Research on Hadoop Job Scheduling Algorithms Based on Dynamic Fusion

Lihong Wang

1 Department of computer and information engineering, Heihe University, Heihe, Heilongjiang, 164399, China

*Corresponding author’s e-mail: 116560629@qq.com

Abstract. Adaptive genetic algorithm and improved ant colony algorithm are combined to solve Hadoop job scheduling problem. Firstly, the global search ability of the adaptive genetic algorithm is used to generate the list of resources allocated by the task. When the search speed of the genetic algorithm gradually decreases, the optimal integration time of the adaptive genetic algorithm and the ant colony algorithm is determined dynamically. The initial pheromone distribution of the ant colony algorithm is generated from the optimal solution solved by the adaptive genetic algorithm. Improve the target node selection strategy of ant colony algorithm, consider the success rate of completing tasks of nodes, and accelerate the speed of ant colony algorithm to solve the optimal solution. Simulation results show that compared with genetic algorithm and ant colony algorithm, hybrid genetic algorithm takes less time, and the more tasks, the more obvious advantages.

1. Introduction

Hadoop[1] is an open source big data processing framework of Apache foundation, The core part consists of HDFS(distributed file system) and MapReduce (parallel computing model) modules[2]. HDFS can efficiently complete the efficient storage of large-scale data sets. MapReduce divides the work to be processed by the application into several small pieces, making it easy for developers to realize distributed application programmers. The combination of the two allows users to write distributed programs at the top without having to know the details at the bottom. Users can make full use of the advantages of clustering for distributed and high-speed storage and computing. The combination of HDFS and MapReduce makes Hadoop more powerful[3].

Job scheduling is an important factor affecting the performance of Hadoop platform. Many experts and scholars have adopted particle swarm optimization (pso)[4], ant colony optimization (aco)[5] and genetic algorithm (ga)[6-7] to solve the Hadoop cluster job scheduling problem. Literature[8] aimed at the slow convergence speed and redundant operation problem of genetic algorithm, adopted dual objective function and optimal solution retention strategy to improve job scheduling efficiency. Literature [9] applies the adaptive job scheduling scheme to the ant colony algorithm and makes use of the advantage of positive feedback of ant colony algorithm to improve the efficiency of Hadoop cluster job scheduling. Literature [10] firstly classifies tasks by using resource signatures, and then iterates the configuration scheme by using genetic algorithm, which significantly improves the utilization rate of the cluster. The Hadoop job scheduling algorithm based on mixed genetic algorithm adopted in this paper does not need the administrator to preset values, so as to enhance the dynamic adaptability and improve the overall performance of the platform.
2. Hadoop job scheduling problem model

Hadoop job scheduling problem is that N tasks are executed on K task-tracker nodes, each Task can only be assigned to execute on one node, and the optimal scheduling sequence of these N tasks to execute on K nodes is to be solved. To describe the Hadoop job scheduling problem model, the following conventions are made:

1. The set of N tasks is \( P = \{ P_1, P_2, \ldots, P_n \} \), \( P_i (i = 1, 2, \ldots, n) \) express the i subtask.
2. The number of K node sets is \( T = \{ T_1, T_2, \ldots, T_k \} \), \( T_j (j = 1, 2, \ldots, k) \) express the j Task-Tracker node.
3. When \( < P_i, T_j > \) is a task node, set \( ETG_{ij} \) as subtask \( P_i \) on the \( T_j \) expected execution time. Then the assignment relation of corresponding tasks on nodes can form an n-by-k matrix, the calculation formula is as follows:

\[
ETC = \begin{bmatrix}
ETC_{11} & ETC_{12} & \cdots & ETG_{1k} \\
ETC_{21} & ETC_{22} & \cdots & ETG_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
ETC_{n1} & ETC_{n2} & \cdots & ETG_{nk}
\end{bmatrix}
\quad (1 \leq i \leq n, 1 \leq j \leq k)
\]  

If \( m \) tasks are assigned to the JTH task-tracker node for execution, then the \( time(j) \) when node \( j \) completes all tasks is:

\[
Time(j) = \sum_{i=1}^{m} < P_i, T_j > = \sum_{i=1}^{m} ETG_{ij}
\]  

(2)

After all tasks are allocated to each resource node for execution, the execution time used is the maximum value of \( time(j) \). Therefore, the total execution time is:

\[
TotalTime = \max(\{ Time(j) \}) \quad (1 \leq j \leq k)
\]  

(3)

