We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Research and Implementation of Parallel Cache Model Through Grid Memory

Qingkui Chen, Lichun Na, He Jia,
Song Lin Zhuang and Xiaodong Ding
School of Computer Engineering,
University of Shanghai for Science and Technology
China

1. Introduction

With the rapid development of the information techniques and the Internet popular applications, the demand for the high performance processing devices is becoming more and more vehement, such as the server construction of HDTV or IPTV for large scale VOD, and the parallel file system of grid computing. These applications need the ability to process the concurrent peak access. The parallel file system (J.Carretero, F. Perez, P. de Miguel& L.Alonso, 1996; P.F. Corbett & D. G, 1996; Craig S. Freedman, Josef Burger, & David J, 1996; Seogyun Kim, Jiseung Nam & Soon-ja Yeom, 2002) has become the very important research areas, and the parallel cache (Horst Eidenberger, 2005; J. Fernandez,J. Carretero, F. Garcia-Carballeira, A. Calderon & J. M. Perez-Menor, 2005; T. Kimbrel et. AI,1996; Pal Halvorsen, Carsten Griwodz, Vera Goebel, Ketil Lund, Thomas Plagemann & Jonathan Walpole,2006) is playing the key roles in these applications. At the same time, the numbers of the Intranet composed of computer clusters are quickly increasing, and a great deal of cheap personal computers are distributed everywhere, but the utilization of their resources is very low(E. P. Markatos & G. Dramitions,1996; Anurag Acharya & Sanjeev Setia,1998). Through mining and adopting these idle resources, we can get a lot of large-scale high performance computation, storage and communication resources which are not special. How to use these idle memories to construct the parallel cache for supporting the large scale applications, such as VOD for hot segments, is very interesting. It not only improves their performance, but also increases the utilization of these idle resources. However, the heterogeneous, dynamic and unstable characteristic of these resources brings a huge obstacle for us. The grid (I. Foster & C. Kesselman, 1999) techniques and multi-agents (E. Osawa, 1993; Wooldridge, M, 2002) become the main approaches to effectively use these resources, and the multi-agents have already become the feasible solutions for grid applications. There are many successful examples (O.F. Rana & D.W. Walker, 2000; J.O. kephart & Chess, D.M, 2003) of applications which are in conjunction with the automatic multi-agents system. But, the research of construction cache model for hot segment through grid idle memory is still not much. At the same time, the high performance cluster techniques (Rajkumar Buyya, 1999) are already mature and can be the foundation for supporting the parallel cache based on grid memory.

This book chapter introduced a Parallel Cache Model Based on Grid Memory (PCMGM) in the dynamic network environment (DNE). Through building an open grid environment, using the idle memory resources of DNE, adopting the multi-agents, the fuzzy theory and the self-learning methods, we designed the PCMGM model that can support the concurrent peak access for the hot segments in large scale internet applications, such as the large scale VOD system. The experimental results show that PCMGM can improve response time of the large scale application system for hot segments. It can be fit for the grid computing and the large scale VOD system in internet.

2. Architecture of PCMGM

PCMGM includes two parts: one is DNE that is the physical computing devices for the parallel cache, and DNE is composed of computer clusters connected through LAN and Intranet, and all the cache computers are not special, the other is the agent system, and we call it GMG (Global Management Group). GMG is composed of a management agent system (MAS), the application agent (AA), and a lots of cache agents (CA). The architecture is presented in figure1, and the definitions of NA, CA, CN, GA, SA are described as the definition 1 ~7 and the section 3.

