A DISTRIBUTED COOPERATIVE TECHNOLOGY FOR SPATIAL GRID COMPUTING

Chenyu Li*, Kunqing Xie, Xiujun Ma, Cuo Cai, and Yanfeng Sun

Department of Intelligence Science, National Laboratory on Machine Perception, Peking University, Beijing, 100871, China

Email: shiningli@gmail.com, *{kunqing, maxj, ccai, sunyf}@cis.pku.edu.cn*

ABSTRACT

Grid computing for resources sharing and distributed computing has been researched widely in the past. As for distributed spatial datasets, the current centralized administrative scheme may become the system performance bottleneck. This paper presents a distributed cooperative grid computing technology to facilitate complex spatial applications by collaboration among distributed spatial resources. A hierarchical spatial index and communication protocol has been designed for the collaboration, which enables a dynamical choice for the best quality nodes for specified subtasks, synchronized execution, and compensation for a failure to execute a subtask. Also, we present an approach for dynamic resource allocation and distributed transaction mechanics to ensure consistency.

Keywords: Grid computing, Spatial application, Distributed spatial computing, Transaction mechanism

1 INTRODUCTION

Grid computing refers to a new form of distributed computing, massively distributed and highly volatile systems for sharing large amounts of resources, including data resources and computational resources (Wehrle, Miquel, & Tchounikine, 2005). Currently, among the systems that feature most prominently in the grid domain are user applications running on top (or at the edge) of the Internet allowing a large group of users to interact and share resources. Most popular are file or content sharing applications that achieve high availability of files and network capacity, such as DataGrid, Napster, and SETI@home (Koszek & Sandrasegaran, 2005). However, little has been done on collaboration a large number of autonomous computational nodes to perform complicated applications, which is important for users to make the most of distributed resources throughout the Internet (Alameh, 2003).

Deferring to the requirement of data collection and management, spatial data is always deposited in locationally distributed nodes, which are usually administrated under spatial databases. On the other hand, spatial applications are always computation-intensive and data-intensive and distributed spatial data manipulating is well organized and of high performance (Dyo & Mascolo, 2005).

Taking the specialties of spatial application into account, a grid-computing model is necessary for centralization in spatial data sharing and functional cooperation (Andrews, Sherwin, & Banister, 2003). First, a decentralized architecture not only eliminates the limitation of a single point of failure and scalability, but it also enhances the system's robustness and availability as nodes increase. Second, spatial data resources and computational resources are enriched when more nodes are participating, thus improving system performance. Finally, nodes may communicate with each other directly, not mediated by a central server, which greatly promotes system efficiency.

This paper focuses on distributed spatial data manipulation issues and describes our implementation of a grid-based distributed spatial cooperative system, which parses a user submitted global spatial query to subtasks identified by the distributed spatial index and then dynamically passes them to appropriate nodes, cooperatively accomplishing complicated spatial tasks.

The remainder of this paper is organized as follows. Section 2 describes the distributed spatial cooperative computing system architecture. Section 3 discusses some issues of the grid execution model. Section 4 presents the implementation of the distributed spatial cooperative computing system prototype based on JXTA platform. Section 5 gives the conclusion to this paper.

