2008 International Conference on Computer Science and Software Engineering

Agent-based Spatial Query Optimization in a Grid Environment

Yufei Bai, Xiujun Ma, Kunqing Xie, Cuo Cai, Ke Li
Department of Machine Intelligence
Key Laboratory of Machine Intelligence (MOE)Peking University
Beijing, 100871, China
{baiyufei, maxj, kunqing, cci, like}@cis.pku.edu.cn

***Abstract*—Resource management and optimization among autonomous spatial databases in a grid environment is always a challenge task. In this paper, we introduce a BDI Agent model for cooperating complex spatial query optimization task in a grid environment. Spatial query agents coordinate with spatial database resourse agents using FIPA auction protocol for dynamic spatial query execution. The expreiment showed that BDI Agents achieve efficiency on complex spatial query optimization in grid envirionment.**

Keywords- BDI Agent; grid; spatial query optimization

I. INTRODUCTION

In geospatial application, as the infrastructure of information management and processing, grid [1] computing provides the ability of integration and sharing of enormous distributed geospatial information resources and autonomous systems [2]. Thus it provides a solution to meet the demand of retrieving geospatial information, sharing and analyzing geospatial data.

Agent technology is being used increasingly in grid system [3]. As Ian Foster and some other people pointed in [4], grid provides the infrastructure of sharing large scale datasets and solving complex tasks, but it needs a flexible management for node resources; autonomous agents have the ability of coordinate with each other [5]—this is just what grid needs.

In recent years, Agent technology has been increasingly used in geospatial research and GIS applications [6]. Resource management and optimization among autonomous spatial databases in a grid environment is always a challenge task. The objective of this paper is to introduce BDI Agent model [7] into developing spatial query optimization system for a large scale spatial data grid application.

This paper is organized as follows. Section 2 describes the multi-agent system architecture that based on a sponsor-mediator-executor structure of Agent pattern. Section 3 presents Agent-based spatial query optimization implementation. The system implementation and experiments are showed in section 4. The conclusion reports some findings of the proposed approach.

II. .ARCHITECTURE

We adopt a sponsor-mediator-executor structure of Agent pattern. The architecture of Agent system is shown in Figure 1. In the system, the sponsor is Query Agent (QA), the executer of a task is Resource Agent (RA), and the mediator is Directory Agent (DA) which provides directory service for theme queries.

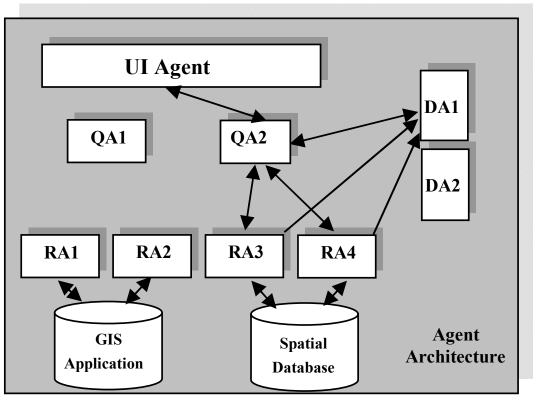

Figure 1. Agent architecture

UI Agent is the initiator of a spatial query. It receives users' queries and transfers it to QA. It also adjusts QA and RA directly

*Query Agen*t is created according to a certain strategy, often on the node with more bandwidth and computational resources. After receiving a spatial query, QA analyzes the query and then transforms it into a spatial query plan. Then QA searches Directory Agent for looking for proper Resource Agent which has the data needed by the spatial query .At last QA distributes the sub query tasks to RA and execution status will be returned to UI Agent.

Resource Agent runs on each node in the grid, which calls the function of spatial database and GIS to execute a spatial task distributed by QA. RA maintains a task queue which will be adjusted dynamically to achieve efficient execution. When a task is completed, RA returns the results or the execution status of the task.

Directory Agent is the container of data information. It receives registration or update of spatial data from RA and

Identify applicable sponsor/s here. *(sponsors)* accepts query and browse requests from other Agents (QA and UI Agent). Data in DA is organized by theme and records represent fragments of a certain theme.

