Cooperative Communication System for Parallel Computing Agents in Multi-cluster Grid

Qingkui Chen, Wei Wang
School of Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
chenqingkui@tom.com

Abstract. The idle computational resources of CSCW environment that is composed of computer clusters are mined to construct the multi-cluster grid in order to support the computation-intensive tasks. For fitting the state changes of idle computing resources during the computing process, the techniques of cooperation and migration agents are adopted. Through the concurrent dataflow, the supper element, and the density factor, the dynamic buffer pool (DBP) was built up. By using of the Grid techniques, the cooperative computing agent, the cooperative communication agent, and DBP, a cooperative communication system (CCMS) for parallel computing agent was designed and implemented. The experimental results show that CCMS can increase the speed-up of parallel computing task. It can be fit for the computation in CSCW based on Internet.

Keywords: cooperative communication agent; dynamic buffer pool; multi-cluster grid; parallel computing

1 Introduction

With the rapid development of information techniques and Internet, the research results based CSCW became the key techniques building enterprise information infrastructure[1].The computer aided design based on cooperative work environments is playing the more and more important role in the business behavior of enterprise today. Especially, CSCW work contains a lot of computation-intensive task to be processed in some high performance computes. On the other hand, Intranet is more and more extended and a great deal of cheap personal computers are distributed everywhere, but the use rate of their resources is very low. The researches of papers [2, 3] point out that many resources is idle in most network environments at a certain time. Even it is the busiest time of a day, still one third of their workstations weren't used completely. So the paper [4] proposed a framework how to min and use the idle computational resources of CSCW environment that composed of multi-clusters connected by Intranet. It uses the idle computer resources in CSCW environment to construct a Visual Computational System (VCE). VCE can support two kinds of migration computations: (1) the Serial Task based on Migration (STM); (2) the Task based on Data Parallel Computation (TDPC).

For adapting these heterogeneous and dynamic environments, we use the Grid [5] techniques, multi-agent [6, 7] techniques and cooperative leaning model of agent [8] and min the idle resource of CSCW environment to construct multi-cluster grid (MCG). Because of the migration of cooperative computing agent in MCG, the communication resources in computing node are changed frequently. So the communication efficiency is decreased gradually along with the increasing of cooperative computing team scale. The study of communication techniques [9~12] becomes very important. Paper [9] proposed a synchronous communication mechanism for parallel computing. Paper [10,11] study the mobile agent communication problem. Paper [12] implements a middleware for agent communication. But all these methods can't satisfy the need of parallel computing in dynamic multi-cluster grid.

This paper proposed a cooperative communication system for parallel computing agents in multi-cluster grid that is composed of many computer-clusters connected by Intranet. By using of dynamic buffer blocks and concurrent dataflow technique, we construct the dynamic buffer pool in every computing node; through the cooperative communication agent, global directory and dynamic buffer pool, we build up a independent cooperative communication system in MCG. This system can support the grid computing and pervasive computing based on task-migratory mechanism, and it can fit the heterogeneous and dynamic network Environment. The experimental results show that this model can increase the speed-up of parallel cooperative computing and use effectively the memory resources for communication buffers.

2 Architecture of MCG

We use of the idle computational resources (e.g. the time that the users take a rest or do the other works and the computer is idle) in CSCW environment, to support the computation-intensive tasks of CSCW. The Computation-Intensive Task is that it needs very long time to run, and it also needs high performance computer to support, such as the finite element analysis. For describing this communication model, we introduce some definitions:

(1) Computing Node (CN) is defined as *CN* (*id*, *CT*, A_m, *CMA*, *AS*), where *id* denotes the identifier of *CN*; *CT* denotes the type of computing node; A_m denotes the main control agent of *CN*; *CMA* is the cooperative communication agent of *CN*; *AS* is the set of the control and computing agents running on *CN*.

(2) Computer cluster (CC) is defined as *CC* (M_a, *CS*), where M_a denotes the main computer of *CC*; *CS*= {CN_1, CN_2…CN_p} denotes the set of all computing nodes which *CC* includes.

(3) Computing Agent (CA) is defined as *CA* (*id*, *PRG*, *BDI*, *KS*, *CE*), where *id* denotes the identifier of *CA*; *PRG* denotes the executable program set of *CA*; *BDI* is the description of its *BDI*; *KS* is its knowledge set; *CE* is its configuration environment.