3. Dynamic Fusion of Genetic Algorithms and Ant Colony Algorithms

3.1. Basic idea of algorithm

The basic idea of this algorithm is to adopt dynamic cohesion strategy to mix Adaptive Genetic Algorithm(AGA) with Adaptive Colony Optimization Algorithm(ACO), form a hybrid optimization algorithm(AGA-ACO). The idea of combining AGA with ACO dynamically is: Before the best time ta, the distribution of pheromone conducted by AGA is taken, then the best solution is generated with ACO. The dynamic strategy is as follows: (1) Set the minimum generation \( Genemin \) and the maximum generation \( Genemax \); (2) Calculate the evolution rate of the best solution \( Genemin- rati \) of offspring in the generation of evolution and set the minimum evolution rate of offspring based on \( Genemin- rati \); (3) If the evolution rate of offspring is less than \( Genemin- rati \) in the continuous generations, it means that optimization speed in GA is low so as to terminate the GA and run AA.

3.2. Adaptive genetic algorithm

3.2.1. Genetic coding and fitness function. Real number coding scheme was used in this passage, the allocation of resources is represented by a list of real numbers, the length of the chromosome is the actual number of tasks. Suppose you have 10 tasks to perform on four task-tracker nodes, task sets \( P = \{ P_1, P_2, \ldots, P_n \} \), resource collection \( T = \{ T_1, T_2, \ldots, T_k \} \), divide the task set into four subsets, The random allocation scheme is: \( T_1 \{ P_1, P_3, P_7 \} , T_2 \{ P_2, P_6 \} , T_3 \{ P_3, P_5, P_9 \} , T_4 \{ P_4, P_8 \} \), The corresponding chromosome is encoded as \{ 1, 2, 3, 4, 1, 3, 4, 2, 3, 1 \}. Because the task \( P_1, P_3, P_7 \) is assigned on the node \( T_1 \), So the chromosome position is set to 1. This direct coding method is simple to decode.

The definition of fitness function is

\[
f = \frac{1}{TotalTime}
\]

(4)
TotalTime is the aim function, the smaller the TotalTime, the greater the fitness of chromosomes.

3.2.2. Adaptive genetic strategy. The adaptive genetic algorithm’s adjustment strategies for crossover probability and mutation probability are as follows:

Original crossover probability is \( P_c \), adjusted crossover probability \( P_c' \) is:

\[
P_c' = \begin{cases} 
P_c \times \frac{f_{c1} - f_{c2}}{f_{c1}} & \text{if } f_{c1} > f_{c2} \\
0 & \text{otherwise}
\end{cases}
\]

(5)

\( f_{c1} \) is the fitness value of the optimal chromosome involved in crossover before crossover operation, \( f_{c2} \) is the fitness value of the optimal chromosome obtained after crossing.

The original probability of variation was \( P_m \), the adjusted probability of variation is \( P_m' \):

\[
P_m' = \begin{cases} 
P_m \times \frac{f_{m1} - f_{m2}}{f_{m1}} & \text{if } f_{m1} > f_{m2} \\
P_m & \text{otherwise}
\end{cases}
\]

(6)

3.3. Ant colony algorithm

3.3.1. Definition of node pheromone. The pheromone of node \( i \) in this algorithm is composed of the resource capacity of the node and the estimated execution time of tasks on each node obtained through genetic algorithm.

The factors that affect the resource capability of a node include the processing capability of CPU, storage capacity and communication capability between nodes. The processing capacity \( P \), memory capacity \( r \), external memory capacity \( s \) and communication bandwidth \( w \) of the CPU of node \( i \) were collected. The pheromone \( \tau_i(0) \) of node \( i \) is the weighted sum of pheromones, The calculation formula is shown in formula (1).

\[
\tau_i(0) = ap_i + br_i + cs_i + dw_i
\]

\[a + b + c + d = 1\]

(7)

By adjusting the weighting factor to focus on the selection of certain types of nodes, the weight of pheromone corresponding to CPU processing capacity can be appropriately increased for CPU-intensive tasks, so that nodes with strong CPU processing capacity can be selected more easily.

The estimated execution time of tasks on each node obtained by genetic algorithm is converted into the pheromone \( \tau_i(t) \) of node \( i \), the value of the pheromone \( \tau_i(t) \) of node \( i \) is:

\[
\tau_i(t) = \tau_i(0) + \tau_i^l(t)
\]

(8)

3.3.2. Pheromone update. If the expected execution time of a task on node 1 is within the specified range, the node is considered to be a valid node. Ant will be affected by pheromone concentration during traveling, and pheromone update is determined by whether the node is a valid node and relevant information brought by backward ant. When task is assigned to the node at time \( t_1 \), the value of the node pheromone is updated to:

\[
\tau_i(t_1) = (1 - \rho)\tau_i(t_1) \quad 0 < \rho < 1
\]

(9)

At time \( t_1 \), \( \tau_i(t_1) \) is the size of the node pheromone. \( \rho \) is the regulatory factor.