III. Agent-based Spatial Query Optimization

The processing of a spatial query task can be divided into two phases: First, a query plan is generated based on the user's request and theme information in DA; Second, dynamic assignment and execution of the query plan through the cooperation between Agents. The procedure is shown in Figure 2.

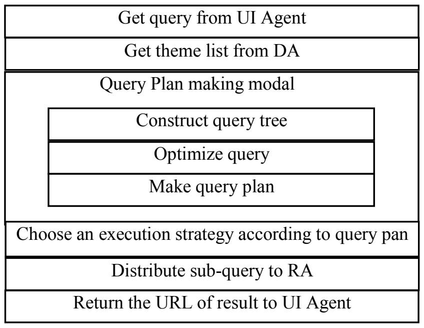

Figure 2. Execution of a query

A spatial query often started when QA received the request send by UI Agent. A spatial query is usually described in spatial SQL, as is shown in Figure 3.

```
SELECT a.name as city_name, b.name as mine_name, a.position as
city_position, b.position  as mine_position
FROM cities a, mines b
WHERE distance(a.position, b.position) < 100
AND overlaps(a.position, rectangle(1,1, 200,200))
```

Figure 3. Example of a spatial SQL

This query retrieves city and its nearby mine position in a specified range, where table cities and mines respectively represent city and mine theme. Such a query task will be decomposed into a series of subtasks by QA which could executed by a single node. The union of the subtasks is called as query plan.

The process for generating the query plan from a spatial SQL contains two steps:

First QA constructs the spatial query tree for the spatial query. Similar with the traditional database, the spatial query tree is constructed mainly for lexical and syntax analyzing. By decomposing the spatial query string, QA identify all the key words in the query which are called *Token*. Based on lexical analyzing, *Tokens* are classified into different categories from which a spatial query tree will be constructed. Also QA could find out whether the query string accords with syntax rules or not. A spatial query tree with good syntax structure provides the foundation for further analysis and optimization.

Then QA optimizes the formed spatial query tree. It searches spatial themes which are related to the query in DA so as to retrieving the fragments list, and the data amount, node position of each fragment. Based on the number of theme fragments retrieved, QA adjusts the table sequence in FROM clause in order to reduce joint and I/O cost. After this, one or more candidate spatial query trees will be constructed.

At last QA converts the candidate query trees to candidate query plans, represented by XML.

According to the generated query plan, a spatial query then be decomposed into sub tasks and distributed to various RA to execute. A task that can't be decomposed any more is called atom task. To RA an atom task is equal to an atom SQL which can't be simplified any more.

A. Auction protocol

Based on FIPA auction protocol[8], we propose two auction protocols for dynamic spatial query execution. In the auction, "Price" is described as cost of execution time or the estimated execution time of a task.

- *"Price inquiry":* This is an auction between QA and the entire candidate RA. QA sends the description of task to RA and calls for proposal, each RA returns whether to accept and estimated execution time before dead line, otherwise the return will be ignored. Based on returns, QA chooses a RA with lowest price and assigns the task to it. After the execution RA returns the result of the task. Price Inquiry protocol is showed in Figure 4.
- *"Marked price":* In this auction QA evaluates a standard query time as marked price based on the data quantity concerning to an atom task, then QA sends the description of task and marked price to RA. Then RA estimates its execution time as price, if lower than the marked price and willing to, RA accepts the task, else refuses and returns its price. QA assigns the task to the first RA who accepts it, or chooses RA with the lowest price. Marked price protocol is showed in Figure 5.