CA is the basic element to execute computation task. If a *CA* could complete independently the task, we call it as the independent computing agent (*ICA*). If a *CA* couldn't complete independently the task, and it must cooperate with others, we call it as the cooperative computing agent (*CCA*).

(4) Cooperation Computing Team (CCT) is defined as *CCT* (*id*, A_m, *CAS*, *BDI*, *CKS*, *CCE*), where *id* denotes the identifier of *CCT*; A_m denotes the main control agent of *CCT*; *CAS* denotes the set of all cooperative computing agents which *CCT* includes; *BDI* is the description of its *BDI*; *CKS* is its knowledge set. *CCE* is its configuration environment. *CCT* can support the data parallel computation in the logical computer cluster.

(5) Global Computing Group (GCG) is defined as *GCG (id*, A_m, *ICAS*, *CCTS*, *GKS*, *GCE*), where *id* denotes the identifier of *GCG*; A_m denotes the main control agent of *GCG*; *ICAS* denotes the set of *ICA* which *GCG* includes; *CCTS* denotes the set of *CCT* which *GCG* includes; *GKS* is its knowledge set. *GCE* is its configuration environment.

Many tasks are executed together in *GCG* during the same time, and the tasks are calculated by a lot of *ICA*s or *CCT*s.

(6) Cooperative Communication Agent (CMA) is defined as *CMA* (*id*, *CN*, *DBP*, *BDI*, *CKS*, *CCE*), where id denotes the identifier of *CMA*; *CN* denotes the computing node on which *CMA* runs; *DBP* is the dynamic communication buffer pool of *CMA*; BDI is the description of its *BDI*; *CKS* is its knowledge set. *CCE* is its configuration environment. Every a computing node has a *CMA*, and *CMA* do all communication work for computing and control in the computing node.

(7) Cooperative Communication System (CCMS) is defined as *CCMS* (*CCAS*, *GD*), where *CCAS* denotes the set of all *CMA*s in a parallel computing environment composed of many computing node ; *GD* denotes the global directory of *CCMS*.

(8) Multi-cluster Grid (MCG) is defined as *MCG* (M_a, *CCS*, *N*, *R*, *GCG*, *CCMS*), where M_a denotes the master computer of *MCG*; *CCS* denotes the set of all computer clusters which *MCG* includes; *N* is the connection network set of *MCG*; *R* is the rules of connections; *GCG* is the global computing group; *CCMS* is the cooperative communication system.

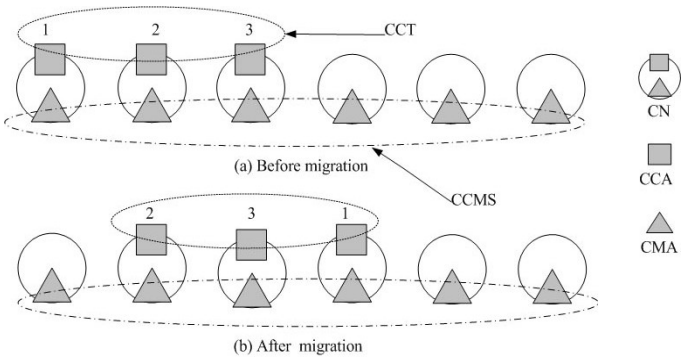

Fig. 1. The example of the relation between CCT and CMAS in MCG.

CMA is the only communication component in a computing node and *CMA* is on this computing node forever. During the computing process, the migration component only is the cooperative computing agent. The figure 1 shows that the cooperative computing agent 1 has been migrated from computing node 1 into computing node 4. But the *CMA* of CN_1 is still in CN_1.

3 CCMS Structure

(1) Buffer block (BB) is defined as a communication unit, and it is composed of a certain size memories. Its size be defined as |B|.

(2) Concurrent Dataflow (CDF) There is a communication relation between cooperative computing agent CCA_i and CCA_j ($i \neq j$), C_i is a *CN* on which CCA_i runs, C_j is a *CN* on which CCA_j runs. CMA_i and CMA_j are two cooperative communication agents which run on C_i and C_j respectively. DF_{ij} is the dataflow from CCA_i to CCA_j, DF_{ji} is the dataflow form CCA_j to CCA_i. During the computing and migration process, there are many dataflow among the computing agents at the same time. These dataflow are call as concurrent dataflow.

