GeoLoc: Robust Resource Allocation Method for Query Optimization in Data Grid Systems

Igor Epimakhov, Abdelkader Hameurlain, Franck Morvan

GeoLoc: Robust Resource Allocation Method for Query Optimization in Data Grid Systems[1]

Igor EPIMAKHOV[a,2], Abdelkader HAMEURLAIN[a], Franck MORVAN[a]

[a] *Institut de Recherche en Informatique de Toulouse IRIT,*
Paul Sabatier University, 118 Route de Narbonne, 31062 Toulouse, France
{Igor.Epimakhov, Abdelkader.Hameurlain, Franck.Morvan}@irit.fr

Abstract. Resource allocation (RA) is one of the key stages of distributed query processing in the Data Grid environment. In the last decade were published a number of works in the field that deals with different aspects of the problem. We believe that in those studies authors paid less attention to such important aspects as definition of allocation space and criterion of parallelism degree determination. In this paper we propose a method of RA that extends existing solutions in those two points of interest and resolves the problem in the specific conditions of the large scale heterogeneous environment of Data Grids. Firstly, we propose to use a geographical proximity of nodes to data sources to define the Allocation Space (AS). Secondly, we present the principle of execution time parity between scan and join (build and probe) operations for determination of parallelism degree and for generation of load balanced query execution plans. We conducted an experiment that proved the superiority of our GeoLoc method in terms of response time over the RA method that we chose for the comparison. The present study provides also a brief description of existing methods and their qualitative comparison with respect to proposed method.

Keywords: Data grid systems, resource allocation, distributed query processing and optimization, incentive-based scheduling, extended classic scheduling, allocation space definition, intra-operation parallelism degree determination.

Introduction

One of the most important problems of query processing in the Data Grid environment is Resource Allocation (RA) - assignment of resources to the query operations. Being sent by the user, a query is processed by one of the Data Grid nodes, which takes the role of scheduler for the query. The problem lies in the fact that for the placement of an operation we need to select a subset of nodes among a set of Data Grid nodes, that the placement on this subset will minimize execution time of the query. In addition, each node in the Data Grid environment has its own static characteristics, such as CPU performance, amount of memory, bandwidth of I/O system and network; and dynamic

[1] This work was supported in part by the French National Research Agency ANR, PAIRSE Project, Grant number -09-SEGI-008.

[2] Corresponding author

characteristics: a current load of each of these mentioned resources. Another important aspect is distribution of relations over the nodes as well as their replication. Large scale of the Data Grid systems is also complicates the problem of RA, practically eliminating a possibility of using of exhaustive search algorithms.

After analyzing a number of papers on the subject [1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 23, 24], we conventionally distinguish two fundamentally different approaches [6]: incentive based [4, 13, 20, 24] and extended classic [1, 2, 5, 7, 8, 9, 10, 16, 19, 21, 23]. Their main difference lies in the cooperation type between the scheduler node and candidate nodes for hosting operations. The first approach is based on the use of some virtual incentives that encourage nodes to participate in query optimization and execution processes. The second approach involves some internal discipline or subordination of the set of candidate nodes.

In [6] we have considered and analyzed a number of proposed methods, highlighting some of advantages and disadvantages of each method. In considered studies were presented various solutions for resolving the RA problem taking into account different specific characteristics of Data Grid environments. We believe that in those studies was paid little attention to such aspects as the definition of allocation space (AS) and determination of the optimal degree of parallelism.

In this paper we propose a RA method named GeoLoc. It uses classical approach with a static RA scheme. The main ideas are:

- Restriction of the set of candidate nodes (allocation space) based on the selection of nodes that are geographically close to data sources.
- The criterion for determining an optimal degree of intra-operation parallelism based on parity between the total capacity of source nodes and the total capacity of nodes performing the join operation.
- The ranking function, which estimates overall capacities of nodes for the execution of query operations.

