Query Optimization in Grid Databases Using with Particle Swarm Optimization

Mahdi Mahjour-Bonab[1] and Javad Sohafi-Bonab[2]

[1]Sama technical and vocatinal training college , Islamic Azad University, Bonab Branch , Bonab , Iran

[2]Department of Computer Engineering, Islamic Azad University, Bonab Branch , Bonab , Iran

Abstract

Query Optimization is one of fundamental problems in grid databases. Especially, when the databases are replicated and stored in different nodes of the network. with regard to the point that query in grid databases can be processed in different sites, The problem of choosing suitable sites to execute query is very important. In this article, to choose the sites particle swarm optimization algorithm has been used. To this purpose one function has been used as fitness function in a way that it takes into account the required memory to execute a certain query. Also, the time needed to execute a query and the cost to do so or both of them have been taken into account to perform a query suitably in certain site which is effective in allocating the site to perform a certain query.

In this article different repetitions on different particles with regard to cost and time needed to execute a query in different sites have been conducted and the simulation results have been compared with each other.

Keywords: *Query optimization, Grid database, Particle swarm Optimization algorithm, Distributed query processing.*

1. Introduction

Query optimization is the process of choosing the most efficient query evaluation plan from among the many strategies usually possible for processing a certain query, especially if the query is complex [1].

Grid database is a new research field which combines those techniques of database with grid which the traditional database queries and their optimization techniques cannot meet the needs of grid databases due to the dynamic features of the grid nodes [2]. The architecture of grid database generally works in an extended environment, distributed multi-institutions, and heterogeneous and autonomous environments. Applications such as earth simulation, weather forecasting, study of global warming, and other collaborative works need to access the data from different sites [3].

Fernandez Beca indicated that the optimized access of the tasks to resources in distributed system is np-complete [4]. By development of Globus Toolkit and GGF and OGSA-DAI standards for grid databases, a lot of research have centered on processing of the query in grid databases [5], For example, Polar [6] and OGSA-DQP[7] are two famous projects which have centered on the processing of query in grid databases.

GrADS[8] and Condor[9] develop the query plan on the basis of Directed Acyclic Graph (DAG). The reference [9] has proposed min-min algorithm for sub query scheduling in query plan. Random sampling algorithm, nearest neighbor, recording the farsthest[10], genetic algorithm[10,11], deterministic search algorithm, iteration improvement[11], simulated annealing[10], two-stage optimization[10], A^* algorithm[12], improved A^* algorithm[12] have been used to optimize the queries.

Particle swarm optimization is an optimization technique which acts on the basis of swarm of early responses. This technique was designed at first by Eberhart and Kennedy in 1995 on the basis of social behavior of flocks of birds and schools of fish [13].

This technique acts like the evolutional computing techniques like genetic algorithm in most cases. In this method also the system starts with a number of early responses and tries to find the optimized response by moving the early responses in successive repetitions. Unlike the genetic algorithms, PSO do not have the evolutionary Functions such as mutation and crossover. The responses of the problem in PSO, are to find the results of the system optimization in research environment.

The remainder of this paper is organized as follows. section 2 formulates the problem. In section 3 the optimized solution on the basis of particle swarm algorithm for query optimization problems, and the fitness function used are explained. section 4 presents simulation of the grid database and the parameter values are illustrated in section 5. In section 6 results of the experiments are explained. Finally, section 7 concludes the paper.

