

Continuance Parallel Computation Grid Composed of Multi-Clusters

Qingkui Chen

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology,
Shanghai, China

Email:chenqingkui@tom.com

Haifeng Wang and Wei Wang

Business school, University of Shanghai for Science and Technology, Shanghai China

Email:gadfly7@126.com

Abstract—For supporting the grid computing in dynamic network environment composed of multi-clusters, a continuance parallel computation grid (CPCG) is proposed in this paper. A series of formal definitions, such as the CPCG architecture, the dynamic network environment (DNE), the management agent system, the independent computing agents (ICA) which support the traditional computing (TC), the cooperation computing team (CCT) which supports the data parallel computing (DPC), and their relations are given. Through DPC, TC, and the migration mechanism, the continuance data parallel computing (CDPC) was constructed. The dynamic learning method, the fuzzy partition technique for the logical computer cluster on which CCT runs, the stage checkpoint mechanism and the migration process are studied. CPCG computing process is described. The experiment results show that CPCG resolves effectively the problems of optimization use of resources in DNE. It can be fit for grid computing.

Index Terms—parallel computation, dynamic network, multi-clusters grid, continuance parallel computation

I. INTRODUCTION

With the rapid development of the information techniques and their popular applications, the demand for the high performance processing devices is becoming more and more vehement. Nowadays, the numbers of Intranet composed of many computer clusters are quickly increasing, and a great deal of cheap personal computers are distributed everywhere, but the using rate of their resources is very low[1, 2]. Mining and adopting these idle resources, we can get alot of large-scale high performance computation, storage and communication resources which are not special. However, the heterogeneous, dynamic and unstable characteristic of these resources brings the huge obstacle for us. The grid [3] techniques and multiagents [4, 5] can become the main approaches to use effectively these resources, and the multiagents have already become the feasible solutions for grid computing. There are many successful examples [6, 7] of applications which are in conjunction with the automatic multiagent system. On the other hand,

the researches of parallel computing based on grid are development quickly, and there are a lot of research results [8-12] today. The data parallel computing (DPC) [13] is an important computation mode in traditional parallel environments, such as the computer cluster. Because the dynamic network environment (DNE), which is composed of many computer-clusters connected by Intranet, is dynamical, the DPC computing can't be completed in one computing phase, and it need many computing phases to complete. Due to the computation resources for DPC may be single computer or a computer cluster, DPC mode have to degenerate into the traditional serial computing (TC) mode. Therefore, the continuance data parallel computing (CDPC) in DNE means that the computing task is executed mainly by the DPC mode and sometimes by TC mode and it need many computing phases to complete. How to use the computation resources in DNE to support the CDPC is a very interest work. The key techniques are as follows : (1)How to partition the logical computational unit for DPC; (2) How to classify the DPC task by their resource demands; (3) How to set the checkpoint [10] [12][14][15][16]; (4) How to increase the intelligence of computing agents by self-learning[17,18];(5)How to migrate the computational tasks [11] [19-22].

This paper introduced a Continuance Parallel Computation Grid (CPCG) in DNE. Building an open grid computing environment; using of the idle computational resources of DNE; adopting the multiagent, the fuzzy theory [23], the self-learning methods, DPC, TC and the migration mechanism, we designed the CPCG model that can support the continuance data parallel computing. The experimental results show that CPCG can increase the percentage of the using resources in DNE and it can improve response time of computation-intensive tasks.

II ARCHITECTURE OF MCG

CPCG includes two parts: one is DNE that is the personal computers for grid computing, and DNE is composed of many computer clusters connected through LAN and Intranet, and all the computers is not special; other is the agent system, and we call it as GCG (Global Computing Group). GCG is composed of a management

Supported by National Nature Science Foundation of China (No.60573108).

agent system (MAS) and a lot of computing agents. The architecture is presented in figure 1.

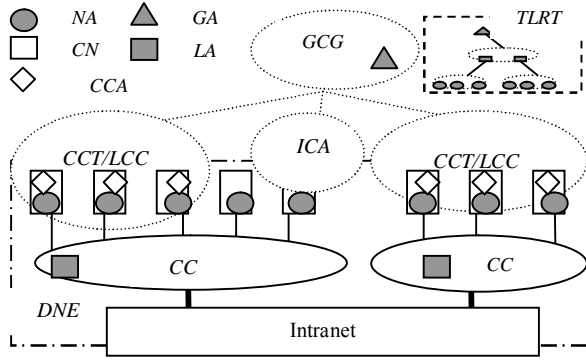


Figure. 1. The architecture of CPCG

A. DNE

Definition.1. Computing node (CN) is defined as $CN(id, CT, AS, RSV, st)$, where id is the identifier of CN ; Let CT be the type of computing node (definition 10); Let AS be set of agents running on CN ; $RSV(r_{cpu}, r_{mem}, r_{disk}, r_{net})$ is its resource support vector, power of its CPU denotes as r_{cpu} , power of its memory storage denotes as r_{mem} , power of its hard-disk denotes as r_{disk} , and power of its network adapter denotes as r_{net} . Let $st \in \{“Local”, “Idle”, “Running”\}$ be its state. “Local” represents that computer is undertaking local tasks, “Idle” means computer of being vacant, “Running” denotes that computer is performing tasks for Grid.

Definition.2. Computer cluster (CC) is defined as $CC(Master, CS)$, where Master is the main computer of CC ; Let $CS = \{CN_1, CN_2 \dots CN_p\}$ be the set of all computing nodes in computer cluster.