The present work deals with a multi-join query. Our algorithm is based on the "greedy" principle of nodes selection using an estimated overall capacity of each node. The scheduler sequentially optimizes each operation in the query plan. The degree of parallelism for each operation is determined by adding nodes one by one until the optimality criterion is fulfilled. We conducted an experiment using our own developed Data Grid simulator, which confirmed that the query execution plan generated by GeoLoc is more efficient in terms of execution time comparing to the method proposed by Gounaris et al. [10] (hereafter “reference method”). However, its optimization process is also more expensive.

The paper is organized as follows: for the first, section 1 presents a brief comparison of our method with existing works presented in the literature. Then, an approach for allocation space restriction will be introduced in section 2. A detailed description of GeoLoc algorithm is provided in section 3. And before presenting our conclusions, we provide in section 4 results of performance evaluations of the proposed method in comparison with the reference method [10].

1. Related Works

In recent years were published several works covers the issue of RA in the Data Grid environment, we will consider some of them. [1, 2, 5, 7, 8, 9, 10, 16, 19, 21, 23]

There are three main strategies that are being studying in literature in parallel [6]: static, dynamic and hybrid, each of which has its own specific characteristics that determine its ability to respond to changes of a dynamic environment and its ability to use different forms of parallelism for query optimization.

First strategy consists in performing RA once before query execution phase basing on the information of resources that is available at the moment of scheduling. In [10] was offered an elegant method with the static strategy that exploits practically all types of parallelism. In the beginning the algorithm obtains an optimized query execution plan with a degree of independent intra-operation parallelism equal to 1 (each operation assigned to a single node for its implementation). Than for the most expensive operations, it improves the parallelism in the loop by appointing one additional node to perform optimizing operation on each step. The loop continues until degree of parallelism of the operation becomes optimal. Another static method presented in [19], where authors proposed a scheduler that uses inter-query and intra-query parallelism. Intra-query parallelism is limited only by partitioned intra-operation and pipeline parallelism. There is iterative algorithm, which takes the query bushy tree and parallelizes each operation on the optimal number of nodes. In order to provide load balancing, scheduler selects firstly the nodes that do not perform any operations with the requested relation at the moment. Very interesting static method was proposed in [16]. Like GeoLoc, it operates with estimated overall capacities of nodes based on its parameters and parameters of queried relations. But it implements only an independent inter-operation parallelism. The optimization algorithm builds a logical tree for a query, then for each relation of the query it selects, as a source, only one best node.

For resolving the RA problem, solutions are often based on the graph theory. Good work in that way of research was presented in [21]. Authors consider as a problem the static RA of a set of independent jobs with intensive usage of large volumes of data. As an objective they take maximization of throughput. Multiple data sources and computing resources are considered independently, even if they are physically located at the same node. A set of data sources and computational resources is represented as a weighted graph. For the calculations algorithm selects computational resources closest to the data sources. The problem was reduced to a Set Covering Problem and was used a well-known algorithm for its solution.

Dynamic strategies of RA in Data Grids are also presented in the literature [1, 2, 23] but does not get much support in the scientific community. The strategy consists in performing separate optimization of each query operation after finalizing the execution of previous ones. This approach deals with dynamicity of a Data Grid environment better than pure static methods, but its capacity to reflect to system changes is limited by the moments between operations when it can modify the plan. Also the method is limited in using of the pipeline parallelism.

The third hybrid RA strategy can be considered as an extension of the static strategy, which proposes to complement an initial static allocation with the dynamic reallocation that could reflect to changes of the environment and modify the plan during the query execution phase. It gained much attention in the recent works [5, 7, 8, 9] and we believe that it is a very promising strategy in the dynamic large scale environment. An interesting solution implementing the strategy was proposed in the work [5]. The

method is based on the greedy RA algorithm, which selects nodes with maximal throughput capacities, known from previous queries executions. The method includes a dynamic load balancing algorithm on the basis of the algorithm Eddies, which allows transferring the load between nodes during operation execution without interrupting the operation.