2. Problem formulation

Suppose that a grid system has R different resources in a way that computational capacities of different resources can be shared by users. Each user expects to have his/her work completed by grid system. Each task is composed of N parallel tasks which is equivalent to the relationship in a grid query, Which can be performed in a parallel way. These independent relationship cannot be decomposed and can be allocated only to one site or host. The resources in grid system are heterogeneous and each can have different processor or memory. Therefore if the relationships are executed in different grid resources, they will have different execution cost and time. Due to lack of data dependency among the parallel tasks for execution, the total execution time of the query will be equal to the maximum execution time of the parallel tasks.
The notations used in formulating the resource allocation problem are listed as follows. In our problem, n job is the n relation of a certain query and resource r of the same site or host is the number of r.
u_r^n: (Decision variable), $u_r^n = 1$ represents task n allocated to resource r for execution; otherwise $u_r^n = 0$
t_r^n: execution time of job n on resource r.
e_r^n: execution cost of job n on resource r.
N: the number of grid tasks.
R: the number of grid resources.
l^n: the length of task n in the grid.
m^n: the required memory for task n in the grid.
m^n: processing capacity of resource r in the grid.
M_r: the memory capacity of resource r in the grid.
C_r: processing cost of the resource r of the grid (Per time unit or capacity unit).
To minimize the total execution time and cost of parallel tasks, each task shows its cost and execution time in function with certain parameters and variables. Since the capacity of the obtained part for each task is in proportion with the tasks in certain resource is the same. The execution time of task n on resource is equal to:

$$t_r^n = u_r^n l^n / P_r \qquad (1)$$

and the execution cost of the task n on resource r is equal to:

$$e_r^n = u_r^n \, l^n C_r \qquad (2)$$

To compute the value of C_r, the distance of each site is divided into 5^{10} and this is for the speed of optical fibers, and then due to commuting of the information, the obtained value from the division is multiplied by 2 to take into account both the cost of going and cost of returning.
The users need a useful model to let them determine the resource and parameter needs with priority. The user wants to minimize both the execution time and cost. To maximize the utility of users, we should certain the value of this utility. For this purpose, We select the Cobb- Douglas utility function, because it reflects the tradeoff between various variables of the model fairly well and is frequently used to express an individual's utility. The utility for a user is a balance between the cost and the time of completing a job. Therefore we want to minimize the following two values:
Time: the required time to be completed.
Cost: the required cost to be completed.
The utility function of Cobb- Douglas is as following:

$$U(Time, Cost) = (Time)^{\theta}(Cost)^{1-\theta} \qquad (3)$$

where *Time* and *Cost* are the quantities consumed and θ is a real number between [0,1] which describes an individualistic preference for the time related to cost. The utility function shown in this method provides a diminishing marginal utility for us. A utility function of Cobb- Douglas for increasing utility is generally on the basis of quantities assumptions. But since we are to minimize both the time and cost of executing a task, we introduce the negative utility form of Cobb-Douglas. A negative utility function can be as following (4):

$$U = \theta \ln(Time) + (1-\theta)\ln(Cost) \qquad (4)$$

Since different tasks on common resource get the computational capacity proportional to their length, the total execution time of the tasks in the common resource r is t_r^n. suppose that the tasks in different resources start at the same time, therefore the job execution time is the maximum execution time of the tasks, $max_{n=1}^{N} \sum_{r=1}^{R} t_r^n$. By replacing $max_{n=1}^{N} \sum_{r=1}^{R} t_r^n$ and $\sum_{n=1}^{N} \sum_{r=1}^{R} e_r^n$ for Time and Cost the equation (4) is written in this form (5):

$$U^i = \theta \ln\left(max_{n=1}^{N} \sum_{r=1}^{R} t_r^n\right) + (1-\theta) * \ln\left(\sum_{n=1}^{N}\sum_{r=1}^{R} e_r^n\right) \qquad (5)$$

The result obtained for allocation of the resources for Resource R and N task is $\{u_r^n\}_{1 \le n \le N, 1 \le r \le R}$ .this optimization problem for grid resource allocation can be formulated as following:

$$\begin{cases} Min\ U = \theta \ln\left(max_{n=1}^{N} \sum_{r=1}^{R} t_r^n\right) + \\ (1-\theta)\ln\left(\sum_{n=1}^{N}\sum_{r=1}^{R} e_r^n\right) \\ s.t.\quad \sum_{r=1}^{R} u_r^n = 1, \qquad \forall n = 1,2,\dots,N \\ \sum_{n=1}^{N} u_r^n m^n \le M_r, \qquad \forall r = 1,2,\dots,R \\ u_r^n \in \{0,1\}, \qquad \forall n,r \end{cases} \quad (6)$$

The objective function in optimal problem (6) shows time utility and the cost of task execution. The first constraint means that each task can only be allocated only to one resource. The second constraint means that the resource memory capacity, satisfy the memory required for all the tasks on that resource, and the third one means that u_r^n is a binary decision variable [14].