2 SYSTEM ARCHITECTURE AND WORKING FLOW

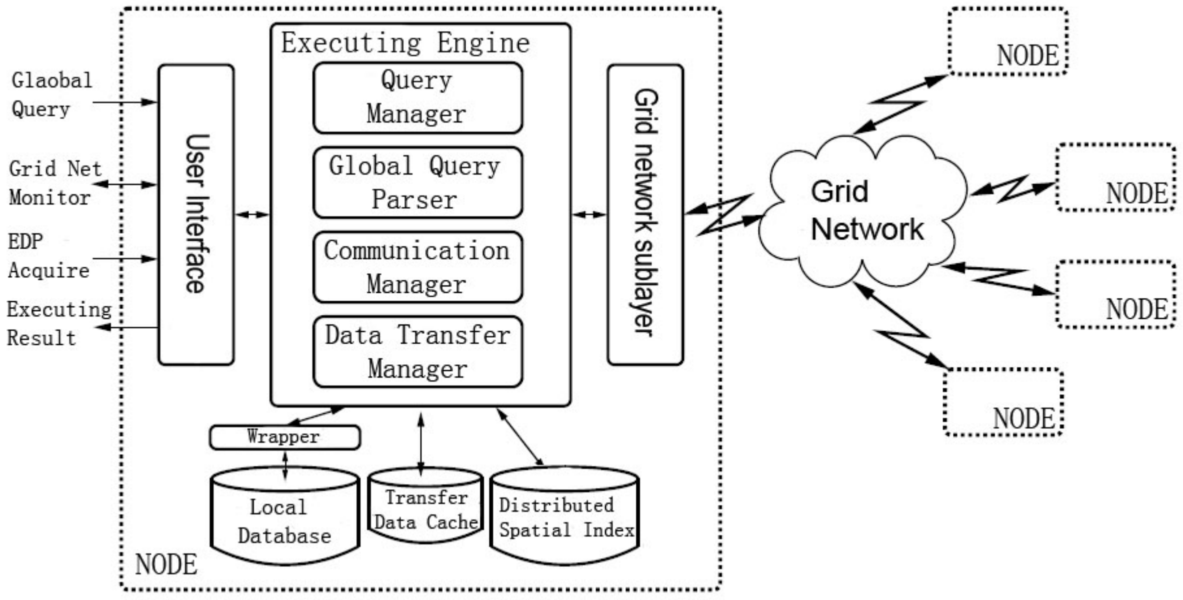


Figure 1. The architecture of a distributed cooperative computing system

Figure 1 illustrates the architecture of a node participating in the distributed cooperative computing system. The system is composed of identical nodes that intercommunicate via a grid network sub layer. A user interface is also implemented to manage a request from the user and eventually return a reply. Each node holds a local database of data resources and distributed spatial indices, which maintain distributed information of data resources and cooperative nodes in the system, along with a transfer data cache for bundle data transfer. However, the core module of the node is the executing engine, which consists of four components:

- Global query parser

Queries are submitted using a user interface and handled first by the global query parser. In this process, the global query parser checks the request's range to locate appropriate resources, generating an Equivalence Distributed Program (further mentioned as EDP), which contains several ordered local subtasks mapped from the global SQL statement and functionally equivalent to it. Then the EDP passes to the query manager for execution.

- Query manager

When an EDP arrives, the query manager combines it with a node state to form the executing sequence. Subtasks in the executing sequence may be relevant to resources disposed locally or globally in the system. In the former case, the subtask will be processed straightforwardly by the query manager; in the latter case, the query manager will dynamically search for the best quality node in the system to process the query. Also, the query manager may be involved in other nodes' query tasks.

- Communication manager

Communications between nodes are frequent, including node information exchange, executing instruction sending and response, and other control messages. The communication manager provides an abstract layer beyond the grid network to facilitate communication among the nodes. However, regarding the large amount and diversity of spatial data, data transfer between nodes is specially dealt with by the data transfer manager.

- Data transfer manager

In principle, data transfer should be avoided, but there are still some circumstances in which it is inevitable, such as delivering result data to the corresponding node. A data transfer manager is designed to directly pass data between databases of specific nodes.

3 DISTRIBUTED SPATIAL COOPERATIVE COMPUTING TECHNOLOGY

3.1 Distributed Hierarchical Spatial Indexes

Hierarchical spatial indexes are designed for efficient location of spatial resources and low control cost. First, the global range is divided into partitions, whose sizes are carefully calculated to ensure that there are neither too many nor too few nodes in one partition (Nam & Sussman, 2005).

Nodes in one partition hold the partition's neighbors' routing information, so a request can be routed to a destination partition swiftly by a greedy method (Fernandes de Mello & Senger, 2006). By partitioning the global range, complicated consistency maintenance and unnecessary message exchange are eliminated, as there is only a small amount of routing information to record among the partitions.

As in the partition, every node holds the same partition index, which is the combination of all partition nodes' spatial resource indices. Thus spatial resources can be located quickly by accessing any node in the selected partition. The cost of refreshing partition indexes in nodes is not too high because the number of nodes in one partition is limited.