Another important characteristic of RA methods is the organization of a control structure of query scheduler. Most of methods are based on the centralized scheduling, relying upon the central Data Grid scheduler [1, 2, 5, 16, 19, 21, 23], using a global catalog that provides complete information about all nodes and relations of the environment or even about network topology and interconnection links [21]. One of advantages of that approach is the possibility to exploit an inter-query parallelism that is impossible in decentralized approaches. But, in our opinion, centralized scheduling is a very risky solution in the large scale dynamic environment, because in that case the functionality of whole system depends entirely on the reliability of the single control element. That is why our method is based on the decentralized approach that we consider the most effective solution in the Data Grid .

Also we believe that it is important to base a method on the realistic hypothesis about distribution of relations in the Data Grid. For example, a number of methods [1, 2, 16, 19, 21, 23] do not take into account distributed relations, considering only its duplication. In contrast, in GeoLoc method we utilized distribution and duplication of relations, which is not only more realistic, but also raises the efficiency of our algorithm. The principle was implemented also in some other methods [5, 10].

In our opinion, the two points of interest got little attention in all reviewed works: definition of allocation space (AS) and criterion of determination of a parallelism degree. For the AS definition, the typical solution is to consider all existing nodes as candidates [21] or to restrict the search space only by nodes that initially contain a requested data [16, 19]. The first solution is not realistic in a large scale environment and the second is too strong restriction that may decrease the efficiency of RA. That is why we proposed our definition of allocation space, considering two categories of candidate nodes: source nodes and nearest nodes.

For determination of parallelism degree, some authors propose to increment iteratively parallelism of an operation, finishing the process when adding another node to allocation plan does not give any more profit [5, 10]. In [19] presented an algorithm of load balancing of query plan, which reassigns nodes among operations from less complex to more complex ones. Our method differs from the above-mentioned by using the criterion of parity between scan and join operations, which permit to generate load balanced query plans with response time close to optimum.

2. Allocation Space

Allocation space (AS) is a set of nodes, which are considered as candidates for the query placement in the Data Grid. In a large-scale environment for solving the problem of RA we cannot consider as candidates the entire set of existing nodes. This would make the task of the optimal placement extremely time-consuming and expensive. The only one solution is to limit the AS by those nodes that are supposed to be the best candidates for the processing query.

Admittedly, as candidates for query operations placement we consider initially the nodes that store fragments of processing relations. Firstly, they naturally act as a data

source for placement of Scan operations. Secondly, placing on them join operations is often an effective way to reduce the amount of data transferred over the network. This is especially important considering that the network connections in a large-scale environment is the most critical resource that often becomes a bottleneck that limits performance of the query. This category of nodes we named source nodes.

The second major category of node which we propose to include in the AS is a set of nodes that are close geographically to the source nodes. We assume that the nodes that are in a geographical proximity to the source nodes have, in general, less extensive communication links with them and therefore the data transferring between them is much faster with fewer loads on the network components. As a consequence, the nearest nodes are attractive candidates for join operations placement because the distribution (or redistribution) on them can be done with a much lower cost. Of course, in our assumption there may be exceptions in particular case, however, we believe that in general it is realistic and will provide substantial benefits on average. This category of nodes, we denote nearest nodes.

Discussing the need to use the nearest nodes, we assume that all nodes store information about their own geographic coordinates, and that other nodes of Data Grid can use it in the resource discovery phase to determine the geographical proximity.

3. Allocation Algorithm

The problem of query placement consists in finding an optimal placement among a large set of possible placements and is proven to be NP-complete [12]. There are two main approaches for resolving this problem: exact methods [22] and heuristics [17]. The first class of methods cannot be applied in large-scale environments because of the huge number of possible placements. So for resolving the problem of RA in Data Grid the second approach is a natural choice. It allows us to find quasi-optimal solution in a short period of time, which is very important because in a dynamically changing environment, the resource data may become obsolete faster than a slow algorithm finds an optimal solution.

