Improving Cache Global Consistency and Hit Ratio in Dependency Objects with Semantic Spatial Locality Correlations

CHING-SHUN HSIEH[1] JUI-WEN HUNG[2]
[1]Department Commercial Technology and Management
[2]Department of Information and Management
[12]Ling Tung University
No.1, Lingdong Rd., Nantun District, Taichung City 408, Taiwan (R.O.C.)
TAIWAN (R.O.C.)
[1]sjsoon@mail.ltu.edu.tw
[2]harng@mail.ltu.edu.tw

Abstract: - On requesting cache data in disk, the distribution of spatial locality is critical to access performance. Unfortunately, spatial locality properties of cached data is largely ignored, and only temporal locality is considered. Besides, an individual disk object might induce different dependency relations in different applications and possibly partial dependency to several distributed original data source (ODS). This interesting property might be neglected. Normally, these situations can be improved by effectiveness of storage caching, prefetching, and prediction of the user navigation behavior, data layout of storage systems and global consistency storage replicas. In this paper, we consider the problem and solve it using data mining techniques and a service routable consistency framework (Global Distributed Hierarchical Cache Consistent Model; GDHCCM). The model based on a scalable routing service algorithm that dynamic reconfiguration forwarding data path within hierarchical enterprise region portals. A novel hypergraph scheme was also proposed to represent the complex object relations among the applications. Instead of a local measure that depends only on common objects among patterns, we propose a global measure process based on the semantic properties of these patterns in the overall data set. The experiments show the effectiveness of the proposed framework. Apply scenario can reduce the global patch service cost; improve performances and minimum the turnaround time in access the scope of computer games or virtual environments (VE).

Key-Words: - cache**,** spatial locality, global consistency, hypergraph, prefetching, service routing, virtual environments

1 Introduction

With the rapid growth of the CPU computation abilities via VLSI development rapid and increasing demand to develop advanced applications. It is apparent that the disk bottleneck effect is worsening in modern computer systems, while the role of the hard disk as the dominant storage device will not change in the foreseeable future, and the amount of disk data requested by applications continues to increase. Unfortunately, spatial locality properties of cached data is largely ignored, and only temporal locality is considered. Besides, an individual object might induce different dependency relations in different applications and partially dependency to several distributed original data source. Several studies are currently focusing on the design and concept of using caching, middleware and database replication technologies [23,24,25] to meet the requirements. However, existing consistency architectures do not meet real-time and globally consistence within hierarchical enterprise region. This paper focuses on using data mining and a service routable consistency framework (GDHCCM) for improving cache replica global consistency and hit ratio in dependency objects with semantic spatial locality correlations.

1.1 Background

The hard drive is the most commonly used as secondary storage device supporting file accesses and virtual memory paging. While its capacity growth pleasantly matches the rapidly increasing data storage demand, its electromechanical nature causes its performance improvements to lag painfully far behind processor speed progress. On the other side, faced with the ever-growing importance and immense popularity of the 3D visualization, especially in the scope of computer games or virtual environments (VE), users expect intelligent processing (such as processors that recognize their true information needs) and a broad

and accurate coverage of all realms of their lives (such as information on "real-world" topics or the possibility to enhance the fidelity of the scenes). Unfortunately, because storage systems are much slower than processors, these applications often end up wasting a substantial fraction of their execution times waiting for storage requests to complete [21,22]. We call this problem as demand-supply-gap problem. Rather than fetching data from disk on demand, predictive prefetching involves fetching data from disk in anticipation of an upcoming attempt to access data. The potential benefit of an predictive prefetching is that, if the anticipated attempt occurs, then the prefetching will have hidden some or all of the time it takes to fetch that data from disk, so the accessing application will waste less time stalled on read/write operations. In other words, the management for swapping objects is dictated by the placement of data [21,22]. Judicious placement of data on storage media is therefore critical, and can significantly affect the overall performance of the storage system.

However, it is not an easy task to exploit the intelligence in storage systems. One primary reason is the system latency between VE applications and storage systems. In such a case, VE do not consider the problem of access times of objects in the storage systems. Recently several researches [11,12,13,5,6,10] reveal that they always simply concerned about how to display the object in the next frame during the very near future (i.e., temporal locality; here, we call it as transient locality). Furthermore, previous works also fail to take into account the relationships between objects. As a result, the VE can only manage data at the rendering and other related levels without knowing any semantic information such as semantic correlations between data. Besides, an individual object might induce different dependency relations in different applications and partially dependency to several distributed original data source. This motivates a more powerful analysis tool to discover more complex patterns, especially semantic patterns, in storage systems. Therefore, the aim of our work is to decrease this latency through intelligent organization of the access data, enabling the clients to perform predictive fetching, and improving cache data real-time correctness and hit ratio in dependency objects.

1.2 Contributions of the paper

In this paper, we consider the problem and solve this using data mining techniques [1]. Two main observations can support the adoption of data mining schemes. First, at a broad level, data mining is the process by which one can extracts accurate and previously unknown information from large volumes of data. This information should be in a form that can be understood, acted on, and used for improving decision processes in scientific applications, especially in 3D visualization applications [13,12,2,5,6,10]. Second, it is clearly when users traverse in a VE, some potential semantic characteristics will emerge on their traversal paths. If we collect the users' traversal paths, mine and extract some kind of information of them, such semantic information can help to improve the performance of the interactive VE. For example, we can reconstruct the placement order of the objects into a storage system according to the common section of users' path. In other word, exploring these correlations is very useful for improving the effectiveness of storage caching, prefetching, data layout, and disk scheduling.

Compared with the previous schemes, our approach has the following different characteristics. (1).Scheme_based: traditional schemes always utilize the tree_based data structure with locality (either parent-child locality (i.e., temporal locality) or spatial locality); (2).Complexity_based: previous schemes manage these data structures to fit the memory constraints without considering the real demand for object; (3).Semantic_based: the semantic property seems very weak compared with ours; (4).Knowledge_based: the knowledge of previous schemes only hold for very near future. If the time went by longer, the prediction may be wrong. Besides, we focus on the process of knowledge construction especially. Knowledge construction is a process that reflects not the end result a user would want, or what the system should produce, but rather the operations a user will perform using the tools of a knowledge construction environment. This is an important distinction because it places the user first in the knowledge construction process, but it also highlights the semantic gap between desired outcomes and the available tools. Existing cache-related environments focus on identifying what kind of information the system needs to discover successfully. However, it is very important to take the design initiative of enabling the users' expertise within a knowledge construction process, rather than attempting to supplant it.

This paper also presents a service routable consistency framework (GDHCCM) that is capable of supporting ripple propagate cache updating copies that existed in distributed heterogeneity of storage. Besides, an individual object might induce different dependency relations in different

applications and partially dependency to several distributed original patch data source (OPDS). By using a new scalable web service routing policies that can dynamic reconfiguration distributed forwarding cache patch data path in hierarchical enterprises storage and improving cache data real-time correctness and hit ratio.