Definition.3. Logical computer cluster (LCC) is defined as $LCC(id, LCS, B, CC)$, where id denotes the identifier of LCC ; Let LCS be the set of computing nodes of LCC ; CC is the computer cluster which comprises LCC . Network bandwidth of LCC denotes as B .

So, the dynamic network environment (DNE) can be defined as $DNE(Master, CCS, N, R)$, where Master denotes the main computer of DNE ; CCS is the set of all computer clusters in DNE ; Let N be its network set; R denotes the connection rules.

B. Management Agents System (MAS)

The main functions of MAS are the computation resource management, the DNE monitoring and the task scheduler. MAS involves three parts. The first part is the global control agent (GA) which should be responsible for management of $CPCG$. The second part, called as local agent(LA), is the agent for managing single computer cluster. The last part is the agents for managing the computing nodes, which are called as the node Agents (NA), and each computing node has a NA . MAS ' structure is the tree-level-ring-tree ($TLRT$) and it is shown in Figure 1.

The main functions of GA are as follows: (1) Control and manage DNE ; (2) Receive and dispatch the computing tasks; (3) control and monitor $CDPC$ process; (4) Construct and remove CCT for DPC ; (5) construct the LCC for DPC ;

The main functions of LA are as follows: (1) Control all the computing nodes in its cluster; (2) Calculate the idle resources of cluster, and report them to GA ; (3) Monitor the states all computing nodes and node agents;

The main functions of NA are as follows: (1) Control the computing node to join or to disjoin the DNE in dynamic; (2) Calculate the idle resources, and report them to LA ; (3) Monitor the states and events in CN , and make the response adjustments; (4) Control the computing agents (ICA or CCA) to complete the computing task.

C. Computing Agents and GCG

There are mainly two types of computing mode in $CPCG$ to support $CDPC$: the traditional serial computing (TC) based on single computer and data parallel computing (DPC) based on logical computer cluster. In order to support the two computing mode, the cooperation computing team (CCT) and the independent computing agent (ICA) must be introduced firstly.

Definition.4. Computing agent (CA) is defined as $CA(id, PRG, BDI, KS, CE)$, where id is the identifier of CA ; Let PRG be the executable program set of CA ; BDI is the description of its BDI ; Knowledge set denotes as KS ; Configuration environment denotes as CE . CA is the basic element to execute computation task;

Definition.5. independent computing agent (ICA), if one CA could complete the task independently, we call it as the independent computing agent (ICA);

Definition.6. cooperative computing agent (CCA), if one CA couldn't complete the task independently. So it must cooperate with others. Then we call it as the cooperative computing agent (CCA);

Definition.7. Cooperation computing team (CCT) is defined as $CCT(id, Am, CAS, BDI, CKS, CCE, LCC)$, where id is the identifier of CCT ; There exists main control agent denoted as Am which is the master of MASTER-SLAVER parallel mode in CCT . CAS is the set of all cooperative computing agents (CCA) in CCT . BDI is the description of its BDI ; Knowledge set denotes as CKS . Configuration environment denotes as CCE . LCC is a logical computer cluster, on which CCT runs.

So, **global computing group (GCG)** is defined as $GCG(id, MAS, ICAS, CCTS, GKS, GCE)$, where id is the identifier of GCG ; MAS gives an indication of the manage agent system of GCG ; $ICAS$ is indicative of the set of ICA which GCG includes; $CCTS$ denotes as the set of CCT which GCG includes; GKS is an indication of knowledge set. GCE indicates its configuration environment. The relations between DNE and GCG are presented in figure 1.

D. CDPC

Definition.8. Task (TSK) is defined as $TSK(id, RDV, DAT, AS, St)$, where id is the identifier of TSK ; $RDV(cpu, mem, disk, net)$ is its resource demand vector; DAT denotes as the data set; AS is an indication of the set of agents to calculate TSK . St represents its state. There exist six states of TSK which are “Committed”, “ready”, “Suspended”, “Migrating”, “Running” and “Completed”. The State graph is presented in Figure 2.

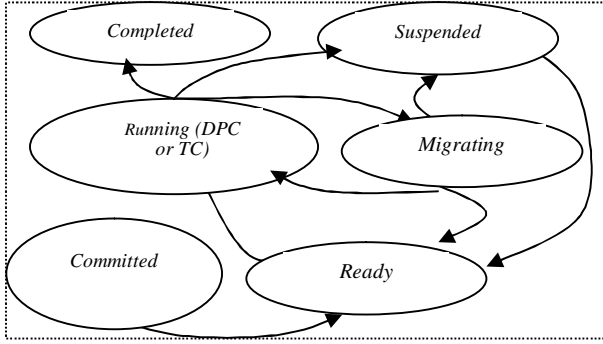


Figure.2. The state graph of task

Obviously, there is no quiescent environment in *DNE*. The computing task, which will run for a long time, can not be completed in one computing phase. Then it must be migrated during the computing process. So, computing task may span many computing phases to be completed. Therefore, Beyond doubt, there exist different computing mode (*ICA* or *CCT*) in the whole period.

Definition.9. Continuance Data Parallel Computing (CDPC) is defined as a computing phase sequence $\{CS_1, CS_2, \dots, CS_{CDPC}\}$, where CS_i is the i^{th} computing phase, and the computing mode of CS_i is *ICA* or *CCT*; If the computing mode is *CCT*, the *CCT (LCC)* scale may be different in different computing phases. All tasks of CPCG are to be executed in the form of CDPC. The continuance means is that the suspended time of tasks is as short as possible.

