International Journal of Grid and Distributed Computing
Vol. 10, No. 2 (2017), pp.1-12
http://dx.doi.org/10.14257/ijgdc.2017.10.2.01

Research on Hadoop Job Scheduling Based on an Improved Genetic Algorithm

Tao Xue*, Xiaoqing You and Minglei Yan

College of Computer Science, Xi'an Polytechnic University, Xi'an 710048, China
**xt73@163.com*

Abstract

Hadoop job scheduling is an important influence factor in the performance of the Hadoop platform. Due to its characteristics, genetic algorithm (GA) has natural parallel advantages in job scheduling. But the scheduling based on the traditional simple genetic algorithm (SGA), in the aspect of the average execution time of job, the convergence speed and the selection of the optimal solution, is lack of consideration, which cannot make full expression of its advantages. We propose an improved genetic algorithm, which adopts the reserved strategy of optimal solution to speed up the convergence speed. The experimental results indicate that the proposed algorithm has a certain improvement on the scheduling efficiency and the performance of platform.

Keywords*: Hadoop, Genetic Algorithm, CHC, The optimal solution*

1. Introduction

Apache Hadoop [1] is an open source distributed platform, mainly composed of two core projects, namely distributed computing framework Map/Reduce [2] and distributed file storage system HDFS [3]. With the rise of big data [4], Hadoop, with the advantages of its own, has attracted a lot of IT companies. After years of research and development, Hadoop has been large-scale used in the field of data processing, but also get the wide attention of the academia.

In the study of Hadoop, job scheduling is always the key point of the research, because it is an important link of impacting the performance of the platform. Hadoop platform itself has three kinds of scheduling methods: Job Queue TaskScheduler based on FIFO mechanism, Fair Scheduler and Capacity Scheduler. In addition, there are many scheduling policies for different environment and different job, such as dynamic scheduling based on heterogeneous load, real-time scheduling, etc. Processing mode of job in Hadoop is to split a big job in small tasks, then different tasks are assigned to each task engine to execute. This is a kind of distributed programming ideas from Google, and Map/Reduce programming model in Hadoop is an open source implementation of the idea. It is very suitable for processing large data set. Its data processing consists of five stages, InputFormat, Map, Partition, Reduce, and OutputFormat. And reduce stage is depending on Map stage. The data processing of Map/Reduce is as follows.

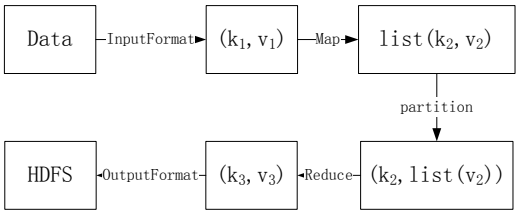

Figure 1. The Data Processing of Map/Reduce

ISSN: 2005-4262 IJGDC

The purpose of job scheduling is how to allocate different tasks to suitable task machine, to make job completion time shorter, and at the same time make utilization of task machine higher. In the process of task allocation, many constraints need to be comprehensively considered, including the network bandwidth, load of each node, failure, performance, data locality, etc. So, this is a kind of combinatorial optimization problem, also a kind of NP problem. Genetic algorithm has intelligent features of self-organizing, self-learning and self-adaption, and parallelism of itself. It is a powerful tool in solving the problems of combinatorial optimization, the job scheduling, etc. Therefore, the combination of genetic algorithm and job scheduling can cope with such difficulties and seek a satisfactory solution. In this article, based on an improved genetic algorithm, which is an improvement of the traditional simple genetic algorithm, we comprehensively consider the optimal solution retention strategy and the constraint conditions that mentioned above, simulating and obtaining approximate optimal solution as soon as possible. In genetic algorithm, this Map/Reduce process will be considered in the form of an overall. So it can make scheduling approach more concise, without too much focusing on the processing of task.

2. Related Work

The process of Hadoop job scheduling is shown in Figure 2. The Client submits a job to the JobTracker and notifies the task scheduler. When TaskTracker asks for new task to the JobTracker, TaskScheduler, with the task request information from JobTracker, return the task list. Finally TaskTracker launches assigned task.