A novel approach is used to discover the promising patterns, cluster similar patterns in a hypergraph form and layout these clusters into the storage systems for efficient predicting runtime access patterns. First, the VSPM algorithm [16] was used to discover semantic patterns in VE. Besides, two clustering methods were proposed to cluster the similar patterns for reducing the access time. One is based on the idea of co-occurrence of transaction data have developed. They are usually measured by Jaccard coefficient SIM(T1, T2)= |T1 ∩ T2| / |T1 ∪ T2| [8]. The other clustering scheme is based on the hypergraph-based model. Our purpose is to propose a semantic clustering learning technique, which collects the frequent patterns and uses hypergraph to represent different but complex relationship among the objects, then obtains the semantic clusters by a semantic-based hypergraph partitioning. In this model, the vertex set corresponds to the distinct objects in the VE and the hypergedges correspond to the frequent sequential patterns. Both of them will make similar objects much closer to be accessed in one time. These result in less access times and much better performance. We also compare the distinctions between them. Moreover, we also have evaluated the benefits of object correlation-directed prefetching and disk data layout using the real system workloads.

On the basis of the ubiquitous service routable framework, several models were built. The OPDS dependency spatial cache usage mining rate is a function of dependency replica operation response time and turnaround time. The updating route policy scheduling rate is a function of usage mining and reservation priority. The production rate of activity represented as a time variable. Different value added update service schema was also analyzed. For example, active updating OPDS pushed by real time, periodic, reservation, and queuing with priority/degradation/preemption etc. Apply scenario can reduce the global cache patch service cost of transnational enterprise storage cache and minimum the turnaround time of cache patch delay.

1.3 Outline of paper

The remainder of the paper is organized as follows. Related works are given in Section 2. In Section 3, we describe our hypergraph model and problem formulation. The suggested clustering mechanism is explained with illustrative examples shown in Section 4. In Section 5, the forwarding Enterprise Portal (EP) agent [26-29] use pipeline mechanism automatically conversion all kinds of dependency patch and update special locality caches with OPDS dependencies. The OPDS hierarchical ripple propagation web service [30-32] routing algorithm discussion is given in Sect. 6. Section 7 presents our experiment results. Finally, we summarize our current results with suggestions for future research in Section 8.

2 Related Works

In this section, we will briefly describe related works about the traditional prefetching and caching mechanisms applied to virtual environments and hypergraph clustering schemes, respectively.

2.1 Prefetching and caching methods in V.E.

Since the navigation in virtual environments consists of many different detailed objects, e.g., of CAD data that cannot all be stored in main memory but only on hard disk. Many techniques were proposed for rendering complex models used today, as [6] cited, the previous researches fall into three categories: First, data-related: it is concerned with the organization of objects, such as object partitioning, connectivity and block size. Like the level-of-detail (LOD) management [20], hierarchical view-frustum and occlusion culling [12], working-set management (geometry caching) [20] are these examples; Second, traversal-related: this focus on reduction of access times for objects. Traditional cache-efficient layouts [10,11] (also called cache-aware), based on the knowledge of cache parameters, utilize the different localities to reduce the number of cache misses and the size of the working set; Finally, another variation is cache-oblivious algorithms [5]. Instead, they do not require any knowledge of cache parameters or block sizes of the memory hierarchy involved in the computation.

However, large polygons of such highly complex scenes require a lot of hard disk space so that the additional data could exceed the available capacities [3,10]. As Chim et. Al. [19] shows that LRU-based and related schemes do not appropriate in a context where objects accessed by a client might change over time. However, the semantics of data access is more important in defining the placement

policy [3,15,17,18,9]. To meet these requirements, an appropriate data structure and an efficient technique should be developed with the constraints of memory consumption.

2.2 Hypergraph_based pattern clustering methods

The fundamental clustering problem is to partition a given data set into groups (clusters), such that data points in a cluster are more similar to each other (i.e., intra-similar property) than points in different clusters (i.e., inter-similar property) [2,4,7].These discovered clusters are used to explain the characteristics of the data distribution [2,4,,7]. However, these schemes fail to produce meaningful clusters, if the number of objects is large or the dimensionalities of the VE (i.e., the number of different features) are diverse and relatively large.

2.3 Cache mutual inconsistent with distributed OPDS

Today's global cache consistent solution have been proposed using data integration middleware [24] techniques like xml-based integration middleware, xml-based data integration platform and xml-powered integration middleware. All these data integration system that enables enterprises to rapidly build web services and applications that can query multiple, disparate data sources and provides a unified result. If cache with partial dependency relationships with specific OPDS object keeps mutual consistency, the following problems will be happened. 1. The Bullwhip Effect will be occurred in dependency cache replicas flow chain operation. 2. Enterprise storages can't make sure the newest version caches. 3. Enterprise knowledge base will occur horizontal information lacks harmony. 4. If OPDS site can't control its cache updating flow, transaction and usage mining results, it will not be able to support different priority and value added updating services. 5. The dependent cache replicas usage mining results can't be sent to OPDS site to make adaptive cache updating route policies and value added updating services.

3 Hypergraph Model and Problem Formulation

Before we explain our hypergraph model, one motivation of example is show below. This can be classified into two different conditions for the relationships among the objects. Under the concern of *intra-similarity*, every *object* represents some importance. For example, the support for frequent pattern *abcd* is 5, but the supports for the object *a, b, c, d* are *5,6,7,8* respectively. Under the concern of *inter-similarity*, shown in Figure 1, every *frequent pattern* represents some importance. Besides, the support for frequent pattern abcd is 5, but the supports for the object *abe, abcde, cd, df* are 5, 4, 6, 8 respectively. From the above observations, these patterns are intertwined with the relationships and should be properly and efficiently managed. Therefore, those observations also motivate us to adopt the HG model for representing the relationships.

In this section, given that the valuable frequent pattern found, we employ a HG-based clustering strategy for our hierarchical placement method. The HG was used for representing both intra-relationships and inter-relationships. Finally, we propose a simple and efficient HG-based partition scheme to cluster these partitions for data layout. There are two problems involved in this situation. One is *persistent clustering* − significant improvements are accomplished, but are at the expense of the quality of the final placement solution. As the operations for clustering increase, the quality of clustering deteriorates. Another problem is the control of physical cluster sizes. During the placement step, the size of clustered objects may be too large relative to decision dimensionality, which results in the degradation of the final placement solution quality. Therefore, our goal of HG_based clustering is to address these two deficiencies. First, we try to generate high-quality clustering solution so that any potential loss of the final placement solution quality is minimized. Second, we take advantage of the hierarchical nature of clustering so that the clustered objects are maximization of intra-similarity and minimization of inter-similarity. The notations used in the HG_based model are as follows.

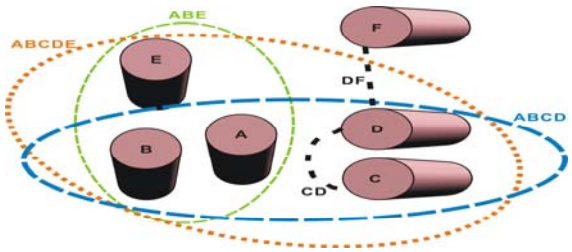

Fig. 1. Demonstration for intra-/inter-relationships among the frequent patterns.

3.1 Notations for hypergarph partition

A hypergraph *H =(V, N)* [3,7] is defined as a set of vertices *V* and a set of *nets (hyper-edges*) *N* among

those vertices. Every net $n_j \in N$ is a subset of vertices, i.e., $n_j \subseteq V$. The size of a net n_j is equal to the number of vertices it has, i.e., $s_j = | n_j |$. Weight (w_j) and cost (c_j) can be assigned to the vertices (v_j) and edges ($n_j \in N$) of the HG, respectively. $K = \{V_1, V_2, ..., V_K\}$is a *K-way* partition of *H* and satisfies the following conditions: (1) each partition is a nonempty subset of *V;* (2) partitions are pairwise disjoint; (3) union of *K* partitions is equal to *V*.