A dataflow DF_{ij} is a sequence of buffer blocks, and it has a density factor k ($k \geq 0$) that is the buffer block numbers from CCA_i to CCA_j in a unit time. When k is 0, DF_{ij} is closed.

(3) Dynamic Buffer Pool (DBP) is defined as *DBP* (m, *RM*, *RFC*, w_r, *SM*, *SFC*, w_s), and its main parts are described as follows:

- m is the numbers of cooperative communication agents which communicate with C_i; there are $2m$ data flows for C_i . There are m data flows for sending and m data flows for receiving.
- *RM* is $m \times w_r$ matrix and it is the receiving buffers that store data received from other m concurrent data flows; it is called as receiving buffer matrix. Its matrix element is called as supper element, and it has two formats: ① null; ② compose of k ($k \geq 0$, k is the density factor) buffer blocks that size is |B|; w_r is the numbers of column of *RM*.

Every a row of *RM* is a dataflow that save data received from other cooperative computing agent. So *RM*[i] ($1 \leq i \leq m$) is the ith receiving dataflow.

- *RFC* is m vector and its element is the density factor of dataflow; Namely, *RFC*[i] is the density factor of dataflow *RM*[i].
- *SM* is $m \times w_s$ matrix and it is the sending buffers that store data sent into other m concurrent data flows, and it is called as sending buffer matrix. Its matrix element is also supper element; w_s is the numbers of column of *SM*.
- *SFC* is m vector and its element is the density factor of dataflow; Namely, *SFC*[i] is the density factor of dataflow *SM*[i].

(4) Global Directory（GD） is defined as a relation table *GD* (CCA_{id}, CN_{id}, CCT_{id}, *sta*), where CCA_{id} denotes the identifier of *CCA*; CN_{id} denotes the identifier of computing node on which *CCA* runs; CCT_{id} is the identifier of *CCT*

which *CCA* belongs to; *sta* is the state of *CCA*, and *sta*∈{'ready', 'computing', 'migration'}.The row numbers of *GD* be written as *p* and its means is that there are *p* cooperative computing agents in *MCG* at present.

(5) Buffer Block Set（**BBS**）is the set of all communication buffer blocks in a computer node, and its size is $|B| \times N_B$.

CCMS structure is shown as the figure 2. There are three cooperative computing agents distributed into three computing nodes. Figure 2 shows that communication method of these cooperative computing agents through CCMS.

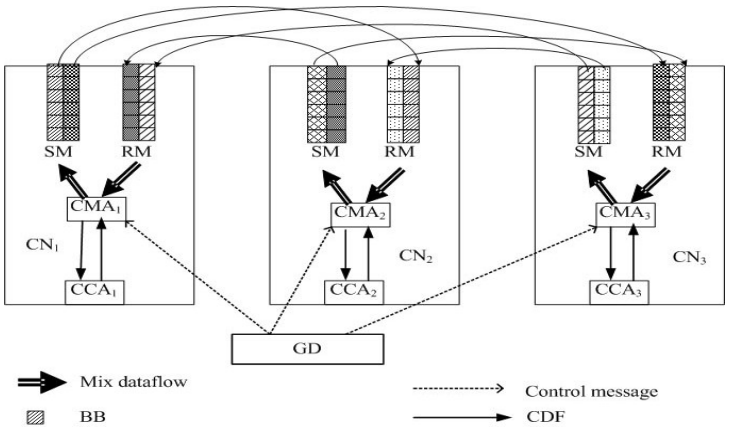

Fig. 2. The structure of CCMS composed of three computing nodes.

4 Descriptions of CCMS Process

4.1 CCMS Start-up

Algorithm 1 (Start-up and Calculation of CCT)

(1) *GCG* gets a parallel job and constructs a *CCT*= {CCA_1, CCA_2,…,CCA_t};the identifier of *CCT* is *cno*;

(2) *GCG* gets a logical computer cluster *LCC*= {C_1, C_2… C_t} from *MCG*;

(3) For $1 \leqslant i \leqslant t$, *GCG* do step(4) repeatedly;

(4) send CCA_i into C_i; append a new tuple *NT* into *GD* and do as follows: { $NT.CCA_{id}$= identifier (CCA_i); $NT.CN_{id}=C_i$; $NT.CCT_{id}=cno$; *NT.sta*='ready';}

(5) *GCG* send 'initialization' to CMA_i ($1 \leq i \leq t$), which CMA_i is the cooperative communication agent in computing node C_i;

(6) CMA_i ($1 \leq i \leq t$) does algorithm 2 in parallel in order to initialize the dynamic buffer pool on computing node C_i;

(7) *GCG* start all cooperative computing agents CCA_1, CCA_2,… CCA_t to complete the parallel job cooperatively;

(8) If the migrations occur during the computing process, *GCG* starts migration process [4] and all *CMA*s do algorithm 3 to adjust CCMS;

(9) Repeat to do step (8) (9) until the parallel job has been finished.