3.2 Parse Global Query

A prerequisite for distributed computing is that tasks can be split into subtasks, which can be processed independently. As for spatial applications, the critical process in distribution is to map one global SQL statement into several subtasks according to spatial data resource distribution.

A distributed spatial data management system should deposit data independently and redundantly. However, data distribution and redundancy creates more complexity; spatial data, which are logically integrated, may be divided into tiles distributed in separated nodes. Therefore the global parsing process needs to access distributed spatial indices for locating resources.

The output of global parsing is an EDP file, that, when executing is equivalent to the global query. One critical property of the EDP file is that there may be no definite candidate nodes for each subtask; however, appropriate nodes will be elected dynamically to perform specified operations during execution.

Then the EDP file will be passed to the query manager, which will combine it with related nodes' states to execute content and sequence. After the query manager checks out tasks for it to achieve and then execute them, another important task is searching for the best quality node for a subsequent task. By communicating with all candidate nodes listed in the EDP file, the most appropriate node is chosen, and then the execution content and sequence are delivered to it. Thus the execution progresses.

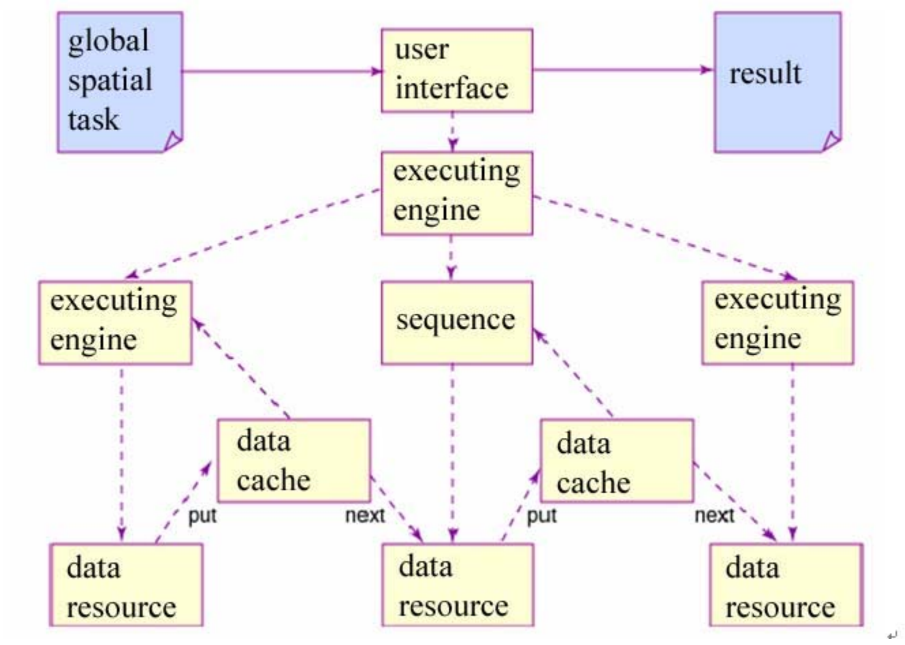


Figure 2. Parsing and executing spatial processing tasks

3.3 Cooperating Messages and Communication Mechanism

Nodes intercommunicate by cooperating messages and thus cooperate and accomplish spatial application tasks. Messages are sent directly to the destination (may be routed by other nodes), without needing to go through intermediate nodes such as the central controller or message server. This direct method thoroughly decreases network communication and accordingly reduces the possibility of congestion.

There are six types of message designed for communication. They are:

1) Start: This is a notification that a node is sending a start signal to ask successive nodes whether they are prepared for a subtask execution.

2) Start response: The successive node that has completed the preconditions check notifies the predecessor whether it is ready to execute subtask. If the response is yes, then the successive node also puts its quality characteristics into the response message.

3) Execution: After receiving the ready start response message from the successors, the predecessor picks out the best one and sends the execution content and sequence to it to start execution of the subtask.

4) Failed: When a subtask execution fails, the node quits the execution and sends a failure notification to its predecessor, which may start another candidate in compensation.