3. the optimal solution on the basis of particle swarm algorithm for query optimization problem

In this part, the method is described and coding of the particles for this problem and the method of this description and calculation of fitness function and the operations conducted for this algorithm are talked about.

Table 1: An example of particle coding

	T1	T2	T3	T4
Allocation=	(r2	r1	r3	r2)
x_i =	(2	1	3	2)

Corresponding with particle coding in Table 1, the optimization problem (6) for grid resource allocation can be rewritten as following. First, a value transfer function sign(x) is described as following:

$$sign(x) = \begin{cases} 1, & if \quad x > 0, \\ 0, & if \quad x = 0, \\ -1, & if \quad x < 0. \end{cases} \quad (7)$$

Now, the decision variable u_r^n can be changed into equation (8) using sign(x):

$$u_r^n = \left||sign(x_{in} - r)| - 1\right| \quad (8)$$

and similarly, time execution of task n in grid on resource r (equation (1)) can be changed into equation (9):

$$t_r^n = \frac{l^n \left||sign(x_{in} - r)| - 1\right|}{P_r} \quad (9)$$

and the execution cost of the task n in the grid on resource r (equation (2)) can be changed into (10).

$$e_r^n = l^n C_r \left||sign(x_{in} - r)| - 1\right| \quad (10)$$

Now, the optimal problem which have been shown under the condition of (equation (6)) for gird resource allocation on the basis of particle can be changed into (11) [14].

3.1. coding of the particles

In PSO, each individual of the population shows potential solution for the optimization problem. In which, each individual has two characteristics:
Position: which determine the function value corresponding with the particle position.
velocity: on the bans of velocity, the particle moves in the search space.
One of the notable characteristics of the PSO is that it needs less parameters to adjust the needs. Any way these parameters have great effect on the efficiency and precision of the algorithm.
In our optimization problem, each particle shows a possible solution for allocation optimization. Since, there are N task in grid work; optimal allocation can be expressed as a N-dimensional vector as $X_i = (x_{i1}, x_{i2}, \dots, x_{iN})$, In a way that each part of x_{ij} shows that j-th task has been allocated for execution to resource x_{ij}. for example, 4 tasks have been allocated for 3 resource respectively. In table 1, we see that $x_{i1} = 2, x_{i2} = 1, x_{i3} = 3, x_{i4} = 2$ have been shown in the allocation by the i-th particle, $x_{i1} = 2$ Means that the task 1 has been allocated for resource 2 for execution.

$$\begin{cases} Min\ U = \theta\ ln\left(max_{n=1}^{N} \sum_{r=1}^{R} t_r^n\right) + \\ (1-\theta) ln\left(\sum_{n=1}^{N}\sum_{r=1}^{R} e_r^n\right) \\ s.t. \sum_{n=1}^{N} m^n \left||sign(x_{in} - r)| - 1\right| \le M_r, \\ \forall r = 1,2,\dots,R \end{cases} \quad (11)$$

3.2 fitness function for allocation problem optimization.