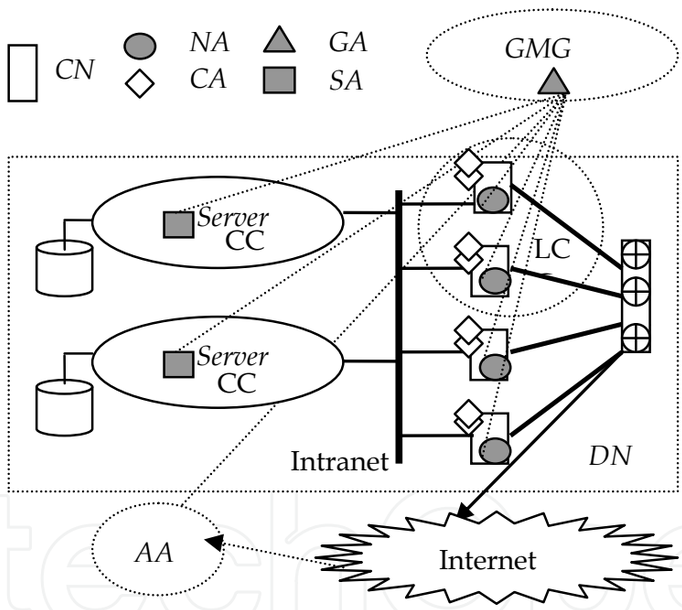


Fig. 1. The architecture of PCMGM

2.1 DNE

Definition.1. Cache node (CN) is a computer and it is defined as CN (id, AS, RSV, st), where id is the identifier of CN; AS is the set of agents running on CN; RSV(r_{cpu}, r_{mem}, r_{disk}, r_{net}) denotes its resource support vector, r_{cpu} is the power of its CPU, r_{mem} is the power of its memory storage, r_{disk} is the power of its hard-disk , and r_{net} is the power of its network adapter; st $\in$ {"Local", "Idle", "Caching"} is its states, "Local" denotes that this computer is working for local tasks, "Idle" denotes that this computer is not working , and "Caching" denotes that this computer is working for Grid in the form of cache.

Definition.2. Computer cluster (CC) is defined as CC (Master, CS), where Master is the main computer of CC; CS= {CN1, CN2 … CNp} is the set of all cache 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; LCS is the set of cache 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, SVS, N, R), where Master is the main computer of DNE; CSS is the set of all computer clusters in DNE; SVS is the set of file servers to storage the files; N is its network set; R is the connection rules. All the free memories of the idle cache nodes in DNE are called the grid memory.

Definition. 4. Basic memory unit (BMU) is the uniform size of basic memory block as the allotting unit in PCMGM, and its unit is MB.

Definition. 5. Grid memory capacity (Gmc) is the total numbers of BMU provided by all cache nodes in DNE. The Gmc value is dynamic changed.

2.2 Management Agents System (MAS)

The main functions of MAS are the computation resource management, the DNE monitoring and the task scheduler. MAS include four parts: The first part is the global control agent (GA) for managing PCMGM. The second part is the agent for managing one file server, and it is called as the server agent (SA). The third part is the agents for managing the cache nodes, and they are called as the node agents (NA), and each cache node has a NA. The last part is the application agent (AA) that is the connection manager between the users and PCMGM. The MAS structure is presented in figure 1.

The main functions of GA are as follows: (1) Control and manage DNE; (2) Receive and dispatch the caches; (3) control and monitor the file access process; (4) load balance; (5) Control all the cache nodes in DNE; (6) Calculate the idle resources; (7) Monitor the states all computing nodes and node agents ;

The main functions of SA are as follows: (1) File operations; (2) The hot segment 27 management; (3) File transfer;

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

2.3 Cache file and cache agent

The large file is always divided into a series of segments (SEG) in the file server and the access frequency of each segment is different, so the high access frequency segment is called as the hot segment. If the hot segment is duplicated into the grid memory, the access efficiency will be improved greatly. For describing this mechanism, we introduce some conceptions.

Definition.6. Duplication Segment (DSEG) is the SEG (definition 5) duplication stored in Grid memory, and it can be defined as DSEG (SEG-id, CA, CN, ST), where SEG-id is the identifier of its SEG; CA is the cache agent for managing and accessing the DSEG; CN is the cache node on which the DSEG is stored. ST∈ {"Memory", "Disk"} is its states, and "Memory" denotes that the DSEG is in the memory of cache node, and "Disk" denotes that the DSEG is in the local disk of cache node. In the follow, we will give the detail definition about SEG.

Definition.7. Segment (SEG) is the file basic unit and it can be defined as SEG (SEG-id, File, DSEGS, LOC), where SEG-id is the identifier of SEG; File is the file that includes this SEG; DSEGS = {DSEG1, DSEG2 … DSEGd} is the set of all its active DSEGs; LOC is its server address. |SEG | is the numbers of DSEGS elements and it is called the duplication width of the SEG.

So, the file in PCMGM can be presented as a SEG sequence {SEG1, SEG2 … SEGf}, and the length of all SEGs is a constant Segl, and it is called the segment length, and its measure unit is BMU. Each file has its private segment length by its characteristic. The SEG duplication stored in Grid memory is called as duplication segment (DSEG). We call this structure file as the cache file.

Definition.8. Cache agent (CA) is defined as CA (id, DSEG, RDV, PRG, BDI, KS, CE), where id is the identifier of CA; DSEG is the duplication segment which is managed by CA; RDV (d_{cpu}, d_{disk}, d_{net}) is the resource demand vector of CA and it is the need rate for CPU, DISK and NET ; PRG is 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 manage DSEG, and a CA can serve for a group of users; the relations between CA, SEG, DSEG, and cache file is presented in figure 2.