We decided to use as the base of our method a so-called “greedy” heuristic, which is widely used in optimization algorithms. We would remind that in our system, we consider all relations as a number of non-intersecting fragments, each of which is in general replicated among multiple nodes. The proposed algorithm receives as an input a set of discovered candidate nodes and as an output it returns a generated execution plan. For each join operation a separate RA is performed, the result of which - the intermediate relation - is used as a source in the join operation of a higher level. Thus, the algorithm passes a logical query tree bottom-up, successively placing resources for each join operation.

The proposed algorithm consists of two main steps:

1. Definition of the allocation space
2. Parallelism degree determination and generation of an execution plan

We will examine them more in detail in the following subsections.

2.1. *Definition of Allocation Space*

As described above, AS includes source nodes and nearest nodes. Preparing of AS in our algorithm starts with the selection of source nodes. Each node in our system we consider as a set of hardware resources, such as:

- Network connection bandwidth (Mb/s)
- I/O performance (Mb/s)
- Amount of memory (Mb)
- CPU performance (MOPS)

For node performance estimation, for using it as a source of relation, we take into consideration only the first two hardware resources (network and I/O). We assume that each used relation fragment will be read from I/O subsystem and transferred to other nodes for executing the join operation. I.e. total bandwidth of the node N we estimate as a minimum between network connection and I/O subsystem bandwidths:

$S_n = Min(Net_{eff}, IO_{eff})$ — Where $\mathrm{Net_{eff}}$ - unoccupied network connection bandwidth, $\mathrm{IO_{eff}}$ - unoccupied I/O bandwidth

$Net_{eff} = (1 - C_{net}) \cdot Net$ — Where $\mathrm{C_{net}}$ - load factor of the network connection, Net – network bandwidth

$IO_{eff} = (1 - C_{io}) * IO$ — Where $\mathrm{C_{io}}$ - load factor of I/O bandwidth, IO – I/O bandwidth

Thus, the node performance index depends directly on the current load of its resources. For placing Scan operation for each fragment of the relation we select the one that have the highest performance index (bandwidth), i.e.
$S_{high} = Max(S_1, S_2, \dots, S_n)$

After defining the set of source nodes, we can place Scan operations and generate AS incorporating both source nodes and its nearest nodes. We suppose that the entire set of nearest nodes for each potential source node is determined at the phase of resource discovery. Consequently, after selecting source nodes, we can determine a subset of its nearest nodes.

2.2. *Generation of Execution Plan*

The algorithm of an execution plan generation is intended for the resources allocation of a query. Firstly, it selects a subset of nodes from a set of candidate nodes for query execution, and then it determines an optimal degree of intra-operation parallelism for each join operation. As an input the algorithm receives a set of candidate nodes (AS) and characteristics of joining relations and its fragments. At the output, it returns the final query execution plan with allocated resources.

GeoLoc algorithm
INPUT: set of candidate nodes, characteristics of relations
OUTPUT: query execution plan

BEGIN

1. **FOR** each join $J \in Q$ **DO**
2. Count the time of source relations read and transferring, *Tscan_exec*
3. **DO**
4. Choose the most efficient node *Neff* from a set of AS for placing join operation
5. Add *Neff* to the join allocation plan, *Pjoin*
6. Estimate the execution time of join, *Tjoin_exec*
7. **WHILE** (*Tjoin_exec* > *Tscan_exec*)
8. Add *Pjoin* to the query allocation plan, *Pquery*
9. **ENDFOR**

END

As a result, the algorithm sequentially adds one node at each iteration until the estimated execution time of the join operation becomes equal to the estimated execution time of Scan operations . Offering the criterion for determining the degree of parallelism, we assumed that the join execution time is limited naturally by the time in which the original relation will be read and send.