In our model, we assign every object to one vertex, and every frequent pattern is represented by one hyper-edge (net). As shown in Figure 1, according to how many objects involved, object a, b, c, d, e, and f are circled together in different line form, respectively. Since there are five different patterns, we plot five different nets for demonstration.

Finally, we will define our problem in two phases. Phase I: given a frequent pattern set $P = \{p_1, p_2, ..., p_n\}$, we design a efficient formulation scheme to bridge two different domain knowledges (i.e., *P* and HG model); phase II: in order to reduce the disk access time, we distribute *P* into a set of clusters, such that minimize inter-cluster similarity and maximize intra-cluster similarity.

4 Our Hypergraph-based Clustering Algorithm

First, as mentioned previously, let the object corresponds to a vertex, and a frequent pattern also corresponds to a hyperedge. The weight ψ_e of a hyperedge e is defined as $1/|e|$, which is inversely proportional to the number of objects that are incident to the hyperedge. Based on main concepts of [2], we propose our semantic-based hypergraph clustering scheme. Let two objects u and v are given, the similarity score *d (u,v)* for between *u* and *v* is defined as

$$d(u,v) = \sum_{e \in E | u,v \in e} \frac{\psi_e}{(m(u) + m(v))} \quad \textbf{(1)}$$

Where *e* is a hyperedge connecting objects *u* and *v*, ψ_e. is a corresponding edge weight, and *m(u)* and *m(v)* are the some interesting measures of *u* and *v*, respectively. As Han. *et. al.* [7] cited that the support measure is not a proper measure used in a hypergraph model. Therefore, in our experiments, shown in next section, adopt the confidence measure. The similarity score of two objects is directly proportional to the total sum of edge weights connecting them and is inversely proportional to the sum of their measures. Suppose N_u is the set of neighboring objects to a given object *u*. We define the closest object to *u*, denoted *c(u)*, as the neighbor object with the highest similarity score to *u*, i.e., *c(u) = v* such that *d (u,v)* = Max $\{d(u,z) | z \in N_u\}$.

Now, we will explain our clustering algorithms. The main ideas come from both *object-based* and *HG-based* mechanisms. Since there are multiple relationships exist in *object-to-object* and *pattern-to-pattern* formats. There are not sufficient to represent such relationships if the non-hypergraph models are used. This is our main motivation for HG-model.

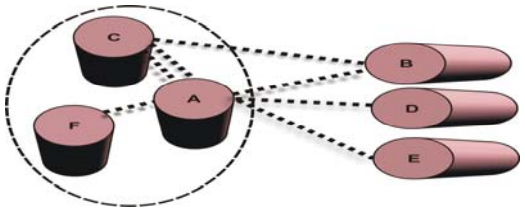

Fig. 2. The initial condition.

In Figure 2, the dot lines denote which vertexes are connected in the hyperedges. The circle is especially for represented one hyperedge $\{A,C,F\}$. Since the multiplicity of hyperedge $\{A,C\}$ is two. Therefore, there are two hyperedges between vertex *A* and *C*.

In order to identify the globally closest pair of objects with the highest score, a data structure with *priority-queue (PQ)* mechanism is implemented. There are two phases in our algorithm. Phase 1: we would like to build the *PQ* structure initially. First, for each object *u* in the *Obj* (Object Set), the closest object *v* and its associated similarity score are found, and inserted into the *PQ* with the key *d*. Note that for each object *u*, only one tuple with the closest object *v* is inserted and maintained. Due to the less complexity in computation, this vertex-oriented *PQ* is more efficient than methods of edge-based. Phase 2: First, we pick up the top tuple *(u, v, d)* in the *PQ* (step 7). If conditions are satisfied, the pair of objects *(u, v)* is clustered and created a new object *u'* (step 8). In step 9 and 10, the new closest object *v'* is found and the T set is updated. Therefore, a new tuple *(u', v', d')* is inserted into *PQ* with *d'* as the new key. Since the clustering changes the vertex-connectivity of HG, some of previously calculated similarity scores might become invalid. Thus, the similarity scores of the neighbors of the new object *u'* need to be re-calculated, and the *PQ* is adjusted accordingly. The following is the pseudo codes of object-based hypergraph clustering algorithm.

- Object-oriented Hypergraph_based Clustering (OHGC) Algorithm

// *D* is the database. *P* is the set of frequent patterns. *Obj* is the set of frequent patterns. *T* is the set of clusters, and is set to empty initially.
Input: *D, P, Obj,* and *T*.
Output: *T*
Begin

```
// Phase 1: Initialization step for
            Priority_Queue (PQ)
1. While (let each object u∈Obj and Obj
    is not empty) do
2. Begin
3.  Find closest object v, and its
    associated similarity score d;
4.  Insert the tuple (u, v, d ) into PQ
    with d as key;
5. End;      // while in Phase 1.

// Phase 2: Hypergraph Clustering based
            on PQ
6. While (user-defined cluster number is
       not reached or top tuple's score
       d >0) do
7.      Pick top tuple (u, v, d ) from PQ;
8.      Cluster u and v into new object u'
       and update the T;
9.      Find closest object v' to u' its
        associated  similarity score d';
10.     Insert the tuple (u', v', d') into
       PQ with d' as key;
11.     Update similarity score  of all
       neighbors of u;
12. End;       // while in Phase 2.
```

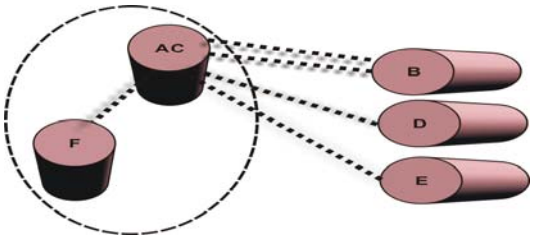

Fig. 3. After the computation of step1 to step4, the vertex *C* was chosen and merged with vertex *A*.

Example 1 (*OHGC*). Assume that the system has 6 objects and 8 frequent patterns. Let *Obj* = {*A, B, C, D, E, F, G*} and frequent patterns ser $P = \{P_1 = AB; P_2 = AC; P_3 = AD; P_4 = AE; P_5 = AF, P_6 = AC, P_7 = BC,$ *and* $P_8 = ACF\}$. Note that the multiplicity of hyperedge {*A,C*} is two. This is one of main differences between other methods and ours. We set up the *level-wise threshold* for the *multiplicity* of frequent patterns. For example, if the support of P_i is less than some fixed constant, say α, then the multiplicity of *Pi* is set up to be 1; otherwise if the support of P_i is less than or equal to some fixed constant 2α but great than α, then the multiplicity of P_i is set up to be 2. This idea will alleviate the complexity of HG-model for future partitioning. The above initial conditions are shown in Figure 3.

5 GDHCC Model Description

Application operation patch caches in intranet have three kinds of models. In Figure 4, patch cache with partial specific remote OPDS dependency relationships were denoted by (a). Patch cache with OPDS role in extranet enterprise flow chain was denoted by (b). A replica with partial dependency with a specific self-enterprise intranet OPDS was denoted by (c). Each EP can be two roles, FP (Forwarding Portal) and FPC (Forwarding Portal Cluster).

