An Agent Structure Model Suitable for Parallel Computing

Hao Liu\textsuperscript{a}, Hong Zhou\textsuperscript{b} and Lei Ma\textsuperscript{c}

Department of Electronic Information Engineering, Jincheng College of Sichuan University, 611731, Chengdu, China

\textsuperscript{a}liuh6679@163.com, \textsuperscript{b}307113521@qq.com, \textsuperscript{c}1774562676@qq.com

Abstract. Network communication system will simultaneously deal with a large number of communication incidents in the operation process, so the agent in the software needs to carry on a plurality of tasks at the same time, this will produce a conflict when it call the system resource. Based on the distributed computing model and agent technology, this paper presents a multi-agent parallel computing model based on KM (Kuhn Munkras) algorithm, this model can better solve the conflict between agents multitasking and the limited system resource, test experiment proved the effect of this model.

1. Introduction

With the development of network technology, distributed computing has played an increasingly important role in today's computer technology, it examines how to divide a single problem that requires enormous computing power into many small parts, then assign these sections to multiple computers or different sections of a computer for calculation to get the final result.

Specifically, distributed computing is the sharing of information between two or more pieces of software that run on the same computer or on multiple computers that are networked together.

Traditional distributed computing technologies are mostly based on C/S mode and communicate remotely via remote procedure call (RPC) or message passing, it is suitable for a stable network environment, far from being able to adapt to the rapid development of today's changing network applications. The agent technology is a new distributed computing model, which largely solved the problem \cite{1, 2}.

2. Brief description of multi-agent parallel computing

Agent is a software entity with autonomy, cooperative ability and goal-driven features. It can running in the network carrying the code and state autonomously from one node to another node, find the appropriate computing resources to complete a specific task \cite{3}. Agent model has the characteristics of autonomy, reactivity, initiative, interaction and mobility, it can effectively reduce the network load in distributed computing, improve communication efficiency, support offline computing, support asynchronous and autonomous interaction, so it becoming the most popular distributed computing mode \cite{4}.

There are many agent parallel processing methods: mobile agent technology; using multi-agent technology to achieve parallel computing \cite{5, 6}; on the basis of agent technology through distributed parallel algorithm to improve data mining capabilities \cite{7, 8}.

Utilizing the idea of geometric graph theory, we constructs an agent model suitable for parallel computing, it can solve the contradiction between the need of agent and the limited system resource.
3. Agent parallel computing model based on KM algorithm

When multiple agents perform distributed parallel computing, they encounter this inconsistency: one agent needs to perform multiple tasks simultaneously, but in the system, different tasks correspond to different system resources, so the same task require the same resource, however, in this multi-tasking system, each agent accepts more than one task, therefore, sometimes there are multiple agents involved in the same task, this can cause conflicts when calling the system repository.

To solve the problem, this paper draws on the algorithm ideas of graph theory in geometry.

3.1. KM Algorithm in Graph Theory

The bipartite graph in graph theory is defined as: Let \( H = (V, E) \) be a set of graphs. If the vertex \( V \) can be divided into two subsets (A, B), and the two vertices \( i \) and \( j \) associated with each edge \((i, j)\) in the graph respectively belong to the two different vertex sets (\( i \) in A, \( j \) in B), then the view \( H \) is a bipartite graph.

Given a sub-graph \( N \) of a bipartite graph \( H \), if any two sides of the edge set of \( N \) are not attached to the same vertex, we call \( N \) is a match. The match that contains the largest number of edges is called the graph's maximum-weight match.

If each vertex in the graph is associated with an edge in the graph, the maximum-weight match is a perfect match.

If \( K \) is a path in graph \( H \) that is connected to two unmatched vertices and alternating between the edges belonging to \( N \) and the edges not belonging to \( N \) (i.e matched and to-be-matched edges) in \( K \), then let \( P \) be an augmenting path relative to \( N \).

We get the maximum weight of the graph by the following method:

**Theorem 1:** Suppose the mark of the vertex \( x_i \) in the bipartite graph \( H \) is \( A[i] \), the mark of the vertex \( y_j \) is \( B[j] \), and the edge weights between vertex \( x_i \) and vertex \( y_j \) is \( W[i, j] \) \( i \in Z, j \in Z \). If all the edges in the sub-graph \( N \) of graph \( H \) are satisfied \( A[i] + B[j] = W[i, j], i \in Z, j \in Z \). \( M \) is called the equivalent sub-graph of \( G \).