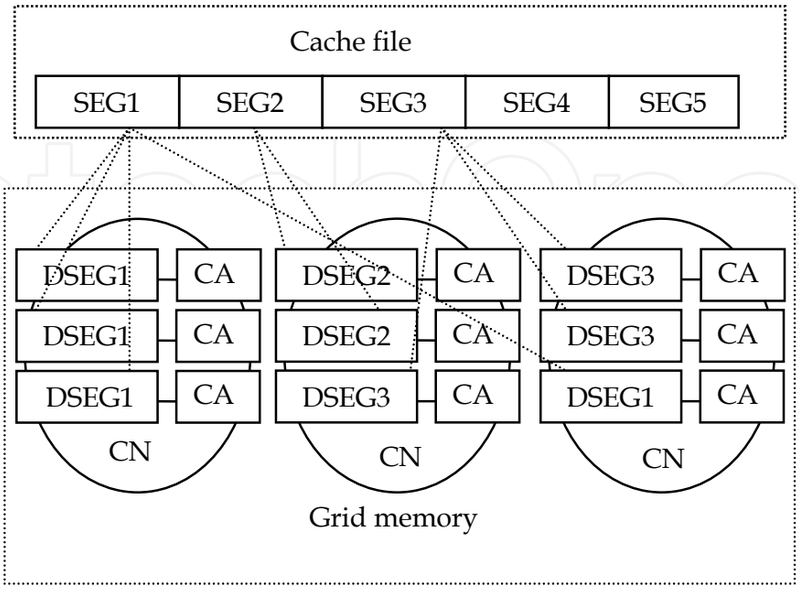


Fig. 2. The relations between CA, DSEG, SEG and the cache file

Defination.9. Cache node ability (CNA) is defined as CNA (CN-id, Ma, Ca), where CN-id is the identifier of CN; Ma is the memory storage ability for storing DSEG in its memory and Ma=$\left\lfloor \frac{CN.RSV.r_{mem}}{\text{BMU}} \right\rfloor$, Ca is its computing ability for supporting the cache agent and

Ca=.minimum $\{ \left\lfloor \frac{CN.RSV.r_{cpu}}{CA.RDV.d_{cpu}} \right\rfloor, \left\lfloor \frac{CN.RSV.r_{disk}}{CA.RDV.d_{disk}} \right\rfloor, \left\lfloor \frac{CN.RSV.r_{net}}{CA.RDV.d_{net}} \right\rfloor \}$

3. Construction of logical cluster

When GA received the information from all cache nodes in DNE, the logical computer cluster will be constructed for Parallel Cache Model. Here PCS={ $CN_i(id_i, CT_i, AS_i, RSV_i, st_i) \mid 1 \le i \le n$ } is the set of all cache nodes in DNE. Actually, as different cache node may provide different resources, we must classify the cache 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. In order to solve this problem, we should firstly discrete the power data, and fuzzy relation theory is employed to represent the data.

3.1 Resource matrix

Suppose that MR= (r_{ij}) $(1 \le i \le n, 1 \le j \le 4)$ 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 cache nodes, and the calculation method is as follows:

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

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

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

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

3.2 Partition for logical computer cluster

It is very important to divide different logical computer clusters, then the LCC division algorithm as follows:

Algorithm 3.1 Partition method for LCC.

1. $MR=(r_{ij})$ $(1\le i\le n, 1\le j\le 4)$ is the resource matrix in DNE, where n is the numbers of all cache nodes; T is a parallel cache mode task.
2. Construct the fuzzy matrix $FM=(f_{ij})$ $(1\le i\le n, 1\le j\le n)$, where $f_{ii}=1$;

$f_{ij}=1-\beta(x_1|r_{i1}-r_{j1}|+x_2|r_{i2}-r_{j2}|+x_3|r_{i3}-r_{j3}|+x_4|r_{i4}-r_{j4}|)$,when $i\neq j$, and $0<\beta<1$, and $x_1+x_2+x_3+x_4=1$, $x_k>0$ $(1\le k\le 4)$;

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 cache nodes of PCS into several equivalence class $LCC_1 \cup LCC_2 \cup \ldots LCC_e$ by FM_c ;
6. Choose a LCC for T according to its resource demand vector by algorithm3.2.

3.3 Choose LCC for parallel cache mode task

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

Algorithm 3.2. Choose LCC method for parallel cache mode task.