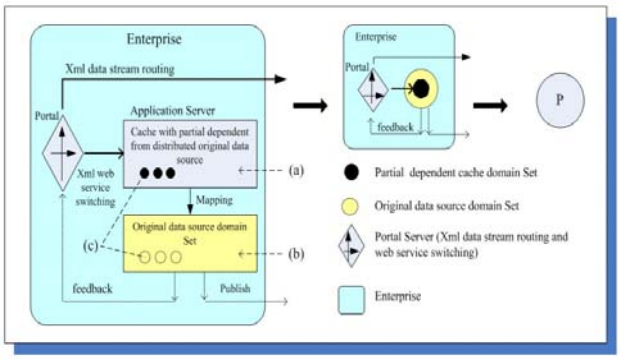

Fig. 4. Enterprise Portal (EP) Model

The ERP (Enterprise Root Portal) can get optimize web service routing table according by receiving dependency replica updating request from low layer EPs. Then ERP configure each EP web service forwarding table based on a shortest path algorithm to create the appropriate distributed data path.

5.1 Problem formulation

For a given data path, a set of EPs are needed to perform web service processing functions from an ingress EP to an egress EP, and these EPs can be physically distributed and can be duplicated. Hence, several paths are possible between ingress and egress EP nodes since multiple duplicated EPs can perform the distributed data path. For a given datapath, DP is used to denote a vector of EP_j involved in the datapath formation:

$$DP = (EP_j) \text{ for } 1 \le j \le F. \quad \textbf{(2)}$$

which is also equivalent to a vector of sets A involved in the data path formation:

$$DP = (A_j) \text{ for } 1 \le j \le F. \quad \textbf{(3)}$$

Figure 5 shows an example of a given FPC data path DP in enterprise intranet and extranet environment.

Symbol description: Enterprise is denoted by E. Routing path is denoted by r. flow path is denoted by *b* and *c*. EP encountered OPDS dependency patch replica flow will generate three kinds of message flow path , as follows.

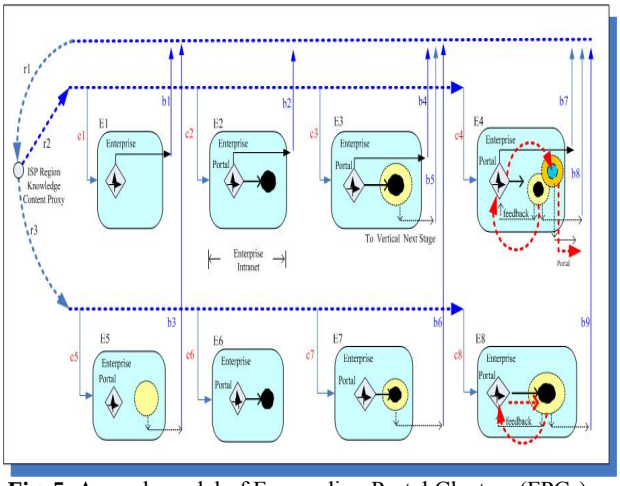

Fig. 5. A graph model of Forwarding Portal Clusters (FPCs)

Apply in enterprise extranet data path, like (4).

$$E5 \rightarrow b3 \rightarrow r1 \rightarrow r3 \rightarrow \{c7|c8\} \rightarrow \{E7|E8\} \rightarrow \{b6|b9\} \rightarrow r1 \rightarrow r3 \quad (4)$$

For redirect updating replica to other hierarchy enterprise, flow path like formula (5).

$$E5 \rightarrow b3 \rightarrow r1 \rightarrow r2 \rightarrow \{c1|c2|c3|c4\} \rightarrow \{E1|\ E2|\ E3|\ E4\ \} \rightarrow \{b1|b2|b4|b7\} \rightarrow r1 \rightarrow r2 \quad (5)$$

For synchronize trigger multiple EP horizontal patch replica consistency, the feedback intranet OPDS updating copy must be built , like formula (6)

$$E8 \rightarrow b9 \rightarrow r1 \rightarrow \{\{r2 \rightarrow \text{same above (a)}\} \mid \{r3 \rightarrow \text{same above (5)}\}\} \quad (6)$$

5.2 OPDS ripple propagation patch routing mechanism

In this section, which consists of OPDS subscribe and ripple propagation message chain shown in Figure 6. When OPDS object was triggered by update event, the message was propagation in active, real time, and automation. Then all patch cache with the specific OPDS dependent relationship could be globally updated.

5.3 GDHCCM model of patch cache consistency

In this paper, a novel framework was proposed which consists of active, real time, automation, and global routing patch replica from OPDS domain to GDHCCM set. See in Figure 7.

When OPDS object occurred add/insert events, according to subscribe lists. New copies will propagation push to all global EP which contained dependency cache replica information.

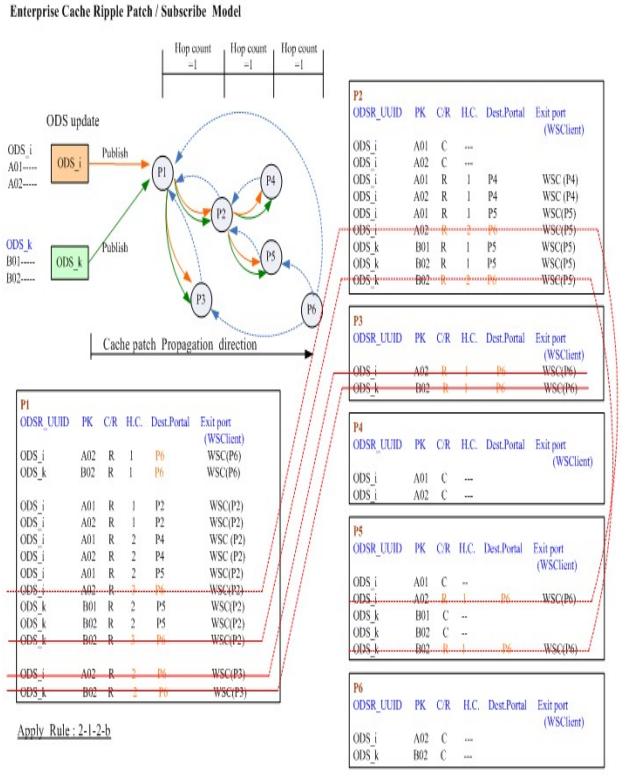

Fig. 6. OPDS ripple propagation patch chain

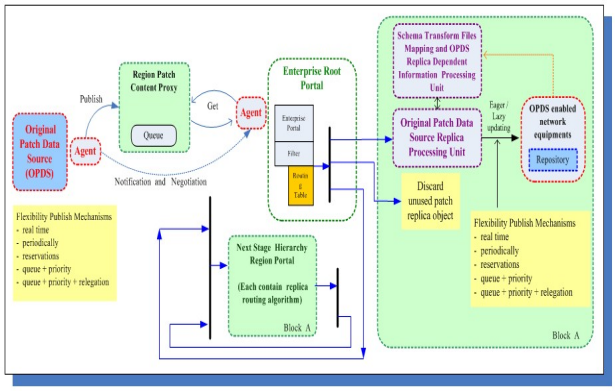

Fig. 7. GDHCCM concept model

The EP then make one of the choices about discard/rerouting to lower hierarchy portal, automatic pipeline processing and schema transformation. All wide area OPDS dependent cache replica consistency and correctness can be automatic maintenance. Enterprise intranet application system need not change any source code.