**Theorem 2:** If all the sub-graphs which consist of the bipartite graph that satisfy \( A[i] + B[j] = W[i, j] \) (equal sub-graph) have a perfect match, then this perfect match is the maximum-weight match of the bipartite graph.

The following is the diagram of seeking augmenting path

![Diagram of seeking augmenting path in the bipartite graph.](http://blog.csdn.net/lc200808)

**Figure 1.** Seeking augmenting path in the bipartite graph.

In this algorithm, in order to make the formula \( A[i] + B[j] = W[i, j] \) established, let's make \( A[i] \) be the maximum- weight of all the edges associated with the vertex \( x_i \) and \( B[j] = 0 \).
If there is no perfect match for the equal sub-graphs of the current bipartite graph, the vertex coordinates will be modified in the following way to expand the equal sub-graphs, modify it again until the equal sub-graph has a perfect match.

The following is the Algorithm flow:

1. Set the value of the mark \( iA \) to be the maximum weight of all edges associated with the vertex set \( X \). Set the value of the mark \( jB \) to 0, then judge the equation \( iA + jB = W[i, j] \) is established or not. If it is true, it indicates that there is a complete match for the equal subgraphs of the current bipartite graph; On the contrary, if it is not established, it indicates that it does not have a complete match, and then enters the second step.

2. Modify the mark and expand the subgraph: Modify the value of the mark in the bipartite graph \( H \), expand the subgraph and make another judgment, the specific method is: In the bipartite graph \( H \), reduce the mark value of all the vertices belonging to the set \( X \) on the augmentation path by \( C \), and increase the mark value of all the vertices belonging to the set \( Y \) on the augmentation path by the same value \( C \), then:
   
   (1) For those edges \((i, j)\), where both ends are in an augmented path, or edges where both ends are not in an augmented path, the value of \( iA + jB \) will keep no change. It means that original edge of the equal sub-graph is still the equal sub-graph, those are not still not.
   
   (2) For those edges \((i, j)\), where the X-end is not in the augmentation path and the Y-end is in the augmentation path, the value of \( iA + jB \) will increase, it means that this edge is originally not in the equal sub-graph, it is still not in the equal sub-graph.
   
   (3) For those edges \((i, j)\), where the X-end is in the augmentation path and the Y-end is not in the augmentation path, the value of \( iA + jB \) will decrease, it means that this edge is originally not in the equal sub-graph is possible to enter this equal sub-graph now, and the equal sub-graph is therefore enlarged.

3. Find the added value \( C \) of each iteration In order to ensure that the equality \( iA + jB = W[i, j] \) be established, and there is one edge (or multiple edges) in the equal sub-graph, \( C \) should equal to:
\[
\text{Min}\{iA[j] + jB[i] - w[i, j]\},
\]
where, \( iA \) in the augmentation path, \( jB \) is not in the augmentation path.

The above is the KM algorithm.

3.2. Agent Parallel Computing Model Based on KM Algorithm

In our system, assume \( Agent = \{Ag_1, Ag_2, \ldots, Ag_n, \ n \in Z\} \), represents the agent collection, \( Lib = \{Lib_1, Lib_2, \ldots, Lib_n, n \in Z\} \), represents the collection of repository. \( \langle Agents, Libs \rangle \) represents the syndication relationship between agent and repository, we mapping this relationship to geometric graph theory and building a model, then \( \langle Agents, Libs \rangle \) corresponding to the two sets of vertices in the bipartite graph, so if we can find the maximum weight matching of the graph, it is to maximize the use of the system resource library without conflicting agents.

According to the KM algorithm, the steps to find the maximum weight match are as follows:

1. Set the mark of the vertex set \( Agent = \{Ag_1, Ag_2, \ldots, Ag_n, n \in Z\} \) to agent \([i]\), let the value of agent \([i]\) be the maximum weight of all edges associated with the vertex agent, the mark of the vertex set \( Lib = \{Lib_1, Lib_2, \ldots, Lib_n, n \in Z\} \) to be zero, the edge weight between the vertex agent and Lib is \( W[i, j] \), during the execution of the following algorithm, for any edge, the equation \( Agent[i] + Lib[j] \geq W[i, j] \) is always true.

