Application of Grid Task Scheduling and Neural Network Algorithm in the Simulation Training in Higher Vocational Colleges

Xueping Yang
Modern Education Technology Center, Information Teaching Applied Technology Extension Center, Chongqing Vocational Institute of Engineering, Chongqing, 402260, China

GuangHai Chen
Information Teaching Applied Technology Extension Center, Chongqing Vocational Institute of Engineering, Chongqing, 402260, China

Yu Liu
Information Teaching Applied Technology Extension Center, Chongqing Vocational Institute of Engineering, Chongqing, 402260, China

Abstract
Grid computing is an active area of research at present, in which the task scheduling is an important part. In accordance with the characteristics of task scheduling and the traditional DVG diagram in the grid environment, the advantages of the neural network are drawn in, and the neural network theory is integrated with the task scheduling feature to establish the grid task scheduling neural network model based on partitioning. On this basis, through the partitioning of the grid neural network, the mapping and scheduling of the tasks for the grid resources is implemented, and a kind of Optimized Neural Network Partitioning Based Grid Task Scheduling Algorithm HNGTS is put forward. The grid task scheduling and the neural network algorithm are applied in the simulation training in the higher vocational colleges, and the simulation results show that: The model put forward in this paper and its algorithm have good performance in the grid task scheduling, which is superior to the performance of the similar algorithms.

Key words: Grid, Task Scheduling, Neural Network, Partitioning.

1. INTRODUCTION

Grid computing is a hot research topic at present. Its purpose is to integrate the entire network into a huge supercomputer, and to achieve the comprehensive sharing of the computing resources, the storage resources, the data resources, the information resources, the knowledge resources and the expert resources (Aron, Chana, and Abraham, 2015). Due to the heterogeneity and wide area of the grid resources, the task scheduling in the grid environment is more complicated (Atef, Hagara, Mahdy, and Janeček, 2017). There are many grid task scheduling algorithms at present, such as the Min-Min algorithm (Pop, Dobre, Comaneci, and Kolodziej, 2016), the Max-Min algorithm (Zarrabi, Samsudin, and Karuppih, 2015), the Max-Int algorithm (Nematollahi, Naghibzadeh, Abrishami, and Yaghmae, 2017), the Fast-Greedy algorithm (Bashir, Susarla, and Vairavan, 2016), the Generation Scheduling algorithm (GS), the DLS (Dynamic Level Scheduling) algorithm and so on. In these algorithms, the majority of them make use of the traditional Directed Vector diagram (hereinafter referred to as DVG for short) to construct the dependent task model, which has certain deficiencies in the representation of the task scheduling situation in the grid environment, thereby leading to the reduction in the execution efficiency of the task scheduling and the impact on the performance of the scheduling algorithms (Aziza and Krichen, 2017; Panda and Jana, 2015). In this paper, the concept of neural network is introduced to extend and popularize the general DVG so as to adapt to the grid scheduling features. And on this basis, the grid task scheduling strategy is implemented by using the partitioning to the neural network (Kianfar, Moslehi, and Yahyapour, 2015; Doostsadigh, Asemi, and Ashena, 2015).

In this paper, the advantages of the traditional DVG diagram are drawn on, and the characteristics of the grid environment are taken into comprehensive consideration to integrate the neural network partitioning theory with the grid environment factor, and a neural network model for the grid task scheduling is established. On the basis of this model, in accordance with the characteristics of the meta task and the dependent task, a kind of Optimized Neural Network Partitioning Based Grid Task Scheduling Algorithm (hereinafter referred to as HNGTS for short) is put forward. The experiments show that this model can effectively reflect the characteristics of the task scheduling under the grid environment and optimize the task scheduling process. And the algorithm put forward in this paper has good performance, which is superior to that of the similar algorithms.
2. OPTIMIZED NEURAL NETWORK PARTITIONING BASED GRID TASK SCHEDULING MODEL

2.1. Grid Task Scheduling Optimized Neural Network Model

In this paper, on the basis of the theory of neural network, the grid task scheduling optimized neural network model is established in combination with the characteristics of the grid environment and the task scheduling.

