DS_Grid: A Database Oriented Data Grid Supporting Dynamic Data Integration in a Domain[1]

Derong Shen, GeYu, Guangqi Wang, Meifang Li

Dept. of Computer
Northeastern University
Shenyang, 110004, China
shenderong@ise.neu.edu.cn
yuge@ise.neu.edu.cn

Abstract: Database oriented data grid becomes a new demand with the development of grid technology, and DS_Grid (DatabaSe oriented Data Grid) is such a grid system supporting dynamic data integration with the idea of service. In DS_Grid, MultiChord is adopted as the grid framework implementing the distributed storage, processing and data integration of data service resources and the domain and its corresponding data resources schema are specified according to text similarity definition to locate the peers data service resources registered, so as to improve the cover range of data service resources discovered, multi-root and multi-peer data resources replica management mechanism to improve the efficiency of discovering data service resources and the reliability and availability of DS_Grid, filter distributed data integration strategy is applied to reduce the traffic cost and the data integration cost, and distributed clustering analysis technology is realized to summarize the huge data information. The experiments have demonstrated the availability of the key technologies adopted in DS_Grid.

1 Introduction

Data Grids[SR05] primarily provides secure and high-performance services and infrastructure for distributed data-intensive science applications to transfer large datasets and a scalable replication on demands, in which the massive amounts of raw data stored in files are dealt with distributed file systems, and performance and replication are usually focused on. The significant efforts are GridFTP[Gri00], SRB[BM98], as well as the European DataGrid (EDG)[GS01,WJ01] etc., which aimed at data in file and standard operation of distributed file system without resolving the problem of integrating database to grid.

Today, the concept of data grid is shifting from file centric one towards that with data stored in databases as data resources[JG05], since thousands of public databases already existed, for example, in areas like biology science, space and e_business. It is obvious

[1] This research is supported by the National High-Tech Development Program (2003AA414210), the National Science Foundation (60573090).

that database integration into the grid will play an important role for many applications such as scientific researches and e_business applications, since these applications are heavily dependent on databases and these typical data intensive grid application processing and storing the data by using flat files cannot benefit from the power that database systems offer[Wa02]. So the need to integrate databases and database technology into the grid has already been recognized in order to support science and business applications as well as to manage metadata, provenance data, resource inventories etc.[Wa02,Ma05], for which significant effort has been contributed into defining requirements, protocols and implementing middleware to access databases in grid environments, such as DAIS-WG[DA03], OGSA-DAI[OG06] and OGSA-DQP[OG06a], while related efforts are GridDB[DM04,DM04a], Oracle 10g[Or03], GDIS[CD05], POQSEC[RT05], Polar*[JA03], CoDIMS-G[FV05] and PALADIN[JG05]. All those efforts above have provided guide for database integration in grid environment, but few studies on supporting dynamic feature of grid environment and the same for those known database-oriented grid system aiming at a special application. Though database-oriented processing technologies under grid environment share many consistent aspects with multi-databases, parallel databases and distributed database, there are still many different aspects due to the uncertainties of grid environment: (1) the uncertainty of data resources and what resources satisfying the user's demand; (2) the size of dataset satisfying the user's demand known only after finishing the query; (3) unable to execute static optimizing processing according to meta-information as the static database query optimization does; (4) unable to anticipate the homogenization of heterogeneous data resources and thus unable to specify the mapping rules etc. Therefore, the uncertainties existed under the grid environment brings about many difficulties such as in query processing, optimization and transaction scheduling of data, and also presents challenges for those data processing researches.

In this paper, we discuss a data grid supported by P2P framework, namely, DS_Grid with database as the primary resources, focusing on effectively integrating the distributed data into grid environment to realize the share of database resources and eliminate resource islands. The main contributions of this paper are: 1) effectively support resource storage, query processing and data integration based on MultiChord system architecture; 2) improve the precision and recall of obtaining resources based on loose service discovery by means of similar domain and data schema matching, and realize the intelligent query processing mechanism based on domain ontology knowledge; 3) reduce data traffic cost based on filter distributed data integration strategy and thus effectively improves the efficiency of query processing; 4) improve the reliability and performance of the grid system through the replica management mechanism of multi-root and multi-peer maintenance; 5) improve the data analysis processing efficiency of huge amount of data by distributed clustering analysis strategy.