At the time \( t_2 \), the task assigned by the node is successfully executed and released, or the task assigned by the node fails, the pheromone is updated to:

\[
\tau_i(t_2) = (1 + \rho)\tau_i(t_1) + \rho \tau_i(t_1) \quad 0 < \rho < 1
\]

(10)
At time $t_2$, $\tau_{i(t_2)}$ is the size of the node pheromone. For the nodes that successfully complete the task, increase the pheromone concentration appropriately, $0 < \rho_t < 1$; the pheromone concentration was appropriately reduced for the failed node, $-1 < \rho_t < 0$.

4. AGA-ACO for Hadoop job scheduling

4.1. AGA-ACO algorithm flow
1. Initialize the parameters of the adaptive genetic algorithm and set the end conditions of the algorithm to classify the task list to be processed;
2. Randomly generate the initial population $A(0)$;
3. Calculate fitness of every entity in $A(0)$ and $ETG_{ij}$;
4. Determine the probability $P_i$ of individual selection in $A(g)$;
5. Perform the following actions for individuals in $A(g)$ until all individuals are selected:
   (1) Choose two parent in $A(g)$ according to roulette selection strategy;
   (2) Generate a random number $r$ between 0 and 1;
   (3) if ($r \leq P_m$) execute mutation to the male parent, insert new entity to the $A(g+1)$;
      else if ($r \leq P_m + P_c$) execute interlace operation to the male parent, insert new entity to the $A(g+1)$;
      else execute multiply operation, insert original male parent to the $A(g+1)$;
6. Calculate $ETG_{ij}$ and fitness of every entity in $A(g+1)$, $g = g + 1$
7. Change crossover probability $P_c$ and mutation probability $P_m$;
8. Repeat step 4~7, Until the end condition of adaptive genetic algorithm is satisfied;
9. Among the individuals obtained by adaptive genetic algorithm, the first 20% individuals with high adaptive value are selected as the set of optimal solutions;
10. According to equations (7) and (8), the pheromone on the node is initialized;
11. Set parameters and end conditions of ant colony algorithm;
12. Before searching with the ant colony algorithm, first check the task list. If the task list is empty, then randomly place the ants on the node, Otherwise, place m forward-facing ants at the recommended node, and select the next node $j$ according to formula (11), put $j$ into $tabuk$;
13. If node $j$ is a valid node, the backward ant is generated at node $j$, and pheromone update is carried out according to equations (9) and (10). If the optimal solution is found successfully, the node is filled into the task list.
14. According to the backward ant information, the success rate of each node and the global pheromone are updated.
15. If it meets the ending condition of ant colony algorithm, then the end of the algorithm is achieved; otherwise, go to step 12.

4.2. Combining self-adaptive genetic algorithm with ant algorithm
1. Set the time for dynamic fusion: (1) reach the maximum generation $Gen_{max}$; (2) $Gen_{die}$ is less $Gen_{min\_ratio}$ by a predefined value.
2. Set the initial pheromone of the ant algorithm: We use the initial distribution of GA. Pheromone on the path $(i,j)$ is defined as follows:
   \[ \tau \tau_{ij} = \tau + \tau \]

---

---
Where $\tau^c$ is the pheromone constant on the path $(i, j)$ given the problem. $\tau^g$ is the pheromone from the results conducted by GA. $\tau^c = 30$, and $\tau^g = 0$.

3. Convert solutions conducted by GA to pheromones: the set $P_{\text{best}}$ of optimized solutions consist of 20% entities with large finesses from the population $(0.2 \times N = 0.2 \times 50 = 10)$. For every solution in $P_{\text{best}}$ as $(s = 1, 2, \ldots, 10)$, if the path $(i, j)$ is traveled, $\tau^g$ is increased by 10.

5. Analysis of experimental results

The simulated environment was 20 resource nodes, and 30-100 tasks were taken as examples for testing. Simulation experiments were carried out on self-genetic algorithm (GA), ACO and AGA-ACO. In order to test the effectiveness of the algorithm, the control parameters of GA, ACO and AGA-ACO are set the same, and the running time and iteration times of the algorithm are compared respectively. The minimum set of tasks is shown in Table 1.