It is necessary to clarify the step 4 of our algorithm. To determine the most productive node we need in the first place to estimate an expected load of the operation on node's calculation resources. We propose the following formulas:

$$NET = NET_{in} + NET_{out} = \frac{R_1 + R_2 - S}{N} + (S * \frac{N-1}{N} + \frac{R_{result}}{N})$$

Where NET is the amount of data sent in both directions over the network connection; R_1, R_2- sizes of the first and the second relations involved in the join; S - size of the fragment stored at this node; N - the current degree of parallelism; R_{result} - size of relation obtained after joining of R_1 and R_2.

$$IO = IO_{write} + IO_{read}$$
$$= \max(R_1 + R_2 - MEM, 0) + \max(R_{result} - \max(MEM - R_1 - R_2, 0), 0)$$
$$+ S + \max(R_1 + R_2 - MEM, 0)$$

Where IO is the number of data reads and writes through the I/O subsystem; MEM is the amount of free available memory.

In these formulas, we assumed that:

- Each relation is distributed by N equal parts.
- Hybrid Hash Join algorithm is used.
- The result of each join is always retransferred from the node.
- All available memory can be used for full or partial storage of source relations and the resulting relation, reducing the data transferred through I/O subsystem.

After calculating an expected load on the resources of the node, execution time estimation is performed taking into account a current load of resources:

$$T_{io} = \frac{IO}{PERF_{io} * (1 - C_{io})}$$

Where T_{io}, $PERF_{io}$ and C_{io} – execution time, performance and load factor of the I/O system.

$$T_{net} = \frac{NET}{PERF_{net} * (1 - C_{net})} + latency$$

Where T_{net}, $PERF_{net}$ and C_{net} – execution time, performance and load factor for the network connection. latency – latency of the connection;

$$T_{result} = \max(T_{io}, T_{net})$$

Where T_{result} - execution time estimation.

After generating an execution plan for a join operation, the algorithm is repeated for a next join until the resources will be assigned for all operations. Meanwhile, nodes that store obtained intermediate relations from previous operations are used as source nodes in subsequent operations.

4. Performance Evaluation

We have conducted an experiment to determine the effectiveness of the proposed RA method. We used our Data Grid simulator, which allows simulating RA methods in a large-scale environment. The main parameters that we used for the simulation are shown in Table 1.

Table 1. System configuration and database parameters.

Parameter	Value
Node CPU performance	10 – 1 000 MIPS
Node I/O performance	10 – 90 Mb/s;
Node memory amount	0,001 – 40 Mb
Node network connection bandwidth	10 – 60 Mbit
Node network connection latency	0.5 s
Relation number of attributes	10
Relation size of attribute	300 Bytes
Relation cardinality of attributes	0.3 – 0.9
Relation size of tuple	3000 Bytes
Relation number of tuples in relation	1000 – 11000
Relation size	3Mb – 33Mb
Relation fragments number	10
Relation duplicates number	10

Simulation model and system parameters discussed in detail in [6]. Measurements were performed by running a series of 100 queries for various levels of complexity. As simple queries we considered queries with 1 and 2 joins, average - 3-5 joins, complex – 6-7 joins.

In our experiment we compared the methods by two principal parameters: optimization time that reflects the duration of RA process and response time that we count as a sum of optimization and execution times. Also was counted a speed-up for each of measured parameters as a biggest value divided by a smallest one.

4.1. Performance Analysis

For our comparison we decided to use the reference method [10], implementing it on the basis of the greedy heuristic. This method uses all kind of parallelism, but its allocation space is limited by source nodes only.

Reference algorithm
INPUT: initial query execution plan (with minimal degree of parallelism), set of candidate nodes
OUTPUT: query execution plan

BEGIN
1. *Plan* = Initial plan
2. **LOOP**
3. *X* = MostCostlyOperation(Plan)
4. **IF** *X* is Optimized **THEN EXIT LOOP**
5. **DO**
6. Increase the degree of parallelism of *X*
7. **UNTIL** PerformanceIncrease(*X*) < Threshold
8. Mark operation *X* as Optimized
9. **ENDLOOP**
END

From our point of view, it is quite representative for the classical approach of RA.