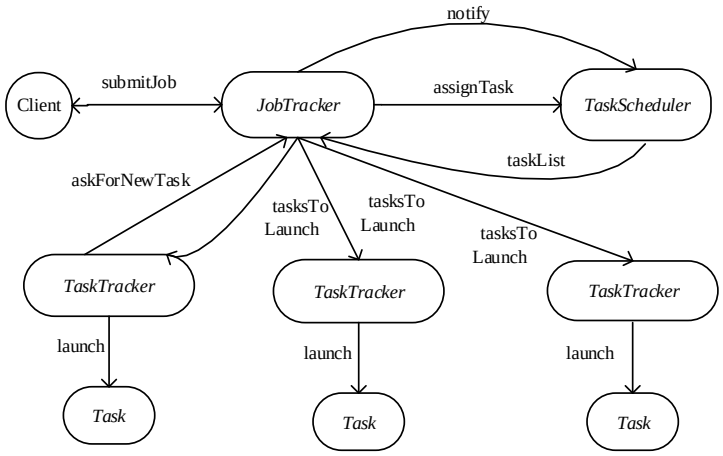

Figure 2. The Process of Hadoop Job Scheduling

The function of job scheduling is to assign computing resources of different TaskTracker nodes to the tasks in cluster by certain strategies. In the Hadoop job scheduling, itself contains three kinds of schedulers, but it is difficult to meet user's quickly changing needs. In Hadoop, task scheduler is designed to be a pluggable module, and it can be flexible to cope with different applications. So it has appeared many new scheduling algorithms and the improved scheduling algorithms. In general, they can be divided into two classes: the non-intelligent algorithm and intelligent algorithm.

Non-intelligent algorithm is mostly aimed at the bottleneck factors for innovation and improvement in the process of scheduling. The algorithm is simple, and has more research results, but the configuration is complex. Intelligent algorithm is simulation and application of biological activity or neural behavior. So, the algorithm is long process, high complexity and less research results, but the configuration cuts, and it is more efficient than the non-intelligent algorithm in the huge amounts of data.

In the non-intelligent algorithms, LATE scheduling algorithm [5], [6] is based on task speculation mechanism. It is more likely to find "dragger" task, launches the backup task ahead of time, and effectively shortens the execution time in a heterogeneous cluster environment. SAMR algorithm [7] is based on the LATE algorithm, using dynamic way to compute task progress, and it improves the accuracy of scheduling information. Delay scheduling algorithm [8] is a scheduling method that it ignores fairness in order to improve the data locality of task. Dynamic priority scheduling algorithm [9] allows the user to dynamically adjust priority of job and allocates task proportionally.

In intelligent algorithm, the simulated annealing algorithm [10] uses the memory function to choose the optimal solution, avoiding the local optimal solution. Artificial fish swarm algorithm [11] codes tasks to be assigned by random key approach, and improves the performance of the job scheduling algorithm in heterogeneous environment. Job scheduling based on genetic algorithm [12] uses parallelism of algorithm and genetic operation to search the optimal solution.

In Hadoop, job scheduling belongs to the category of the combinatorial optimization, and is a multi-objective problem, also a typical NP problem. Intelligent algorithm usually has a good performance on the NP problems. Genetic algorithm is a kind of excellent intelligent algorithm, and meets the optimality principle. Its solving way is simulating population evolution mechanism, generational circulation, finally retaining the optimal solution of set. This is the basic theory of a Hadoop job scheduling using genetic algorithm. According to this, in this paper we propose a Hadoop job scheduling algorithm based on CHC genetic algorithm.

3. The Problem of Job Scheduling Based on Traditional Genetic Algorithm

The process of Simple Genetic Algorithm(SGA) is as follows.

Algorithm1: Simple Genetic Algorithm

```
define global variable;
void initPopulation();
void calInitFit();
void calFit();
void generation(){
    int select();
     int crossover();
     void mutation();
     }
main()
```

The SGA algorithm is making simply parallel processing for task, screening the optimal solution only by the genetic operations. So there are the following questions, 1) The evolutionary generations is long, and convergence speed is slow. 2) For the optimal solution is not to do special processing, it may be lost in the process of evolution. 3) The SGA algorithm is only considering the total completion time of job, ignoring the differences between big jobs and small jobs, lack of consideration for average completion time of job.