III CONSTRUCTION OF LOGICAL CLUSTER

When *GA* receives the information from all computing nodes in *DNE*, the logical computer cluster will be constructed for DPC mode. Here let $PCS = \{CN_i (id_i, CT_i, AS_i, RSV_i, st_i) \mid 1 \leq i \leq n\}$ be the set of all idle computing nodes in *DNE*. Actually, as different computer node may provide different resources, we must classify the computer nodes to different power logical computer clusters according to their power. Therefore, the principle of classification is based on their power of CPU, Memory, Disk, and Net adapter. For solving this problem, we should firstly discrete the power data. And fuzzy relation theory is employed to represent the data.

A. Resource Matrix

Suppose that $MR = (r_{ij}) (1 \leq i \leq n, 1 \leq j \leq 4)$ is the resource matrix in *DNE*. F_i denotes as CPU frequency of CN_i , here it is measured in MHZ; M_i denotes as memory capacity of CN_i , and it is measured in MB; R_i indicates Disk speed of CN_i , and it is measured in RPM; N_i indicates communication power of CN_i , and it is measured in MBS; The resource matrix elements are determined by the real ability of computing nodes, and the calculation method is as follows:

$$r_{i1} = \left\lfloor \frac{F_i}{500\text{MHZ}} \right\rfloor + 1;$$

$$r_{i2} = \left\lfloor \frac{M_i}{128\text{MB}} \right\rfloor + 1;$$

$$r_{i3} = \begin{cases} 1, & \text{when } R_i = 5400\text{RPM}; \\ 2, & \text{when } R_i = 7200\text{RPM}; \\ 3, & \text{when } R_i = 10000\text{RPM}; \end{cases}$$

$$r_{i4} = \begin{cases} 1, & \text{when } N_i = 10\text{MBS}; \\ 2, & \text{when } N_i = 100\text{MBS}; \\ 3, & \text{when } N_i = 1000\text{MBS}; \end{cases}$$

Partition for logical computer cluster

It is very important to divide different logical computer clusters. Then the *LCC* division algorithm as follows:

Algorithm3.1 Partition method for LCC.

- (1) $MR = (r_{ij}) (1 \leq i \leq n, 1 \leq j \leq 4)$ is the resource matrix in *DNE*, where n is the numbers of all idle computing nodes; T is a DPC mode task.
- (2) Construct the fuzzy matrix $FM = (f_{ij}) (1 \leq i \leq n, 1 \leq j \leq n)$, where

$$f_{ii} = 1;$$

$$f_{ij} = 1 - b (x_1 |r_{i1} - r_{j1}| + x_2 |r_{i2} - r_{j2}| + x_3 |r_{i3} - r_{j3}| + x_4 |r_{i4} - r_{j4}|), \text{ when } i \neq j, \text{ and } 0 < b < 1, \text{ and } x_1 + x_2 + x_3 + x_4 = 1, x_k > 0 (1 \leq k \leq 4);$$
- (3) build the fuzzy equivalence matrix;
 Repeat do

$$FT = FM \odot FM;$$
 // \odot is the operation theorem to take the maximum and minimum
 If $FT = FM$ then goto (4);

$$FM = FT;$$
 End do;
- (4) Calculate the c -cut matrix FM_c ;
- (5) Divide the computing nodes of *PCS* into several equivalence class $LCC_1 \cup LCC_2 \cup \dots \cup LCC_e$ by FM_c ;
- (6) Choose a *LCC* for T according to its resource demand vector by algorithm3.2.

B. Choose LCC for DPC mode task

Suppose that $LCCS = LCC_1 \cup LCC_2 \cup \dots \cup LCC_e$ is the set of all logical computer clusters which are built through algorithm3.1, and T is a DPC mode task.

Algorithm3.2. Choose LCC method for DPC mode task.

- (1) Get the resource demand vector $RDV (w_1, w_2, w_3, w_4)$ of T ;

$$mine = \infty;$$
- (2) While $LCCS \neq \emptyset$ do
 - { Get $S \in LCCS$; /* S is a logical computer cluster */
 - Calculate total resource vector ($ar_{Cpu}, ar_{mem}, ar_{disk}, ar_{net}$) of all computing nodes of S , it is as follows:

$$\{ ar_{Cpu} = \sum_{CN \text{ in } S} CN.RSV.r_{cpu};$$

$$ar_{mem} = \sum_{CN \text{ in } S} CN.RSV.r_{mem};$$

$$ar_{disk} = \sum_{CN \text{ in } S} CN.RSV.r_{disk};$$

$$ar_{net} = \sum_{CN \text{ in } S} CN.RSV.r_{net};$$

$$y_1 = ar_{Cpu} (ar_{Cpu} + ar_{mem} + ar_{disk} + ar_{net});$$

$$y_2 = ar_{mem} (ar_{Cpu} + ar_{mem} + ar_{disk} + ar_{net});$$

$$y_3 = ar_{disk} (ar_{Cpu} + ar_{mem} + ar_{disk} + ar_{net});$$

$$y_4 = ar_{net} (ar_{Cpu} + ar_{mem} + ar_{disk} + ar_{net});$$

Construct the vector $Y (y_1, y_2, y_3, y_4)$;
 $e = |y_1 - w_1| + |y_2 - w_2| + |y_3 - w_3| + |y_4 - w_4|$;
 if $mine < e$ then $\{ LCC = S; mine = e \}$;
 $LCCS = LCCS \cup \{S\}$;
 $\}$; // end while

- (3) Choose LCC as the logical computer cluster for T ;
- (4) End.