2. Find the perfect match of the sub-graph formed by the vertex set agent \([i]\) and lib \([j]\):
   
   1. Set the sub-graph \( N \) of the bipartite graph \( H \) to null.
   2. Find an augmentation path \( P \) in the sub-graph \( N \), with the XOR algorithm inverts it and try to find a bigger match \( N' \) in the bipartite graph \( H \) instead of \( N \)
3. Repeat (2), iteratively again, until you no longer find the augmentation path to replace the previous one.

(3) If it still fails to find the best match, we reduce the value of the mark of the vertex agent \([i]\) that already exists in the augmentation path by the value \(d\) and increase the value of the mark of the vertex \([j]\) by the value \(d\), \(d\) must meet 
\[
\text{Agent}[i] + \text{Lib}[j] - w[i,j] \leq 0,
\]
among them, agent \([i]\) is in the augmentation path and \([j]\) is not in the augmentation path.

(4) Repeat algorithm 2, 3, and iterate until it finds the complete match of the equivalent sub-graph, this is the maximum weight matching of the original graph. It means we establish an agent parallel processing model in the system, solve the contradiction between the multi-task handling agent and limited system resource library to the maximum extent.

4. Experiment
We used the following experiment to verify the advantage of this parallel structure.

First, we develop a middleware software, which is an independent system software with a corresponding level between the application software and the system infrastructure software. It can play an intermediate “bridge” between different parts of the system. This software can use the parallel agent structure described above, in the experiment, we use this middleware to adopt the parallel agent structure and the non-parallel agent structure respectively, test the time spent by middleware for system messaging in both cases to determine the effect of this parallel agent structure.

Second, we create a test environment as shown below.

![Testing environment](image)

**Figure 2.** Testing environment

In the experiment, the middleware was deployed in the server and the server was connected to multiple clients at the same time. We use RPC (Remote Procedure Call) to send various messages from the client to the server, and then test the average reception time. Because there are multiple clients sending messages to the server at the same time, the server needs to handle multiple events at the same time, so, we can verify the parallel processing capabilities of the middleware.

The experiment is divided into two parts. First, multiple clients send message queues to the server at the same time. The specific formats of the messages include text, pictures, objects, streaming media files, byte stream data, etc. The second is that the server sends message queues to multiple clients at the same time. The specific format of the messages also includes text, pictures, objects, streaming media...
files, and byte stream data. The time spent sending and receiving messages from the server was tested using our parallel Agent architecture and using the generic Agent architecture, respectively.

The experimental results are shown below:

![Figure 3. Average time for the server to receive client messages without parallel structure](image1)

![Figure 4. Average time for the server to receive client messages with parallel structure](image2)

It can be seen from the experiment that the software with the parallel agent structure reduces the systems time-consuming in message delivery.

5. Conclusion
The multi-agent parallel computing model based on KM algorithm solves the contradiction between agent multi-task processing and limited system resource library to some extent. This paper gives the method of realization of this computing model. Further optimization and practical application need further study.

Acknowledgments
This work was financially supported by Sichuan Provincial Education Department research fund (18ZB0357)

References
[1] Feng H, Wang L, Wang J H. A distributed parallel computing mode based on mobile agent. Journal of Computer Technology and Development [J]. 2012, Vol21, No.2, 70-74.
[2] Attlys H, Welch, J. Distributed computing [M], Second Edition. Beijing, Electronic Industry Press, 2008.
[3] Chen S B, Chao J, Li M L. Research of distributed software system framework based on Agent
[4] Yu W G, Wang W P. Li Q. Review of parallel agent simulation [J]. Journal of System Simulation, 2012 Vol.24, No.2 245-250.

[5] Liu D Y, Foreword of special issue of agent theory and technology [J]. Journal of Software, 2011 No.11.

[6] Zhou W P, Tang H X, Ji Z Z. Research progress in simulation of parallel groups. Journal of Computer Application, 2013 Vol.30 No.1 30-35.

[7] David C, Colin H, Aaron K. Mobile Agents: Are They a Good Idea?. Research Report, IBM Research Division, T.J.Watson Center, 1995.

[8] Mi Z K, Jiang X H, Lei L. A stable parallel distributed frequent set mining algorithm and its application [J]. Computer application and software, 2011, 28 (3): 83 - 85, 124.