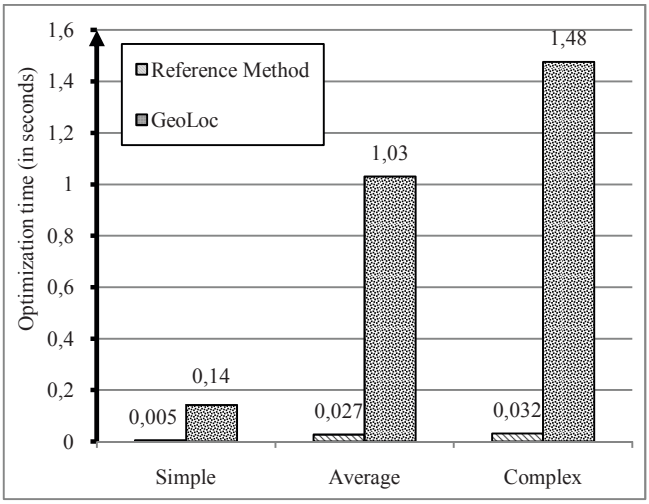

Figure 1. Optimization time.

As seen in Fig. 1, GeoLoc method has considerably higher computational complexity. This is because it uses a much larger set of candidates including not only

nodes that store data, but also nearest nodes, which firstly complicates the search and secondly allows to use a higher degree of parallelism.

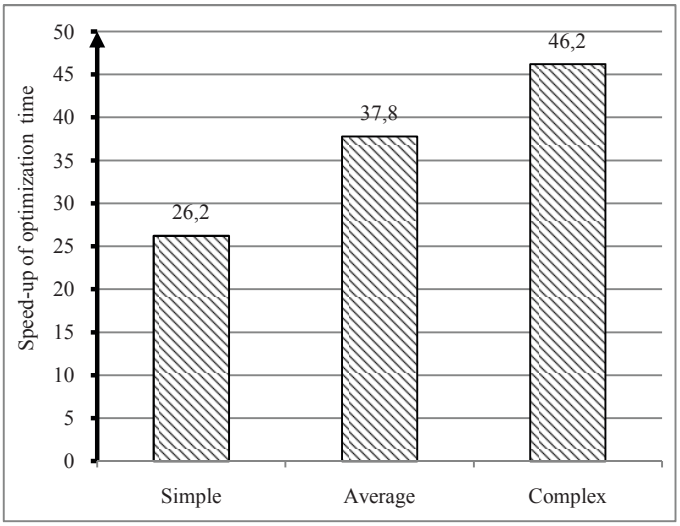

Figure 2. Speed-up of optimization time.

Fig. 2 shows a slight increase in the speed-up between the methods with increasing complexity of the query. We can conclude that for queries of all complexities the method of GeoLoc requires significantly more time in the optimization phase.

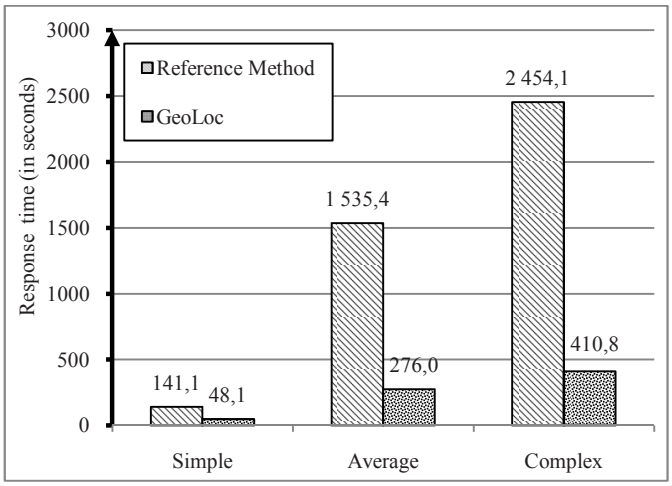

Figure 3. Response time.

