Research Journal of Applied Sciences, Engineering and Technology 5(22): 5267-5271, 2013
DOI:10.19026/rjaset.5.4275
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 22, 2012 Accepted: December 28, 2012 Published: May 25, 2013

Research Article

Based on the MapReduce Model for Data-intensive Computing of Energy Scheduling Algorithm Strategy

[1, 2]Yuqiang Sun, [1]Xin Gao, [1]Huanhuan Cai, [1]Xianmei Chang and [1]Lei Li
[1]International Institute of Ubiquitous Computing, Chang Zhou University, Chang Zhou 213164, China
[2]School of Computer and Information Technology, Henan Normal University, Hennan, Xinxiang 453007, China

Abstract: In this study, based on the consideration of energy consumption, we take to improve the strategy of the MapReduce job scheduling algorithm, in order to reduce the average response time for task scheduling of interactive jobs in the network. In accordance with the job priority grouping to adjust the scheduling task response time which can reduce the impact of network congestion, with good results that increase the throughput of the system transferring data and computing power.

Keywords: Data-intensive, MapReduce, scheduling algorithm

INTRODUCTION

The development and application of Internet technology make the need to deal with the amount of data show the combinatorial explosion situation. Based on network data-intensive computing about massive data processing, require mass storage, high-performance computing platforms to achieve this process. Nevertheless, data-intensive computing is usually not available locally to provide services, so it is an effective and natural way that network services provide the application interface. The network has become the carrier of a large-scale data processing (Li *et al*., 2011).

However, it is different from the traditional high performance computing, users' application requirement on huge amounts of data calculation in the network may include from data acquisition to pre-treatment and then the whole process of data analysis. There are some complex processes, therefore service interface for data-intensive computing applications based on these network services interaction functions. In the network environment, a series of interactive behavior like storing, calculation and retrieve, energy consumption of them is particularly prominent, to some extent. Energy performance indicators are demonstrated between each data node in the network, including utilization of network bandwidth, response time/waiting time of the running job and throughput of the system (Laura *et al*., 2010; Xiao-Xia and Zhong-He, 2011; Chen *et al*., 2009a). Energy problem is becoming a hot issue of network data center. Therefore, the traditional centralized or small-scale distributed parallel technology, which takes corresponding storage, retrieval and computing technology should also change, so that it may be in the fast growing amount of data to meet the requirements of response time, throughput and scalability. MapReduce (Chen and Qian-Ni, 2009b) was initially developed at Google as a platform to parallel processing of large datasets. It is suitable for large-scale computer clusters to analysis processing with data-centric. Likewise, it is a core computing model in the cloud currently. Based on MapReduce energy consumption, we'll study this point of view from job scheduling (Matei *et al*., 2009).

In this study, based on the consideration of energy consumption, we take to improve the strategy of the MapReduce job scheduling algorithm, in order to reduce the average response time for task scheduling of interactive jobs in the network. In accordance with the job priority grouping to adjust the scheduling task response time which can reduce the impact of network congestion, with good results that increase the throughput of the system transferring data and computing power.

MAPREDUCE PARALLEL MODEL FRAMEWORK

MapReduce is a calculational model which can support a very large scale keys/ value pairs; it hides the complex distributed processing details of parallelization, fault tolerance, data distribution, load balancing, etc. Its processing mechanism can be exempted from the limitations of the data scale, with

Corresponding Author: Yuqiang Sun, International Institute of Ubiquitous Computing, Chang Zhou University, Chang Zhou 213164, China

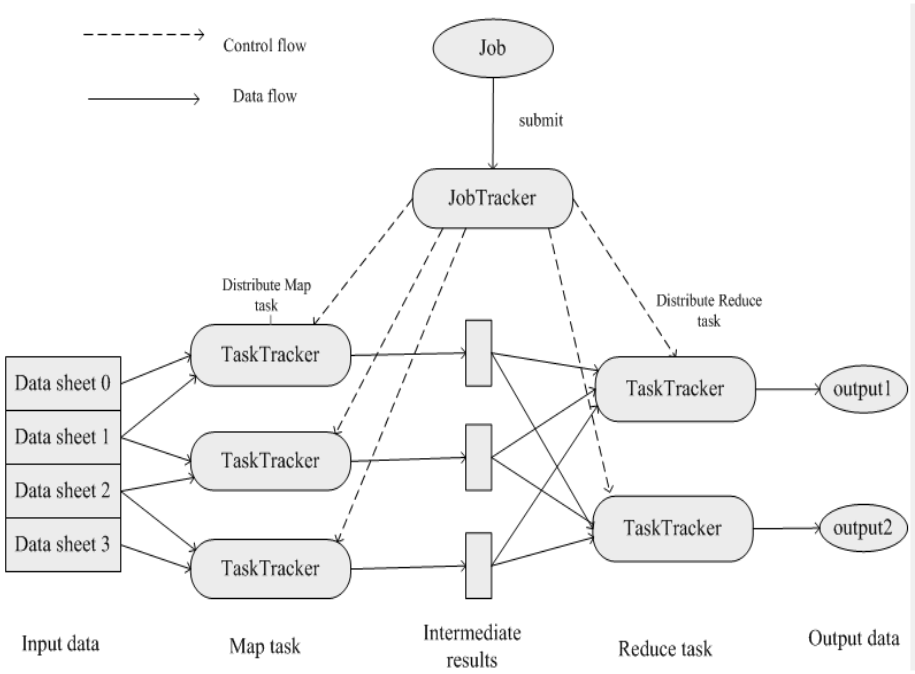


Fig. 1: MapReduce basic framework

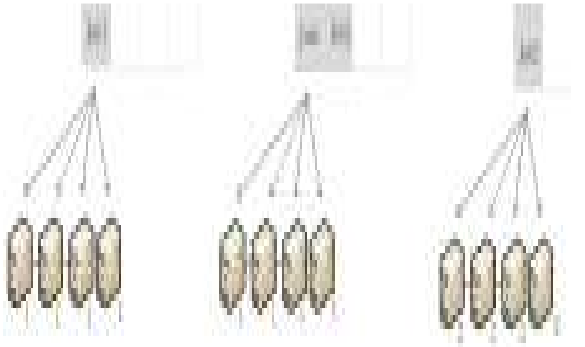
superior scalability. Users 'needs are handled by the distributed parallel computing on massive data sets on clusters of computers. In a MapReduce framework, a calculation request of every user as a job, the main server JobTracker is responsible for coordinating the operation of this job. Each job will be split into many tasks, then the tasks are assigned to different slave servers TaskTracker by JobTracker, TaskTracker is responsible for carrying out specific tasks. Different TaskTrackers execute Map tasks and Reduce tasks assigned. At its core, MapReduce has two user-defined functions. The Map function takes in a key-value pair and generates a set of intermediate key-value pairs. The Reduce function takes in all intermediate pairs associated with a particular key and emits a final set of key-value pairs. Figure 1 shows the basic framework of MapReduce model from the perspective of scheduling.

Based on MapReduce implementation mechanism, when some data that the user requires are running in the network, the user submits the job to the scheduling system, each job contains a series of tasks and scheduling system will allocate these tasks on more than one machine available in a cluster. However, MapReduce splits data to achieve parallel computing on each node, there are some restrictions. First, the data cannot be infinitely spilt, if each data is spilt too small, then the calculated energy costs by MapReduce framework itself will be relatively larger. And if the cluster nodes become larger, the network will overload, because the network bandwidth between the racks is far less than the total bandwidth of each rack local, it will cause the energy overhead of communication between nodes increase. Therefore, in order to fully use network bandwidth resources, allocate resources reasonably in the job scheduling process, we propose the following optimization strategy to the schedule algorithm based on energy performance.

MAPREDUCE JOB SCHEDULING ALGORITHM

MapReduce supports data-intensive computing process. Between the stages of Map task and Reduce task, the whole implementation process is serial synchronous, Reduce operation do not begin until the last Map operation finished. This interdependent relationship will lead to resource utilization and idle too much in the implementation of mission operations. If a long-running job gets many Reduce slots on the machine, it will release them only after the end of Map stage, while other small workload in its reserved slot has been in a state of "hunger" and lack of a chance to run. Like this the process will cause a large number of idle workload and long queuings delay, resulting in waste of resources in time and space. It is so important to schedule effectively for data-intensive cluster computing because of the shared cluster resources and network bandwidth are very expensive, as well as difficult to move. Now data integrated scheduling reflects the important value.

For in all the MapReduce computation to achieve, we choose an open source platform Hadoop. In Hadoop, job scheduling processes are implemented to allocate tasks to the corresponding TaskTracker by JobTraeker. Multiple tasks are running at the same



(a) Job1 enter the queue (b) Job1 run, Job2 wait (c) Job2 begin to run

Fig. 2: FIFO algorithm

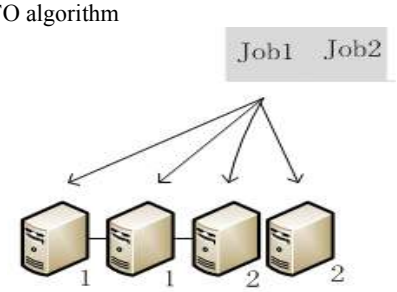


Fig. 3: Job1 and Job 2 run together, share cluster resources

time, the order in which jobs are executed and the allocation of computing resources will affect the performance for Hadoop platform and the energy efficiency of system resources. In data processing, job scheduling algorithms are generally used as follows.

The first scheduling algorithm is a default scheduling algorithm, FIFO. FIFO algorithm is with a priority and all the users' jobs are submitted to a queue only. Scan the entire job queue in accordance with the priority level and submitted in chronological order; choose one job to implement tasks allocation, as shown in Fig. 2. FIFO algorithm is relatively simple with less scheduling overhead of the cluster. However, it has its limitations that the algorithm ignores jobs demand difference among different users. In the case of large operations, small jobs' response times are relatively poor. For massive data processing, it is inevitable to carry out interactive tasks in different applications operating. To a certain extent, waste the resource of the system cluster, make that parts of the resource use frequently, while another is in an idle state during period of time.

Another common scheduling algorithm is Fair Scheduler. This algorithm could access to cluster resources averagely as time goes by. When a single job is running, it will get the resource of the entire cluster. When new users submit jobs in the queue, the system will assign the task slots to these new users, allowing each user to share resources evenly. By default, each user has an independent resource pool, which means that all running tasks can get the same amount of the shared resources, regardless of whether they submit the number of jobs. The advantage of this algorithm is shorter time-consuming jobs can be completed within a reasonable period of time; meanwhile longer time-consuming job will not be chronically hungry. Shown in Fig. 3. However, the amount of job queue exceeds the system carrying capacity, so that it does not improve the total throughput of the system. Instead, it will cause working nodes in the system are in a state of overload. The quantity of tasks which are allocated by TaskTrackers is oversize, or the numbers of running multiple jobs are excessive at one time, there may be overload. As a result, this will reduce the job response time and system processing capabilities. For batch tasks have been running with computing resources which are not completed, the server will not be able to recycle these resources, so it will make some jobs for prolonged lack of response. In conclusion, the algorithm does not take into account the actual load of the nodes, although scheduling the jobs proportional, it leads to the nodes load are imbalance.

The third scheduling algorithm is Capacity Scheduling. In the algorithm it defines multiple queues. Assign a certain amount of system capacity for each queue by file configuration; a task will be randomly placed into a queue. Limit the percentage of the resources available to jobs submitted by the same user, so that belong to a user's job resource cannot be exclusive. Idle resources can be dynamically assigned to the heavy loads of the queues. However, queue settings and queue selection cannot be free to choose, users need to know a lot of system information.

OPTIMAL SCHEDULING ALGORITHM BASED ON ENERGY CONSIDERATIONS

Analysis of the energy index: Based on the above scheduling algorithms process, none of the scheduling algorithms consider energy consumption in the transmission and scheduling process. Running calculation and storage problems for mass of data are growing fast. More and more alternate switching execution scheduling problems between long jobs and small jobs. In this study, it is a prominent place that combines the energy performance and job scheduling algorithms to optimize the better algorithm. Have a means of maximizing the utilization of system resources and throughput; meanwhile reduce latency and response time of the jobs.

There are a lot of inputs and outputs data needs of users during transmission, so the network bandwidth is the scarcest resource in the system. Task scheduling and allocation cannot be concentrated in the same node; scheduling algorithms need to consider the data input/read. Data cluster resource is shared and the nodes transfer data between two-way communications.

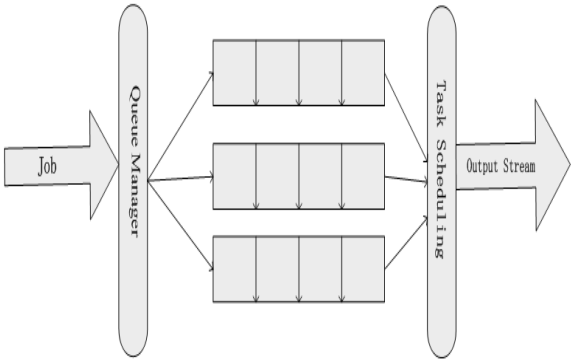


Fig. 4: Algorithm chart

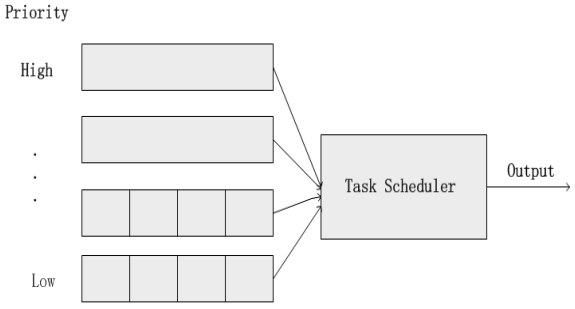


Fig. 5: Scheduling algorithm process

A large number of data transfer or pending transfer will take up a lot of network resources, scheduling inappropriate will affect the response time of a single task, ultimately affect the overall system throughput and the total response time.

The task response time is that the completion time which a single task runs minus the time which the tasks submitted to the system. It is a two-tuple, described as T = <U, V>, U represents delay factor sets, V = {job segmentation number N, job length J, tasks number I}. It reflects the ability of interaction between MapReduce model and user. The shorter of the response time, the stronger of the system interacted with the users. The total response time reflects the computing capacity and throughput of the entire system, a shorter response time indicates that this model can use a small amount of computers and network resources to accomplish the same task in a large number of parallel executions of tasks. Throughput is the number of jobs which are processed by cluster system in unit time. The more of throughput, the more jobs processed in unit time.

Algorithm process improvement: In this study, the strategy adopted by combination of these algorithms is the use of priority, as well as weights and equitable use of shared resources. Each queue is seen as a separate resource pool, the system allocates shared resources. Reduce the transmission of network data as a starting point, for decentralized demand of jobs on different nodes. On the basis of Fair Scheduler, use the priority grouping strategies to reduce the response time of interactive jobs in the network. Above all, segregate jobs into segments, according to segmentation to determine priority. Priority of each task depends on the length of specific job, define two priorities. 1 indicates a high priority and 0 indicates a low priority. When master server sends tasks to TaskTracker, the queue manager carries out jobs segmentation, form task groups. If job segmentation is only composed of a single segmentation task, it displays as 1, identified as high priority. If form multiple segments task, it displays as 0, identified as low priority. The algorithm structure diagram and process diagram as follows. As shown in Fig. 4 and 5.

The pseudo codes of the corresponding job scheduling are as below:

```
Job scheduling begin
    Initialize job queue JobList
    Traverse all the queues, define the queue priority
{int high-priority-queue;
    int low-priority-queue;
    The jobs are submitted to the queue manager, job
segmentation;
    int t[N]; //Define the storage of job queues, N is the
number of job segments
    t< =  total number of job queue
    if N = 1
      {Jobs are mapped to the high priority queue;
      put this job into the queue position t [1] and put
job segmentation into high-priority-queue team;
      }
      else
        {Jobs are mapped to the low priority queue;
      for (i = 1; i<N; i++)
        {put this job into the queue position t [i], put
    job segmentation into low-priority-queue team;}
    }
}
```

That is, when there is only one group task, the priority is 1 and then adds it to the end of the high priority queue; otherwise it is added to the end of the low priority queue. When TaskTracker is idle, the queue manager always selects a first packet from a high priority queue to transmit. When the high priority queue is empty, select another first packet from a low priority queue to send. The algorithm sets a dynamic priority, neither exists a certain priority queue that is progressed in a long time, nor job overload. Timing to a point in the low priority queue, a packet will be removed from it and added to the end of the high-priority for processing. In the scheduling process, the whole system has always maintained a balance in each node running.

CONCLUSION

Optimization of job scheduling algorithm based on the MapReduce energy aware is from the macro aspects to execute performance improvement. When multiple

tasks are running at the same time, adjust the order of tasks to run, as well as the allocation of computing resources. Make a distinction between the uses' job requirements, meanwhile ensure a certain degree of fairness that no user for a long time to occupy most of the resources system. The optimization algorithm also reduces the waiting time of majority tasks and condenses jobs response time, thus it improves the work efficiency of MapReduce. The next step is from MapReduce internal proceeded to optimize the operating efficiency of a single job. We will focus on the task scheduling strategy in a distributed system; make use of the reciprocity between nodes to select the most appropriate node to execute the corresponding tasks. So as to reduce the amount of data transmission system and decrease the overhead of network bandwidth, consequently achieve the goals of improving overall system performance.

ACKNOWLEDGMENT

Supported by The project of general office of Broadcasting and Television (GD10101) and Natural Science Fund in JiangSu (BK2009535) and Natural Science Fund in ZheJiang(Y1100314).

REFERENCES

Chen, Q. and D. Qian-Ni, 2009b. Self-adaptive MapReduce scheduling under heterogeneous environment [J]. Comp. Eng. Sci., 31(A1).

Chen, Y., L. Keys and R.H. Katz, 2009a. Towards energy efficient MapReduce [J]. Technical Report No. UCB/EECS-2009-109.

Laura, K., R. Suzanne and D.D. John, 2010. The search for energy-efficient building blocks for the data center [J]. Proceeding of the International Conference on Computer Architecture. Springer-Verlag, Berlin, pp: 172-182.

Li, M., G.H. Xu and Y. Ji, 2011. Application research on MapReduce programming model in network-I/O-intensive programs [J]. Appl. Res. Comp., 29(9): 3372-3374.

Matei, Z., B. Dhruba, S.S. Joydeep, E. Khaled, S. Scott and S. Ion, 2009. Job scheduling for multi-user mapreduce clusters [J]. Technical Report No. UCB/EECS-2009-55.

Xiao-Xia, L. and Z. Zhong-He, 2011. The research for optimization of the main task scheduling algorithm in cloud computing [J]. Comp. Technol. Automat., 30(4): 108-110.