In view of the above three drawbacks, in this paper we propose a job scheduling method based on improved genetic algorithm. The algorithm sacrifices simplicity of SGA algorithm in exchange for a better parallelism genetic effect [13], at the same time emphasizing the optimal solution retention strategy and accelerating the convergence speed.

4. Job Scheduling Based on Improved Genetic Algorithm

4.1. Related Definition

The goal of improved genetic algorithm proposed in this paper is that the total completion time is shortened and the average completion time is short as far as possible. At the same time, giving consideration to the optimal solution, and achieving rapid convergence.

Before job is scheduled, *JobTracker* node calculates the average task completion time by the table 1 and table 2. Suppose there are m Jobs and n *TaskTracker* nodes. In which l_i represents the splitting number of each job, b_i represents the maximum number of *TaskTracker* that is available for each job. By default every b_i value is equal. When the failed node arising or dynamically adding or removing nodes in the cluster, it will change, $1 \leqslant i \leqslant m$. S_i represents the number of parallel slots in $TaskTracker_i$ node. It can be dynamically read from the configuration file and dynamically changes as the stage of task, which is Map process or Reduce process, $1 \leqslant i \leqslant n$.

Table 1. Job Table

Job	Splitting number	Maximum number of TaskTracker
Job_1	l_1	b_1
Job_2	l_2	b_2
…	…	…
Job_m	l_m	b_m

Table2. The Configuration of Tasktracker

TaskTracker	Number of parallel slots
TaskTracker1	*S1*
TaskTracker2	*S2*
……	……
TaskTrackern	*Sn*

JobTracker calculates the data of table 1 and table 2, and obtains the average execution time TT of Job_m on $TaskTracker_n$.

Table 3. The Average Execution Time of Task

Job	TT_1	TT_2	…	TT_n
Job_1	$\bar{t}_{11}$	$\bar{t}_{12}$	…	$\bar{t}_{1n}$
Job_2	$\bar{t}_{21}$	$\bar{t}_{22}$	…	$\bar{t}_{2n}$
…	…	…		…
Job_m	$\bar{t}_{m1}$	$\bar{t}_{m2}$	…	$\bar{t}_{mn}$

Among them, the scheduling order of different jobs on each task machine is allocated in the following way. Assume that there are three jobs, the allocation of subtask on $TaskTracker_1$ is as the following,

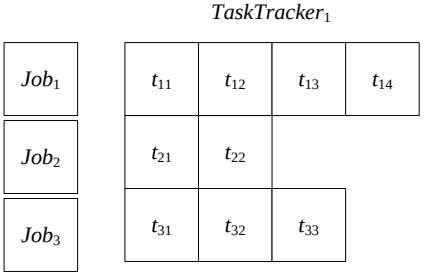

Figure 3. The Example of Job Distribution

Considering the differences between different jobs and the fairness of scheduling, *TaskTracker* complies with the order $\{t_{11},t_{21},t_{31},t_{12},t_{22},t_{32},t_{13},t_{33},t_{14}\}$ to schedule jobs. So that we can ensure a timely response of small job and execution time of big job [14].

Because the processing speed of the machines is different, the type of tasks is different, and the tasks group by parallel slots, so the sequence of completion of task is uncertain. It is more accurate to solve completion time of task with waiting time adding execution time, but the uncertainty of the sequence of completion can also led to the uncertainty of solution. So choosing the average execution time is an effective method. In the process of scheduling, to get the average task execution time by the final completion moment of each job minus the starting time, divided by the norm of the task.

$$\bar{t}_{mn} = \frac{t'_{mn} - t_0}{\|T_{mn}\|} \tag{1}$$

Among them, $\|T_{mn}\|$ represents the norm of the subtask set Job_m of $TaskTracker_n$. t'_{mn} represents completion time of final task in subtask set Job_m of $TaskTracker_n$. t_0 represents the beginning execution time of the first task in $TaskTracker_n$. By the above definition, we can get objective function and the corresponding constraints according to the scheduling goal.