1. Get the resource demand vector RDV (w1, w2, w3, w4) of T;

mine= ∞ ;

2. While $LCCS\neq\phi$ 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 cache nodes of S, it is as follows:

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

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

$$y_1=ar_{Cpu}/(ar_{Cpu}+ar_{mem}+ar_{disk}+ar_{net});\quad y_2=ar_{mem}/(ar_{Cpu}+ar_{mem}+ar_{disk}+ar_{net});$$

$$y_3=ar_{disk}/(ar_{Cpu}+ar_{mem}+ar_{disk}+ar_{net});\quad y_4=ar_{net}/(ar_{Cpu}+ar_{mem}+ar_{disk}+ar_{net});$$

};
Construct the vector Y（y1,y2,y3,y4）;
e=|y1-w1|+|y2-w2|+|y3-w3|+|y4-w4|;
if mine<e then { LCC=S; mine=e };
LCCS=LCCS-{S};
}; // end while

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

3.4 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 $NSET = \{N_i\ (bw_i) \mid 1 \le i \le m\}$ 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 $NM = (s_{ij})$ $(1 \le i \le m, 1 \le j \le m)$ that is composed of the factors of the network skip distance and bandwidth. The construction method for *NM* is as follows:

$s_{ii}=1$;

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

Where,

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 cache nodes of *LCC* into several equivalence class $SLCCS = SLCC_1 \cup SLCC_2 \cup \ldots \cup SLCC_e$ by FM_c ;
5. *SLCCS* is the set of optimized *LCC* according to the network factors.

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

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 $NTS = \{NT_1, NT_2, \ldots, NT_{nt}\}$.

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 *GMG* includes;
2. **Dynamic rule (dr)** is defined as dr (ct, nt, br, rul, w, sta, life), where ct ∈ CTS, nt ∈ NTS, br ∈ BRS; rul is the formalization description of dr; w is the its weight value; and sta is its state, and sta ∈ {"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 *GMG,* 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 *GMG,* we call *dr* as **castoff rule (cr)**. Its state is "*Castoff*".

The state graph of rules is presented in figure 3.

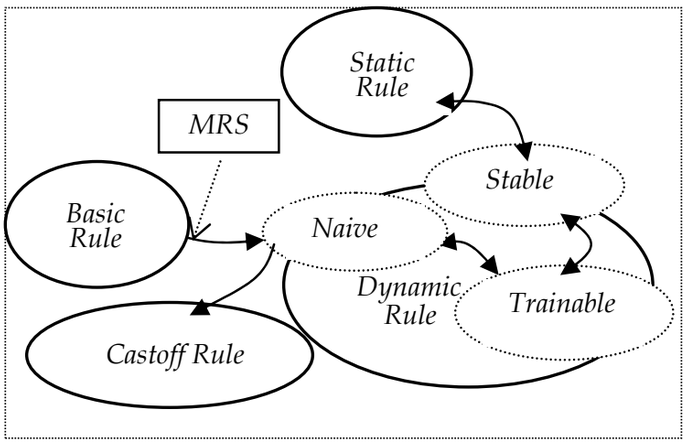


Fig. 3. The state graph of rules

The dynamic knowledge is the set of all the dynamic rules in *GMG*. The static knowledge is the set of all static rules. The basic knowledge can be earned by passive learning. To adapt the variety of cache resources, the dynamic rules must be generated at the start of the cache and revised during the caching process. Therefore, reinforcement learning can be adopted in the revising mechanism. Resources utilization for cache 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 cache 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 >= 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".

5. Cache allocation methods

For designing the allocation methods, we introduce some parameters firstly.

5.1 Parameters

Suppose that there are p cache nodes CN_1, CN_2 … CN_p, and their memory storage ability are Ma_1, Ma_2 … Ma_p, their computing ability are Ca_1, Ca_2 … Ca_p; so, the total memory storage ability is Gmc= $\sum_{1\le i\le p} Ma_i$, and the total computing ability is Gcc= $\sum_{1\le i\le p} Ca_i$; and, there are m cache files {CF_1, CF_2 … CF_m} in PCMGM and their segment lengths are {$Segl_1$,$Segl_2$ … $Segl_m$}. The parameters are as follows:

Average segment length is Asl= ($\sum_{1\le i\le m} Segl_i$)/m;

Total cache ability of Grid memory is Tc= $\left\lfloor \frac{Gmc}{Asl} \right\rfloor$, and Tc denotes that the total numbers of DSEG can be stored by grid memories at one moment;

Average cache ability of cache node is Ac= $\lfloor Tc / p \rfloor$, and Ac denotes that the average numbers of DSEG can be stored by one cache node memory at one moment;

Average computing ability demand of cache agent is Ada that can be determined by the RDV of all cache nodes.

Total cache agent ability is Tca= $\left\lfloor \frac{Gcc}{Ada} \right\rfloor$, and Tca denotes that the total numbers of cache agent can be supported by PCMGM at one moment;

Average cache agent ability of cache node is Aca= $\lfloor Tca / p \rfloor$, and Aca denotes that the total numbers of cache agents can be supported by one cache node at one moment;

Hot segment threshold is a constant H in PCMGM, and if the duplication width of a segment is more than H, we call it as the hot segment. The file which includes the hot segment is called as the hot file.