C. Optimized LCC

Obviously, the LCC may span many different networks in DNE , so LCC can be optimized through the network skip distance, network bandwidth, and the resource demand vector of T . Suppose that $NSET = \{N_i (bw_i) | 1 \leq i \leq m\}$ is the network set in LCC . bw_i indicates bandwidth of N_i , and m indicates the numbers of network. $RDV (w_1, w_2, w_3, w_4)$ indicates the resource demand vector of task T . The process is described as follows:

- (1) Construct network matrix $NM = (s_{ij}) (1 \leq i \leq m, 1 \leq j \leq m)$ that is composed of the factors of the network skip distance and bandwidth. The construction method for NM is as follows:

$s_{ii} = 1$;
 $s_{ij} = 1 - g (distance(N_i - N_j) + |bw_i - bw_j|)$, when $i \neq j$,
 and $0 < g < 1$;

When the network skip distance between N_i and N_j is 1, distance $(N_i, N_j) = 1$;

When the network skip distance between N_i and N_j is 2, distance $(N_i, N_j) = 3$;

When the network skip distance between N_i and $N_j > 2$, distance $(N_i, N_j) = 6$;

When the bandwidth of N_i is 10MB, $bw_i = 1$;

When the bandwidth of N_i is 100MB, $bw_i = 3$;

When the bandwidth of N_i is 1000MB, $bw_i = 6$;

If $N_i.bw_i = 1$ and $(N_j.bw_j = 3$ or $N_j.bw_j = 6)$ then distance $(N_i, N_j) = 6$

If $N_i.bw_i = 3$ and $(N_j.bw_j = 1$ or $N_j.bw_j = 6)$ then distance $(N_i, N_j) = 6$

If $N_i.bw_i = 6$ and $(N_j.bw_j = 1$ or $N_j.bw_j = 3)$ then distance $(N_i, N_j) = 6$

- (2) Build the fuzzy equivalence matrix:

Repeat do
 $FT = NM \odot NM$; // \odot is the operation theorem to take the maximum and minimum

If $FT = NM$ then goto (4);

$NM = FT$;

End do;

- (3) calculate the c -cut matrix FM_c by w_4 of resource demand vector of T ;

- (4) Divide the computing nodes of LCC into several equivalence class

$SLCCS = SLCC_1 \cup SLCC_2 \cup \dots \cup SLCC_e$ by FM_c ;

- (5) $SLCCS$ is the set of optimized LCC according to the network factors.

IV DESCRIPTION OF AGENT LEARNING MODEL

Because of the difference resources which CC , CN , and the network provided in DNE , their types must be considered. These types are described as follows:

Definition.10. Computing node type (CNT) can be defined by $RSV (r_{cpu}, r_{mem}, r_{disk}, r_{net})$ of the computing node. According to the real conditions of each CN in DNE , the CN types can be divided into the *computing node type set* $CTS = \{CT_1, CT_2, \dots, CT_{ct}\}$.

Definition.11. Network type (NT) is defined as $NT (B, PRT)$, where B denotes as the network bandwidth of CC ; PRT indicates network protocols of CC . According to the real condition of networks, network types can be divided into the *network type set* $NTS = \{NT_1, NT_2, \dots, NT_m\}$.

The agent rules are described as follows:

- (1) **Basic rule (br)** is defined as $br (id, rul, MRS)$, where id is its identifier; rul is the description of br ; MRS is the meta-rules set for revising br ; The **basic rule set (BRS)** is the set of all basic rules that GCG includes;
- (2) **Dynamic rule (dr)** is defined as $dr (ct, nt, br, rul, w, sta, life)$, where $ct \in CTS$, $nt \in NTS$, $br \in BRS$; rul is the formalization description of dr ; w is the its weight value; and sta is its state, and $sta \in \{“Naive”, “Trainable”, “Stable”\}$; “Naive” denotes that the dr is a new rule; “Trainable” denotes that the dr is revising rule; “Stable” denotes that the dr is a mature rule; $life$ is the its life value;
- (3) If dr is a dynamic rule and $dr.w > MaxWeight$, which $MaxWeight$ is a constant in GCG , we call dr as the **static rule (sr)**, its state is “Static”;
- (4) If dr is a dynamic rule and $dr.w < MinWeight$, which $MinWeight$ is a constant in GCG , we call dr as **castoff rule (cr)**. Its state is “Castoff”.

The state graph of rules is presented in figure 3.

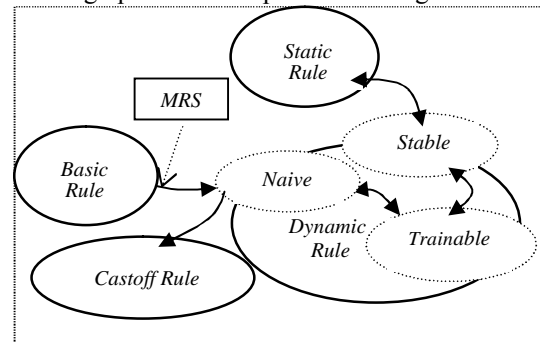


Figure. 3. The state graph of rules

The dynamic knowledge is the set of all the dynamic rules in GCG . The static knowledge is the set of all static rules. The basic knowledge can be earned by passive learning. To adapt the variety of computing resources, the dynamic rules must be generated at the start of the computing and be revised during the TSK computing process. Therefore, reinforcement learning can be adopted in the revising mechanism. Resources utilization rate for TSK is very important reinforcement factors.