4.2. The Mathematical Model of Improved Genetic Algorithm

4.2.1. Objective Function

The proposed algorithm is based on the total completion time and the average completion time, the mathematical equation as below.

$$\begin{cases} f(i,j) = \min\left[\max_{j=1}^{v}\sum_{i=1}^{u} t_{T_{ij}}\right] \\ f(i) = \min_{i=1}^{u}\left[\max_{j=1}^{v} \bar{t}_{ij}\right] \end{cases} \tag{2}$$

Where $t_{T_{ij}}$ represents execution time of subtask set of job u on $TaskTracker_v$.

4.2.2. Constraint Condition

(1) The total tasks of Job_i assigned to all *TaskTracker* nodes cannot exceed the number of splitting l_i.

$$\sum_{i=1}^{m} \|Job_i\| \le l_i \tag{3}$$

(2) The concurrent tasks of each *TaskTracker* cannot exceed setting value, at the same time, the total number of concurrent cannot exceed 90% of the total slots in cluster. A part of resources are set aside, in the node failure or other unexpected circumstances supplied for *TaskTracker*.

$$\begin{cases} \left\| TaskTrac\ker_j \right\| \le s_j \\ \sum_{j=1}^{n} \left\| TaskTrac\ker_j \right\| \le 0.9 * \sum_{j=1}^{n} s_j \end{cases} \quad (4)$$

4.3. Improved Genetic Algorithm

4.3.1. Encoding and Decoding

In this paper, we adopt a direct encoding way that it is distributed between *Job-TaskTracker* according to the task set of job. A chromosome is expressed in the form of matrix, $Y = [T_{ij}], i = 1,\cdots,m, j = 1,\cdots,n$, Which i represents the number of job involved in the scheduling, j represents the number of *TaskTracker* node in cluster. T_{ij} is a subtask set, represents allocated subtask set of job i on node j.

Table 4. Coding and Decoding Table

Job	TT_1	TT_2	…	TT_v
Job_1	T_{11}	T_{12}	…	T_{1v}
Job_2	T_{21}	T_{22}	…	T_{2v}
…	…	…		…
Job_u	T_{u1}	T_{u2}	…	T_{uv}

Which T_{uv} is subset of taskset $\{t_{u1},t_{u2},\ldots,t_{un}\}$ of job Job_u [15]. There are constraints on the job as shown in equation 5.

$$\begin{cases} \bigcup_{i=1}^{v} T_{uv} = Job_u \\ \bigcap_{i=1}^{v} T_{uv} = \varnothing \end{cases} \quad (5)$$

4.3.2. The Generation of Initial Population

As there is no any prior conditions about the allocation of the job, so we adopt the randomly generated method, and to satisfy the constraint conditions. Here, the population size is 100.

The generation step of initial population is as follows.

Step1: Make all elements in the matrix **Y** empty, namely $T_{ij} = \varnothing$.

Step 2: In order to keep the balance of assigned tasks, each *TaskTracker* divides task according to its own parallel number of slots in the proportion of total number of slots. Firstly to determine the size of the set of tasks, and then randomly assign task according to the size.

Step 3: Checking *Job* and *TaskTracker* whether meet the constraint (3) (4) (5).

Step 4: Looping execute the first two steps until the population size.

4.3.3. Fitness Function

Fitness function decides the merits of the individual, so that more superior individual is selected in the next round of evolution.

Therefore it is critical to select fitness function. And it is closely related with the convergence speed, if you can find out the optimal solution. Here directly to the objective function as the fitness function, namely, $F(x) = f(x)$.

4.3.4. Genetic Operation

(1) Individual selection

Firstly, individual selection is based on the optimum solution, namely the locality of task. In the algorithm it is as the first factor to consider. Here individual selection is based on sorting method. Firstly for each round scheduling of individual, to calculate the objective function value. Then ascending sort according to the scheduling result, the corresponding individual of the larger objective function value is eliminated (elimination rate 10%). So that we can eliminate bad genes, as soon as possible to ensure the rapid concentration of good genes, to speed up the convergence. Figure 4 and Figure 5 is a scheduling of two parent individuals.