Definition 1: BP neural network for the Grid task scheduling

In which, \( V = \{v_0, v_1, v_2, \ldots, v_n\} \) stands for the grid task set, \( n = |V| \) is referred to as the order of the neural network, that is, the number of tasks, \( v_i (i \in [0, n-1]) \) stands for the \( i \)-th task. \( v_i = \{tID, rID, tG, tServ, tData, tType\} \). And the meaning of each attribute is as the following:

1) \( tID \) stands for the task identification.
2) \( rR = \{rC_1, rC_2, \ldots, rC_k\} \) stands for the grid resource requirements of the task, \( k \) stands for the number of resource capabilities required for the task, \( rC_j (j \in [0, k-1]) \) stands for the capabilities that are required for the resources to complete the task, and the grades of capacities are arranged in a descending order.
3) \( tGain \) stands for the \( Gain \) value of the task. \( Gain \) value is an important basis for the movement of the nodes in neural networks. If a node is about to be moved from the current partition sub sets to other partition sub sets, the number of the vectors that will be changed after the movement in the current Cut – Set is the \( Gain \) value. The larger the \( Gain \) value is, the greater the influence of the node movement on the association among the partition sub sets is, and therefore the higher the priority of movement is.
4) \( tServ \) stands for the grid services called by the tasks, which includes the WSName, WSMethow, WSQ and so on.
5) \( tData \) stands for the data related to the task, which includes C, inData, outData and so on, where C, inData and outData stands for the computing volume, the input data and the output data of the tasks, respectively.
6) \( tType \) stands for the node movement type, which can be divided into BC and FC.

BC (Base Cell) stands for the basic node, which refers to the unit nodes that have been selected to move to the other partition sub set.

FC (Free Cell) stands for the free node, which refers to the unit nodes that have not been locked except by the BC.

The allocation of the task resources is on the basis of the capability model of the resources. The capability set \( RM \) of the resources can be described as: \( RM = \{R_i\} \), in which, \( R_i (i \in [0, rt - 1]) \), \( rt \) stands for the number of the resource types) stands for the resources, \( R_i = \{RID, RC, RER, RN\} \), \( RID \) stands for the resource identifier, RC stands for the capability set of the resources, RER stands for the expected readiness time of the resources, and RN stands for the quantity of the resources.

In the process of the resource selection, if the capacity requirement set \( rR \) of a task \( v_i \) is contained within the capability set of a grid resource object R, the task can be selected to be mapped to the resource.

2.2. Grid Task Scheduling Neural Network

Definition 2: AT: that is, the expected time of data transmission. \( AT(i, j) \) stands for the time that the data are transmitted from the resource assigned to the task \( i \) to the resource assigned to the task \( j \). And the equation to calculate AT is as the following:

\[
AT(i, j) = \sum W_{UD}(i, j) \times \frac{TD}{UD} \quad (1)
\]

\[
W_{UD}(i, j) = \frac{tp}{RBW} + RTT + FS \quad (2)
\]

In which, \( tp \) stands for the transmission parameter, \( RBW \) stands for the transmission bandwidth between the resources, \( RTT \) stands for the network delay, \( FS \) stands for the format conversion time, \( TD \) stands for the...
transmission data size, $W_{UD}[i,j]$ stands for the communication time for the transmission of the data blocks with the size of UD between the resources $R_i$ and $R_j$, which includes the various hardware and software overheads in the communication between resources, the network transmission delay, the time required for the conversion of the formats and so on.

Definition 3: AW: that is, the expected time of waiting. $AW(i,j)$ stands for the time of waiting on the resource $R_j$ before the execution of the task $v_i$.

$$AW = \frac{TD}{LBW} + \frac{QL}{sv} \quad (3)$$

In which, LBW stands for the bandwidth of the local resource I/O, QL stands for the length of the waiting queue in the resource, and sv stands for the service rate of the resource.

Definition 4: AE: that is, the expected time of execution, which means that the expected time of execution of the task $v_i$ on the resource $R_j$ is $AE[i,j]$, where AE stands for the matrix expected to be executed.