5.2 Construction for cache demand matrix

According to the parameters described in Section 5.1, we can construct the cache demand matrix, which is the total cache needing of all cache files, and as followings:

1. For CFi (1≤i≤m) do
{Calculate the duplication width vector

DWVi (|SEGi1|, |SEGi2| … |SEGiw|) for CFi by Segli , where iw is the numbers of the segment of CFi, and |SEGij| is the duplication width of j th segment of CFi};

2. Look for the cache file which has the maximal segment number in all cache files, and the maximal segment number is k;
3. Through the vector set DWV1, DWV2 … DWVm and k, we construct the cache demand matrix: CDM= (qij) (1≤i≤m, 1≤j≤k), Where qij is the duplication width of the jth segment of CFi, and qij ≥0; If the jth segment of CFi is not existed, then qij=0.

In CDM, the sum value Tcd= $\sum_{1\le i\le m,\ 1\le j\le k} q_{ij}$ is the total cache demand of all cache files, and we hope the formula (1) is true. In fact, it is not possible.

$$Tcd \le Tc \tag{1}$$

When $Tcd \ge Tc$, we can processes the *CDM* by the techniques of cut-matrix and compress in order to make the formula (1) is true.

5.3 Cut-matrix and compress technique

The *cut-matrix* is a special $m \times k$ matrix $CM=(cq_{ij})$ $(1 \le i \le m, 1 \le j \le k)$, where $cq_{ij}=Cut\text{-}v$ and *Cut-v* is a integer. The process is presented as follows:

```
NCDM=CDM; /* NCDM is the cut CDM that can
                satisfy the formula (1)*/
Cut-v=0;
Repeat do
   Cut-v++;
   Set all elements of CM as Cut-v;
   NCDM=NCDM-CM;
     /* - is the subtraction of Matrixes and if
       NCDM=0, then NCDM-CM=0, 0 is zero-matrix
       */
   Calculate the Tcd by NCDM;
          Until the formula (1) is true;
```

The *NCDM* is the new cache demand matrix and all its *DSEG*s can be cached in grid memory. The demand of *CDM-NCDM* will be save the disk. The cut-matrix is presented in the figure 4.

According to the *CA* conception, the corresponding relation between a *CA* and a *DSEG* is one to one. But if a set of duplications of the same segment of be stored in the same cache node, the duplication will be stored one time in the cache node. The corresponding cache agent is established many times in the cache node, so this mechanism can improve the use rate of memory resources, but the computing ability of the cache node may be overload. The average ratio of the *CA* numbers and the *DSEG* numbers is called as the *cache agent density* (CAD). The compress principle is presented in the figure 4.

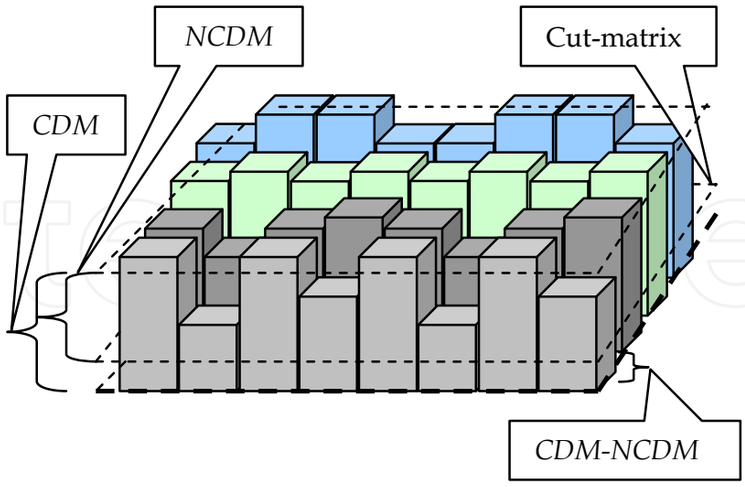


Fig. 4. The relations between CDM, Cut-matrix, and NCDM

Strategy 1. *d-density* **allocation strategy** means that the allocation ratio between the numbers of *CA* and the numbers of their *DSEG* for one segment in the same cache node is *d*. Namely, one *DSEG* has *d CA*s in the same cache node. *d* is the average in PCMGM.

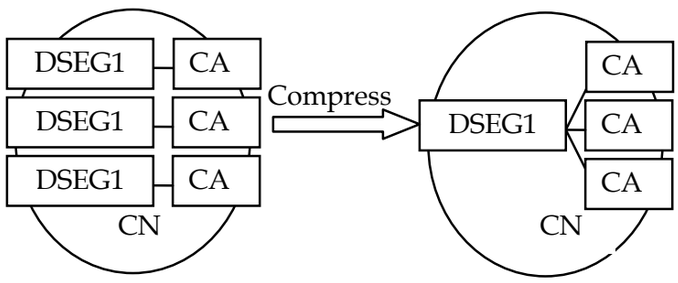


Fig. 5. This is the compress storage in a cach node for the same duplication and it is the 3-density allocation strategy.