	$TaskTracker_1$	$TaskTracker_2$	$TaskTracker_3$	...	$TaskTracker_{n-1}$	$TaskTracker_n$
Job_1	t_{11}, t_{14}, t_{17}	t_{12}, t_{16}, t_{19}, t_{1a}	t_{13}, t_{18}	...	t_{1b}, t_{1c}, t_{1d}	t_{15}, t_{1e}, t_{1f}
Job_2	t_{21}, t_{24}, t_{25}	t_{22}	t_{23}, t_{29}, t_{2c}	...	t_{2a}, t_{27}	t_{26}, t_{28}, t_{2b}
Job_3	t_{33}, t_{34}	t_{31}, t_{37}	t_{32}	...	t_{36}	t_{35}
Job_4	t_{47}	t_{43}, t_{44}		...	t_{41}, t_{42}, t_{46}	t_{35}, t_{38}
Job_5		t_{51}, t_{54}, t_{57}	t_{58}, t_{59}	...	t_{52}, t_{53}, t_{56}, t_{5a}	t_{55}
Job_6	t_{61}, t_{67}		t_{64}, t_{66}	...		t_{62}, t_{63}, t_{65}

1 2 3

Figure 4. Parent1

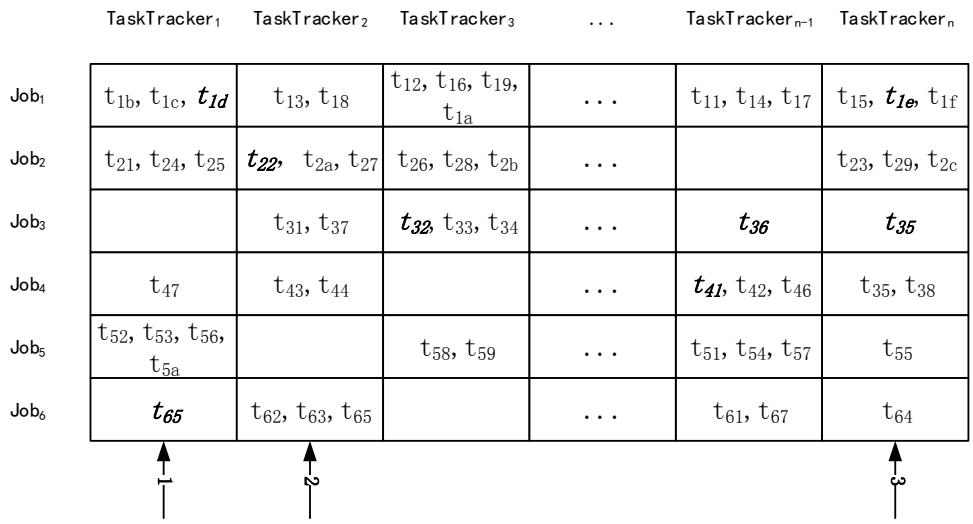

Figure 5. Parent2

(2) Crossover operation

Due to the crossover operation has constraint conditions on Job rows and TaskTracker columns, and to ensure retention strategy of the optimal solution. Therefore, we use a method of multipoint crossover. Crossover columns are randomly selected (see Figure 4, Figure 5 mark 1, 2, 3 column), and these columns meet the following conditions: The column including locality task does not participate in cross; The column removing locality node, participate in cross columns contain equal tasks, to calculate by row; If there is no such column, skip this round of crossover; At the same time, the order of the job in a row is strictly determined, remaining the same in the process of crossover.

(3) Mutation operation

Individual matrix is used for mutation operation, randomly selecting two column to exchange (see Figure 6), which the column of data locality will not be involved.

So that the optimal solution is retained for each generation, and obtains a quick convergence.