2 Overview of DS_Grid

Data grid with databases as primary data resources is proposed for realizing the effective sharing of database resources to provide additional services on user's demand. According to the dynamic, self-adaptive and high efficient processing features of grid,

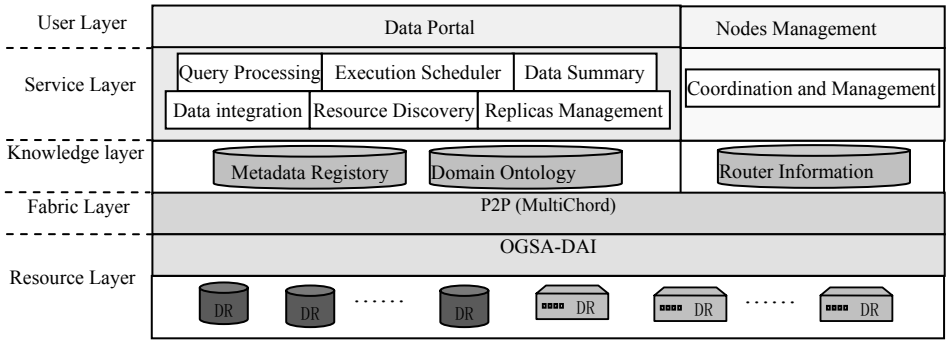

Fig.1 DS_Grid Architecture

DS_Grid mainly focuses on the following prototypes: 1) adopt the service-oriented idea, wrap the data resources as Grid service; 2) realize service resource management based on P2P architecture, fully make use of the distributed resources within grid and improve the efficiency of data processing; 3) define the global data schema based on domain ontology and effectively implement the heterogeneous data integration; 4) specify query and data integration strategy considering the dynamic, autonomous and heterogeneous features of grid resources; 5) guarantee the reliability of grid and the efficiency of locating data service resources by means of replica management strategy; 6) realize the information visualization of huge data amounts based on data mining technology. DS_Grid system can be applied to any domain for supporting effective data integration within it, such as in dynamic allied enterprises. The architecture of DS_Grid is shown as Fig. 1, in which database resources are wrapped as grid services based on OGSA-DAI, and actually, any kind of data resources as grid services are applied; JXTA is used to construct MultiChord framework; Shunsaku XMLManager is adopted as service resource repository. The functions of main service components are as follows.

Resource Coordination and Management maintain the availability and integrity of P2P MultiChord structure and the service resource meta information as grid services joining in and quitting of peers.

Query Processing extends global query semantics based on domain ontology knowledge, decomposes it into global sub-query schema, rewrites query based on the discovered service resources, and generates the executing plan composed of multiple sub-queries.

Service discovery takes the global sub-query schema information as a unit, implements domain and schema matching based on domain ontology knowledge to discover suitable services.

Scheduler schedules the corresponding executing site where each sub-query of the whole query executing plan is, and globally coordinates the execution of each sub-query.

Data Integration optimizes the data integration processing of a sub-query according to FDDI strategy (See Section 6).

Replica Management Multi-replicas exists according to the principle of multi-root and multi-peer replication management to guarantee the integrity and effectiveness of the metadata of service resources, and to further enhance the efficiency of discovering grid data services.

Grid Portal submits query request and presents the integrated results to the user.

3 MultiChord Architecture

Chord algorithm[SG01] has drawn many attentions for its simplicity and validity. However, as a data query request may contain multiple schemas within a data grid, we probably have to execute many times to locate these service resources if we adopt Chord algorithm, which results in low efficiency and unable to distinguish domain information as well. Therefore, according to the idea of managing the service resources based on domain for reducing the time cost of discovering service resources, MultiChord[PM03] is proposed to improve the query efficiency by means of decreasing the total number of hops in a query, and thus lessen the system overload.

MultiChord architecture divides the query process into two steps. First, it locates the domain information, reduces the searching scope of schema mapping for next step and acquires the schema information within the cut down scope. Detailed query process is shown as Fig. 2. Given a MultiChord with 20 peers, the notation of each peer composed of 5 bits and all peers compose the big chord while peers with the same first two bits of the peer notation compose the small chord, which results in two chords, and each peer has a preceding and following finger table to locate its neighbors.

Given N peers, each with K bits of the domain keyword notation, and 2^k small chords, assume one query request needs S schemas, then the average hop is $S*log_2N$ in Chord algorithm while $log_2N+S*log_2(N/2^k)$ in MultiChord algorithm. When $S>(log_2N/k)$, the efficiency of MultiChord is higher than Chord if N remains unchanged.