In order to compare the execution time and cost of AGA, ACO and AGA-ACO in executing larger task sets, a larger task set is set up as shown in Table 2. The control parameters of the three algorithms are the same, and the execution time and execution cost are compared respectively.

| Table 1. The minimum set of tasks | Task | The number of task | [30,100] | The length of task | [500,5000] | General type of expediency | 0,1,2 | Expected time (ms) | [10,50] | Expected bandwidth (bit) | [300,1500] | Expectation cost | [5,20] |
|----------------------------------|------|-------------------|---------|-------------------|----------|--------------------------|------|-------------------|--------|----------------------|----------|---------------------|------|

| Table 2. The larger set of tasks | Task | The number of task | [100,500] | The length of task | [5000,15000] | General type of expediency | 0,1,2 | Expected time (ms) | [50,500] | Expected bandwidth (bit) | [300,1500] | Expectation cost | [10,50] |
|----------------------------------|------|-------------------|----------|-------------------|-----------|--------------------------|------|-------------------|--------|----------------------|----------|---------------------|------|

As can be seen from figure 1, when the number of tasks is relatively small, GA has a strong search ability and takes a short time, but with the increase of the number of tasks, the time increases rapidly. ACO algorithm rapidly converges to the optimal solution with the accumulation of pheromones, and the time increase is smaller than AGA algorithm. For AGA-ACO algorithm, after combining the advantages of the two algorithms, the advantages of the algorithm are not obvious when the number of tasks is small. As the number of tasks increases, the advantages of the algorithm become prominent.
Figure 2 shows the AGA, the ACO and AGA - the ACO three algorithms in different task the number of iterations required to obtain a optimal solution of the comparison, the comparison result shows that the number of iterations required AGA algorithm, so the algorithm is the most time-consuming, AGA - the ACO rapidly in early use of the global search ability of GA algorithm to obtain the initial pheromone distribution, the optimal fusion point of reusing the ACO algorithm of positive feedback mechanism, finding the optimal solution quickly the number of iterations required less than GA and ACO algorithm.

![Figure 3. Comparison of Task Execution Time](image3.png)

![Figure 4. Comparison of Task Execution Time](image4.png)

6. Conclusion
Aiming at the Hadoop job scheduling problem, this paper integrates the adaptive genetic algorithm and ant colony algorithm to solve the optimal solution of task allocation. In the early stage, AGA's global fast search ability is used; in the later stage, ACO's positive feedback mechanism is used to form complementary advantages. In the process of algorithm implementation, the resource capacity of the node and the estimated time of task completion are considered comprehensively, and then the task is allocated to the most suitable node for execution. Compared with the three common schedulers of Hadoop, this algorithm improves the efficiency of job scheduling and saves the cost of administrator management node.

Acknowledgments
Young Schloars with Creative Talents of Heilongjiang Education Department (2017-KYYWF-0360).

References
[1] Xu H.L., Zhai L. (2015)Research on MapReduce Scheduling on Hadoop Platform. Computer Applications and Software, 32(5):1-6.
[2] Apache Hadoop Next Gen Map Reduce (YARN) [J/OL].2013, http://hadoop.apache.org/docs/r2.2.0/hadoop-yarn/hadoop-yarn-site/YARN.html
[3] LIU J.F,PENG J.(2011)Task scheduling algorithm based on improved genetic algorithm in cloud computingenviroment.Journal of Computer Applications, 31(1):184-186.
[4] LIU P.H. (2015)Research of scheduling algorithm for Hadoop based onparticle swarm optimization . Journal of Hebei University of Engineering( Natural Science Edition) ,32(1):23-25.
[5] ZHANG H.Y., WU F. (2014)Research of Hadoop resource-aware scheduler based on ant colony algorithm. Computer Engineering and Applications, 50(15):65-71.
[6] Lou T,Du W.C. (2015)Hadoop Clusters of Job Scheduling Based on MixedAnt Algorithm Genetic Algorithm . Natural Science Journal of Hainan University, 33(4):138-143.
[7] BAO D.H. (2015)Improved Genetic Algorithm Based on Hadoop. Journal of Inner Mongolia Normal University(Natural Science Edition). 44(1):62-65.
[8] Xue Tao,Yan Minglei. Research on Hadoop Job Scheduling Based on CHC Genetic Algorithm.Computer Engineering, 2016,42(3) :1-68
[9] QIN J, DONG Q.Q. (2017) Improvement of Algorithm for Cloud Task Scheduling Based on AntColony Optimization and Simulated Annealing. Computer Technology and Development, 27(3) :117-121.

[10] Ma Y, Yu C.Y. (2017) Hadoop parameter automatic tuning system based on resource signature and genetic algorithm. Application Research of Computers, 34(11):25-31.

[11] WAN B, DUAN X. (2014) Hadoop job scheduling algorithm based on resource prefetching. Application Research of Computers, 31(6):1639-1643.

[12] GUO C.C. (2016) Improvement of large data clustering method based on Hadoop MapReduce and coarse grain parallel genetic algorithm. Journal of Engineering of Heilongjiang University, 7(3):87-91.