4.2 Operation for CCMS

Algorithm 2 (Initialization of DBP)

CCT= {CCA_1, CCA_2…CCA_t} is a cooperative computing team and LCC= {C_1, C_2… C_t} is the logical computer cluster which is supporting CCT. GD is global directory. For initializing the dynamic buffer pools of CCMS, all CMA_i ($1 \le i \le t$) do as follows cooperatively each other.

(1) For ($1 \le j \le t \wedge j \ne i$) do (2)~(5);

(2) CCA_i calculates the data size ds which will be sent into CCA_j ;

(3) CCA_i calculates the density factor k_{ij} of dataflow from CCA_i to CCA_j by ds; CMA_i constructs dataflow DF_{ij} (from CCA_i into CCA_j) according to ds and adds DF_{ij} to SM of CMA_i; CMA_i sends ds into CMA_j; CMA_j constructs dataflow DF_{ij} (from CCA_i into CCA_j) according to ds and adds DF_{ij} to RM of CMA_j;

(4) CCA_j calculates the density factor k_{ji} of dataflow from CCA_j to CCA_i by ds; CMA_j constructs dataflow DF_{ji} (from CCA_j into CCA_i) according to ds and adds DF_{ji} to SM of CMA_j; CMA_j sends ds into CMA_i; CMA_i constructs dataflow DF_{ji} (from CCA_j into CCA_i according to ds ; CMA_i adds DF_{ji} to RM of CMA_i;

(5) CMA_i gets the tuple NT form GD by CCA_i and inform CCMS to replace $NT.sta$ with ‘ready’;

(6) End.

Algorithm 3 (Adjustment for CCMS after migration)

CCT={CCA_1, CCA_2… CCA_t} is a cooperative computing team and LCC={C_1, C_2… C_t} is the logical computer cluster which is supporting CCT. GD is global directory. Suppose that MIG={CCA_{m1}, CCA_{m2}… CCA_{mg}} is the subset of CCT and all cooperative computing agents in MIG will be migrated from $LEAVE$= {C_{m1}, C_{m2}… C_{mg}} into new computing node set NLC={NC_{m1}, NC_{m2}…NC_{mg}}. There is $LEAVE \subset LCC$. For simple description, suppose that CCA_{mi} ($1 \le i \le g$) will be migrated into computing node NC_{mi}. When all cooperative computing agents in MIG finished their migration process, for adjusting the dynamic buffer pools of CCMS, all CMA_{mi} ($1 \le i \le g$, CMA_{mi} is the cooperative communication agent of computing node C_{mi}) do as follows cooperatively each other.

Set $NMIG$=CCT-MIG; /*agent set of no-migration*/

S_{nmig} is the set of cooperative communication agents of computing nodes in LCC-$LEAVE$; S_{mig} is the set of cooperative communication agents of computing nodes in NLC; S_{leave} is the set of cooperative communication agents of computing nodes in $LEAVE$;

/*Delete the old dataflow of migration agents from CCMS*/

(1) For $CMA_j \in S_{leave} \wedge 1 \le j \le g$ do (2)(3);

(2) CMA_j gets its CCA_j from MIG and informs CCMS to replace $GD.sta$ (which CCA_j has) with ‘migration’;

(3) For $CMA_i \in S_{nmig} \wedge 1 \le i \le t\text{-}g$ do {CMA_i delete the dataflow DF_{ij} from its *DBP*; CMA_j delete the dataflow DF_{ji} from its *DBP* ;}

/* Construct new dataflow among CCAs in *MIG* */

(4) For $CMA_i \in S_{mig} \wedge 1 \le i \le g$ do (5);

(5) For $CMA_j \in S_{mig} \wedge j \ne i \wedge 1 \le i \le g$ do (6) (7);