5) Commit: When the last node in the execution sequence finishes the last activity, it sends a commit notification to all the predecessors to release computing resources. Until a commit or an abort message is received, nodes participating in the current task are not permitted to quit.

6) Abort: This is a notification to abort the whole task execution. When receiving an abort notification, all the executing nodes will quit activities after rollback.

Execution content and sequence specifies tasks and orders and, with the messages introduced above, constitutes the foundation of node collaboration. Nodes use execution content and sequence to obtain the execution information of complicated spatial tasks and use messages to communicate, thus making it possible to accomplish spatial tasks.

3.4 Node Collaboration and Compensation

Once a spatial executing sequence is generated, a number of nodes are allocated to be in charge of initiating, controlling, and monitoring the specified subtask.

Figure 3 illustrates the collaboration process of a sequential execution among nodes. E_i and E_j are two sequential subtasks of a spatial executing sequence. Supposed that node e_{i1} has executed the specified subtask E_i successfully. If no exceptions or errors occur, the successor will perform a normal execution process as follows:

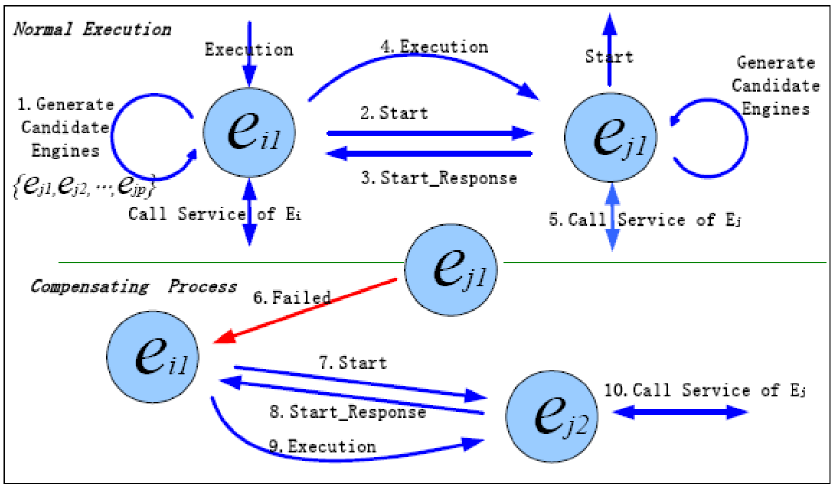


Figure 3. Engines in the collaboration and compensation process

1. Node e_{i1} checks the executing sequence to get the direct successor of subtask E_i. If no successor subtask (e.g. E_i is the last subtask of the sequence) is provided, it will send a *commit* message to all other nodes participating in the sequential execution and return results to the user. Currently, the direct successor is E_j. The node e_{i1} searches all nodes that delegate E_j and generates the candidate nodes list (Step 1 of Figure 3).

2. Then node e_{i1} picks out the best quality node e_{j1} to execute the specified subtask of E_j and sends a *start* message to inquire of e_{j1} whether it is able to execute the specified subtask of E_j (Step 2 of Figure 3).

3. After receiving the *start* message, node e_{j1} checks system resources (e.g., usage of storage, CPU and Memory) to decide whether to accept the executing request. The decision will be encapsulated in a *start_response* message and sent back to node e_{i1}. Here, we suppose that e_{j1} accepts the request. (Step 3 of Figure 3).

4. When node e_{i1} receives the *start_response* message, it will send an *execution* message to node e_{j1}. The *execution* message includes execution content and sequence (Step 4 of Figure 3).

5. When node e_{j1} gets an *execution* message, it begins to execute the specified subtask (Step 5 of Figure 3). After it finishes executing and gets a result, it repeats the process from 1 to 5 and passes the execution content and sequence to other node (s).

Also, an alternative participant approach for failure handling of the sequential execution is provided. During the sequential execution, one node chooses the appropriate participant node based on its state and keeps others in a candidate list. If the chosen participant fails, the node can choose another candidate node from the list. Then the whole composition goes on. In many situations, this saves an expensive rollback or abort. In the worst circumstance, where all the candidate nodes for a critical subtask fail, it does allow the whole execution to abort.