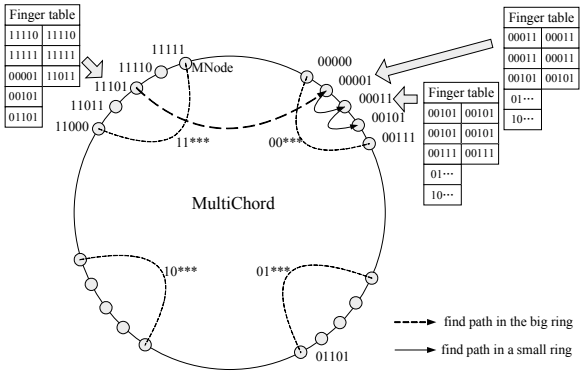

Fig. 2 Structure sketch map of MultiChord

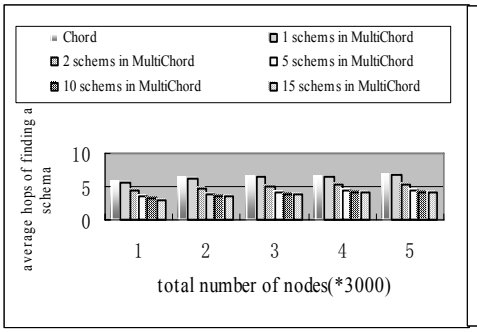

Fig.3 Average hops of querying 1 schema in a small chord with 50 nodes

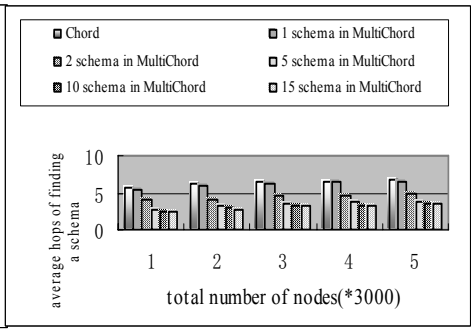

Fig.4 Average hops of querying 1 schema in a small chord with 150 nodes

We have carried out emulation tests on MultiChord and Chord respectively. X-axis is the whole peer number of the system, and y-axis is the average needed number of hops in finding a schema, we have tested changes in hops in small chords with 50 and 150 nodes respectively, shown as Fig. 3 and Fig. 4. It is better to adopt MultiChord than Chord if a query contains multiple schemas since the number of small chords has little influence on the average hops needed in finding every schema increment.

Since the discovery process of data service resources in DS_Grid often involves many schemas as well as many attributes, that is, all resources discovered are realized based on multi-element hashing, and thus it is favorable to adopt MultiChord.

4 Intelligent Query Processing Mechanism Based on Ontology

To meet the user's demand and provide valuable integration information, we specify the database within a domain as service resources, define global schema based on domain ontology so as to acquire high quality data service resources on user's demand.

4.1 Global Ontology Schema Definition

Ontology is a formulated specification on the shared conceptual schema, aiming at capturing the related domain knowledge and providing a common understanding on it. The paper defines the global schema based on ontology[Us98,ES99], carries out the heterogeneous transformation from XML heterogeneous data resources to the mapping of global schema, and thus realizes ontology based intelligent query.

The paper defines the ontology of global schema as a 4-gram tuple O= (C, R, A, r), where C is a conceptual set, R is the bidirectional 2-gram relation between conceptions, A is a set of attributes and r is semantic relation set between conceptions. Ontology, as a domain specification, is used to describe conceptions within that domain and their relations[BB04, MF01]. The semantic relation rules of data schema are defined as follows: 1) Equivalent class relation. Two classes can be declared as equivalent, if they refer to the same conception. 2) Inherited class relation. Class A inherits all the attributes

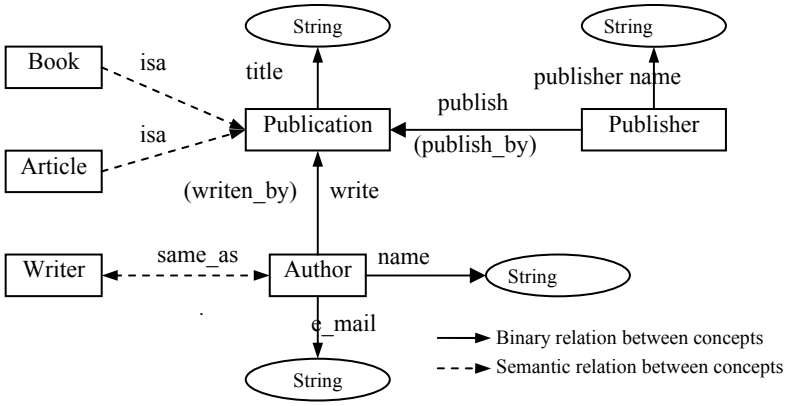

Fig.5 Publication domain ontology

of class *B* if class *A* is the sub-class of class *B*. 3) Inverted relation. Given P_1 is declared as the inverted attribute of P_2, if *X* is associated to *Y* by P_2, then *Y* is associated to *X* by P_1. 4) Transitive relation. Given (*x*,*y*) is an instance of transitive attribute *P*, if (*y*,*z*) is also an instance of transitive attribute *P*, then (*x*,*z*) is an instance of transitive attribute *P*. 5) Symmetrical relation. The relations can be declared symmetrical, provided that if (*x*,*y*) is an instance of symmetric attribute *P*, then (*y*,*x*) is also its instance.