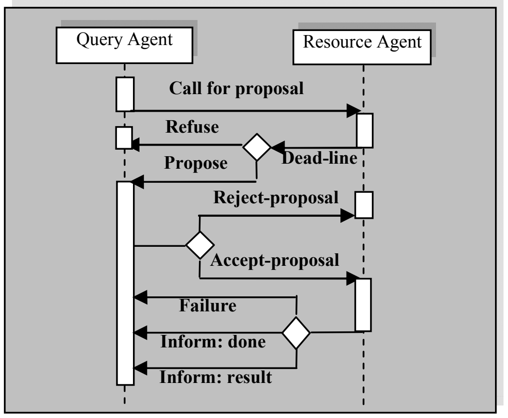

Figure 4. Price inquiry protocol

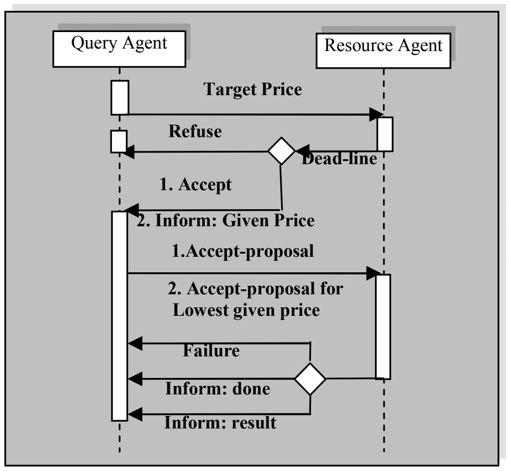

Figure 5. Marked price protocol

Number 1 and 2 represents the two situations QA met in the protocol.

The differences between the two protocols are that in price inquiry model, the auction finished only after all the RA returns price or over the dead line. If the network is unstable or the task has a short execute time, the time cost of auction is probably the same as execution time of task, this is low efficient. Marked price model can shorten time cost of auction for it needs only one RA's return, but it often has a high standard which is not easy to find a perfect executor for those tasks with high time cost.

When the estimated execution time is longer than threshold or the system idleness rate is less than threshold, QA adopts the price inquiry model, when the estimated execution time is shorter than threshold or the system idleness rate is larger than threshold, QA adopts the other.

B. Optimization of query plan

RA is the executor of the query plan. The query plan execution (QPE) can be divided into four situations:

- *Choose.* QA evaluates plans in Choose structure and generates the sub-QPE for the one with shortest execution time. The chosen sub-QPE is also a query plan which can be called query plan block.
- *Parallel.* QA generates a sub-QPE for each sub plan in Parallel structure, sub-QPEs are independent with each other.
- *Sequence.* QA generates a sub-QPE for each sub plan in Sequence structure, sub- QPEs have dependence relationship and must be executed sequentially.
- *SQL.* Based on current status, QA chooses a RA which can complete the task with fewest time cost.

When sub-QPE finished or atom SQL completed successfully, RA returns the results, and marks finished.

QA could run an arbitrary number of tasks at the same time (if system permits), but only for task scheduling. RA organizes all its tasks as a queue, and adjusts the queue according to the execution status.

During the execution of query plan, we have two choices: first is choosing the optimal RA to complete the atom SQL; second is choosing the optimal query plan in Choose structure. Thus QA needs to evaluate the execution time of an atom SQL or a sub-QPE.

To the first we adopt the proposed *Price Inquiry Protocol* and *Marked Price Protocol* explained in previous section to decide which RA will the task be assigned. Atom tasks in a Sequence structure will be assigned to the same RA, while price inquiry model is adopted.

To the second we choose the optimal sub-QPE in a Choose structure, by evaluate execution cost of each sub task and data quantity. Each sub-QPE has a total cost: For the Sequence sructure, total cost is the sum of each part; for the Parallel structure, total cost is the max of all parts; for the Choose structure, total cost is the min of all parts.

C. Dynamic execution of query plan

RA executes the atom tasks distributed by QA and manages the execution queue, mainly in two aspects.

First, RA decides whether to accept a task and provides estimated execution time to QA. There are several factors affect the decision: if the length of current task queue is less than the given threshold, then RA accepts the atom task, else only accepts the repeated task. A repeated task is that if it fit the existing result stored by RA, or there has been a same atom task in the queue.