As seen in Fig. 3, the method of GeoLoc has significantly less response time than the reference method. During the experiment we found out that our algorithm uses much higher (by order of magnitude) degree of intra-operation parallelism then the

reference method because of larger set of candidate nodes. This is the main reason of its advantage in response time. Another reason is the using of nearest nodes for join operation placement that provides faster data transmission.

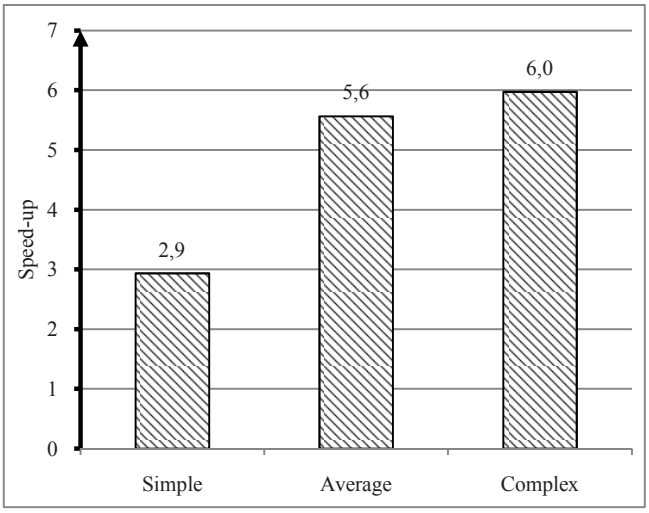

Figure 4. Speed-up.

Fig. 4 shows the increase of the speed-up between the methods with increasing complexity of the query. For queries of any complexity, our method shows a significant advantage over the reference method by the response time.

5. Conclusion

In the present study we proposed the GeoLoc method that consists of two parts. First part is an algorithm for determining the AS based on geographic proximity to the source nodes of relations. Second part is an algorithm of execution plan generation that uses our proposed parallelism degree determining criterion. In the experiment our method shows a significant advantage over the classical method for performance evaluation in terms of response time. However the query optimization time increased tenfold. Nevertheless, we believe that the level of complexity of the GeoLoc algorithm is acceptable for use in a large-scale Data Grid.

In this paper we have dealt with two of the three [11] characteristics of Data Grid: a large scale and heterogeneous nodes. In the future we will extend the proposed method for resolving the third significant problem of Data Grid – dynamicity.