Suppose that Y_1 is the **castoff threshold**, and Y_2 is the **mature threshold**; $Q (urt)$ denotes as the **reinforcement function**, and $Q (urt) > 0$. urt indicates resources utilization rate; $MaxWeight$ is the maximum of the rule weight, and $MinWeight$ is the minimum of the rule weight, and let $MinWeight < Y_1 < Y_2 < MaxWeight$; $MaxLife$ be the maximum of life value. The revising process is as follows:

- (1) Suppose that a computing agent CA adopted a dynamic rule dr of $CA.KS$;

- (2) $dr.life++$; //increase the value of life
- (3) wait for the urt from MAS ;
- (4) If $urt > 0$ then $dr.w = dr.w + Q(urt)$; //increase weight
If $urt < 0$ then $dr.w = dr.w - Q(urt)$; //decrease weight
- (5) If $dr.w > MaxWeight$ then $dr.w = MaxWeight$;
If $dr.w < MinWeight$ then $dr.w = MinWeight$;
- (6) If $dr.w < Y_1$ and $dr.life > MaxLife$ then $dr.sta = "Castoff"$;
//Castoff rule
If $Y_2 < dr.w < MaxWeight$ then $dr.sta = "Stable"$;
//Stable rule
If $dr.w \geq MaxWeight$ then $dr.sta = "Static"$; //Static rule
If $Y_1 < dr.w < Y_2$ then $dr.sta = "Trainable"$; //Trainable rule
If $MinWeight < dr.w < Y_1$ then $dr.sta = "Naive"$.

V Process of CPCG

The CPCG computing process is as follows:

- (1) The user agent commits the tasks to GCG , and their states is "*Committed*";
- (2) GA of GCG gets a task TSK that state is "*Ready*";
- (3) GA does the initialization works for TSK ;
Set $C_{phase} = 0$; // computing phase counter
 GA decides the computing mode for TSK by the conditions of DNE ;
- (4) GA allots a computing device PE (that is a CN or a LCC) for TSK ; if this computing phase for TSK is DPC mode, GA get a LCC (That is constructed by GA through fuzzy partition method) for TSK ; if this computing phase for TSK is TC mode, GA get a single computing node CN for TSK ; Set $C_{phase} = C_{phase} + 1$;
- (5) LA and NA , which PE (CN or LCC) includes, constructed computing Agents (ICA or CCT) for TSK ;
- (6) All computing agents (ICA or CCT) for TSK construct the dynamic knowledge;
- (7) All computing Agents calculate the TSK in cooperative, and they construct the stage checkpoint (Section 6.1) during the computing process, and they start the rule revising functions to revise the dynamic knowledge;
- (8) If the TSK must be migrated, GA starts the migration process (Section 6.2 and 6.3) and then does
{Set $C_{phase} = C_{phase} + 1$; // next computing phase
Go step (6) to continue the next computing phase ;}
- (9) If TSK be finished, GA receives the TSK from all NA of PE and save the new knowledge into its knowledge base.

VI MIGRATION PROCESS

A. Checkpoint for DPC

In order to support the CDPC, we discuss the effective checkpoint mechanism. The communication checkpoint [10] [12][14][15][16] is very important for parallel computing, and it may cause the domino effect. The *coordinated checkpoint* [15] and the *independent checkpoint* [16] are the possible techniques. But the

additional spending of coordinated checkpoint is very high, and the independent checkpoint is not fitting for the DPC mode. We propose a *stage checkpoint* approach to support the CDPC.

The DPC process in CCT can be divided into a series of computing stages by the global synchronization point in CCT . We design two types of checkpoints in each computing stage: the first one is *local checkpoint* that is the state information of local computing process of CCA and it is the independent, and the failures of communication can be restore through the fault-tolerant communication protocols; the second one is *synchronization checkpoint* which is the backups in the synchronization point of CCT , and it is the process that the middle results (or final results) are gathered. The stage checkpoint is presented in the figure 4. The stage checkpoint mechanism avoids the additional spending of coordinated checkpoint and the incomplete characteristic of independent checkpoint. The stage checkpoint can be described through XML language.