5.4 System sequential diagram

In Figure 8 describe the UML sequential diagram about the concept model of system object.

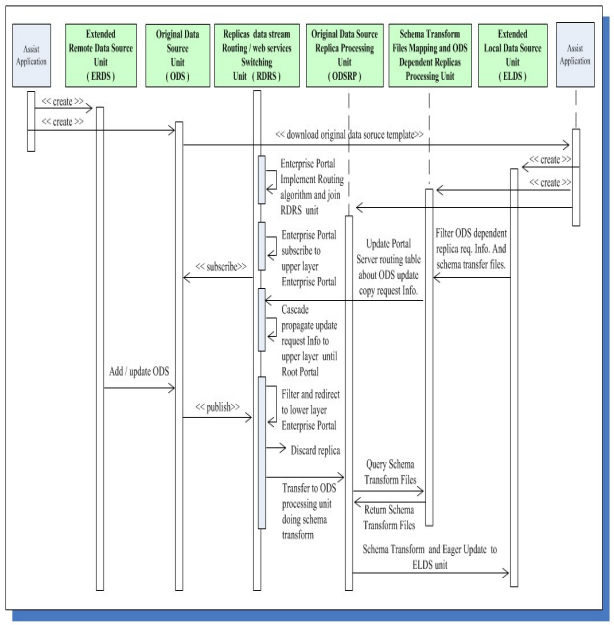

Fig. 8. System object sequential diagram

6 Ripple Propagation OPDS Routing Algorithm

Routing algorithm was built into each EP for assisting the xml data stream route path choosing.

6.1 Basic principle

The routing table fields description include: 1. OPDS dependency replica updates request information always maintained by EP. 2. Information Included UUID value, PK value , Hop count, web service client etc. need to log in portal routing table from lower hierarchy enterprise request issues. 3. Portal's replica update routing information can come from intranet and extranet enterprise. 4. When a OPDS update replica publish to Portal, it will discard, send to processing unit, or reroute to lower portal depend on routing table check results. 5. For each route request from lower hierarchy portal will add a HC value to routing table's Hop Count value field.

Table 1. Routing table field's definition

Routing Table (RT) fields	Description
action state flag (ASF)	denote when add/delete OKMS dependent replicas in enterprise LKS
OKMS template ID (UUID)	OKMS template identify defined by OKMS site
OKMS replica Primary key value(PK)	instance of specific OKMS template
local connect / remote connect (LC / RC)	dependency replica update request issued by local enterprise or lower level enterprise portal
Destination Portal ID (DPID)	original enterprise Identify that issues the update request
Hop Count value (HC)	the distance between the enterprise contained the newest routing information and request enterprise
WSC_port	call next stop portal web service

6.2 Routing algorithms

Apply the rules to each route request. The *ASF(Action State Flag)* is sent from lower layer EP by portal agent and web services. By modifying Bellman-Ford Equation (*DV* algorithm) :

Let $d_x(y)$ is the minimum path cost from node x to y, $c(x,v)$ is a moving path from x to v.

$$d_x(y)=min_v\{c(x,v)+d_v(y)\} \quad \textbf{(7)}$$

In distributed and asynchronous algorithm, each node will transfer its distance vector copy to all adjacencies at random time.

```
If (ASF==add) {
 // (rule.2-1) : ASF == add
   If (route request info.(UUID、PK、DPTD) sent from lower portal does not exist){
     //(rule.2-1-1)
     add this information into RT and upper cascade propagate this add event registry}
   Else If (route request info (WSC_port) identical){
        //(rule. 2-1-2-a)
          replace with the new route request info. and cascade propagate to upper portal when
          TTL trigger.}
   Else If (route request info (WSC_port) not identical){
       ///(rule. 2-1-2-b)
       If (new HC value < RT's HC){
        //(rule. 2-1-2-b-I)
          replace with the new route request info. and cascade call back to lower hierarchy portal
          to disable the route with the same DPID when its WSC_port <> null }
       Else {
        //(rule. 2-1-2-b-II)
          Local portal do nothing then cascade call back to lower hierarchy portal to disable the
          route with the same DPID when its WSC_port <> null }

 }
Else {
// (rule.2-2) : ASF == delete
   If (route request info does not exist in RT){
    //(rule. 2-2-1)
    Local portal do nothing }
   Else {
    //(rule. 2-2-2)
     Find and delete the record in RT, and upper cascade propagate delete event registry }
}
Recursive()
```

Fig. 9. Cache WS ripple patch routing algorithms

The following; Figure 10; is a demonstration for using above algorithm to adjust the P1 portal routing table. Figure 11,12,13 is a demonstration for enterprise portal add/change/delete its position issues.