Fig. 5 represents the ontology of the publication domain, words within the bracket represent the name of inverted relation. When submitting a query within a publication domain, one can adopt it as global schema, deduct it by means of the above semantic rules and implement semantics extension based on domain ontology knowledge.

4.2 Intelligent Query Processing Based on Global Ontology Schema

We should receive the query request based on the global schema and execute semantic processing of the query request facilitated by domain ontology knowledge. For instance, in Fig 5, Author and Writer have the same conception since Writer shares the whole features of Author. Book and Article is the sub-class of Publication, and thus queries on Publication can be automatically extended to all the sub-classes according to the semantic information. For example, searching books belong to computer category, then books on database and network should also be found. Steps of ontology based intelligent query processing are executed as follows.

(1) Receive the user submitted query based on conception (global schema), and generate a set of all associated conceptions according to the semantic relation between conceptions of ontology, that is $C=\{c_1, c_2, \ldots\}$.
(2) Standardize and generalize all the conceptions in *C* according to their semantic relations in *r*, such as equivalent conceptions, inherited relation, including and included relation etc., and extend the query conditions.
(3) Establish a 2-gram relation set R according to their 2-gram relations for all the conceptions in *C*, that is $R=\{r_1, r_2, \ldots\}$.
(4) Infer from the specified semantic rules for all relations in *R*, such as inverted,

symmetrical and transitive relation, to obtain a deeper semantic relation.

(5) Generate rewritten query set after the semantic extension by means of the above four steps.
(6) Take a rewritten query as a unit to discover corresponding service resources and decompose it into a set of multiple sub-query schemas.
(7) Rewrite sub-queries based on the mapping information between ontology schema and the XML resources schema discovered and generate query execution plan.

5 Service Discovery

Service discovery is the guarantee of finding grid data services in a data grid. According to MultiChord structure, service metadata of the same domain is saved in the same small chord, where every peer maintains the service metadata containing different schemas. Service discovery can be divided into three steps. First, find the corresponding small chord based on domain information; secondly, find the corresponding peers of schema information in a small chord and thirdly, match the schema information on peers, and find suitable service resources.

Domain Matching. Let $DO=\{(D_1, DD_1), (D_2, DD_2),\ldots, (D_n, DD_n)\}$ be the information set of domain description, DD_i be the domain description information, D_i be the corresponding keyword, DR be the query request description information, Dr be domain keyword matched, then $Dr=\{D_i||DR,DD_i|=max\{|DR, D_1|,|DR, D_2|,...,|DR, D_n|\}$, where $|DR,DD_i|$ is the similarity distance[SY06].

Schema Matching. Schema description information adopts XML schema description, executes the equivalent semantic extension of elements at first, and then defines the schema matching degree based on the string matching by making use of edit distance, and thus locates the peer of the service resources stored. The idea of realizing replicated publishing based on loose-peer-location is adopted, and detailed description is as follows.

Let $SO=\{S_1, S_2,..., S_n\}$, $S_i=\{s_{i1}, s_{i2},..., s_{in}\}$, $SR=\{sr_1, sr_2,..., sr_m\}$, where SO represents the ontology schema set defined by domain experts, S_i is an ontology schema, such as books, and SR is the query request schema. Compute the matching degree of schema name($\sigma(S_i, SR)$) and the matching matrix $M(S_i,SR)$ between S_i and SR according to string and structure matching[NO03], then let MS be the matching schema set, $MS=\{S_i \mid \sigma(S_i, SR) >= \sigma_s \exists m_{ij}(m_{ij} \in M(S_i, SR) \wedge m_{ij} \geq \sigma_a)\}$, where (σ_s, σ_a) are the matching pair threshold given, represented as (σ_s, σ_a), specified (0.5,0.8) or (0,0.9) in DS_Grid, thus the peers of service resources are located based on the threshold (σ_s, σ_a).

