Task Scheduling Model Based on Multi-Agent and Multi-Objective Dynamical Scheduling Algorithm

Zhanjie Wang and Ting Fang
Department of Computer Science, Dalian University of Technology of China, Dalian China
Email: wangzhj@dlut.edu.cn

Abstract—With the increasing number of nodes in distributed systems, the complexity of task scheduling also increases. Therefore, how to schedule tasks reasonably is becoming more and more significant. Most traditional algorithms only consider a single condition without thinking over dynamic characteristic of system and tasks and lack of comprehensive measures. Therefore they cannot meet the needs of distributed systems. To solve these problems, we establish a distributed task scheduling model based on multi-agent in this paper, build a negotiation scheduling mechanism based on the model and propose distributed multi-objective dynamical scheduling algorithm (DMOD). In the algorithm, each node is capable of independent decision-making and dynamical evaluation rules make a comprehensive evaluation of task completion time, system load and communication traffic. DMOD, MinMin and the algorithm based on tree structure (BTS) are compared through simulation experiments. Experimental results show that DMOD reduces communication traffic without increasing task completion time, avoids performance degradation caused by sharp increase of system load and communication traffic in distributed system and therefore improves system stability and task execution efficiency.

Index Terms—Distributed System, Dynamical Task Scheduling, Multi-Agent, Multi-Objective

I. INTRODUCTION

With the rapid development of computer technology and the improving requirement of computing power, distributed computation has been widely used because of its excellent computing power. Distributed systems have many advantages, such as dynamic characteristic, reliability, real-time and so on. However, the big number of nodes in large-scale distributed system will cause some corresponding problems. For task scheduling problem, as the number of nodes increases, the complexity will also increase. So how to schedule tasks reasonably in distributed systems is very significant.

The task scheduling problem in distributed computing system is allocating a set of tasks or a group of jobs which form the workloads to several nodes according to some scheduling rules, policies and specific execution sequence, and therefore to achieve a better performance [1]. Task scheduling problem is an NP-complete problem [2], and the answer is non-deterministic. In distributed systems, dynamic characteristic and uncertainty should also be concerned, which certainly will also increase the difficulty of solving the problem and make scheduling algorithms become more complicated. Therefore task scheduling problem has become a hot issue. In this paper, we give a solution to solve the problem. For some specific task scheduling problems, traditional scheduling algorithms such as list scheduling algorithm [3] [4] [5] and genetic algorithm [6] [7] have done a good job. However, they still have some restrictions when considering dynamical changes of system and tasks. With further research in recent years, some new algorithms based on traditional algorithms have been proposed. Task scheduling problem has achieved some progress and development, yet there are still many problems. For example, the conditions considered in such algorithms are relatively simple. Most of them only consider task completion time or communication traffic [8] [9] [10] without taking comprehensive measures of multiple factors. With the increasing number of nodes and scale of applications, scheduling algorithms should also be able to adapt to dynamical changes of the system structure. Traditional centralized scheduling strategies cannot meet the demands of large-scale applications in dynamic characteristic and reliability, so research of distributed scheduling strategy has become more and more significant.

We introduce the concept of multi-agent when studying task scheduling problems. The research of agent originated in the field of artificial intelligence. It is a computer program with certain intelligence which can run autonomously, provide corresponding service and imitate human behavior and relationships [11]. The independence, autonomy and interactivity of multi-agent provide a new direction for studying dynamical task scheduling in distributed system. Many researchers apply it to task scheduling problems and achieve some results [12] [13]. Multi-agent system is an intelligent system with different agents which have different functions and can solve problems collaboratively. Multi-agent system can divide a large-scale and complicated problem into sub-problems. Each agent in the system only considers the solution of one sub-problem and several agents accomplish a task through collaboration. With the development of distributed system, it’s a good solution to apply agent
technology to distributed systems. Although scheduling tasks to different nodes has improved degree of parallelism to some extent in distributed systems, resource utilization of system is not very high. Multi-agent technology can make resource allocation more intelligent. Each agent encapsulates the solution of a subproblem which increases degree of parallelism of tasks while also improves system resource utilization significantly. The autonomy, reactivity and initiative of agent can adapt to the dynamic characteristic and uncertainty of distributed system better. However, most scheduling strategies are still centralized in current systems. With the increasing number of nodes, the computing and communication capabilities of master node have become bottleneck that reduces the reliability of the system.