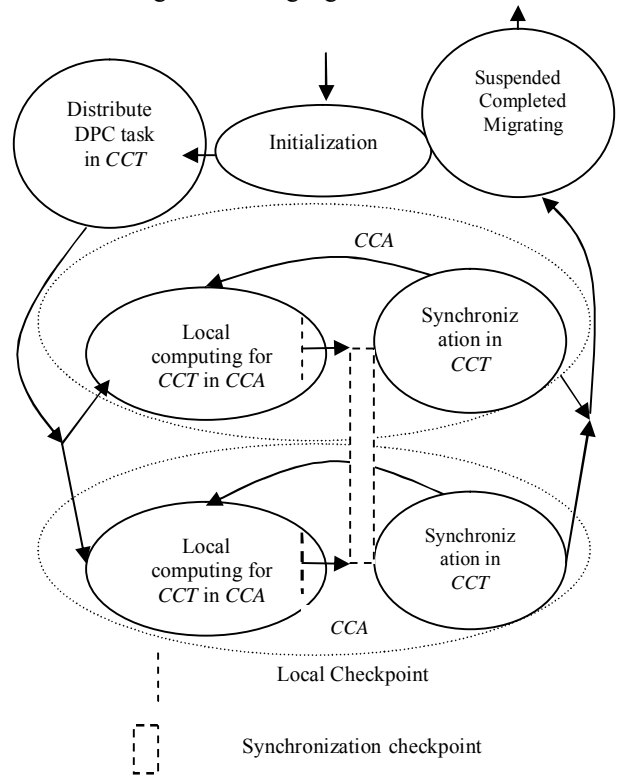


Figure.4. CCT process and the stage checkpoint

B. Migration Strategies

CDPC process involves a series of migrations, and the migration process can be classified as four sorts: (1) $ICA \rightarrow ICA$; (2) $ICA \rightarrow CCT$; (3) $CCT \rightarrow ICA$; (4) $CCT \rightarrow CCT$. The former three sorts are very simple, but the last one must be discussed in detail.

Now suppose that CCT runs on a logical cluster LCC and $CCT = MIG \cup NMIG$, where MIG is the set of the computing agents that will be migrated; $NMIG$ indicates the set of the non-migration computing agents;

- (1) If $NMIG$ is f , all $CCAs$ of CCT will be migrated into a new logical cluster $NLCC$, and the $NLCC$ is optimized, so the migration process is simple;
- (2) If $NMIG$ is not f , all $CCAs$ of MIG will be migrated into a new logical cluster $NLCC$, and CCT will run on two logical computer clusters $NLCC$ and $RLCC$ ($RLCC$ is the subset of LCC which $NMIG$ runs). $NLCC \cup RLCC$ could not be optimized, so we construct the migration functions:

$$MF = \frac{Z1 \times TotalPower(NLCC) \times MBw(RLCC, NLCC)}{Z2 \times Diff(RLCC, NLCC) \times Dist(RLCC, NLCC)}$$

Where $TotalPower(NLCC)$ indicates total computing power of $NLCC$; $Diff(RLCC, NLCC)$ denotes their difference of RSV ; $Dist(RLCC, NLCC)$ denotes network skip distance between $RLCC$ and $NLCC$; $MBw(RLCC, NLCC)$ gives the indicative of minimal network bandwidth which the path from $RLCC$ to $NLCC$ includes; $Z1$ and $Z2$ are the coefficient.

If $MF > Y4$ ($Y4$ denotes as **migration threshold**), the migration will be executed; if $MF < Y4$, the sub-tasks from the computing agents of MIG will be retractible, and the CCT ($NMIG$) will run on $RLCC$.

C Migration Process

We present the process that ICA to ICA migrates and CCT to CCT migrates.

Algorithm6.1. Migration of ICA to ICA

An ICA running on the computing node CN_i must be migrated into the computing node CN_j , so the migration and learning process is presented as follows:

- (1) ICA commits its CE and KS to NA in CN_i , and NA in CN_i saves the KS and CE ;
- (2) ICA asks for a new computing node CN_j from GA , and it consults with NA of CN_j , and migrates into CN_j ;
- (3) ICA gets the current knowledge about CN_j and the other agents from NA of CN_j ;
- (4) ICA and NA of CN_j learn each other, and they resolve the conflicts, and they refresh their KS and CE ;
- (5) CN_i and CN_j report their KS and CE into LA and GA ;
- (6) ICA continues the next computing phase and Learning process on CN_j at the newest stage checkpoint.

Algorithm6.2. Migration of CCT to CCT

Suppose that $CCT(id, Am, CAS, BDI, CKS, CCE)$ is a cooperation computing team running on the logical computer cluster LCC , and $CAS = MIG \cup NMIG$, where MIG indicates the set of the computing agents that will be migrated; $NMIG$ indicates the set of the non-migration computing agents. The migration process involves three sub-algorithms.

Algorithm.6.2.1

While $NMIG \neq f$ and $Am \in NMIG$, the migration process is as follows:

- (1) All the computing agents of MIG migrate into their new computing nodes severally, and learn the new knowledge according to the algorithm6.1;
- (2) For all $CCA \in MIG$ do (3)(4)(5):
- (3) CCA learns the cooperation knowledge from Am of $NMIG$, and consults to resolve the conflicts;
- (4) If the conflict consultation was failure, CCT will retract the task t from CCA and expel CCA ; $*MIG = MIG - \{CCA\} */$
- (5) If the conflict consultation is successful, then $\{CCA$ refreshes its KS ; $NMIG = MING + \{CCA\}$; $MIG = MIG - \{CCA\} \}$;
- (6) CCT continues for executing the next computing phase at the newest stage checkpoint.

Algorithm.6.2.2

While $NMIG \neq f$ and $Am \notin NMIG$, the migration process is as follows:

- (1) All the computing agents in $NMIG$ cooperate to elect a new main control agent Anm from $NMIG$, Am submits the cooperation knowledge about CCT to Anm ;
- (2) Do the algorithms.6.2.1;

Algorithm.6.2.3

While $NMIG = f$, the migration process is as follows:

- (1) Am migrates into the new computing node according to algorithm6.1, and notices GA ;
- (2) Am broadcasts its new conditions to all members of CAS of CCT ; The other members of CAS receive the messages from Am , and they do the migration by algorithm 6.1, and they submit the learning results to Am ;
- (3) After all the cooperation computing agents finish the migration process, CAS cooperate to produce the new CKS by Am control;
- (4) CCT continues the next computing phase at the newest stage checkpoint.