6 Filter Distributed Data Integration Strategy (FDDI)

As there are huge dynamic and autonomous data resources and even huge redundant

information in grid environment, the discovered data resources may have a mass of redundant information. If they are unfiltered to the initial peer for data integration, it will transport much redundant information, which may result in bandwidth waste and increase the execution time of the task. Therefore, we propose a strategy of filtering the replicated data based on keywords, namely, Filter Distributed Data Integration strategy based on keywords. The basic idea[FK02,PR99] is to extract keywords of the results and to transport them to the initial peer rather than transport the XML data results directly between peers, and after the filtering of the initial peer, retrieve the keywords of needed data to the peer, that is, decrease the transported data amount and time by means of transporting data on demands. Detailed description is as follows.

Suppose there are n peers participating in one task, the XML data result of each peer is D_i, the keyword set is K_i, the keywords of initiating peer is K_1, then the keywords of result set is $K = K_1 \cup K_2 \cup \ldots\ldots \cup K_n$, and K_i' is the keywords needed for retrieving the XML data from the *ith* peer, $K_i' = \left(K - \bigcup_{j=1}^{i-1} K_i\right) \cap K_i$, $i=2,3,\ldots,n$, where the sequence of K_i is ordered according to the retrieval order of the XML data results, with faster peers transmitting more data, which can greatly decrease the data amount transferred by the slower peers. So the data amount and its responding time of network transportation can all be decreased by means of filtering the overlapping keywords.

By experiment, we have made a comparison between FDDI strategy and other two integration strategies, respectively centralized data integration strategy (CDI) and distributed data integration strategy (DDI). CDI is to execute recalling and integrating of all data services of one task in one peer while DDI is to execute them in multi-peer and re-transport the calling result data to the initial peer for data integration. Fig.6, Fig.7 and Fig.8 respectively represent the needed executing time of one task under the redundancy of 0%, 10% and 20%. From the figures, evidently, FDDI outperforms the other two as the increasing of redundancy, namely, it effectively shortens the execution time through decreasing the large amount of data transportation.

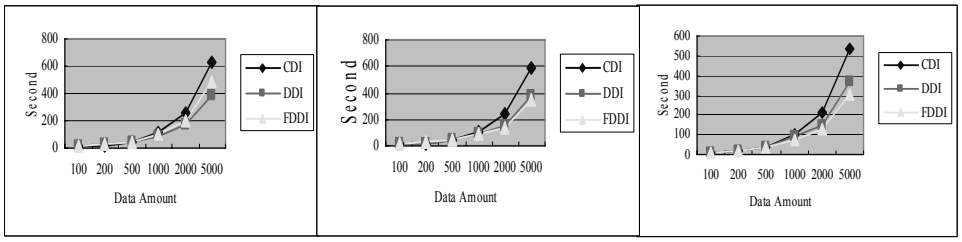

Fig. 6 Comparison with no redundancy

Fig. 7 Comparison with 10% redundancy

Fig. 8 Comparison with 20% redundancy

7 Multi-root and Multi-peer Replica Management Mechanism

Data replication[ZJ01] is an important approach aiming at improving the reliability and performance of the system. In structural P2P system, the multi-root[Ka01] approach is

presented to solve the single peer failure. DS_Grid adopts multi-root multi-peer replica strategy, the main idea of which is that every peer maintains $M(M >= 1)$ peers with the nearest distance to its own *guid* (peer *id*), namely M neighbor set, and then add its own indices onto those peers, by which means, one peer index will be maintained at most on $M+2$ peers because its own neighbors may not belong to the M neighbor set. If the original peer is still in the system, object locating and index publishing will be routed to that peer, and the primary functions of all peers remain the same only with the root peer timely maintain the M neighbor set and republish the peer index information. If the original root peer exits, the routing process will automatically locate to the new root as long as the new root belongs to the M neighbor set of the former one, and accomplish the whole process of object locating seamlessly. When a new peer maintains its M neighbors, it will disperse its index information to all peers in its neighbor set. Thereby, the system will ensure high efficiency of object locating in the dynamic network as long as it well manipulates the time period to maintain M neighbor set.

To acquire an appropriate number for multi-peer and multi-replica, we have carried out a series of experiments based on NS simulation network environment, and obtained the similar successful locating rate of single peer with respectively *2* and *4* multi-replica(M) (figures omitted). Hence, we choose $M=2$ and $M=4$ and test the co-influence of root peer and peers it maintains, shown as Fig.9, Fig.10, Fig.11 and Fig.12. The x-axis represents the frequency of replicas maintaining for neighbor republishing, with second as unit, while the y-axis is the rate for successful object locating. The figures demonstrate that multi-root and multi-peer maintenance has great advantages in improving successful locating rate, and it is more favorable when $M=4$, with little change in effect during the increasing of maintaining period. When there are *2* or *4* roots, the result curve is less