In this paper, with considering dynamic characteristic and uncertainty of distributed systems, we establish a distributed task scheduling model based on multi-agent to make resource utilization more reasonable and improve intelligence of the whole system. In this model, negotiation scheduling mechanism is accomplished by collaboration and interaction of multi-agent. The corresponding distributed multi-objective dynamical scheduling algorithm (DMOD) is proposed on the basis of the mechanism. Each node in the model has independent decision-making ability, which will not only improve reliability of the system, but also satisfy the service requests of large-scale applications in the distributed system. The dynamical evaluation rules in DMOD consider task completion time, system load and communication traffic comprehensively. Performance degradation caused by sharp increase of system load and communication traffic is avoided. Consequently, system stability and task execution efficiency is improved. Feasibility and effectiveness of the model have been proved through simulation experiments and therefore a new direction for related research has been provided.

II. TASK SCHEDULING MODEL BASED ON MULTI-AGENT

A. Task Scheduling Model and Related Concepts

Before establishing the model, we set the following constrains:

1. Local communication traffic between tasks is not included in communication cost, which means local communication traffic is negligible.

2. Communication between tasks is free from constraints of precedence relations when tasks need to communicate with other tasks.

The definitions of set of nodes and set of tasks based on above constrains in distributed system are:

For a distributed system with n nodes, the set of all nodes is defined as H while \( h_i (1 \leq i \leq n) \) represents different nodes, hence \( H = \{ h_1, h_2, \ldots, h_n \} \). When m tasks need to be processed, the set of all tasks is defined as T while \( t_j (1 \leq j \leq m) \) represents different nodes, hence \( T = \{ t_1, t_2, \ldots, t_m \} \).

In order to describe the task scheduling model proposed in this paper, we give the following concepts.

1. The interdependency \( c_{ij} \) between task \( t_i \) and node \( h_j \)

In the task scheduling model, the interdependency \( c_{ij} \) between task \( t_i \) and node \( h_j \) is defined as the proportion of communication traffic when communicate with node \( h_j \) compared to communicating with all nodes. For the task set \( T, T_j \) represents the tasks that have been allocated to node \( h_j \). \( T_j \) is a subset of \( T \). For any task \( t_k \) and \( t_{\ell} \), \( e_{\ell k} \) is defined as the communication traffic between these two tasks. Then the definition of the interdependency \( c_{ij} \) between task \( t_i \) and node \( h_j \) is:

\[
\begin{align*}
\begin{cases}
\sum_{t \in T_j} e_{jt} \geq 0, & (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n) \\
0; \sum_{t \in T} e_{jt} = 0
\end{cases}
\end{align*}
\]

(1)

\( t_i \) represents a random task in \( T, \sum_{j \in H} e_{jt} \) represents total communication traffic between task \( t_i \) and tasks allocated to node \( h_j \), and \( \sum_{i \in T} e_{jt} \) represents total communication traffic between task \( t_i \) and all tasks. Allocating tasks to the node which has maximum interdependency can reduce communication traffic effectively.

2. Load of node \( h_j \)

Load \( w_j \) of node \( h_j \) can be comprehensively measured through different parameters, such as network traffic \( S_i \), CPU utilization \( U_j \), memory utilization \( M_j \) and disk utilization \( D_j \). To consider the influence of different parameters on the system simultaneously, the definition of the load \( w_j \) of node \( h_j \) is defined as:

\[
w_j = x_i S_j + x_j U_j + x_k M_j + x_k D_j
\]

(2)

And there is:

\[
x_i \geq 0 \text{ and } \sum x_i = 1
\]

(3)

\( x_i \) is the weight of different parameters. Real-time tracking of these parameters can obtain running state of node \( h_j \) effectively.