References

[1] Huajun Chen and Zhaohui Wu. Dartgrid III: A semantic grid toolkit for data integration. In Proceedings of the First International Conference on Semantics, Knowledge and Grid, SKG ’05, pages 12–, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Huajun Chen, Zhaohui Wu, Yuxin Mao, and Guozhou Zheng. Dartgrid: a semantic infrastructure for building database grid applications: Research articles. Concurr. Comput. : Pract. Exper., 18:1811–1828, 2006.
[3] Rogério Lus de Carvalho Costa and Pedro Furtado. Scheduling in grid databases. Ín Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops, pages 696–701, Washington, DC, USA, 2008. IEEE Computer Society.
[4] Rogerio Luis de Carvalho Costa and Pedro Furtado. Runtime estimations, reputation and elections for top performing distributed query scheduling. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, pp. 28–35, Washington, USA, 2009. IEEE.
[5] V. F. V. Da Silva, M. L. Dutra, F. Porto, B. Schulze, A. C. Barbosa, and J. C. de Oliveira. An adaptive parallel query processing middleware for the grid. Concurrency and Computation: Practice and Experience, 18(6):621–634, 2006.
[6] Igor Epimakhov, Abdelkader Hameurlain, Tharam Dillon, and Franck Morvan. Resource scheduling methods for query optimization in data grid systems. In Johann Eder, Maria Bielikova, and A Tjoa, editors, Advances in Databases and Information Systems, volume 6909 of Lecture Notes in Computer Science, pages 185–199. Springer Berlin / Heidelberg, 2011.
[7] Anastasios Gounaris, Norman W. Paton, Rizos Sakellariou, Alvaro A. A. Fern, Jim Smith, and Paul Watson. Modular adaptive query processing for service-based grids, 2007.
[8] Anastasios Gounaris, Norman W. Paton, Rizos Sakellariou, and Alvaro A. A. Fernandes. Adaptive query processing and the grid: Opportunities and challenges. In DEXA Workshops, pages 506–510, 2004.
[9] Anastasios Gounaris, Norman W. Paton, Rizos Sakellariou, Alvaro A. A. Fernandes, Jim Smith, and Paul Watson. Practical adaptation to changing resources in grid query processing. In Proceedings of the 22nd International Conference on Data Engineering, ICDE '06, pages 165–, Washington, DC, USA, 2006. IEEE
[10] Anastasios Gounaris, Rizos Sakellariou, Norman W. Paton, and Alvaro A. A. Fernandes. Resource scheduling for parallel query processing on computational grids. In GRID, pages 396–401, 2004.
[11] A. Hameurlain, F. Morvan, and M. El Samad. Large scale data management in grid systems: a survey. In Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008, pp 1 –6.
[12] Oscar H. Ibarra and Chul E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. J. ACM, 24(2):280–289, April 1977.
[13] Hesam Izakian, Ajith Abraham, and Behrouz Tork Ladani. An auction method for resource allocation in computational grids. Future Gener. Comput. Syst., 26:228–235, February 2010.
[14] Hesam Izakian, Ajith Abraham, and Václav Snásel. Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. In CSO (1), pages 8–12, 2009.
[15] Congfeng Jiang, Cheng Wang, Xiaohu Liu, and Yinghui Zhao. A survey of job scheduling in grids. In Proceedings of the joint 9th Asia-Pacific web and 8th international conference on Advances in data and web management, APWeb/WAIM'07, p. 419–427, Berlin, Heidelberg, 2007. Springer-Verlag.
[16] Shuo Liu and Hassan A. Karimi. Grid query optimizer to improve query processing in grids. Future Gener. Comput. Syst., 24:342–353, May 2008.
[17] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.
[18] Xiao Qin. Design and analysis of a load balancing strategy in data grids. FGCS, 23:132–137, 2007.
[19] Khin Mar Soe, Aye Aye Nwe, Than Nwe Aung, Thinn Thu Naing, and Ni Lar Thein. Efficient scheduling of resources for parallel query processing on grid-based architecture. In Information and Telecommunication Technologies, 2005. APSITT 2005 Proceedings. pp. 276 –281, 2005.
[20] Michael Stonebraker, Paul M. Aoki, Witold Litwin, Avi Pfeffer, Adam Sah, Jeff Sidell, Carl Staelin, and Andrew Yu. Mariposa: a wide-area distributed database system. The VLDB Journal, 5:048–063, 1996.
[21] Srikumar Venugopal and Rajkumar Buyya. An scp-based heuristic approach for scheduling distributed data-intensive applications on global grids. J. Parallel Distrib. Comput., 68:471–487, April 2008.
[22] Gerhard J. Woeginger. Exact algorithms for np-hard problems: A survey. In Combinatorial optimization - Eureka, you shrink!, pages 185–207. Springer-Verlag New York, Inc., 2003.
[23] Zhaohui Wu, Huajun Chen, Changhuang Changhuang, Guozhou Zheng, and Jiefeng Xu. Dartgrid: Semantic-based database grid. In International Conference on Computational Science, pages 59–66, 2004.
[24] Lijuan Xiao, Yanmin Zhu, L.M. Ni, and Zhiwei Xu. Incentive-based scheduling for market-like computational grids. Parallel and Distributed Systems, IEEE Transactions on, 19(7):903 –913, 2008.