In order to use the storage of grid memory effectively, we hope *CDM*=*NCDM*. Through the *d-density* allocation strategy**,** we can make the formula (2) is true, where *Tcd*(*NCDM*) means that the total cache demand of *NCDM*.

$$\mathrm{Tcd(NCDM)} < \mathrm{Tc} \times \mathrm{d} \tag{2}$$

5.4 Cache allocation

For all cache nodes CN_k ($1 \le k \le p$) , we get a subset of *NCDM* elements SE to make the formula (3) is true. ε is a small constant. For each *DSEG*s, we adopt the d-density allocation strategy to appoint CA.

$$\sum_{1\le i\le p,} Ma_i - \varepsilon \le \sum_{q_{ij}\in SE,} q_{ij} \le \sum_{1\le i\le p,} Ma_i + \varepsilon \quad (3)$$

The *DSEG*s in *CDM-NCDM* will be written into the local disk of cache node.

5.5 Dynamic adjusting mechanism for cache node

Because of the unstable of the cache node, the cache node can be joined or disjoined in dynamic. So the grid memory is changed in PCMGM. The dynamic adjusting mechanism for cache node must be discussed. We call the cache node which can't work for grid as the *failure cache node* and the new joined computer as the *new cache node*.

Definition.8. Computing density vector is defined as $CDV = (cd_1, cd_2 \ldots cd_p)$, where cd_i is the value of density allocation strategy of CN_i. If $cd_i < d$, we call CN_i as the *low-density cache node*; If $cd_i > d$, we call CN_i as the *high-density cache node*;

Definition.9. Failure storage matrix is defined as $FSM = (fq_{ij})$ $(1 \le i \le m, 1 \le j \le k)$, where fq_{ij} *is DSEG* numbers of the *j* th segment of CF_i, which *DSEG*s are stored in the *failure cache node*.

Definition.10. effective storage matrix is defined as $ESM = (eq_{ij})$ $(1 \le i \le m, 1 \le j \le k)$, and *ESM=NCDM-FSM*.

Definition.11. Migration storage matrix is defined as $MSM = (mq_{ij})$ $(1 \le i \le m, 1 \le j \le k)$, where mq_{ij} is *DSEG* numbers of the *j* th segment of CF_i, which *DSEG*s are stored in the *failure (or overload) cache node* , and the *DSEG*s of *MSM* will be migrated.

5.5.1 Failure process

The failure process description is as follows:

1. Calculate the *DSEG*s which the current failure cache nodes include, and Construct the failure migration storage matrix *MSM*;
2. Look for the low-density cache node by *CDV*, and get the free cache resources for *MSM*; Calculate the acceptable cache matrix *AMSM* by these resources, and *AMSM* will be migrated the new cache node; *MSM – AMSM* will be added into the failure storage matrix *FSM* and *FSM* will be stored into the local disk which they are.
3. Revise the computing density vector *CDV*;

5.5.2 Joining process of new cache node

The new cache joining process description is as follows:

1. Calculate the ability of *new cache nodes*;
2. If the failure storage matrix is existed, we construct the migration storage matrix *MSM* by *FSM*;
3. Look for the high-density cache node by *CDV*, and add their *DSEG*s information into *MSM*;
4. *MSM* will be migrated into the new cache nodes;
5. Revise the computing density vector *CDV*;

6. Process of PCMGM

The PCMGM process is as follows:

1. AA do the statistics work from user information and report them to GA; GA calculates the cache demand matrix CDM by the statistics information, and get a LCC (That is constructed by GA through fuzzy partition method);
2. All NAs, which LCC includes, calculate its memory storage ability Ma and its computing ability Ca;
3. GA receives the idle resource information from LCC and calculates the total computing ability Gcc and the total memory storage ability Gmc;
4. GA constructs the NCDM according to the cut-matrix and d-density allocation strategy;
5. GA allots DSEGs for all CNs;
6. All NAs of CN transfer the segments form the file server into grid memory in order to form the DSEGs; then, construct CAs for each DSEG;
7. All CAs in PCMGM start the hot segment service;
8. All users commit its access sequences to AA, AA redirect the connections to build the relations between the users and CAs;
9. All CAs and users start the file operation in Parallel and PCMGM start the dynamic adjusting mechanism by the migration.