VII EXPERIMENTS

We have performed a number of experiments to verify the CGCP model in the DNE, which is composed of 24 computers and 3 physical computer-clusters connected by Intranet. All the computers are classified as four groups depending on the types of CPU, memory, disk, and net adapter. These groups are RSV (3000MHZ, 512MB, 7200RPM, 100M), RSV (2600MHZ, 256MB, 7200RPM, 100M), RSV (1600MHZ, 256MB, 5400RPM, 1000M), and RSV (1200MHZ, 128MB, 5400RPM, 100M). The operating systems of the computers are the Windows series or Linux. For the performance evaluation, there are three sets of continuance data parallel computing tasks which are the matrix operations, the linear programming and JOIN operation that is a most important operation of parallel relation in the parallel relation database. We programmed the DPC edition and TC edition for these operations in order to support CDPC. The Matrix operation RDV is (0.3, 0.3, 0.2, 0.2), and the linear programming RDV is (0.5, 0.3, 0.1, 0.1), and JOIN

operation *RDV* is (0.2, 0.3, 0.25, 0.25). The intranet clock is synchronous through GTS protocol. The initialization basic rules include 24 rules for *ICA* and 7 rules for *CCT*, and the parameter values are as follows: *MaxLife*=43200(s), $Y_1=15$, $Y_2=80$, $Y_3=0.3$, *MaxWeight*=100 and *MinWeight*=0.

The experimentation includes seven times, and each time has 12 hours, and the total amount is 84 hours. The tests adopt a random function to choose some tasks (the matrix operation, the linear programming, and JOIN operation) in each time. In order to make the tasks to migrate as far as possible in the *DNE*, We make use of the random migration function *RandMigration()* to form the migration strategies during the test processes. Through the average values of the test information, we observe the test results of CPCG. The experiment results are as follows:

Experiment1. As shown in table 1, we can see the distribution characteristic that *LCC* spans many networks. Obviously, JOIN operation depended on the network bandwidth greatly.

Table I. The distribution characteristic that *LCC* spans many networks

Network	1	2	3
Matrix operation	0.81	0.19	0.00
Liner programming	0.12	0.78	0.10
JOIN operation	1.00	0.00	0.00

Experiment2. We tested the effective scales of the logical computer cluster (*LCC*) for three types of DPC. The test results are presented in Table2.

Table II. The effective scales of *LCC* for different DPC

DPC	Matrix operation	Liner programming	JOIN operation
Effective scale	About 8	About 10	About 5

Experiment3. Fig 5 presents the comparison the distribution of dynamic rules, which are generated during the learning process. The solid line denotes the distribution of dynamic rules that their state is always “*Naive*” during their life period. The dotted line denotes the distribution of the “*Trainable*” dynamic rules that their sate had become the “*Stable*” during their life period. It can be seen from the figure that the efficiency of this learning model is increased gradually along with computing process.

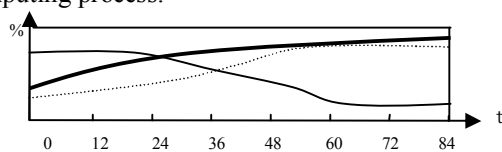


Figure. 5. The tests results of migration learning model and the use rates of *DNE*

Experiment4. As shown in fig 5, the thick solid line denotes the average resources utilization rate of *DNE*. No

doubt, CPCG can raise resources utilization rate of *DNE* consumedly.

Experiment5. Table 3 presents the comparison the average time rates of computing mode (TC, DPC) for three kind CDPC computing task. The test result show that JOIN operations depend on its large data and its migration cost is very high; the others can fit for this migration model.

TableIII. The rates of time

Operations	The time rate of DPC	The time rate of TC	Suspended time rate	The time rate of migration
Matrix operation	0.66	0.15	0.04	0.15
Liner programming	0.77	0.12	0.02	0.09
JOIN operation	0.35	0	0.37	0.28

VIII CONCLUSIONS

Because of the heterogeneous resources, the different power of computers, and the migration of computing task, the effective use of resources is very difficult for the grid computing in the dynamic network environment. Through the techniques of *ICA*, *CCT*, the dynamic learning and the logical computer cluster partition based on fuzzy theory, the stage checkpoint, and cooperation migration, CPCG can support CDPC. These approaches can raise the resources utilization rate in *DNE*. It can fit for the grid computing in *DNE*.

ACKNOWLEDGMENT

We would like to thank the support of National Nature Science Foundation of China (No.60573108), the Innovation Program of Shanghai Municipal Education Commission (No. 08ZZ76, 07ZZ92), and Shanghai leading academic discipline project (S30501).