/* CCA_i is active operator */

(6) CCA_i calculates the data size *ds* which will be sent into CCA_j ; CCA_i calculates the density factor k_{ij} of dataflow from CCA_i to CCA_j by *ds*; CMA_i constructs dataflow DF_{ij} (from CCA_i into CCA_j) according to *ds* and adds DF_{ij} to *SM* of CMA_i; CMA_i sends *ds* into CMA_j; CMA_j constructs dataflow DF_{ij} (from CCA_i into CCA_j) according to *ds* and adds DF_{ij} to *RM* of CMA_j;

/ * CCA_j is active operator */

(7) CCA_j calculates the density factor k_{ji} of dataflow from CCA_j to CCA_i by *ds*; CMA_j constructs dataflow DF_{ji} (from CCA_j into CCA_i) according to *ds* and adds DF_{ji} to *SM* of CMA_j; CMA_j sends *ds* into CMA_i; CMA_i constructs dataflow DF_{ji} (from CCA_j into CCA_i) according to *ds* and adds DF_{ji} to *RM* of CMA_i; }

/* Construct new dataflow between *NMIG* and *MIG* */

(8) For $CMA_i \in S_{mig} \wedge 1 \le i \le g$ do (9);

(9) For $CMA_j \in S_{nmig} \wedge 1 \le i \le t\text{-}g$ do (10);

(10) CCA_i calculates the data size *ds* which will be sent into CCA_j ; CCA_i calculates the density factor k_{ij} of dataflow from CCA_i to CCA_j by *ds*; CMA_i constructs dataflow DF_{ij} (from CCA_i into CCA_j) according to *ds* and adds DF_{ij} to *SM* of CMA_i; CMA_i sends *ds* into CMA_j; CMA_j constructs dataflow DF_{ij} (from CCA_i into CCA_j) according to *ds* and adds DF_{ij} to *RM* of CMA_j; CCA_j calculates the density factor k_{ji} of dataflow from CCA_j to CCA_i by *ds*; CMA_j constructs dataflow DF_{ji} (from CCA_j into CCA_i) according to *ds* and adds DF_{ji} to *SM* of CMA_j; CMA_j sends *ds* into CMA_i; CMA_i constructs dataflow DF_{ji} (from CCA_j into CCA_i) according to *ds* and adds DF_{ji} to *RM* of CMA_i;

(11) CMA_i ($CMA_i \in S_{mig} \cup S_{nmig}$) gets the tuple *NT* form *GD* by CCA_i and inform CCMS to replace *NT.sta* with 'computing';

(12) End.

4.3 CCMS communication process

CCT= {CCA_1, $CCA_2 \ldots CCA_t$} is a cooperative computing team and *LCC*= {C_1, $C_2 \ldots C_t$} is the logical computer cluster which is supporting *CCT*. *GD* is global directory. CMA_i ($1 \le i \le t$) is the cooperative communication agent of computing node C_i. Suppose that CCA_i ($1 \le i \le t$) had been allotted on C_i. Set *SCMA*= {CMA_1, $CMA_2 \ldots CMA_t$}.

Algorithm 4 (Sending data process)
The sending data process includes two threads: one is the data gather thread (DGT) from local CCA; another is the data sending thread (DST) through network adapter. DGT process is as follows:

(1) Do step (2) (3) repeatedly;

(2) CCA_i does the local computing work and produces dada d; CCA_i transfer d to local cooperative communication agent CMA_i;

(3) CMA_i decides the destination $CCA_j(1\leq i\leq t,\ j\neq i)$ by d and gets k (k is density factor)buffer blocks B_1, $B_2 \ldots B_k$ from BBS ; CMA_i saves d into B_1, $B_2 \ldots B_k$ and builds up a supper elements B={ B_1, $B_2 \ldots B_k$ }; CMA_i adds B into dataflow DF_{ij} (that is dataflow from CCA_i to CCA_j);

DST process is as follows:

(1) Do step (2) (3) repeatedly;

(2) Scan all dataflow DF_{ij} ($1\leq i\leq t$, $1\leq j\leq t$) of SM in DBP;

(3) If DF_{ij} is not NULL then {get data d from DF_{ij} ; send d into CMA_j which runs on C_j; free the buffer blocks;}

Algorithm 5 (Receiving data process)
The receiving data process includes two threads: one is the data receiving thread (DRT) from network; another is the data transfer thread (DTT) that transfer data to local CCA. DRT process is as follows:

(1) Do step (2) ~ (4) repeatedly;

(2) Receive data d from the network adapter;

(3) Decide the data source CCA_j by d;

(4) get the density factor k of dataflow DF_{ji} from CCA_j to CCA_i; get k buffer blocks B_1, $B_2 \ldots B_k$ from BBS; Save d into B_1, $B_2 \ldots B_k$ and build up a supper elements B= {B_1, $B_2 \ldots B_k$} ; add B into dataflow DF_{ji} of RM;

DTT process is as follows:

(1) Do step (2) (3) repeatedly;

(2) Scan all dataflow DF_{ij} ($1\leq i\leq t$, $1\leq j\leq t$) of RM in DBP;

(3) If DF_{ij} is not NULL then {get data d from DF_{ij} and free the buffer blocks which d used; transfer d into CCA_i which runs on C_i;}

4.4 Computation for density factor

Suppose that CCA_i and CCA_j is a pair of cooperation computing agents and DF_{ij} is the dataflow from CCA_i to CCA_j. The calculation process for density factor k of DF_{ij} is as follows:

(1) CCA_i estimates the communication data size s from CCA_i to CCA_j per unit time;

(2) $k=\lfloor s / |B| \rfloor+1$.

5 Analysis and Experiments

5.1 Analysis for CCMS

CCMS has many characteristics, they are as follows:

(1) Any a row of the matrix *SM* and *RM* of *DBP* is a concurrent dataflow that is a logical channel between a pair of cooperative computing agents. CCMS use the Matrix *SM* to implement the one-to-many communication; CCMS use the Matrix *RM* to receive message from other many cooperative computing agents which runs on other computing node. So CCMS can support the parallel computing.

(2)Every concurrent dataflow has a density factor. The size of density factor is the communication data quantity of concurrent dataflow. We can adjust the value of density factor to control parallel dataflow.

(3)The element of *SM* and *RM* is called as the supper element, because it can be composed of *k* buffer blocks. When *k*=0, the dataflow had been closed.

(4)*SM* and *RM* share the *BBS* space, so it can increase the use rate of memories for buffer blocks.

5.2 Experiments

We built a *MCG* that is composed of 16 computers and 4 computer-clusters that connected by Intranet. The computing tasks provided by *MCG* are the matrix operations and the linear programming. The *CCT* algorithms (Parallel algorithms based on computer cluster) for the matrix operations and the linear programming are given. The intranet clock is synchronous by GTS protocol. In order to make the tasks to migrate as far as possible in the *MCG*, We make use of the random migration function *RandMigration*() and form the migration strategy during the test processes. The description for cooperative rules [8] is given. The experimentation includes seven times, and each time has 12 hours, and the total amount is 84 hours. Through the average values of the test information, we observe the operation results of CCMS. We implement two communication models: One is CCMS which had been implemented according to CCMS; another is BCMS (binding communication system) which communication agent will be migrated with their cooperative computing agent together and it doesn't adopt the DBP technique. The experiment results are as follows:

There are four types of *CCT* scale: (1) *CCT* is composed of 2 *CCA*s; (2) *CCT* is composed of 4 *CCA*s; (3) *CCT* is composed of 8 *CCA*s;(4) *CCT* is composed of 16 *CCA*s.

Experiment 1. The CCMS and BCMS speed-up, which are along with *CCT* scales change, have been tested. The test result is shown as in the figure 3. This test result shows that CCMS keeps linear speed-up and CCMS can support parallel computation effectively.

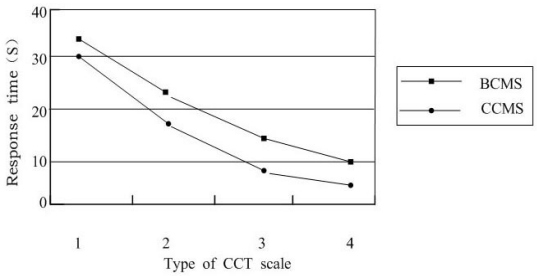

Fig. 3. The speed-up of CCMS and BCMS.

Experiment 2. The changes of memory resources for communication buffer in CCMS and BCMS have been tested. The test result is shown as in the figure 4. This test result shows that the numbers of CCMS memory resources become stable when *CCT* scale type is 2. CCMS can increase the use rate of communication buffers.