When the errors or exceptions occur, the executing node will launch the compensation process. Compensation follows the principle that the predecessor node, which allocates a successor node to run, is responsible for the compensation process when the successors fail. The compensation process is as follows.

As the continuance of the normal execution process (see Figure 3), we suppose that node e_{j1} encounters a failure.

6. Node e_{j1} detects the failure and sends the *failed* message to the predecessor e_{i1} (Step 6 of Figure 3).

7. After receiving the *failed* message, node e_{i1} checks the candidate list of subtask E_j. Here, we suppose node e_{j2} is the second best quality node. Then node e_{i1} will allocate e_{j2} responsibility to execute the specified subtask of E_j. The allocation procedure is similar to the process from 1 to 3 mentioned above.

8. The compensation process from 6 to 7 will be repeated until one node compensates it successfully or the candidate list is empty.

The result of the compensation process is either a success or a failure. In a successful compensation, the whole sequence will be executed as normal; otherwise, the execution of the whole sequence will be aborted, and an *abort* message will be sent to the nodes.

3.5 Distributed Transaction Mechanics

A two-phase commit protocol is chosen for transaction control. Each subtask in the executing sequence is assigned a state, which may be one of *waiting, abort, failed, ready, commit,* and *executing*.

When executing, each node examines its subtask. If successfully executed, the subtask is assigned *ready,* replacing *executing*. When the last subtask of the executing sequence is assigned *ready*, that node sends a commit message to all the nodes involved in current task. When a commit message arrives, nodes alter their state to *commit* and commit transaction.

If the execution has failed, the failing node alters its state to *abort* and rollbacks the transaction; also, all other nodes will be notified through an abort message. Nodes receiving an abort message set their state to *abort* and then rollback the transaction and quit.

4 SYSTEM IMPLEMENTATION

This paper has described the design and implementation of a grid-based distributed spatial cooperative computing system, the main characteristics of which are:

- Global query parsing

The first step is to construct a query tree of the global query submitted, and then the query is equivalently transformed to eliminate type conversion and I/O operation for a table scan. According to the optimized query tree, the global query parser integrates node and tile distribution information from a spatial resource index to generate an EDP file.

- Spatial index administration

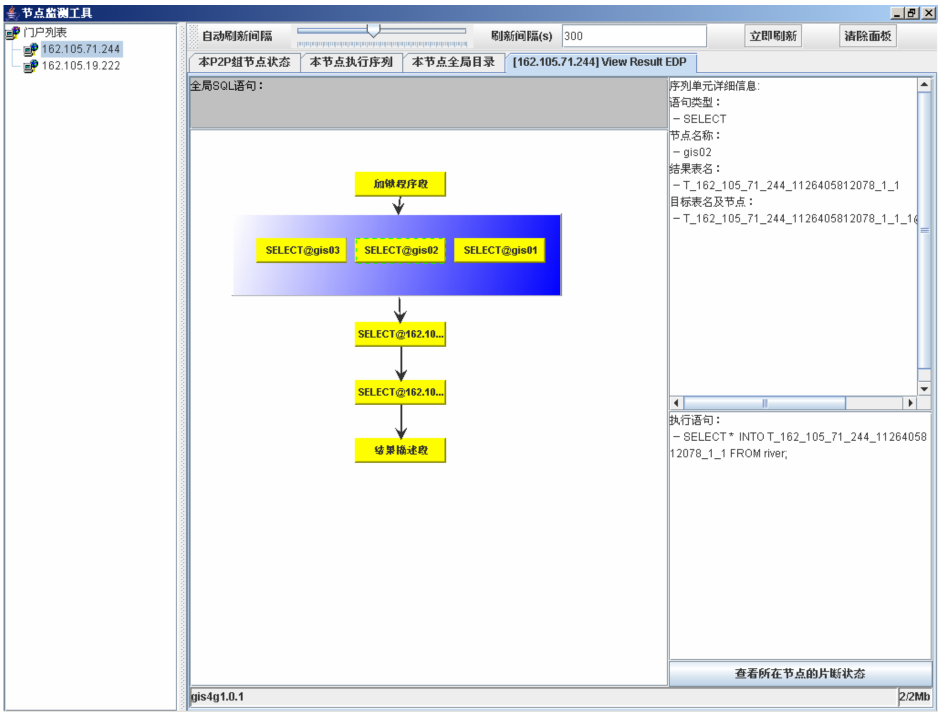
Nodes register with the distributed spatial index and publish the spatial resources they delegate. The node index maintains a set of registered nodes and their resources. When a node joins, it is first routed to a corresponding partition according to its spatial resource range, and then it publishes its resource information through cooperating messages and gains the updated partition index from any existent node. In contrast, when one of the nodes attempts to quit, it should find another node as its surrogate to publish its quit information. However, the distributed spatial index is an active index in the sense that it maintains live information about node availability. To achieve this, it continuously interrogates registered nodes, ensuring that they are active.