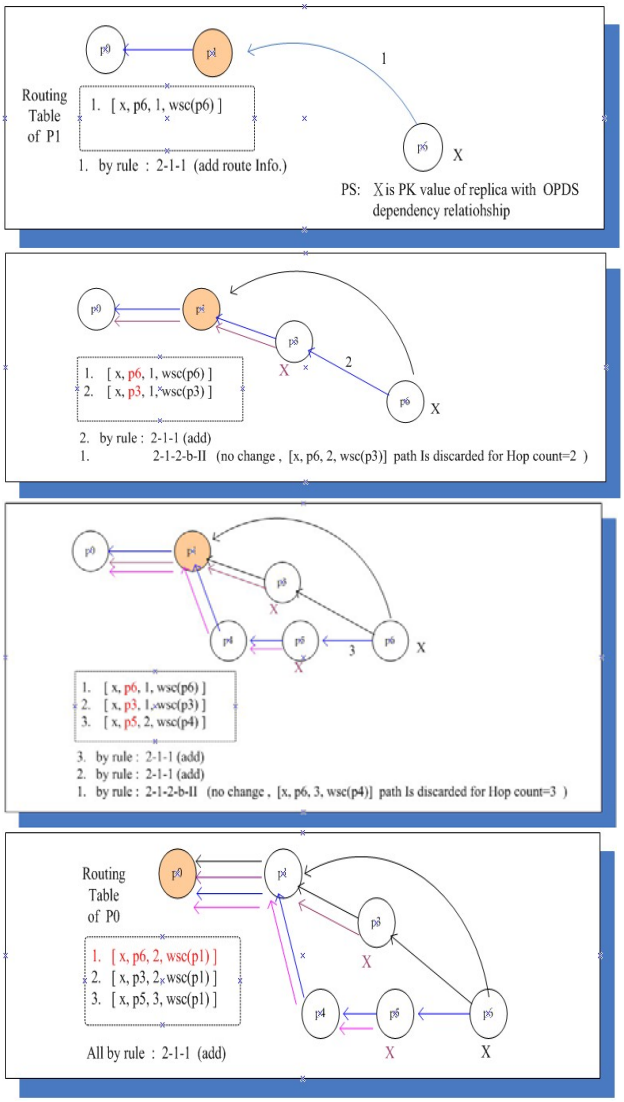

Fig. 10. Demonstration for routing algorithm

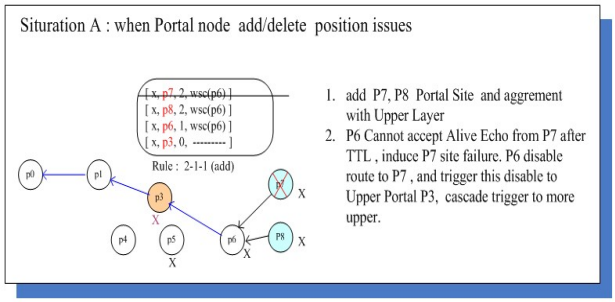

Fig. 11. Demonstration when portal node add/delete its position issues

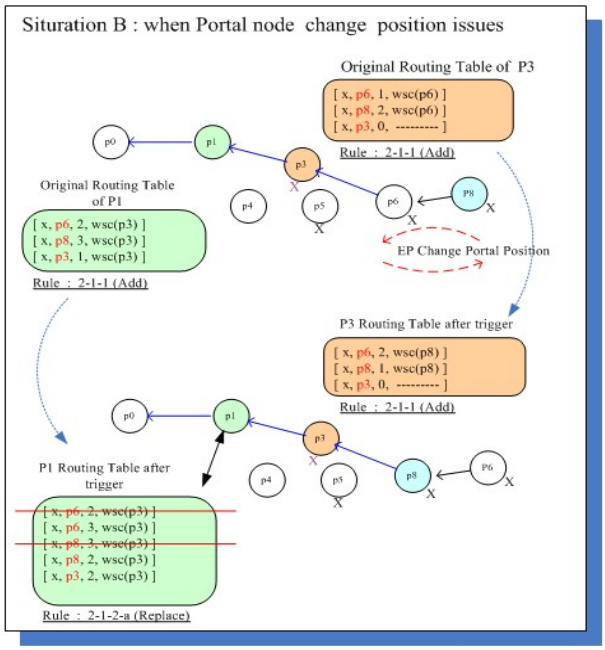

Fig. 12. Demonstration when portal node change its position issues

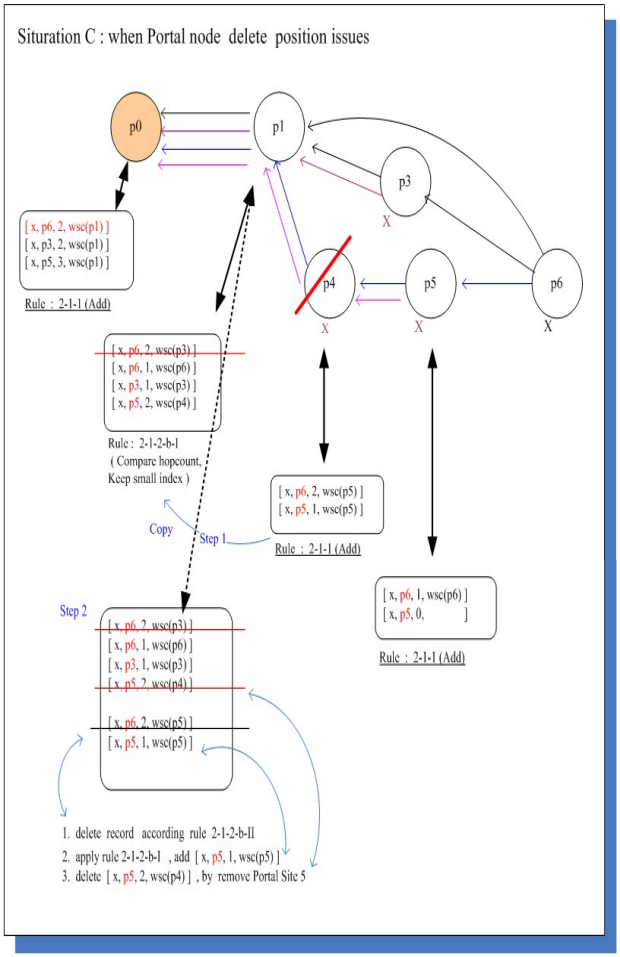

Fig. 13. Demonstration when portal node delete its position issues

6.3 Estimate of routing algorithms for distributed consistency

In this section we estimate three kinds of distributed consistency schemas applied in hierarchical and distributed storage environment like schools. . 1. P2P push based eager replica updating. 2. Proxy pull based lazy replica updating. 3. ODS push based and active web service propagate routing algorithm for global consistency. Simulation environment estimate experimental setups are discussed as follow.

6.3.1 Simulation goal

All administrative information on the hard disk containing the ODS copys of partial dependency set. Through a set of comprehensive, proactive and real-time update mechanism can achieve consistency of the global convergence goals. So that all the schools' decentralized system, will no longer have access to inconsistent or outdated information.

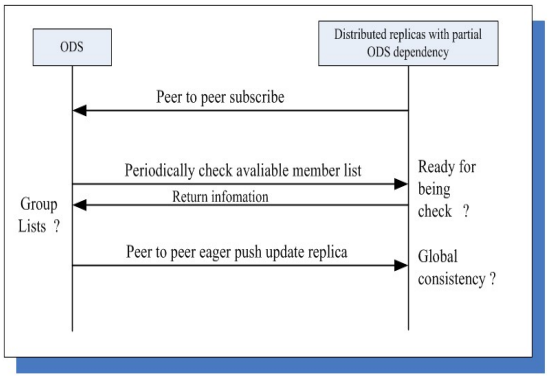

Fig. 14. P2P push based eager replica updating

6.3.2 The scope of simulation

The scope of simulation is based on Ling Tung university campus administrative group, access the information on disk which was part of dependency with Personnel part ODS copy. At present, for improving the application performance by the latest copy, accuracy and timeliness toward 2 directions. Including；1. Cache of time and space to improve regional exchange of information on the frequency problem. 2. Accuracy and real-time upgrade distributed dependency copy.

The following will discuss 3 kinds' global consistency models for evaluating and comparing the effectiveness of its operations.

(a) P2P push based eager replica updating

The uses of P2P subscribe/publish eager push mode send updated copy of the personnel changes to subscribed agency. The advantage is immediately, one-way dissemination to all registered administrative units. The shortcomings is the Personnel must to set up individual Peer to Peer's relations subscribe lists. Personnel must also be effective at any time to verify the transmission list of groups available. How to ensure to meet the goal of global convergence consistency is the most difficult. The operation is shown in Figure 14.

(b) Proxy pull based lazy replica updating

Personnel sent the staff-to-date update copy to school proxy. If all administrative units of application required the ODS dependency copy must through pull and data integration mechanism way. The advantage of lazy changes is to reduce non-essential message conversion and the time to send a message. The disadvantage that proxy server is a single point of failure risks. That wills not real-time changes all the dependency replicas on the copy of the application easy access to information inconsistent results. The operation is shown in Figure 15.