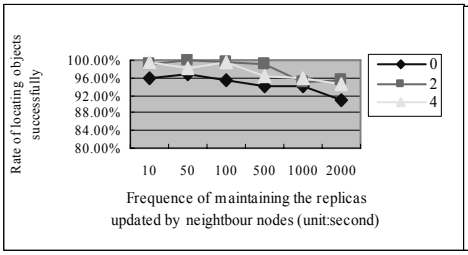

Fig. 9 Index of object republished per 100 sec, M=2

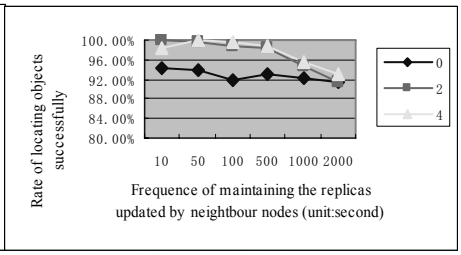

Fig. 10 Index of object republished per 1000 sec, M=2

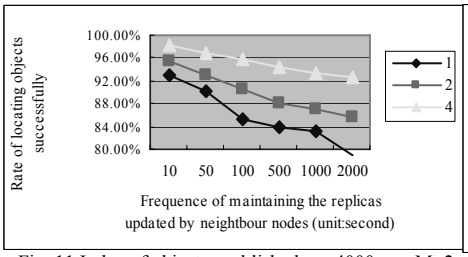

Fig. 11 Index of object republished per 4000 sec, M=2

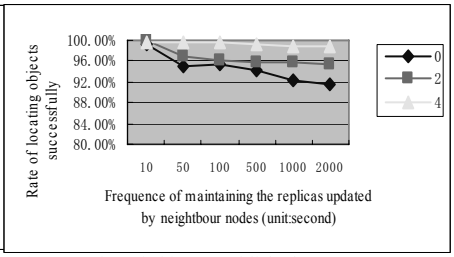

Fig. 12 Index of object republished per 4000 sec, M=4

affected by the change of republished time than that of in a single peer. While $M=4$, with *4* roots, the system maintains *16* data replicas, which, though improves the robustness and successful locating ratio, also results in higher cost for maintaining the consistency of replicas if there are huge replicas.

Fig.13 shows the average maintaining cost of different number of replicas and peers. The x-axis is the replication policy, for example, (*1, 0*) represents *1* root and *0* maintained peer; y-axis is the cost. Provided P is the cost for each time data locating, t is the republished time for a neighbor peer, $N/2^k$ is the number of small chords, M is the number of replicas, and N_r is the number of roots, then the average cost in evaluation model is defined as $\Omega=\sum(C(t)*\sigma(t))$. In experiments, t=[*10*,*50*,*100*,*500*, *1000*,*2000*], with second as unit and N_r= [*1*,*2*,*4*], M=[*0*,*2*,*4*].

By simulation experiments, the proper replica strategy is with $N_r=4$ and $M=2$ for the cost curve is smooth, and the strategy with $N_r=4$ and $M=2$ is adopted in DS_Grid.

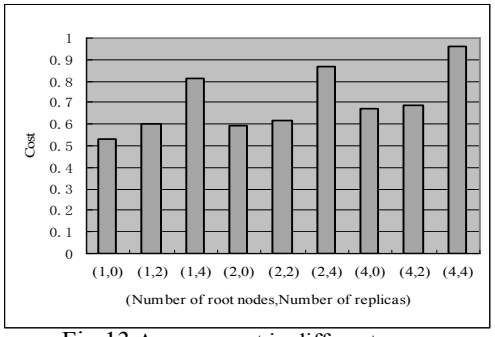

Fig.13 Average cost in different cases

8 Data Visualization Based on Distributed Clustering Analysis

DS_Grid adopts clustering analysis based on k-means with specialized attributes, and then presents the results to the user. Since clustering has high cost in data processing, and the performance of the system will fall sharply as the amount of data and number of concurrent users increase if we resort to centralized clustering analysis. Thereby, DS_Grid proposes distributed clustering analysis processing strategy[KP03,MP01], and divides the data processing into data synthesizing layer and analysis layer, the former of which carries out the data integration, guarantees the aggregated data meeting up with the user's demand, and then it implements 1-time and 2-time clustering analysis, and thus alleviates the centralized processing bottleneck problem and improves the efficiency of data processing in grid by making use of P2P distributed computing capability. As shown in Fig. 14, n-layer and n-1-layer implements 1-time clustering and 2-time clustering respectively, while multi-layer (1~n-2) in data synthesizing layer under n-1-layer carries out data integration and guarantees the integral schema of n-1-layer. Let $S(n)$ be the data schema of n-layer, then $S(n)= S(n-1)$, $S(i)=\overset{m}{\underset{l=1}{\infty}} S_l(i-1)$, that is, integrate the