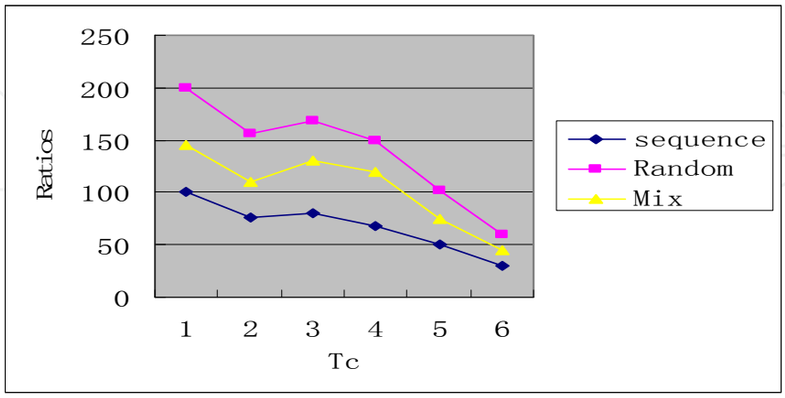
7. Experiments

In order to test this model, we built a *DNE* that is composed of 2 file servers and 8 cache nodes (personal computer), and 16 client computers, and they are connected by LAN. Each server has 4 disks, and the segments of large file are distributed in all disks in balance. There are 4 files and the length of all them is about 1GB. The segment length is 48MB. The cache nodes are classified into 2 types according to their types of cpu, memory, disk, and net adapter. The types of computers are *RSV* (2400MHZ, 768MB, 7200RPM, 100M) and *RSV* (1800MHZ, 512MB, 7200RPM, 100M). The operating systems of the computers are the LINUX. In order to test the peak access ability about the hot segment, we ignore the net factors. So, the test agent running on client end only sends the access command timely and the *CA* does the file operations and the data will be not transferred to the client end. The average value of *d*-density allocation strategy is 6.

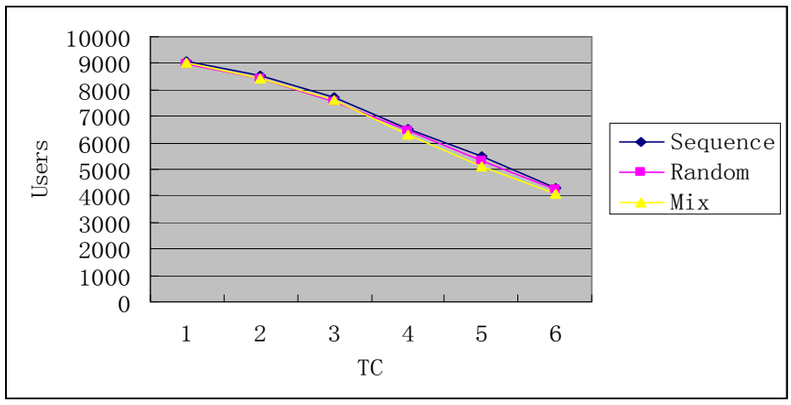
Experiment 1. The experimentation includes two kinds: the one is the response time testing in the condition that all *DSEG*s are stored in the file server disks; the other is the response time testing in the condition that all *DSEG*s are stored in Grid memory. We calculate the average ratios of response time of this two kinds. The tests include 6 times according to the *DSEG* scales. The total number of *DSEGs* is the *Tc*, 2*Tc*… 6*Tc* during the 6 times testing. In each test, we adopt the three access method for the *DSEG*s: the random access, the sequence access, and the mix access composed of the random and sequence. The test results are shown in figure 6(a), and the results show that PCMGM is high efficiency compared with the disk cache method.

Experiment 2. We had tested the variety of PCMGM performance during d is 1, 2, 3, 4, 5, and 6 (The total number of DSEGs is the Tc, 2Tc… 6Tc). The test results are shown in figure

6(b), and the results show that the key influence factor about PCMGM performance is the capacity of Grid memory. So, mining large scale idle computer to construct the grid memory can improve the concurrent access response time for the hot segments during the peak access.



(a)



(b)

Fig. 6. The test results

8. Conclusions

For supporting the concurrent access for the hot segments of large files during the peak access phase in Internet, a parallel cache model based on the grid memory (PCMGM) was proposed in this paper. The concurrent accessing of hot segments depends on the disk performance greatly during the peak access. Through the grid technique, we can mine a lot of idle computers to construct the grid memory in order to store the duplications of hot segments. Because of the heterogeneous resources and the unstable catachrestic, the effective cache model is very difficult in the dynamic network environment. Through the techniques of *CA*, *DSEG*, the dynamic learning and the logical computer cluster partition based on fuzzy theory, PCMGM can support the parallel cache. These approaches can raise the peak access performance for hot segment of large files. It can fit for the grid computing and the large scale VOD in internet.