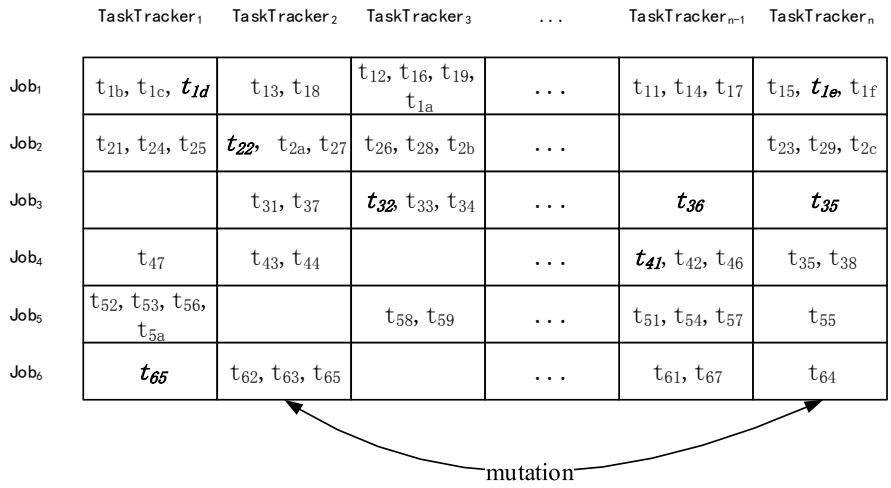

Figure 6. The Process of Mutation Operation

4.3.5. The Determination of The Optimal Solution

After the genetic operation is completed, all individual of father and son will be a new species. Back to the stage of fitness testing and sorting for the operation of the population, if the evolution of the population does not reach the generations, the algorithm continue the genetic operation. Otherwise, the algorithm selects first element in the scheduling queue into the decoding stage, namely the T_{ij} set is one-time assigned to the corresponding *TaskTracker* node.

4.4. The Overall Process of Job Scheduling by Improved Genetic Algorithm

The process of the improved genetic algorithm are almost consistent with the SGA algorithm. The main difference lies in the selection and retention strategy of the optimal solution in every step of evolutionary process. The overall process of job scheduling by improved genetic algorithm is as follows.

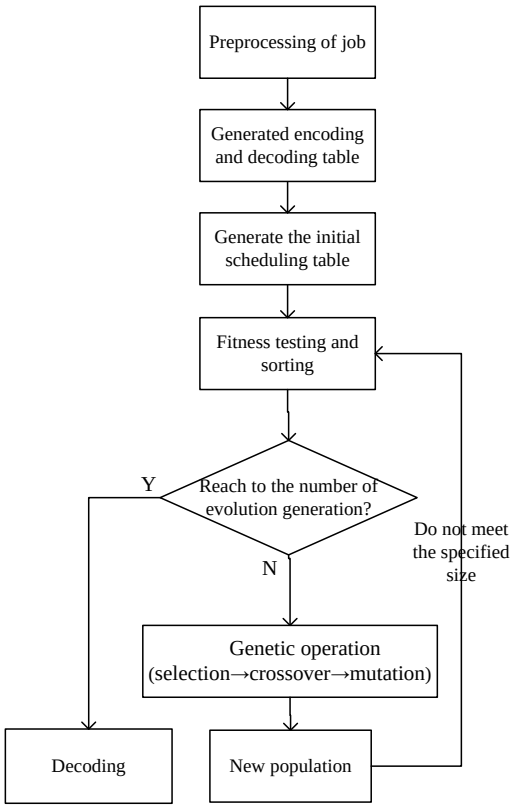

Figure 7. The Overall Process of Job Scheduling by Improved Genetic Algorithm

5. Experiment

There are four different configurations in cluster with a gigabit switch connected, as shown in table 5, and this ensures the heterogeneity of the cluster [16],[17], simulating the evolution of the individual and the actual cluster environment. The whole cluster consists of a *JobTracker* node and 13 *TaskTracker* nodes. The configuration of each node are shown in table 5. The software version are *Ubuntu* 12.04 operating system and Hadoop 1.0.4. Choosing *RandomWriter* to randomly generate test data, and benchmark test program of using *TeraSort* to sort.