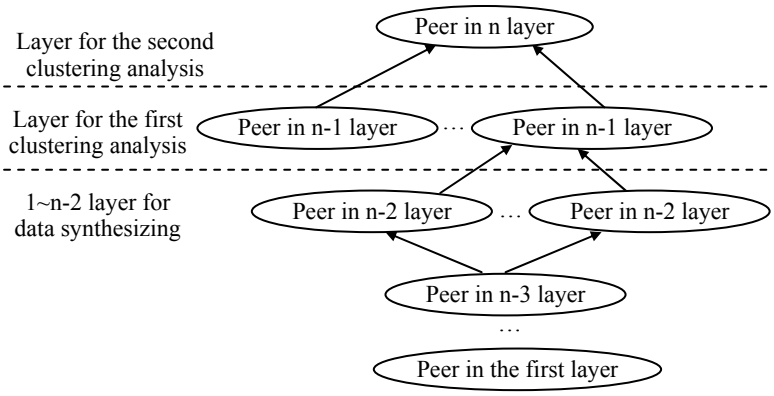

Fig. 14 Distributed clustering analysis structure

data under n-2-layer, and cluster when data result meets the schema demand at the higher layer, and finally acquire the final clustering analysis results to user. Detailed steps are as follows:

(1) Accomplish clustering according to traditional k-means algorithm.
(2) Traverse the whole 1-time clustering results and modulate the results by the maximum and minimum values of three dimensions. If two times 1-time clustering not belongs to the same 2-time clustering, for example, there is overlap in the values of the two clusterings, and thus the two clusterings can be modulated to one 2-time clustering.
(3) Abstract the result of 2-time clustering and extract the maximum and minimum values of each dimension.
(4) Take the record number of each 1-time clustering as weight, get the average weight of 3-dimension and acquire the final 2-time clustering average value.
(5) Take the sum of the records for each 1-time clustering as the record for 2-time clustering.

A simulation test is carried out under local network environment, in which we process clustering analysis with 10000 data and test the corresponding time, with the test results being shown as Fig. 15 and Fig. 16. The two figures show that distributed clustering analysis has advantages in time with the increasing of the record number of data set, which is more provable in internet environment where the transporting cost of data file is comparably more expensive.

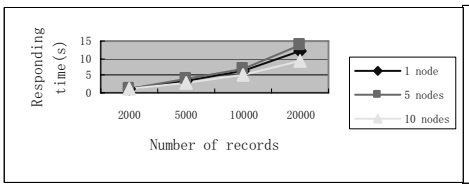

Fig.15 The comparison of the time of cluster analyzing （5 clusters）in different Peer number

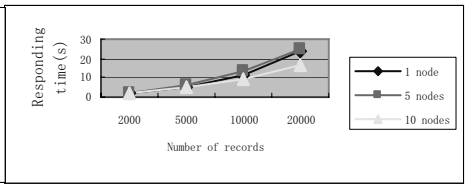

Fig.16 The comparison of the time of cluster analyzing （10 clusters）in different Peer number

9 Conclusions

The paper has introduced a database resources oriented data grid-DS_Grid, which adopts MultiChord as the grid framework and realizes the distributed storage, processing and data integration of data service resources. In DS_Grid, database resources are wrapped as grid services; domain ontology and ontology schema the data resources belong to is specified based on domain ontology knowledge by means of text similarity degree during service publishing and discovery, for locating the resources to the corresponding peers of a small chord, so as to discover suitable grid services rapidly; ontology as domain global schema during query processing, and heterogeneity resolution, semantic extending as well as query rewriting are implemented based on domain ontology knowledge and reasoning rules; sub-query decomposed by query rewritten according to the data resources discovered is scheduled to a peer which carries out the data integration of the sub-query by adopting filter distributed data integration strategy based on keywords at the data synthesizing layer, and finally carries out visualization of huge dataset based on the distributed data mining technology. DS_Grid is supported by national 863 project, developed by the Institute of Computer Software and Theory of Northeastern University in China.

Next, we will focus on data integration indices, self-adaptive query optimization based on multi-objects and the summary of huge data amount.

References

[BM98] Baru C., Moore R.. The sdsc storage resource broker. CASCON ’98, 1998.

[CD05] Carmela C., Domenico T. XML Data Integration in OGSA Grids. VLDB DMG 2005, LNCS 3836, pp. 4–15, 2005.

[DA03] DAIS-WG https://forge.gridforum.org/projects/dais-wg,2003.