REFERENCES

- [1]. E. P. Markatos and G. Dramitions: Implementation of Reliable Remote Memory Pager. In proceedings of the 1996 Usenix technical Conference, (1996) 177-190
- [2]. 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, (1998)278-279
- [3]. I. Foster, C. Kesselman: The Grid: Blueprint for Future Computing Infrastructure. San Francisco, USA: Morgan Kaufmann Publishers, (1999)
- [4]. E. Osawa: A Scheme for Agent Collaboration in Open MultiAgent Environment .Proceeding of IJCAI'93, (1993)352-358
- [5]. Wooldridge, M.: An Introduction to Multivalent System, John Wiley & Sons (Chichester, England). ISBN 0 47149691X, (2002)
- [6]. O.F. Rana and D.W. Walker: The Agent Grid: Agent-based resource integration in PSEs, In proceedings of the 16th IMACS World congress on Scientific Computing, Applied mathematics and Simulation, Lausanne, Switzerland, (2000)
- [7]. J.O. kephart, Chess, D.M: The Vision of Autonomic Computing, IEEE Computer, (2003) 41-50

- [8]. Cahon, S. Melab, N.Talbi, E.-G.: An enabling framework for parallel optimization on the computational grid. Proceedings of IEEE International Symposium on Cluster Computing and the Grid, (2005) (2) 702- 709
- [9]. Hee-Khiang Ng¹ , Dudy Lim¹ , Yew-Soon Ong¹ , Bu-Sung Lee¹ , Lars Freund² , Shuja Parvez² and Bernhard Sendhoff² : A Multi-cluster Grid Enabled Evolution Framework for Aerodynamic Airfoil Design Optimization. Proceedings of Advances in Natural Computation: First International Conference, ICNC 2005, Changsha, China, August 2005, Springer Berlin / Heidelberg, LNCS, 3611 / 2005
- [10]. Yudith Cardinale, Emilio Hernandez: Parallel Checkpointing on a Grid-Enabled Java Platform. Proceedings of Advances in Grid Computing - EGC 2005: European Grid Conference, Amsterdam, Netherlands, February 2005, Springer Berlin / Heidelberg, LNCS 3470 / 2005.
- [11]. Jozsef Kovacs and Peter Kacsuk: A Migration Framework for Executing Parallel Programs in the Grid. Proceedings of Grid Computing: Second European AcrossGrids Conference, AxGrids 2004, Nicosia, Cyprus, (2004).
- [12]. Youhui Zhang, Dongsheng Wang, Weimin Zheng: Parallel Checkpoint/Recovery on Cluster of IA-64 Computers. Proceedings of the 2nd International Symposium on Parallel and Distributed Processing and Applications (ISCA2004), Hongkong, (2004).
- [13]. Rajkumar Buyya: High performance Cluster Computing Architectures and Systems, Prentice Hall, (1999)
- [14]. Raphael Y. de Camargo: Checkpointing-based rollback recovery for parallel applications on the InteGrade grid middleware, ACM International Conference Proceeding Series: Proceedings of the 2nd workshop on Middleware for grid computing table of contents, Toronto, Ontario, Canada , (2004) 35 - 40
- [15]. J. M. Helary, A. Mostefaoui, R. H. B. Netzer, and M. Raynal: Preventing Useless Checkpoint in distributed computation. Proceedings of the 27th international Symposium on Fault-Tolerant Computing Systems, (1997) 68-77
- [16]. E. N. Elnozahy and W. Zwaenepoel: On the USE and Implementation on Message Logging. Proceedings of the 24th international Symposium on Fault-Tolerant Computing Systems, (1994) 298-307
- [17]. L. P. Kaelbling, M. L. Littman, and Moore: Reinforcement learning: A survey. Journal of Artificial Intelligence Research, Vol.4 (2), (1996)237-285
- [18]. F. Zambonelli, N. R. Jennings, M. Wooldridge: Organizational rules as abstractions for the analysis and design of multi-agent systems. International Journal of Software Engineering and Knowledge Engineering, Vol.11 (3), (2001) 303-328
- [19]. Frechette, S. Avresky, D.R.: Method for Task Migration in Grid Environments. Proceedings of Fourth IEEE International Symposium Network Computing and Applications, (2005) 49- 58
- [20]. Vadhiyar, S. and Dongarra, J: Performance Oriented Migration Framework for the Grid. Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), Tokyo, Japan, (2003) 130-137
- [21]. Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn: Job Scheduling Strategies for Parallel Processing, 10th International Workshop, JSSPP 2004, New York, NY, USA, June 13, 2004
- [22]. Rafael Fernandes Lopes and Francisco Jose: Migration Transparency in a Mobile Agent Based Computational Grid. In: Proceedings of the 5th WSEAS International Conference on Simulation, Modeling and Optimization. 1st WSEAS International Symposium on Grid Computing, Corfu, Greece, (2005)
- [23]. L. A. Zadeh: The concept of a linguistic variable and its application to approximate reasoning, American Elsevier Publishing Company. Inc, (1975)

Qingkui Chen was born in 1966. He is a Professor and Ph.D supervisor of Scholl of Optical-Electrical and Computer Engineering at University of Shanghai for Science and Technology (USST), Shanghai, P. R. China. He is the Vice Dean of School of Optical-Electrical and Computer Engineering. Qingkui Chen is a senior member of China Computer Federation (CCF); is a member of Professional Committee of Open System of CCF; is a member of Professional Committee of Computer Support Cooperative Work of Shanghai Computer Federation in China. His research interests include network computing, parallel computing, parallel database and computer network. He is the head of many programs which were supported by the Natural Science Foundation of China (NSFC) and the Shanghai Natural Science Foundation of China. Prof. Chen also served as program committee member of IFIP International Conference on Network and parallel Computing (NPC) and as the Technical Program Committee Co-Chairs of ICISE.

Haifeng Wang was born is 1976. He is a Ph.D.Candidate of Business school at University of Shanghai for Science and Technology, P. R. China. His current major research interests are in the areas of parallel computing, network computing.

Wei Wang was born is 1983. He is a postgraduate of Scholl of Optical-Electrical and Computer Engineering at University of Shanghai for Science and Technology (USST), Shanghai, P. R. China. His research interests include network computing, parallel computing.