The execution time of the task $v_i$ on the resource $R_j$ is as the following:

$$AE(i,j) = C(v_i) * rp * RC(R_j) \quad (4)$$

In which, rp stands for the calculation parameters, $C(v_i)$ stands for the computing volume of the task $v_i$, $RC(R_j)$ stands for the computing capacity of the resource $R_j$. And $(rp * RC(Rj))$ stands for the computing time of the unit computing volume of $v_i$ on the $R_j$.

Definition 5: AC: that is, the expected time of completion, which means that the expected time of completion of the task $v_i$ on the resource $R_j$ is $AC[i,j]$, where AC stands for the matrix expected to be completed as the following.

$$AC(i,j) = \max_{k \in Prad(i)}(AW(i,j) + AT(R(k),j)) + AE(i,j) \quad (5)$$

In which $R(k)$ stands for the resources corresponding to the task $v_k$.

Definition 6: CS: that is, Cut-Set, the cut set, which refers to the set of all the vectors that connect the set that is partitioned. The vector source set contains the sub set in which the nodes and the tail nodes are located in different partition sub sets.

Definition 7: FB: that is, From Block. The FB of a basic node refers to the set of the nodes that are connected to the vector $v_i$ through the vector $e_j$. And the set is the moving source partition sub set located at $v_i$.

And the set is denoted as $FB(v_i, e_j)$. $|FB(v_i, e_j)|$ stands for the number of nodes that are contained in the FB, that is, the size of the set FB.

Definition 8: TB: that is, To Block. The TB of a basic node $v_i$ refers to the set of the nodes that are connected to the vector $v_i$ through the vector $e_j$. And the set is located at the moving set of the sibling partition sub set of $v_i$. The set is denoted as $TB(v_i, e_j)$. And $|TB(v_i, e_j)|$ stands for the number of nodes that are contained in the TB, that is, the size of the set TB.

3. DESCRIPTION OF THE GRID TASK SCHEDULING AND THE NEURAL NETWORK ALGORITHM

3.1. Initialization of the Neural Network

First of all, in accordance with the tasks that have already been known, the resources and the other related parameters, the initialization grid task scheduling BP neural network model is generated.

In BP neural network, if two tasks $v_x$ and $v_y$ $(x, y \in [0, n-1])$ belong to the source set of the same vector, there is no dependency relation between the two, and the resource capacity requirement set of the two are both included in the capability set of the same grid resource object, then the two tasks shall be merged. The
task resource capability requirement set after the merging is the union of the resource requirement sets of the task \( v_i \) and \( v_j \). That is to say, \( \forall v_i, v_j \in \text{Pred}, x, y \in [0, n-1], q \in [0, m-1], v_i \notin \text{Pred}(v_j), v_j \notin \text{Pred}(v_i), v_i \in R_x, R_y, v_j \in R_x, R_y, z \in [0, rt-1] \). Then \( v_i \) and \( v_j \) are merged. After optimization, the number of task nodes is reduced appropriately, which is conducive to reducing the time complexity of the scheduling and simplifying the calculation.

In the process of the generation of the BP neural network model, it is necessary to calculate the initial Gain value of each node. And the algorithm is as the following.

```c
Gain_Calculation()
{
    for (every node \( v_i \) in the GHG)
        \( v_i \).tGain=0;
    for (each vector \( e_j \) that is connected with \( v_i \) )
    {
        // * If only one node \( v_i \) in the vector is located in the FB, this indicates that the vector belongs to CS.
    
    Therefore, the movement of the node \( v_i \) will lead to the reduction of the vectors in the CS, that is, the decrease in the Pcost, thereby increasing the Gain value of the node \( v_i \ )*/
        if ( \( | \text{FB}(v_i, e_j)| = 1 \) )
            \( v_i \).tGain=\( v_i \).Gain+1;
            // *If there are no nodes that are located at the TB in the vector, it indicates that the vector does not belong to the CS. Therefore, the movement of the node \( v_i \) will lead to the addition of the vectors in the CS, that is, the increase in the Pcost, thereby reducing the Gain value of the node \( v_i \ )*/
            if ( \( | \text{TB}(v_i, e_j)| = 0 \) )
                \( v_i \).tGain=\( v_i \).Gain-1;
    }
    }

In this paper, the initialization of the neural network is implemented using the initial function.

### 3.2. Computing Process of the Neural Network

In this paper, the dichotomy method is adopted to carry out hierarchical partitioning to the neural network. There are two goals in the neural network: to reduce the communication overhead and to ensure the load balancing between the partition sub sets.

1. First of all, the nodes are selected in accordance with the Gain value of each node to move between the initial partition sub sets, so as to change the distribution of the nodes in the partition sub sets. The purpose is to reduce the partitioning overhead Pcost and thereby reducing the communication overhead among the partitioned sub sets. This step is implemented through the function \( \text{cellmove}(PX, PY) \), and the process is as the following:

```c
\text{cellmove}(PX, PY) 
{
    // pc stands for the communication overhead between partition sub sets of PX and PY
    pc=Pcost(PX, PY);
    while (PX \neq \text{null})
    {
        // The node \( mv \) with the highest Gain value in the PX is taken as the basic node and the
        \( mv=\text{HG} \) \( \text{select}(PX) \) is locked;
        // \( mv \) is moved forward, and the Gain value is updated
        for (each vector \( e_j \) that is connected to \( mv \) )
        {
            if ( \( | \text{TB}(mv, e_j)| = 0 \) )
                The Gain value of all the free nodes is increased by 1 on \( e_j \);
            else if ( \( | \text{TB}(mv, e_j)| = 1 \)
                    and the node at the TB is a free node)
                The Gain value of the free node located at the TB is reduced by 1 on \( e_j \);
        }
        // \( mv \) is moved from PX to PY
    }
}
```
PX=PX-{mv} ;
PY=PY+{mv} ;
// After the mv is moved, the values of the original FB(mv, ej), TB(mv, ej) and Gain are updated for (each vector associated with mv) {
FB(mv, ej)=FB(mv, ej)-{mv} ;
TB(mv, ej)=TB(mv, ej)+{mv} ;
if(i | FB (mv, ej ) |=0)
The Gain value of all the free nodes is reduced by 1 on ej ;
else if(i | FB (mv, ej ) |=1
and the node located at the FB is a free node)
The Gain value of the free node located at the FB is increased by 1 on ej ;
}
if (Pcost (PX, PY)< pc)
pc= Pcost (PX, PY);
if (the number of the Pcost (PX, PY) is not reduced to reach the value of the threshold)
break;
}
When the moving end condition is met, the movement of the node is stopped. The node movement in this step does not take the changes in the load of each partition sub set after the movement into consideration; hence the task load may be unbalanced. Therefore, after this step is completed, it is necessary to make further adjustment to the task distribution of each partition sub set, so as to meet the needs of the load balancing.

(2) The second basic step is mainly carried out on the basis of the weight value PW of each partition sub set. In order to meet the load balancing rules between the sub sets, the nodes are moved between the partition sub sets in accordance with the calculated sub sets of weights. The nodes are moved from the partition sub set with large weight value to the partition sub set with small weight value. And it stops when the balance rule is met or there are no free nodes in the Ph that can be moved. This step is implemented by the function loadbalance (PX, PY) as the following:
loadbalance (PX, PY){
Let Ph be PX, the partition sub set with relatively high PW in the PY, and Pl stands for the partition sub set with relatively low PW;
pd=│PW (Ph)- PW (Pl)│ ;
/*pdThr stands for the threshold value of pd, pd exceeding pdThr indicates that the load among the sub sets of the partitions is not balanced*/
while(pd>pdThr) {
// The node lv with the highest Gain value in the FS is taken as the basic node
lv=HGain_select(Ph);
// Quit if the movement of the node will lower the load balance
if(pd<│PW(Ph-{lv})-PW(Pl∪ {lv}) )
break;
// lv is moved forward and the Gain value is updated
for (each vector ej that is connected to lv) {
if(TB(lv, ej ) |=0)
The Gain value of all the free nodes is increased by 1 on ej ;
else if(TB(lv, ej ) |=1
and the node located at TB is a free node)
The Gain value of the free node located at TB is reduced by 1 on ej ;
}
// lv is moved from Ph to Pl
Ph=Ph-{lv} ;
Pl=Pl+{lv} ;
}

In this paper, experiment is carried out on the basis of the GridSim, a grid simulator developed by the University of Melbourne in Australia. Compared with the other simulation tools, GridSim is more superior in the simulation of the grid computing. It can simulate the establishment of the distributed grid heterogeneous resources and the network connections, and can implement the functions of resource searching, task establishment, processing and so on. The object oriented technology is adopted for the construction, with clear framework. And the virtual time is used, which allows it to be independent of the host performance.

Since the algorithm put forward in this paper and the LNT algorithm are both heterogeneous scheduling algorithms that take the dependent tasks into consideration, the experimental environment is similar. Therefore, simulation scheduling experiments are conducted on the algorithm put forward in this paper and the LNT algorithm respectively to make comparison. The related parameter values are generated randomly in the experiments to generate the random task diagram and the grid environment, and then to further generate the grid task scheduling neural network on this basis.

In the experiment, task scheduling diagram is generated randomly for different numbers of tasks to carry out the task scheduling, and the algorithm running time and the average scheduling length Makespan is obtained under different numbers of tasks. The experimental results are shown in Figure 1 and Figure 2. In Figure 1, the horizontal coordinates stand for the number of tasks and the vertical coordinates stand for the running time of
the algorithm. The experimental results show that the time of execution of the algorithm put forward in this paper is shorter than that of the LNT algorithm under the condition of different number of tasks. And the more the number of tasks is, the more significant the difference is, and the more superior the performance is. In Figure 2, the horizontal coordinates stand for the number of tasks, and the vertical coordinates stand for the scheduling length Makespan. It can be seen from the figure that the Makespan of the algorithm put forward in this paper is smaller than that of the LNT algorithm, and the performance superiority of the proposed algorithm is shown more significantly with the increase in the number of tasks.

At the same time, the task relation diagrams are generated randomly for different number of resources to carry out the task scheduling, and the algorithm running time and the average scheduling length Makespan is obtained under different numbers of resources. The experimental results are shown in Figure 3 and Figure 4. In Figure 3, the horizontal coordinates stand for the number of resources, and the vertical coordinates stand for the running time of the algorithm. It can be seen from the experimental results that the time of execution of the algorithm put forward in this paper is shorter than that of the LNT algorithm under the condition of different number of resources. And the difference is more significant with the increase in the number of resources. In Figure 4, the horizontal coordinates stand for the number of resources, and the vertical coordinates stand for the scheduling length Makespan. As can be seen from the figure, the Makespan of the algorithm put forward in the paper is smaller than that of the LNT algorithm. In addition, the more the number of resources, the more superior the proposed algorithm is, which has reflected the good performance of the algorithm put forward in the paper.

![Figure 1. Comparison of run time (different numbers of tasks)](image1)

![Figure 2. Comparison of Makespan (different numbers of tasks)](image2)
5. CONCLUSIONS

In this paper, the grid task scheduling neural network model is constructed through the combination of the neural network theory and the grid task scheduling, and a kind of optimized neural network partitioning based grid task scheduling algorithm HNGTS is put forward. The experimental results show that the algorithm put forward in this algorithm is a kind of effective grid task scheduling algorithm. On this basis, the focus of the next step is to further improve and expand the grid task scheduling neural network model in accordance with the changes in the grid environment and the characteristics of the neural network theory. And based on the improvement and expansion, a series of effective grid task scheduling algorithms are put forward.

ACKNOWLEDGEMENTS

This work was supported by New Technology Promotion Project of Higher Vocational and Technical Colleges in Chongqing "Development and Application of Smart Campus New Technology Based on the Mobile Internet and Big Data Support" (No.: GZTG201609); Science and Technology Research Project of Chongqing Municipal Education Commission "University Information Innovation Teaching Integrated Platform Research and Practice Based on Cloud Technology" (No.: KJ1732434); and Higher Education Teaching Reform Research Project in Chongqing "Teaching Resource Library Platform Construction Research and Practice Based on Cloud Technology" (No. 151026).

Figure 3. Comparison of run time (different numbers of resources)

Figure 4. Comparison of Makespan (different numbers of resources)
REFERENCES