[DM04] David T. Liu , Michael J. Franklin. GridDB: Data-Centric Services in Scientific Grids. Technical report, UC Berkeley, 2004,3, http://www.cs.berkeley.edu/.

[DM04a] David T. Liu, Michael J. GridDB: A Relational Interface to the Grid. In SIGMOD, 2003.

[ES99] Erdmann M., Studer R. Ontologies as Conceptual Models for XML Documents, In Proceedings of the 12th Workshop for Knowledge Acquisition, Modeling and Management (KAW'99), Canada, 1999.

[FK02] Foster I., Kesselman C.The Physiology of the Grid: An open Grid services architecture for distributed systems integration[J], http://www.globus.org/research/papers/ogsa.pdf, 2002.

[FV05] Fabio P., Vin′ıcius F.V. An Adaptive Distributed Query Processing Grid Service. VLDB DMG 2005, LNCS 3836, pp. 45–57, 2005.

[Gr03] Grid physics network (griphyn) white paper, 2003. http://www.griphyn.org/ documents/white_paper/.

[Gri00] Grid Forum Remote Data Access group. Gridftp: Ftp extensions for the Grid. 2000, Vol.6, 12-14.

[GS01] German C., Steve M. The DataGrid Architecture. Technical report, DataGrid-ATF-01,

2001,7.
[JA03] Jim S., Anastasios G. DISTRIBUTED QUERY PROCESSING ON THE GRID. The International Journal of High Performance Computing Applications,Volume 17, No. 4, Winter 2003, pp. 353–367
[JG05] J¨urgen, G¨ores. Towards Dynamic Information Integration. VLDB DMG 2005, LNCS 3836, pp. 16–29, 2005.
[Ka01] Kandogan E. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001.
[Ka01] Kandogan E. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001.
[KP03] Keim D.A., Panse C. Pushing the Limit in Visual Data Exploration: Techniques and Applications. Lecture in COMPUTER SCIENCE, 2003.
[Ma05] María A. Nieto-Santisteban1, Jim Gray2, Alexander S. When Database Systems Meet the Grid. In Proceedings of the 2005 CIDR Conference.
[MF01] Manolescu I., Florescu D. Answering XML queries over heterogeneous data sources [J], VLDB 2001.
[NO03] Ng W.S., B. Ooi C., Tan K. PeerDB: A P2P-based System for Distributed Data Sharing. In Proceedings of the 19th International Conference on Data Engineering (ICDE 2003),2003,633–644.
[OG06] OGSA-DAI http://www.ogsa-dai.org.uk/.2006.
[OG06a] OGSA-DQP http://www.ogsa-dai.org.uk/dqp/.
[Or03] Oracle Co. Oracle Database 10g: The Database for the Grid. An Oracle White Paper, 2003,11.
[PM03] Papazoglou, Mike P. Leveraging Web -Services and Peer-to-Peer Networks, Dublin Core Metadata, 2003,1.
[PR99] Plaxton C. G., Rajaraman R. Accessing nearby copies of replicated objects in a distributed environment. Theory of Computing Systems, 32:241-280, 1999.
[RT05] Ruslan F.,Tore R. Framework for Querying Distributed Objects Managed by a Grid Infrastructure. VLDB DMG 2005, LNCS 3836, pp. 58–70, 2005.
[SR01] Stoica, Robert M. Chord: A Scalable Peer-to-peer lookup service for internet applications[C], Technical Report, TR-819, 2001,3.
[SG03] Sandholm, T., Gawor, J.: Globus toolkit 3 core—A grid service container framework. Globus Toolkit Core White Paper (2003), http://www-unix.globus.org/.
[SR05] Srikumar V., Rajkumar B. A Taxonomy of Global Data Grids. Http://www.chinagrid.net/,2005,4.
[SY06] SHEN D., YU G., NIE T. An Effective Service Discovery Model for Highly Reliable Web Services Composition in a Specific Domain.APweb2006, LNCS 3841, pp.886-892.
[Us98] Uschold M.. Knowledge level modelling: concepts and terminology [J], The Knowledge Engineering Review, Vol. 13:1, 1998.
[Wa02] Watson P.. Databases and the Grid. e-Science Programmer Technical Report KeS-2002-01,
ttp://www.nesc.ac.uk/technical_papers/PaulWatsonDatabasesAndTheGrid.pdf
[WJ01] Wolfgang H, Javier J.M. Data Management Architecture Report. Design, Requirements and Evaluation Criteria. Technical report, DataGrid-02-D2.2, 2001,9.
[ZJ01] Zhao B.Y., John K. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report No. UCB/CSD-01-1141.
[ZJ01] Zhao B.Y., John K. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report No. UCB/CSD-01-1141.