- Query performance

Users can submit a global spatial query to any node in the system, which consequently locates correlative nodes in terms of a distributed spatial index and then effectively performs the query via node collaboration and compensation mechanics. A distributed query communication model is implemented based on JXTA. Nodes intercommunicate between pipes using the virtual communication paradigm in JXTA networks (http://www.jxta.org/).

- System monitoring

Each node holds a monitor that inspects all other nodes in the system to determine whether the system is busy or if there are enough resources for scheming tasks. The monitor may publish its state information and discovery about other nodes at time intervals designated by the user. Correspondingly, as a node joins or quits, it automatically publishes its information to the system. Figure 4 illustrates the monitor view, which displays all participants of a currently executing task, and users can further select one of these nodes to examine its detailed information.



序列单元详细信息:
语句类型：
－SELECT
节点名称：
－gis02
结果表名：
－T_162_105_71_244_1126405812078_1_1
目标表名及节点：
－T_162_105_71_244_1126405812078_1_1_1

执行语句：
－SELECT * INTO T_162_105_71_244_1126405812078_1_1 FROM river;

Figure 4. Node resource and state monitoring interface

- Data consistency:

This system supports spatial data duplication and backup. Further, a distributed transaction mechanics is implemented to support spatial data consistency.

5 CONCLUSION

This paper presents a grid-based distributed spatial cooperative computing technology for spatial data manipulation and distribution to accomplish complex spatial applications by collaborating spatial databases that are distributed in different nodes. Important issues discussed in this paper are: distributed hierarchical spatial indexes, global query mapping, node communicating and monitoring, and data consistency in distributed systems. Also we describe a distributed cooperative computing system prototype implemented to demonstrate the main principles of the approach. The mechanics of the grid avoids network congestion and single point of failure, which are caused by a centralized execution model and therefore improves the availability and reliability of spatial data and computing wealth.

6 ACKNOWLEDGEMENTS

This research is sponsored by the China National Advanced Science and Technology (863) Plan under Grant SQ2006AA12Z214047 and the Beijing Science Foundation under Grant 4062019.

7 REFERENCES

Alameh, N. (2003) Chaining geographic information Web services. *IEEE Internet Computing 7(5*): 22-29.

Andrews, P., Sherwin, T., & Banister, B. (2003) A Centralized Data Access Model for Grid Computing. *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03).*

Dyo, V. & Mascolo, C. (2005) Adaptive Distributed Indexing for Spatial Queries in Sensor Networks. *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05).*

Fernandes de Mello, R. & Senger, L. (2006) A Routing Load Balancing Policy for Grid Computing Environments. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06).*

JXTA, The Project JXTA web site. Retrieved from the WWW September 12, 2007: http://www.jxta.org/.

Koszek, P. & Sandrasegaran, K. (2005) Grid Architecture Storage - Utilising Grid Computing for Dynamic Data Storage. *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05).*

Nam, B. & Sussman, A. (2005) Spatial Indexing of Distributed Multidimensional Datasets. *IEEE International Symposium on Cluster Computing and the Grid.*

Wehrle, P., Miquel, M., & Tchounikine, A. (2005) A Model for Distributing and Querying a Data Warehouse on a Computing Grid. *Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05).*