9. 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 Shanghai Education Committee Foundation (No. 06QZ002, 07ZZ92).

10. References

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

Craig S. Freedman, Josef Burger, and David J. Dewitt: SPIFFI-a Scable Parallel File System for the Intel Paragon. IEEE Transactions on Parallel and Distributed System, Vol.7 (11), (1996):1185-1200

E. Osawa: A Scheme for Agent Collaboration in Open MultiAgent Environment .Proceeding of IJCAI'93, (1993):352-358

E. P. Markatos and G. Dramitions: Implementation of Reliable Remote Memory Pager. In proceedings of the 1996 Usenix technical Conference, (1996):177-190

Horst Eidenberger: Gigabit Ethernet-Based Parallel Video Processing, Proceedings of 11th International Multimedia Modelling Conference (MMM'05), IEEE Computer Society, (2005):358-363

I. Foster, C. Kesselman: The Grid: Blueprint for Future Computing Infrastructure. San Francisco, USA: Morgan Kaufmann Publishers, (1999)

J. Fernandez,J. Carretero, F. Garcia-Carballeira, A. Calderon, J. M. Perez-Menor: New Stream Caching Schemas for Multimedia Systems, In Proceedings of First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05),IEEE Computer Society,(2005): 77-84

J. O. Kephart, Chess, D. M: The Vision of Autonomic Computing, IEEE Computer, (2003): 41-50

J. Carretero, F. Perez, P. de Miguel, and L.Alonso: ParFiSys: A Parallel File System for MPP. Operating System review, Vol.30 (2), (1996):74-80

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)

P. F. Corbett and D. G. Feitelson: The Vesta Parallel File System. ACM transaction on Computer Systems, Vol.14 (3), (1996):225-264

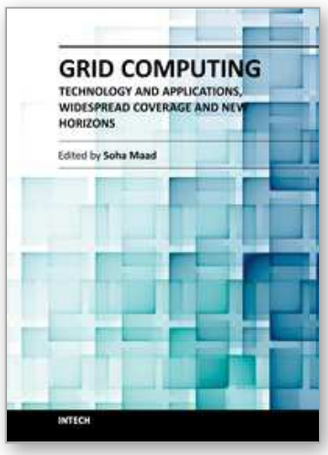
Pal Halvorsen, Carsten Griwodz, Vera Goebel, Ketil Lund, Thomas Plagemann, Jonathan walpole: Storage System Support for Continuous Media Applications, Part2: Multiple Disks, Memory, and Integration, IEEE Distributed systems online 1541-4922,vol.5(2), (2004)

Rajkumar Buyya: High performance Cluster Computing Architectures and Systems, Prentice Hall, (1999)

Seogyun Kim, Jiseung Nam, Soon-ja Yeom:Effective delivery of Virtual Class on Parallel Media Stream Server, Proceedings of the International Conference on Computers in Education, IEEE press,2002

T. Kimbrel et. AI: A Trace-Driven Comparison of Algorithms for Parallel Prefetching and Caching. In Proceedings of the 2nd International Symposium on Operating System Design and Impletation, USENIX Association, (1996):19-34

Wooldridge, M.: An Introduction to Multivalent System, John Wiley & Sons (Chichester, England). ISBN 0 47149691X, (2002)



Grid Computing - Technology and Applications, Widespread Coverage and New Horizons

Edited by Dr. Soha Maad

ISBN 978-953-51-0604-3
Hard cover, 354 pages
Publisher InTech
Published online 16, May, 2012
Published in print edition May, 2012

Grid research, rooted in distributed and high performance computing, started in mid-to-late 1990s. Soon afterwards, national and international research and development authorities realized the importance of the Grid and gave it a primary position on their research and development agenda. The Grid evolved from tackling data and compute-intensive problems, to addressing global-scale scientific projects, connecting businesses across the supply chain, and becoming a World Wide Grid integrated in our daily routine activities. This book tells the story of great potential, continued strength, and widespread international penetration of Grid computing. It overviews latest advances in the field and traces the evolution of selected Grid applications. The book highlights the international widespread coverage and unveils the future potential of the Grid.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Qingkui Chen, Lichun Na, He Jia, Song Lin Zhuang and Xiaodong Ding (2012). Research and Implementation of Parallel Cache Model Through Grid Memory, Grid Computing - Technology and Applications, Widespread Coverage and New Horizons, Dr. Soha Maad (Ed.), ISBN: 978-953-51-0604-3, InTech, Available from: http://www.intechopen.com/books/grid-computing-technology-and-applications-widespread-coverage-and-new-horizons/research-and-implementation-of-parallel-cache-model-through-grid-memory

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen