establish a connection with the back-end server. Compared with the traditional big data processing, there is generally just a parallel task load of hundreds in the remote sensing image preprocessing, with a small task allocation workload in the front-end node of the Web cluster and a large operation load in the back-end graphic workstation. In practical application, the front-end task manager shall be independently configured with special hardware device whenever possible in order to prevent graphic operation from affecting the task allocation mechanism due to excessive consumption of system resources.

Reliable information communication is realized between the front-end and back-end nodes through a heartbeat detection mechanism. At the moment of each heartbeat, the back-end graphic workstation will returns its own status message to the front-end node through a status reporter, and the process scheduler will adjust the process quota of the current graphic workstation in a dynamic way according to the control command returned from the front-end node.

The data processing request of each image is regarded as a data processing job in each link of image processing. Each job is arranged in the task submission order; the task submitted first will be processed firstly, and the task submitted late will queue up for processing. To prevent each computer from overload, the number of jobs that each processing node is able to process in parallel is designed for the rear-end node by using the thread per connection, and set as a adjustable initial preset quota according to different data processing task types in combination with the processing performance of rear-end processing nodes. When all back-end processing nodes reach the maximum quota, the job submitted late will queue up until the processing node finishes the previous job, so as to ensure each processing node is not overloaded.

In order to realize the performance isolation of services with different priorities, a queue priority is arranged for each job when a task is submitted. When the queue priorities are the same, the jobs in the queue will be executed one by one by the submission time; where the queue priorities are different, the jobs with a high priority will be processed preferentially.

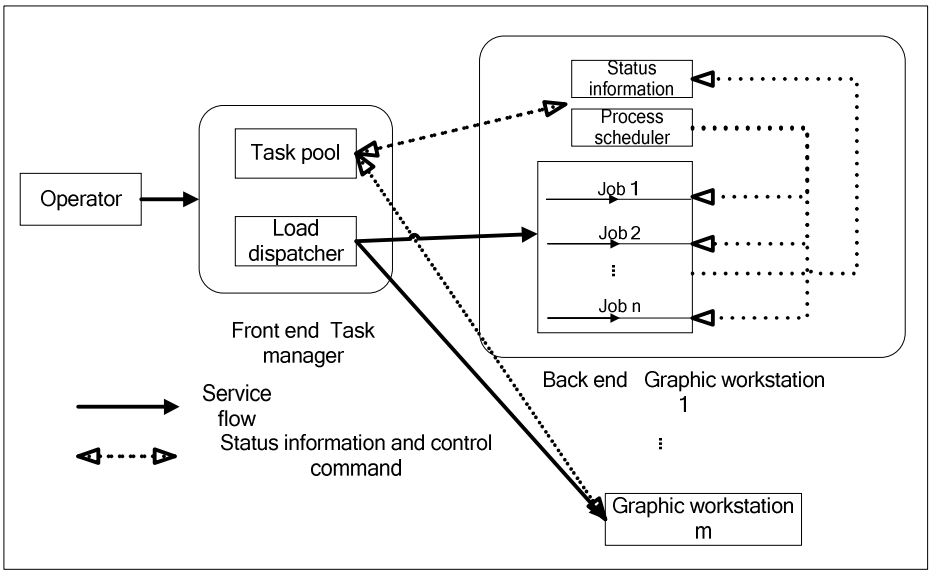

Figure 1 Load Balancing Model

Information exchange mechanism between front end and back end

This paper uses Apache as server software. The operator end is connected with the front-end task manager by http protocol, and the front-end task manager is connected with the back-end graphic workstation by TCP protocol[3]. Figure 2 shows the information exchange between the front end and back end of the cluster architecture. The system transfers the data request submitted by the operator to the database through Apache service in a way of HTTP and XML. All graphic workstations configured respectively establish a connection with the front end. At the moment of each heartbeat, each graphic workstation actively sends a status message to the front end; at the same time, in terms of the job queue to be processed in the front-end database, the jobs to be finished by each graphic workstation are returned to each processing node, according to the preset job priority and the parallel job quota control for the tasks of different job types.

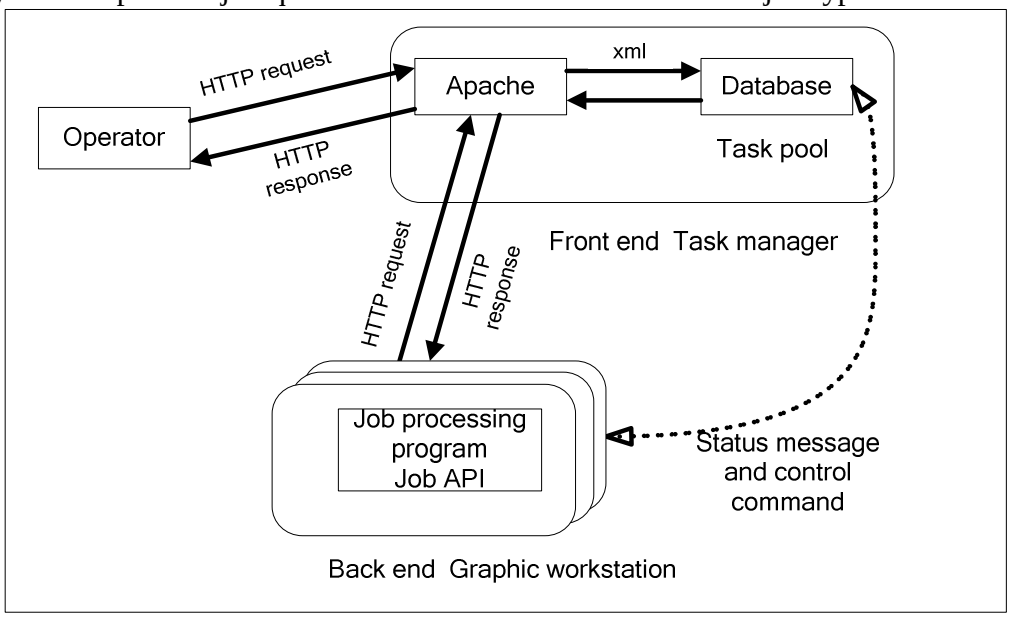

Figure 2 Information Exchange between Front End and Back End of Cluster Architecture

Load balancing

Resource allocation is a core issue for parallel processing and distributed computing. The load balancing technique is the key part of the cluster system and affects the performance of the cluster system [4] [5]. The load balancing technique of the server cluster is aimed to allocate the external

requests to the server cluster in some way through the rational allocation by load balancing strategy while ensuring the current server cluster network architecture is unchanged. After receiving the requests allocated by the load balancer, the processing server in the cluster processes them and feed the processing results back to the front end.

Load balancing algorithm

The common load balancing algorithms include Round Robin, Weighted Round Robin, Least Connection, Weighted Least Connection and Fastest. The study on the above algorithms is mainly based on the Internet application in which the number of request at the client is huge while the processing operation load of each request is small [1]. Whereas, large-scale remote sensing image processing is significantly different from Internet data processing. The data size of a single remote sensing image processing job is generally a scale of megabyte to gigabyte. Such data scale only allows each graphic workstation node to process in parallel several jobs concurrently, and the data processing time of each job ranges from tens of seconds to tens of minutes. When users submit too many data requests, there are a large number of data processing jobs to queue up.

The utilization rates of CUP, memory and disk by different job processing types at different stages can be obtained through deep analysis on the links of remote sensing image preprocessing. Generally, the processing program will read the data to be processed from the disk first, following by the operation link featured in high CUP occupation and then by subsequent pyramid building and result copying process. When the job allocation mechanism with initial preset quota of processing tasks is used for quota control, the quota is generally pre-computed in a manual way and initially preset according to the bottleneck link of data processing. For example, in image fusion, the number of parallel jobs for CPU computation is generally an image fusion processing bottleneck, so the quota control of job allocation is also set according to the parallel computation ultimate capacity of CPU. However, the data processing will enter a long and slow pyramid building and result copying stage (about 20% to 40% in all job stage) when the operation with high CPU occupation is over. At the moment, the utilization rates of CPU, memory and disk are low. But as the current job is not finished and the task quota is not released, the subsequent jobs in a queue will not be executed even though the CPU, memory and disk are in relatively idle status. The processing time of pyramid building and result copying stage is generally longer than the strong operation time of CPU, which causes waste of computing resources and reduced efficiency of data processing. As a result, an improved load balancing algorithm applicable to remote sensing image processing shall be designed for the load balancing job of remote sensing image preprocessing to dynamically adjust the preset quota value for the ongoing data processing job. When the data processing enters the relatively idle status of system resources, the new dynamic balancing algorithm can dynamically adjust the preset parameter value according to the value obtained, enabling all computing nodes to stay in relatively full status all the time until all job tasks in the queue are finished.

At the time of designing a new algorithm, it shall be considered that the time of single remote sensing image processing job is long and the CPU, memory and disk read/write is a long, dynamically changing process. When computing the CPU, memory and disk read/write of computing nodes, it is necessary to consider the status at some time point as well as the status within the recent past time period. Based on the actual testing, it is appropriate to control this time period within 5s to 10s (determined according to different hardware configurations and different jobs), and the load balancing algorithm will initiate the status operation every 5s.

In addition, as the data processing job is executed in cluster environment, it is necessary to take the network mapping disk read/write capacity into consideration when the disk read/write indicator is considered, rather than local disk read/write capacity.

Improved load balancing algorithm

When computing the overall performance indicator of each computing node over the some past time period, it is necessary to read the average performance indicator of the CPU, memory and network disk read/write of each server over a certain past time period, then make statistics on the

average performance indicator of each processing node, find the maximum value from them, obtain the contrast value by comparing the average performance indicator of each processing node with the corresponding maximum value, and calculate the current comprehensive indicator of the processing node by the weight formula [6].

Calculation of the comprehensive performance of each processing node within a certain time

It is assumed that there are m processing nodes in the cluster, and they can form the set N{N1,N2,…Nm}.

1) The processing ratio of CPU in the ith processing node: CSi

$$CS_i = INT(\frac{C_i}{\max(C_1,C_2,\ldots,C_m)} \times 100) \quad (1)$$

Where, Ci is the average processing capacity of the CPU of the ith server over a certain past time, and max(C1, C2, …, Cm) is the maximum processing capacity of the CPU of the server in the cluster.

2) The processing ratio of memory in the ith processing node: MSi

$$MS_i = INT(\frac{M_i}{\max(M_1,M_2,\ldots,M_m)} \times 100) \quad (2)$$

Where, Mi is the average processing capacity of the memory of the ith server over a certain past time, and max(M1, M2, …, Mm) is the maximum processing capacity of the memory of the server in the cluster.

3) The processing ratio of network disk read/write in the ith processing node: DSi

$$DS_i = INT(\frac{D_i}{\max(D_1,D_2,\ldots,D_m)} \times 100) \quad (3)$$

Where, Di is the average processing capacity of the network disk read/write of the ith server over a certain past time, and max(D1, D2, …, Dm) is the maximum processing capacity of the network disk read/write of the server in the cluster.

To facilitate calculation, all ratio ranges are set as the positive integers from 0 to 100. The value reflects the degree of the server to control idle resources at present. The above indicators must be taken into comprehensive consideration to determine whether some server can initiate the next job. As CPU is the bottleneck for quota allocation in terms of most graphic processing operations, different weight values must be assigned to CPU, memory and network disk read/write when considering the comprehensive indicator. As a result, the current comprehensive indicator $GS(N_i)$ of the ith processing node is obtained as

$$GS(N_i) = W_c \times CS_i + W_m \times MS_i + W_d \times DS_i \quad (4)$$

Where:

W_c -- Weight coefficient of CPU indicator;

W_m -- Weight coefficient of memory indicator;

W_d -- Weight coefficient of network disk read/write.

Calculation of comprehensive load indicator

Over a certain past time, the average values of CPU, memory and network disk read/write are respectively Ci, Mi and Di; the actual maximum processing capacities of this server are respectively MaxCi, Max Mi and Max Di; the weighted values of various indicators are respectively W_c, W_m and W_d. The load parameter $LS(N_i)$ of this server over a certain past time can be calculated as per the formula below:

$$LS(N_i) = INT(W_c \times \frac{C_i}{MaxC_i} \times 100) + INT(W_m \times \frac{M_i}{MaxM_i} \times 100) + INT(W_d \times \frac{D_i}{MaxD_i} \times 100) \quad (5)$$

Comprehensive indicator parameter at the processing node

According to the calculated comprehensive performance and comprehensive load indicators over a certain past time at various processing nodes above, the comprehensive indicator parameter $F(N_i)$ at the processing node I can be calculated as follow:

$$F(N_i) = \frac{GS(N_i)}{LS(N_i)} \quad (6)$$

The larger value of $F(N_i)$, the higher comprehensive performance indicator, and the more callable resources in the current computer. After the load balancer calculates the comprehensive indicator parameter of each server in the cluster as per the formula above, the relatively large parameter value indicates that the server is able to accept new processing tasks when the current job haven't been completed. As a threshold is set to the parameter value, when the parameter value exceeds this threshold, the load balancing algorithm changes the preset quota parameter so that this processing node can accept the next processing job in queue. After the processing node accepts the new job, the load balancing algorithm calculates a new parameter value at a fixed interval. When the parameter value exceeds this threshold again, the processing node accepts the next processing job in queue repeatedly in this way until all jobs in queue are processed.

Experimental results

1). In the test, we set up a small Web service cluster processing system, including one front-end task manager and two back-end processing nodes. Distributed cluster processing is conducted to 40 sets of data fusion jobs.

With the new architecture, the system can operate for 7*24 hours without interruption under the ideal conditions of data, and the initiation response time of each new job is preset within 1s.

2). To test the improved load balanced algorithm,we selected different size imagery from different sensor. Each group has 20 sets of imageries.

Table 1. The test data

Group	Sensor	Input_Average_Size (GB)	Output_Average_Size (GB)	sets of imageries
A	KOMSAT-DE2	0.55	1.69	20
B	KOMSAT-MSC	1.43	4.29	20

Each group imageries will be tested in different test plan:

Test 1: Data fusion with Round Robin method;

Test 2: Data fusion with imporved load balance algorithm.

Compare the consuming time in different test plan. We got the table:

Table 2. Comparation of different test plan.

Group	time-consuming(minutes)		Efficiency improvement
	Test 1	Test 2	
GroupA	39.73	36.58	7.93%
GroupB	99.42	87.07	12.42%

According to the experiment, imporved load balance algorithm could improve the processing efficiency about 10%. And with the increase of the amount of single imagery, there is more space for the improvement in efficiency.

Conclusion

This paper, based on the cloud processing thinking, introduces the clustering concept into the field of remote sensing image processing, and designs the automatic allocation mechanism of image processing jobs as well as a new load balancing algorithm and mechanism. It makes full use of the computing resources of graphic workstation processing nodes while ensuring the processing node of

the cluster is not overloaded. An ideal outcome has obtained from the practical testing.

References

[1]. LI Kun,WANG. Baijie. Research on Load Balancing of Web·server System and Comparison of Algorithms [J]. Computer and Modernization.2009,(8):7-15.

[2]. WangliPing. Research and Improvement of the Load Balancing Technology based on Nginx Server Cluster [D]. ShanDong:ShanDong University,2015.

[3]. Gao Ang,Mu De-jun,Hu Yan-su. Differentiated Service and Load Balancing in Web Cluster [J]. Journal of Electronics & Information Technology.2011,(3):555-562.

[4]. LiuKun. Reaearch on Load_balanced Strategy in Cloud Computing [D].Ji Lin: JiLin University,2016.

[5]. PengYuhang,Wujijing,Shenyue .Distributed Computing ModeI and Supporting Technologies for the Dynamic Allocation of Internet Resources[J]. Journal of Computer Research and Development.2011,48(9):1580-1588.

[6]. RONG Hang.Improved Dynamic Load—balancing Algorithm in the Cluster [J]. Wireless Communication Technology.2015,(3):34-37.