3. Earliest completion time \( t_{end_j} \) of task \( t_i \) on node \( h_j \)

For calculating completion time of all tasks in a node, we define the node as \( h_j \). The length of ready queue is defined as \( \text{Queue\_Length}_h \). Task execution time of \( \text{Queue\_Allo} \) is defined as \( t_{\text{Process}_k} \) \((0 \leq k \leq \text{Queue\_Length}_h)\). Then the completion time \( h_{\text{end}} \) of node \( h_j \) is defined as:

\[
h_{\text{end}} = \sum_{k=1}^{\text{Queue\_Length}_h} t_{\text{Process}_k}
\]

(4)

The release time of task \( t_i \) is defined as \( t_{\text{release}_i} \), and time of transferring task \( t_i \) to node \( h_j \) is defined as \( t_{\text{trans}_j} \). Then the earliest completion time \( t_{\text{end}_j} \) of task \( t_i \) on node \( h_j \) is:
\[ t_{\text{end}_j} = t_{\text{release}_j} + h_{\text{end}_j} + t_{\text{Process}_j} + t_{\text{trans}_j} \]  

(5)

## B. Task Scheduling Model Based on Multi-agent

The task scheduling model based on multi-agent established in this paper includes resource agent (RA), task management agent (TA), information interaction agent (IA), evaluation agent (EA) and negotiation scheduling agent (NA). The task scheduling model is shown in Fig.1.

![Task scheduling model based on multi-agent](image)

Figure 1. Task scheduling model based on multi-agent

Then specific functions of each agent will be introduced.

1. **RA**
   - RA is responsible for collecting resource information of the node and calculating the load of the node. RA obtains various parameters related to computing power of the node timely, including network traffic \( S_j \), CPU utilization \( U_j \), memory utilization \( M_j \) and disk utilization \( D_j \). And then RA calculates load of each node by (2) according to these parameters.

2. **TA**
   - TA is the agent that triggers the negotiation scheduling mechanism in the model. The functions of TA includes managing tasks, computing the interdependency between task and node and the earliest completion time of tasks, transferring task scheduling information, receiving the number of target node, transferring tasks and receiving results.

   TA is responsible for supervising two task queues which are queue of tasks to be scheduled (Queue_Wait) and queue of tasks that has been allocated to the node (Queue_Allo). TA is also responsible for recording information of tasks, such as communication traffic, computation load and so on. According to the sum of communication traffic between the task to be scheduled and tasks in Queue_Allo, TA calculates the interdependency. With resource information from EA, TA calculates the earliest completion time of tasks. TA also cooperates with NA to send task scheduling information and receive the number of target node. And TA cooperates with IA to transfer tasks and receive results.

3. **IA**
   - IA is responsible for not only receiving task scheduling information, transferring information and results of the task. Between different agents in the same node, IA is responsible for not only receiving task scheduling information from NA and transferring information from TA, but also sending information received from other nodes to the corresponding agent according to its category. If it is task scheduling information, then it will be sent to NA for further process. And if it is result of a task, then the information will be sent to TA.

4. **EA**
   - The functions of EA include updating parameters timely, adjusting coefficients of parameters in dynamical evaluation rules, calculating evaluation value and sending it to NA. EA is responsible for updating the load, earliest completion time of the task and the interdependency. EA is also responsible for adjusting dynamical evaluation rules according to the running state and resources. EA obtains the load from RA, the earliest completion time of the task and the interdependency from TA. Then EA calculates the value of evaluation according to the information and sends it to NA for negotiation scheduling.

5. **NA**
   - The functions of NA include realizing DMOD and sending task scheduling information. When Queue_Wait is not empty, TA notifies NA to initiate negotiation scheduling. NA obtains evaluation value of the task which NA uses to negotiate with other nodes. Then NA gets the number of target node according to DMOD and sends it to TA.