Second, RA dynamically manages its task queue, when executing an atom task, RA will download the input data of the next task. Periodically RA informs the new estimated complete time to QA that corresponding to the atom tasks in queue. After the execution, RA returns the results status to QA, the status will be success or failure.

If the new estimated complete time is longer than RA estimated initially or the task failed to complete, QA will start another auction in all the other candidates for current atom task, and reassigns this task.

IV. System Implementation and Experiment

We develop Agent by Jadex [9] (extension of Jade) and PostGIS for spatial database server. We adopt three IBM x306m with RHEL AS4 U3 as server, and have part of the traffic net data in three Northeast Provinces. The data has such spatial properties: station (6061 records), road (6285 records), and district (210,000 records).

By simulating 50 RA (node), 4 DA, we test 10 classes and a total 50 queries' performance in different strategies: A. nodes have same process ability and bandwidth; B. same bandwidth and abilities with Normal Distribution; C. same bandwidth and abilities with Mean Distribution; D. both abilities and bandwidth with the same Mean Distribution; E. abilities and bandwidth with independent Mean Distribution. Assume that nodes process equally to all the queries. The result is showed in Figure 6.

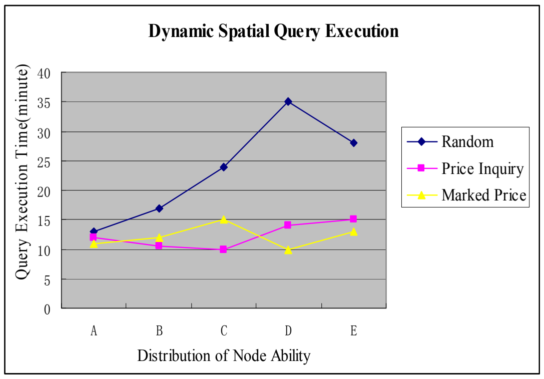

Figure 6. Experiment result

We can see that if nodes have unbalanced distribution of ability and bandwidth, it is inefficient for random assignment while marked price model brings a better efficient — for it shortens the waiting time before assignment and considers more about the effect of bandwidth. Price inquiry model applies the situation that nodes have the same distribution of bandwidth, for it can completely evaluate the nodes.

V. Conclusion

We have developed an Agent-based spatial query coordinate execution mechanism. QA receives a spatial query from UI Agent and generates a spatial query plan based on the theme fragments registered in DA and data quantity, by using the two proposed auction protocol, QA distributes the decomposed tasks to the optimal RA. Our experiments show that this approach is applied to the situation having different bandwidth and computational abilities. There are some future works, for example, how to estimate the execution time more exactly.

Acknowledgment

This research is sponsored by the Beijing Science Foundation under Grant 4062019 and China National Advanced Science &Technology (863) Plan under Grant 2006AA12Z217.

Reference

[1] http://bk.baidu.com/view/10755.htm, Grid, 2007.

The Earth System Grid II: Turning Climate Model Datasets Into Community Resources. Annual Meeting of the American Meteorological Society, 2002

[2] Tsou Minghsiang , Buttenfield Barbara P. Agent based mechanisms for distributing geographic information services on the Internet, GIScience 2000 [OL]

[3] Foster, I., Jennings, N. R. and Kesselman, C. (2004) Brain meets brawn: why Grid and agents need each other. In: 3rd International Conference on Autonomous Agents and Multi-Agent Systems, 2004, New York, USA.

[4] Koch Andreas. Linking multi agent systems and GIS : Modeling and simulating spatial interactions [OL]

[5] Junwei Cao, Stephen A. Jarvis, etc, ARMS: An agent-based resource management system for grid computing, Scientific Programming 10 (2002), pp135–148

[6] Omar Rana, Michael Winikoff, Lin Padgham, and James Harland, Applying Conflict Management Strategies in BDI Agents for Resource Management in Computational Grids, Proceedings of the Australasian Conference on Computer Science, Melbourne, January, 2002.

[7] http://www.fipa.org/, FIPA Auction-English Protocol, 2005

[8] http://vsis-www.informatik.uni-amburg.de/projects/jadex/, Jadex, 2007