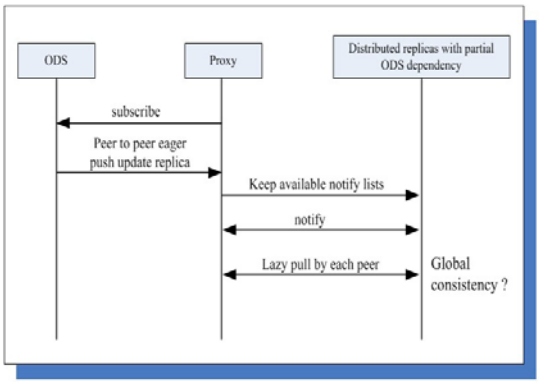

Fig. 15. Proxy pull based lazy replica updating

(c) *ODS push based and active web service propagate routing algorithm*

The goal of routing algorithm proposed is using Subscribe/Publish and active XML web service propagates routing Mechanism for global consistency. In a comprehensive, active and hierarchical automation ripple propagation way to reach global consistency of dependency information copy in school. All administrative units' application is able to integrate information at the lowest cost. Advantage: 1. No longer based on individual applications repeat investment in different database middleware system. 2. There is no need to periodically pull to obtain a copy of dependency-to-date information. For the shortcomings: 1. Root portal and business units to be implemented with replicas routing algorithm. 2. Enterprise portals have to implement ODS update replica schema transformation. The operation is shown in Figure 16.

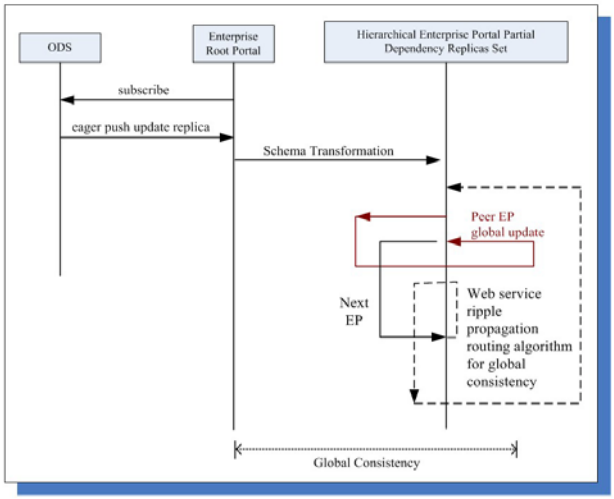

Fig. 16. ODS push based and active web service propagate routing algorithm

Assume in measure the global consistency convergence time is ignored the following metric including processing delay, queuing delay, transmission delay. There are considered the storage replica updating delay and network propagation delay. In practically, then transmission delay is usually between several μs. and ms.

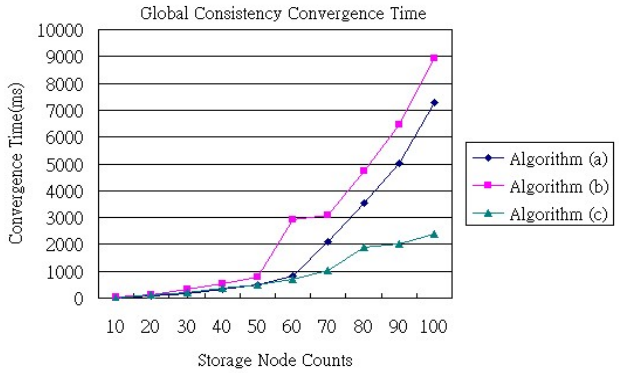

Fig. 17. Comparison of different algorithms on global consistency convergence time

7 Performance Analysis

In this section we describe our experimental results and use cache object-oriented layouts to improve the performance of one 3D visualization − virtual power plant. Moreover, we used different measure metrics for comparing the performance under different circumstances.

7.1 Experimental setups

In this section, the effectiveness of the proposed clustering algorithm is investigated. All algorithms were implemented in Java. The experiments were run on a PC with a Pentium 4 1.60GHz and 512MB megabytes main memory, running Microsoft Windows 2000 server. The *Total Objects per View, Response Time (in ms), Cluster Cohesion, Intra-Cluster Similarit, Number of Retrieval Files per View-radius* and *Response Time per View-radius* were used as measure metric. In order to compare with traditional tree-like data structures [6], the mechanism of *Access Path Predictor* (APP) was implemented for performance comparisons. The detailed experimental results were shown in the following diagrams.

In the meanwhile, we select the different support threshold for comparison. Figure 18 and 19 show the results. Algorithms with clustering outperform other algorithms without clustering. Since the clustering mechanisms can accurately support prefetching objects for future usage. Not only the access time is cut down but also the I/O efficiency is improved. Note that HG_clustering represents the Hypergraph clustering scheme.

In summary, we can determinate that our HG_clustering algorithm is better overall at cluster cohesion and inter-cluster similarity. This means that our HG_clustering algorithm can groups more similar patterns together and do more improvements on the efficiency of storage systems.

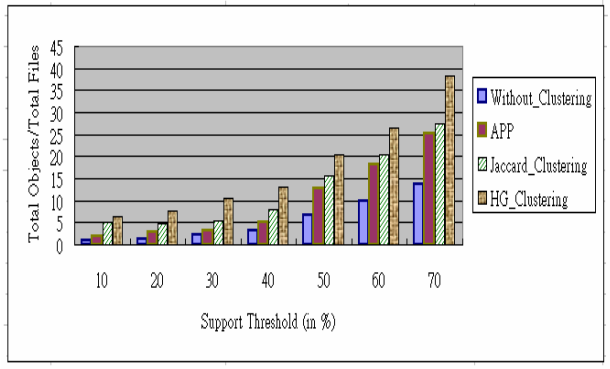

Fig. 18. Comparison of different algorithms on the number of objects retrieved under the same view_radius.

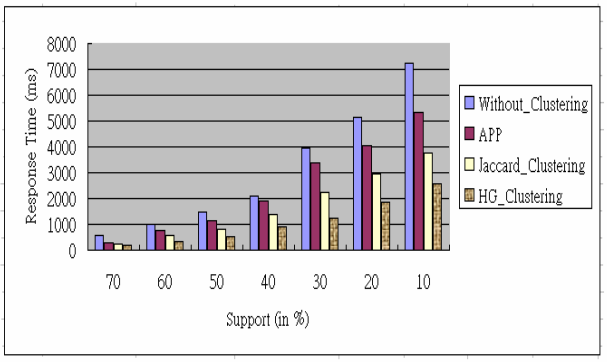

Fig. 19. Comparison of different algorithms on system response time under the same view_radius.

8 Conclusions and Future Work

This paper presents a hypergraph based clustering technique for accesses to 3D pattern clusters. In other words, the underlying premise of our approach is that in the case of cluster accesses, the next objects requested by users of the VE are typically based on the current and previous objects requested. Furthermore, if the requested objects have a lot of links to some "important" object, that object has a higher probability of being the next one requested. An experimental evaluation of the clustering mechanism is presented using real VE traces. The results show the hypergraph based scheme does better random prefetching for clustered accesses, with access reduction of 28 % in almost cases.

This work represents a novel application of association rule discovery techniques for identifying previously hidden knowledge in the fields of 3D visualization. A full automatic strategy for discovering new knowledge about traversal sequences has been developed based on identification of association rules between object correlations.

The paper also propose a service routable consistency framework (GDHCCM) that is capable of supporting ripple propagate cache updating copies that existed in distributed heterogeneity of storage. Besides, an individual object might induce different dependency relations in different applications and partially dependency to several distributed original patch data source. By using a new scalable web service routing policies that can dynamic reconfiguration distributed forwarding cache patch data path in hierarchical enterprises storage and improving cache data real-time correctness and hit ratio.