   By cooperating with other agents, each node has independent decision-making ability. Each node owns complete negotiation scheduling mechanism and dynamical evaluation rules, which will not change with the increasing or decreasing number of nodes. The model can adapt to dynamical changes of the system and make the system have good reliability. In this model, RA provides a reliable basis by collecting resource information of nodes timely and makes the decision-making node achieve more accurate results.

## C. Negotiation Scheduling Mechanism

In the task scheduling model based on multi-agent, RA, TA, IA, EA and NA accomplish the negotiation scheduling mechanism cooperatively. The mechanism is triggered by TA. When the Queue_Wait is not empty, which means that there are tasks need to be scheduled, the mechanism is triggered to schedule the task.

When Queue_Wait is not empty, TA sends information of task \( t_i \) which needs to be scheduled to NA. And then NA releases task scheduling information to call for bids. Meanwhile, TA of the node obtains resource information and load from RA and calculates the interdependency and the earliest completion time of the task. As DMOD accomplished by NA selects target node based on the evaluation value, EA must obtain load from RA and get the interdependency and the earliest completion time of the task.
the task from TA when waiting for information of bidders. Then EA updates parameters in dynamical evaluation rules, calculates the evaluation value and sends it to NA. After receiving information of bidders, NA selects target node according to DMOD and sends the result to TA. Finally TA sends the task to target node through IA. When the task is finished, IA receives the result and sends it back to TA.

D. Dynamical Evaluation Rules

In the negotiation scheduling mechanism, evaluation values of bidders form a set. When NA is scheduling tasks, NA needs to refer to the set to accomplish the final decision. Therefore, dynamical evaluation rules are important for DMOD.

If the load or communication traffic exceeds the threshold, it will lead to an overall decline in system performance and task execution efficiency will be reduced. Meanwhile, in order to make tasks finish as soon as possible, following principles should be satisfied when making dynamical evaluation rules.

(1) Principle of reducing communication traffic
When communication traffic of the system is large, the influence of interdependency should be increased when calculating the evaluation value. It means that tasks should be allocated to the node which has the maximum interdependency in order to reduce communication traffic. Otherwise the system will be busy in communicating for a long time which will decrease communication speed and system performance.

(2) Principle of load balancing
When carrying on evaluating, load balancing should be considered to avoid some nodes being too busy or free which will waste resources. Therefore, the influence of load should be increased when calculating the evaluation value if the load is unbalanced. It will allocate tasks to the node which has lighter load.

(3) Principle of achieving minimum task completion time
The ultimate goal of scheduling is to finish tasks as soon as possible. Tasks should be allocated to the node which make tasks have minimum completion time. Meanwhile the load and communication traffic should also be considered.

According to above principles, we propose the dynamical evaluation rules with adjustment strategy. The definition of evaluation function is:

$$f_y = y_c + y_2 \cdot \frac{1}{t_{-end}} + y_3 \cdot \frac{1}{w} \cdot \left( y_1 + y_2 + y_3 = 1 \right)$$  (6)

$$f_y$$ represents the evaluation value of task $$t$$ on node $$h$$,
$$c$$ represents the interdependency between task $$t$$ and node $$h$$,
$$t_{-end}$$ represents the earliest completion time of task $$t$$ on node $$h$$,
$$w$$ represents load of node $$h$$. And $$y_1$$, $$y_2$$ and $$y_3$$ represent the coefficients of $$c$$, $$t_{-end}$$ and $$w$$ respectively.