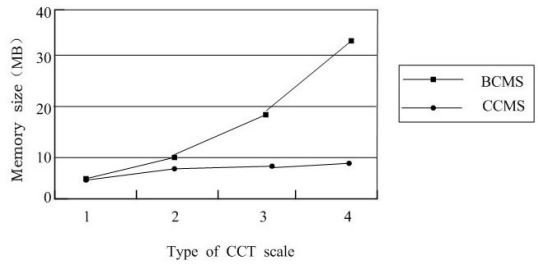

Fig. 4. The memory size for communication buffer in CCMS and BCMS.

Experiment 3. The response time along with the changes of density factor k=1, 2, 3, 4 have been tested. There are |B| =1MB. The test result is shown as in the figure 5. This test result shows that CCMS can effectively support the communication work for high density dataflow and it can be implemented by increasing the density factor of dataflow.

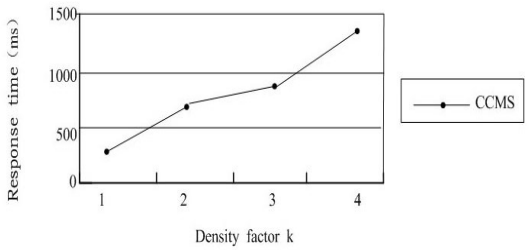

Fig. 5. The changes of response time along with the changes of density factor.

6 Analysis and Experiments

Because of the heterogeneous resources, the state changes of idle computing resources during the computing process and the migration of cooperative computing agent, the communication problem of agent become very important. Constructing cooperative communication system, which is independent system from the computing agent system, we can increase the speed-up of parallel computing task and effectively use the memory resources for communication buffer.

Acknowledgement

We would like to thank the support of National Nature Science Foundation of China (No.60573108), Shanghai Leading Academic Discipline Project (No.T0502) and the Innovation Program of Shanghai Municipal Education Commission (No. 08ZZ76, 07ZZ92).

References

1. Elaine M. Raybourn, Julian Newman : WETICE 2002 Evaluating Collaborative Enterprises Workshop Report. Proceeding of the Eleventh IEEE international Workshops on enabling technologies: Infrastructure for Collaborative Enterprises, pp.11-17. IEEE Press, New York (2002)
2. E. P. Markatos, G. Dramitions : Implementation of Reliable Remote Memory Pager. In proceedings of the 1996 Usenix technical Conference, pp.177-190 (1996)
3. Anurag Acharya and Sanjeev Setia : Using Idle Memory for Data-Intensive Computations. In proceedings of the 1998 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp.278-279 . ACM press, New York (1998)

4. Qingkui Chen, Lichun Na : Grid Cooperation Computational System for CSCW. In proceedings of the 10th IEEE International Conference on Computer Supported Cooperative Work in Design, pp.894-899. IEEE Press, New York (2006)
5. I. Foster, C. Kesselman : The Grid: Blueprint for Future Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco (1999)
6. E. Osawa : A Scheme for Agent Collaboration in Open MultiAgent Environment . Proceeding of IJCAI'93 August 1993, pp.352-358(1993)
7. Wooldridge, M.: An Introduction to Multivalent System. John Wiley & Sons, Chichester, England (2002)
8. Qingkui Chen :Cooperation Learning Model of Agents in Multi-cluster grid. In proceedings of the 11th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD2007), pp.418-423. IEEE Press, New York (2007)
9. H. Franke, W. Dangelmaier, B. Klopper, and C. Kosters : Synchronous Communicating Agents for Parallel Improvement in Transport Logistics. IN Artificial Intelligence and Applications(AIA 2004),Innsbruck, Austria (2004)
10. Shivakant Mishra, Peng Xie : Interagent Communication and Synchronization Support in the DaAgent Mobile Agent-Based Computing System. IEEE transactions on parallel and distributed system, Vol. 14, No. 3 ,pp. 290-306(2003)
11. Salvatore Venticinque , Beniamino Di Martino , Rocco Aversa , Gregorio Vlad and Sergio Briguglio : Mobile Agents Based Collective Communication: An Application to a Parallel Plasma Simulation. Lecture Notes in Computer Science,Vol.4330, pp. 724-733. Springer, Heidelberg (2006)
12. Patrick Riley : MPADES: Middleware for Parallel Agent Discrete Event Simulation. Lecture Notes in Computer Science, Vol.2752 , pp.162-178. Springer, Heidelberg (2003)