Table 5. The Experiment Environment

Cluster	Configuration	Number	Role
VM_1	MM:4 GB CPU: 2.00 GHz×4	1	*JobTracker*
VM_1	MM:2 GB CPU: 2.00 GHz×2	4	*TaskTracker*
VM_2	MM:2 GB CPU: 2.68 GHz×2	5	*TaskTracker*
Desktop	MM:2 GB CPU: 3.00 GHz×2	3	*TaskTracker*
Laptop	MM:2 GB MM:4 GB	1	*TaskTracker*

This experiment mainly tests the performance of algorithm from two aspects, total completion time and the average completion time of job. The test results are shown in Figure 8 and Figure 9. From Figure 8, we can see completion time of the job falling faster in the beginning. This is due to the bad gene is knocked out, good genes quickly gathered process. With the increase of evolution generations, the pace of decline is in gradually reducing. This is due to the accumulation of the good genes, and algorithm increase rate will slow down gradually. Finally at about the 40th generation basically achieve stability. This is due to the reservation of optimal solution, to speed up the accumulation of good genes. To greatly reduce evolution generations of reaching the optimal solution.

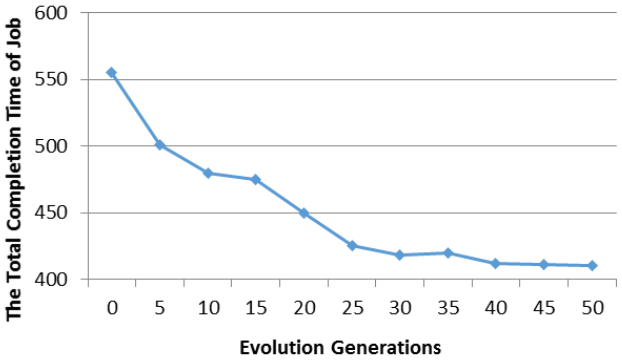

Figure 8. The Total Completion Time of Job

From Figure 9, we can see that the decreasing trend of the average completion time is basically as well as the total completion time. Evolved to around 20 generations it has a small recovery. It is because of the effect of many factors, such as the uncontrollable variation, the performance of the machine, the network situation, and many other conditions. But this is only a small probability event. The final trend is still down and stable.

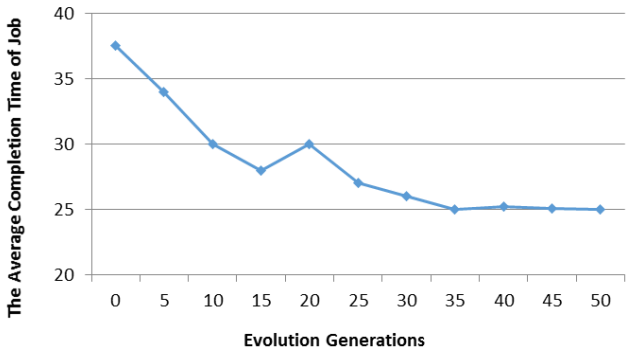

Figure 9. The Average Completion Time of Job

6. Conclusion

In this paper, on the basis of simple genetic algorithm, we propose a Hadoop job scheduling algorithm based on improved genetic algorithm. With reserved strategy of the optimal solution, setting different constraint detection, speeding up the convergence speed of the algorithm, and the job in the two aspects of total completion time and the average completion time has certain promotion. At the same time, the resource utilization of *TaskTracker* node is also improved. But the weight coefficient in the constraint condition is set by the experience value, not necessarily applying to the heterogeneous cluster. Therefore, in the next step we will research the problem of the dynamic allocation of weight coefficient.