EA sets initial values of the coefficients of $$c$$, $$t_{-end}$$ and $$w$$ according to running state and task information. Meanwhile EA sets the threshold of communication traffic $$C_{max}$$ and the threshold of load difference between nodes $$D_{max}$$. EA adjusts the coefficients dynamically according to the changes of communication traffic and load to accomplish dynamical scheduling. The adjustment strategy of coefficients in evaluation function is as follows:

(1) When communication traffic increases dramatically and exceeds $$C_{max}$$, reducing communication traffic between nodes should be considered. In this case, the interdependency between task and node becomes the main parameter that influences system performance. So we should increase its influence on the final evaluation value. Therefore when this case happens, $$y_2$$ will be increased by 0.1, meanwhile $$y_1$$ and $$y_3$$ will be decreased by 0.05 respectively. The coefficients will be adjusted iteratively until communication traffic drops below $$C_{max}$$. When communication traffic is below $$C_{max}$$ and $$y_2$$ is not the initial value, $$y_2$$ will be increased by 0.1 and $$y_3$$ will be decreased by 0.1, and $$y_2$$ and $$y_3$$ will be increased by 0.05 respectively until $$y_2$$ reaches the initial value.

(2) When the task has high requirements of real time, we should increase the influence of the earliest completion time on the final evaluation value. In this case, $$y_1$$ will be increased by 0.1 and $$y_2$$ and $$y_3$$ will be decreased by 0.05 respectively. The coefficients will be adjusted iteratively until the requirement of task finish time is satisfied. When the real-time need is met and $$y_2$$ is not the initial value, $$y_1$$ will be increased by 0.1 and $$y_3$$ will be decreased by 0.1, and $$y_1$$ and $$y_3$$ will be increased by 0.05 until $$y_2$$ reaches the initial value.

(3) When load difference between nodes exceeds $$D_{max}$$, load becomes the main parameter that influences system performance. In this case, $$y_2$$ will be increased by 0.1, meanwhile $$y_1$$ and $$y_2$$ will be decreased by 0.05 respectively. The coefficients will be adjusted iteratively until load difference drops below $$D_{max}$$. When load difference is below $$D_{max}$$ and $$y_3$$ is not the initial value, $$y_3$$ will be increased by 0.1 and $$y_1$$ and $$y_2$$ will be increased by 0.05 respectively until $$y_3$$ reaches the initial value.

The dynamical evaluation rules avoid performance degradation caused by sharp increase of communication traffic and can make load balanced. And it also considers the completion time without reducing task execution efficiency and improves the comprehensive performance of system.

III. DMOD

A. Descriptions of DMOD

In this paper, we propose a scheduling algorithm based on fastbid algorithm [14] combined with characteristics of agent. In this algorithm, the node which releases tasks is defined as a tenderer, waiting for information from other nodes which are defined as bidders.

In DMOD, if there are tasks to be scheduled, the tenderer releases bidding information to other nodes to call for bidding in order to select the target node. Bidders decide whether they participate in this round of bidding according to bidding strategy. If the node participates in this round of bidding, it sends bidding information to the tenderer and becomes a bidder. Then the tenderer selects the winner of bidders to be the target node and sends the task to it. When the task is completed, the bidder sends results back to the tenderer.
When the number of nodes in a distributed system is very huge, a large number of bidders in a round will make communication traffic of the system increase sharply and most of the bidders have small chance of winning the bidding. In this situation, system performance will be reduced. Therefore limiting the number of bidders in a round is very important. An effective bidding strategy is needed to make nodes determine whether they participate in current round of bidding effectively. In this paper, we present the bidding strategy as follows:

1. When bidding information arrives, the bidder estimates whether its current resource can meet resource needs of the task. If the needs can be satisfied, the bidder will carry on further estimation, or give up current round of bidding.

2. When resource needs can be satisfied, the bidder estimates whether to participate in this round of bidding according to running state of system. If there is no running task, then the bidder will participate in this round of bidding. Otherwise it will carry on further estimation.

3. If there are tasks running on the bidder, running state of the bidder will be analyzed according to the completion time of last task. In order to measure running state of the bidder, tolerance is defined as the threshold of good running state. When the ratio of actual task execution time to expected execution time of the task exceeds tolerance, we consider that performance of the bidder declines. The actual execution time of last task is defined as $t_{real}$ and the expected execution time of last task is defined as $t_{respect}$. When $(t_{real} - t_{respect})/t_{respect}$ is larger than tolerance, we consider that the task is not completed as expected, which means running state of the bidder is not very well. So the bidder gives up this round of bidding, otherwise the running state of the bidder is good enough to participate in this round of bidding.

The specific process of DMOD is:

1. Receive task scheduling information. When Queue_Wait of a node is not empty, the node is a tenderer and NA of the node receives task scheduling information from TA.

2. Initiate negotiation scheduling. When NA receives task scheduling information, NA initiates negotiation scheduling of task $t_i$ and sends the information to other nodes through IA.

3. Get the evaluation value on current node. EA updates parameters in dynamical evaluation rules according to $c_p$, $t_{end}$, and $w_i$ received from RA and TA. Then EA calculates the evaluation value by (6) and sends it to NA.

4. Determine whether to bid. The nodes that receive task scheduling information determine whether to participate in this round of bidding according to the bidding strategy. If the answer is yes, EA of the bidder updates parameters in dynamical evaluation rules according to $c_p$, $t_{end}$, and $w_i$ received from RA and TA. Then EA calculates the evaluation value by (6) and sends it to the tenderer. Otherwise the node gives up this round of bidding.

5. Select the target node. The tenderer finds the maximum evaluation value $f_u$ from the bidding information received from bidders. The node $u$ is determined to be the target node and the result will be sent to TA.

6. Task transmission. TA gets target node number $u$ from the received information. If node $u$ is the tenderer, the task will be executed on the tenderer and returns the result directly, then the scheduling ends. Otherwise the tenderer sends the task to node $u$ through IA.

7. Return the result. When the task is completed on target node, the result will be sent to TA of the tenderer. The scheduling ends.

B. DMOD

NA of each node is responsible for accomplishing DMOD.

Algorithm. DMOD

1. begin
2. while Queue_Wait!=NULL
3. get task $t_i$ from Queue_Wait
4. TA sends task information to NA
5. NA initiates negotiation scheduling
6. do
7. EA updates parameters, calculates the value of evaluation and sends it to NA
8. wait for bidding information from bidders, define the set of bidders as $N$
9. while $N$=!NULL
10. find the maximum $f_u$ from $f_i$ which is received
11. make node $n$ be target node
12. allocate task $t$ to node $n$
13. wait for the result of the task and send it to TA
14. end

IV. RESULTS AND DISCUSSION

A. Parameter Settings

Based on the model established in this paper, we make simulation experiments of DMOD. In the same conditions, we compare DMOD with MinMin and the algorithm based on tree structure (BTS) [15]. The number of nodes is defined as $n$ and the number of tasks is defined as $k$. When $n=10$, we set the number of tasks between [100, 300]. When $n=30$, we set the number of tasks between [200, 400]. And when $n=50$, we set the number of tasks between [400, 600].

During simulation experiments, we set the same number of nodes and tasks for three different algorithms. When setting parameters of nodes, computing power is set between [5GFlops, 50GFlops] and bandwidth is between [30Mbps, 50Mbps]. When setting the parameters of tasks, computation load is set between [100GFlop, 500GFlop], the number of tasks with communication is between [3, 7] and communication traffic with other tasks is set between [1M, 3M]. When making comparative experiments, three different algorithms are used to schedule the same set of tasks in same conditions.

B. Comparison of Task Completion Time

Establish the set of nodes and the set of tasks according to the ranges of parameters and compare task completion time of DMOD with MinMin and BTS. When $n=10$, the results are shown in Fig. 2. When $n=30$, the
results are shown in Fig. 3. And when \( n=50 \), the results are shown in Fig. 4.

From the figures, we can see that DMOD has better task completion time than MinMin except for the cases when \( n=10 \), \( k=160 \) and \( n=30 \), \( k=340 \). MinMin only considers task completion time without considering the influence of communication traffic and communication time. Therefore, DMOD obtains better results by considering task completion time, communication traffic and load comprehensively. However, because of the randomness of the set of nodes and the set of tasks, cases when \( n=10 \), \( k=160 \) and \( n=30 \), \( k=340 \) appear in a certain probability, which make MinMin get better task completion time than DMOD. In the scheduling process, BTS selects target node without considering computing power of nodes globally. So DMOD and MinMin get better task completion time than BTS.

From the figures we can also see that with the increase of tasks, DMOD and MinMin have a stable upward trend. But BTS has relatively large fluctuations, and along with the increase of nodes and tasks, the difference of task completion time between BTS and DMOD or MinMin gets larger.

Above all, DMOD proposed in this paper has better task completion time than MinMin and BTS by considering task completion time, communication traffic and load comprehensively. And with the increasing number of nodes, DMOD has a stable upward trend. Meanwhile, in the scheduling process, consideration of system load makes the system obtain a more stable running state.

C. Comparison of Communication Traffic

After the comparison of task completion time, we compare communication traffic of DMOD with MinMin and BTS base on the same set of nodes and tasks. The communication traffic includes not only the communication traffic between tasks, but also the communication traffic of sending tasks to the target node. Results of \( n=10 \) are shown in Fig. 5, results of \( n=30 \) are shown in Fig. 6 and results of \( n=50 \) are shown in Fig. 7.
From the figures we can see that DMOD has less communication traffic than MinMin. This is because in the scheduling process, MinMin only considers task completion time when selecting target nodes while DMOD considers not only task completion time but also the interdependency and the load. The larger the interdependency between task and node is, the higher the probability of allocating tasks to the node with little communication traffic is. Therefore, DMOD obtains less communication traffic than MinMin. And from the above comparison of task completion time, we can see that DMOD has less communication traffic while having better task completion time.

From Fig. 5 we can see that the communication traffic of DMOD is less than MinMin but larger than BTS when \( n=10 \). But along with the increasing number of nodes, the communication traffic of DMOD is less than both MinMin and BTS. This is because when the number of nodes is small, BTS has great probability of allocating the task to node which releases the task. So BTS gets less communication traffic. However, with the increasing number of nodes, the probability will decrease. So BTS no longer has an advantage in communication traffic. And from the comparison above, we can see that task completion time of BTS is much worse than the other two algorithms, which means it has higher time cost.

From the figures we can see that with the increasing number of nodes, three algorithms have a stable upward trend on communication traffic. When the number of nodes is determined, the communication traffic of system will increase with the increasing number of tasks.

In summary, DMOD proposed in this paper has less communication traffic than MinMin and BTS without sacrificing task completion time. By adding the load as a parameter in evaluation function and dynamically adjusting the coefficients of different parameters, better scheduling results can be achieved.

V. CONCLUSION

In this paper, we established a distributed task scheduling model based on multi-agent. The model includes RA, TA, IA, EA and NA. Each agent is responsible for different functions and the agents accomplish scheduling of tasks by negotiation scheduling mechanism. Then we proposed DMOD based on the model. Each node has independent decision-making ability in the algorithm, which can not only increases reliability of the system, but also meets dynamic characteristic and uncertainty of the system better. The dynamical evaluation rules in negotiation scheduling mechanism consider task completion time, communication traffic and the load comprehensively. Therefore sharp increase of communication traffic is avoided as well as performance degradation of the system. System stability and task execution efficiency are improved. In the future, the negotiation scheduling mechanism in the model remains to be further optimized and improved. Meanwhile when scheduling tasks the influence of dependencies between tasks should be concerned.

REFERENCES


Zhanjie Wang is an associate professor in the Department of Computer Science and Technology at Dalian University of Technology. His research interests include networks, file system, task scheduling and image processing. He has worked at Dalian University of Technology since 1986. He has filed several patents in China and has published many papers.

Ting Fang is a postgraduate in Dalian University of Technology. She majors in Computer Science and Technology. She earned her bachelor's degree in Computer Science and Technology from Dalian University of Technology in 2007. Her research interests are networks and task scheduling.