Our approach have the following different characteristics: (1).Scheme_based: traditional schemes always utilize the *tree_based data structure with locality* (either parent-child locality (i.e., temporal locality) or spatial locality); (2).Complexity_based: previous schemes manage these data structures to fit the memory constraints without considering the real demand for object; (3).Semantic_based: the semantic property seems very weak compared with ours; (4).Knowledge_based: the knowledge of previous schemes only hold for very near future. We are currently building a distributed VR system. This work is also investigating the appropriate depth and breadth thresholds for cluster identification. We are investigating the type of VR models that can benefit from the hypergraph based approach and its variations.

References:

[1] S. Chakrabarti, Mining the Web Discovering Knowledge from Hypertext Data, *Morgan Kaufmann Publishing*, 2003.

[2] G-J. Nam, S. Reda, C.J. Alpert, P.G. Villarrubia, A.B. Kahang, A Fast Hierarchical Duadratic Placement Algorithm*., IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.25, No.4, 2006, pp. 678-691.

[3] E. Ohbuchi, A Ral-Time Defraction Venderer for volume objects using a polygon-rendering scheme, *Proceeding of Computer Graphics*, 2003, pp. 190-195.

[4] J. Comg and S.K.Lim., Edge Separability-based Circuit clustering With Application to Multilevel Circuit Partitioning., *IEEE Transaction Computer-Aided Design Integrated Circuits and Systems*, Vol. 24, No.3, 2004, pp.346-357.

[5] S-E Yoon, P. Lindstrom, V. Pascucci, D. Manocha.,: Cache-Oblivious Mesh Layouts., *ACM Transactions on Graphics*, Vol. 24, No.3, 2005.

[6] D. Chisnall, M. C. Risualization. *IEEE-VGTC Symposium on Visualization*, 2006.

[7] E-H. Han, G. Karypis, V. Kumar and B. Mobasher., Clustering based on association rule hypergraph. ,*Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[8] P. Jaccard. ,The distribution of the flora of the alpine zone. ,*New Phytologist*, 1912, pp. 37-50.

[9] M. Sivathanu, V. Prabhakaran, F. Popovici, T.E. Denehy, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau., Semantically-smart disk systems*., Proceedings of the Second USENIX Conference on File and Storage Technologies*, 2003.

[10] W. T. Correa, J. T. Klosowaki, and C. T. Silva., Visibility-based prefetching for interactive out-of-core rendering., *IEEE Symposium on Parallel and Large-Data Visualization and Graphics (PVG'03)*, 2003, pp.2-8.

[11] S.E. Yoon, D. Manocha., Cache-efficient layouts of bounding volume hierarchies. , *Eurographics*, Vol.25, No.3, 2006.

[12] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton. , Batched multi triangulation. , *IEEE Visualization*, 2005, pp. 207-214.

[13] C.V. Apte, S.J. hong, R. Nataraian, E.P.D. Pednault, F.A. Tipu, and S.M. Weiss., Data-intensive analytics for predictive modeling*., IBM Journal of Research and Development*, Vol.47, No.1, 2003, pp. 17-23.

[14] Tesegay, Y., Yurpin, A., and Zobel, Justine,: Weiss., Data-intensive Analytics for Predictive

Modeling., *IBM Journal of Research and Development*, Vol.47, No.1, January, 2003, pp. 17-23.

[15] B. Hu and M.M. Sadowaks., Fine Granularity Clustering-based Placement., *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.23, No.4, 2004, pp. 527-536.

[16] S. Sun and X. Zhou., Semantic Caching for Web-based Spatial Applications., *In:APWeb 2005, LNCS 3399*, 2005, pp. 783-794.

[17] S.S. Hung, and D.S.M. Liu., Using Predictive Prefetching to Improve Interactive Walkthrough Latency., *Computer Animation and Virtual Worlds Journal*, Vol.17, No.3, 2006, pp. 469-478.

[18] R. Lario, R. Pajarola, F. Tirado., Cached Geometry Manager for Xiew-dependent LOD rendering., *Proceedings International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2005, pp.9-16.

[19] C. Zhang, C. Ding, Y. Zhong, Y. Wu., A hierarchical model of data locality., *In: Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Charleston, 2006.

[20] Zhu, Y. , Uniform Remeshing with an Adaptive Domain: A New Scheme for View-Gependent level-of-detail rendering of meshes., *IEEE Transactions on Visualization and Computer Graphics*, Vol.11, No.3, May-June, 2005, pp. 306-316.

[21] Ding, X.., Jiang, S., Chen, F., Davis K., and Zhang, Z.: DiskSeen, Exploiting Disk Layout and Access History to Enhance I/O Prefetch., *In: Proceedings of the 2007 USENI Annual Technical Conference, (USENIX'07)*, Santa Clara, California, June 17-22, 2007.

[22] Chen, S., Shen, B., Wee S., and Zhang, X.: SProxy, A Caching Infrastructure to Support Internet Streaming., *In. IEEE Transaction on Multimedia*, Vol. 9, No. 5, August 2007, pp. 1062-1072.

[23] R. Ladin, B. Liskov, L. Shrira, and S. Ghemawat., Providing high availability using lazy replication., *ACM Transactions on Computer Systems*, Vol.10, No.4, 1992.

[24] M. Patiño-Martinez, R. Jimenez-Peris, B. Kemme, G. Alonso., Consistent Database Replication at the Middleware Level"., *ACM Transactions on Computer Systems (TOCS)*., 2003.

[25] Thanasis Loukopoulos, Ishfaq Ahmad, Static and Adaptive Data Replication Algorithms for Fast Information Access in Large Distributed Systems, *20th IEEE International Conference on Distributed Computing Systems (ICDCS'00)*, 2000, p. 385.

[26] Yung Bok Kim and Soon Woo Lee, Performance Evaluation of Mobile Agents for Knowledge-Based Web Information Services KES-AMSTA, *Springer Lecture Notes in Computer Science*, Vol. 4496, 2007, pp. 209-218.

[27] Ruey-Kei Chiu and Kuo-Chin Tsai and Chi-Ming Chang and S. C. Lenny Koh and Kuan-Chih Lin, The implementation of an agile information delivery system in building service-oriented e-healthcare network, *International Enterprise Network Management Inderscience Publishers*, Vol. 1, ,March 14 2007., pp. 283-298.

[28] Ákos Hajnal and David Isern, Knowledge Driven Architecture for Home Care CEEMAS, *Springer Lecture Notes in Computer Science*, Vol. 4696, 2007, pp. 173-182.

[29] Aurora Vizcaíno and Juan Pablo Soto and Javier Portillo-Rodríguez and Mario Piattini, A Multi-agent Model to Develop Knowledge Management Systems, *IEEE Computer Society*., HICSS, 2007, p. 203.

[30] Mahmoud Brahimi and Mahmoud Boufaïda and Lionel Seinturier, Integrating Web Services within Cooperative Multi Agent Architecture, *IEEE Computer Society*., AICT/ICIW, 2006, p. 197.

[31] Nicolas Singer and Jean-Marie Pecatte and Sylvie Trouilhet, Combining Web Services and Multi-Agent Technology to Increase Web Cooperative Capacities, *IEEE International Conference on Internet and Web Applications and Services (ICIW'07)*, Mauritius, 2007.

[32] Oh Byung Kwon, Multi-agent system approach to context-aware coordinated web services under general market mechanism, *Decision Support Systems*, Vol.41, No.2, 2006, pp. 380-399.