References

[1] Apache Hadoop [Online]. http://hadoop.apache.org/, **(2016)**.
[2] L. Jian-jiang, C. Jian and W. Dan, "Survey of MapReduce Parallel Programming Model", Acta Electronica Sinica, vol. 39, no. 11, **(2011)**, pp. 2635-2642.
[3] F. Azzedin, "Towards a scalable HDFS architecture", Proceedings of International Conference on Collaboration Technologies and Systems (CTS), **(2013)**; San Diego.
[4] S. Dilpreet and C. K. Reddy, "A survey on platforms for big data analytics", Journal of big data 2.1, **(2015)**, pp. 1-20.
[5] M. Zaharia, A. Konwinski and A. Joseph, "Improving MapReduce Performance in Heterogeneous Environments", Proceedings of the 8th Symposium on Operating Systems Design and Implementation, **(2008)**; New York, USA.
[6] M. Zaharia, D. Borthakur and J. S. Sarma, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling", Proceedings of the 5th European Conference on Computer Systems, **(2010)**; New York, USA.
[7] C. Quan, Z. Daqiang and G. Minyi, "SAMR: A Self-adaptive MapReduce Scheduling Algorithm in Heterogeneous Environment", Proceedings of IEEE International Conference on Computer and Information Technology, **(2010)**; Washington D.C., USA.
[8] T. Sandholm and K. Lai K. "Dynamic Proportional Share Scheduling in Hadoop", Proceedings of the 15th Workshop on Job Scheduling Strategies for Parallel Processing, **(2010)**; New York, USA.
[9] K. Kamal and K. Anyanwu, "Scheduling Hadoop Jobs to Meet Deadlines", Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, **(2010)**; Washington D.C., USA.
[10] T. Ming, C. Jun-jie and Q. Yan, "Map Reduce Scheduling Algorithm Based on Simulated Annealing", Computer Engineering, vol. 38, no. 19, **(2012)**, pp. 45-48.
[11] J. Peng-fei, Q. Jian-dong and Z. Wen-fei, "Application of improved artificial fish swarm algorithm in Hadoop scheduling algorithm", Application Research of Computers, vol. 31, no. 12, **(2014)**, pp. 3572-3574,3579.
[12] L. Jian-feng and P. Jian, "Task scheduling algorithm based on improved genetic algorithm in cloud computing environment", Journal of Computer Apllications, vol. 31, no. 1, **(2011)**, pp. 184-186.
[13] L. Di Geronimo, F. Ferrucci and A. Murolo, "A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites", Proceedings of the 5th International Conference on Software Testing, Verification and Validation (ICST), **(2012)**; Montreal.
[14] S. Lian-sheng, W. Lei and H. Cheng-zhen, "Research on Scheduling Strategy of Hadoop for Streaming Media Oriented", Journal of Chinese Computer Systems, vol. 35, no. 4, **(2014)**, pp. 728-733.

[15] R. Kondekar, A. Gupta and G. Saluja, "A MapReduce Based Hybrid Genetic Algorithm Using Island Approach for Solving Time Dependent Vehicle Routing Problem", Proceedings of International Conference on Computer & Information Science (ICCIS), **(2012)**; Kuala.
[16] D. Pletea, F. Pop and V. Cristea, "Speculative Genetic Scheduling Method for Hadoop Environments", In Proceedings of 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, **(2012)**; Timisoara.
[17] Z. Xiao-wei, X. Ming and Z. Da-wei, "Hadoop Cluster Performance Parameters Auto Tuning Information Library System Build", Journal of Chinese Computer Systems, vol. 35, no. 3, **(2014)**, pp. 538-542.

Authors

Tao Xue, He was born in Shaanxi Province, China, in 1973. He received his B.S. and M.S. degrees from Xi'an Jiaotong University of China (XJTU) and Northwest University of China (NWU) in 1995 and 2000, respectively, and received the Ph.D. degree in computer software and theory from the XJTU in 2005. He is currently an associate professor in College of Computer Science at Xi'an Polytechnic University. His research interests include Cloud Computing, Big Data and Content-based Networking.

Xiaoqing You, He was born in Henan Province, China, in 1990. He received the B.S. degree in Computer Science and Technology from Luoyang Normal University of China, Luoyang, in 2012. He is currently pursuing the M.S. degree with the Department of Computer Science, Xi'an Polytechnic University. His research interests include Cloud Computing, Big Data, Smart Search and NoSQL Database.

Minglei Yan, He was born in Shanxi Province, China, in 1988. He received the M.S. degree in Computer Technology from Xi'an Polytechnic University of China, Xi'an, in 2015. His research interests include Cloud Computing, Distributed System, and Big Data.