In our PSO algorithm, each particle shows a resource allocation search for optimal position, in a possible combined space from resource allocation. There is a fitness function for each particle to evaluate the allocation scenario. Therefore the particle shows that the best fitness function is optimal allocation. Along the repetition process, the particle moves to the best position obtained previously. And also it accelerates toward the best particle in to topological neighborhood. Therefore, through modification and adjusting of those position, each particle can fly in the combined space of the allocated resource, and at last gets to an optimal solution. Clearly, the goal function from problem (11) can be uses to evaluate the particles. Anyway, algorithm using objective function as fitness may produce such result that cannot satisfy the memory constraint. Therefore an effective Rockafellar multiplier method is used to change the optimal problem with inequality constraint for an optimal problem with out limitation. The fitness function of final particle is equal to:

$$f(x) = \theta\, ln\left(max_{n=1}^{N} \sum_{r=1}^{R} t_r^n \right) + (1-\theta) ln \left(\sum_{n=1}^{N} \sum_{r=1}^{R} e_r^n\right) + \frac{1}{2c} \sum_{r=1}^{R} \left\{ \left[min\left(0, \mu_r + c\left(M_r - \sum_{n=1}^{N} m^n \left| |sign(x_{in} - r)| - 1 \right| \right)\right)\right]^2 - \mu_r^2 \right\} \qquad (12)$$

fThe μ_r coefficient is produced with the following repetition equation (13) [14]:

$$\mu_r^{(k+1)} = min\ (0, \mu_r^{(k)} + cg_r(x^{(k)}) \qquad (13)$$

3-3 PSO operation to optimize the allocation problem

PSO operation is as following. Each particle shows one possible solution of optimization problem in a N – dimensional space (N- number of tasks). Suppose that $X_i = (x_{i1}, x_{i2}, \dots, x_{iN})$ is the position of the i-th visited particle. lbest is the personal best position (local) found by the i-th particle. And gbest is the best global position discovered by each particle in social and $V_i = (v_{i1}, v_{i2}, \dots, v_{iN})$ is the velocity of the particle. During each repetition, each particle accelerates in the direction of lBest and similarly well in the direction of gbest. Suppose that $X_i(t)$ apply to the current position in search space, and $V_i(t)$ is the current velocity and the velocity of each particle in swarm is updated by using the following equations:

$$V_i(t+1) = \omega . V_i(t) + \sigma_1 . rand. \left(lBest(t) - X_i(t)\right) + \sigma_2 . rand. \left(gBest(t) - X_i(t)\right) \qquad (14)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \qquad (15)$$

As rand is used to maintain the population diversity and is distributed uniformly in the range of [0, 1] and parameters σ_1 and σ_2 are the acceleration coefficients, the proper values for these two acceleration coefficients can make the convergence results faster and reduce getting to local minimum. ω is the inertia coefficient whose value is $0.1 \le \omega \le 0.9$. A proper value for the weight ω usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution. The value of $V_i(t)$ can vary in the range of $[V_{min}, V_{max}]$ to have one constraint to control the global exploration capacity of the PSO algorithm. The range described for also reduces the likelihood of the leaving the search space by particle. Note that this is not a limitation for value of $X_i(t+1)$ in the range of $[V_{min}, V_{max}]$; but only determines the maximum distance that a particle can move during a repetition. The best local allocation for each particle is updated with the following equation:

$$lBest_i(t+1) = \begin{cases} lBest_i(t) & if \ \ f\left(X_i(t)\right) > f\left(lBest_i(t)\right) \\ X_i(t+1) & if \ \ f\left(X_i(t)\right) \le f\left(lBest_i(t)\right) \end{cases} \qquad (16)$$

Function f is the fitness function which is used to adjust the updating. If the new position is better, it allocates that to lBest, otherwise it remains in its previous position [14]. The best global position, gbest, is the best position among all particles in the society during the previous stages, that is:

$$gBest_i(t+1) = \arg max_i \ f\left(lBest_i(t+1)\right) \quad \forall i \in N \qquad (17)$$

Generally, the PSO algorithm terminates when the condition of maximum iteration is satisfied.

4. simulation of grid database

In this experiment we use the simulated data for AReNA Project[5] form PlanetLab [12] which simulates a grid environment on the basis of real data. These data are simulate for a grid database due to the variety in network communication, different geographical positions, dynamism and heterogenetty of individual sites. There are 12 hosts in this grid environment which are distributed in Asia, Europe and North America. The characteristics and hardware configuration of each host have been shown in table (2). Four databases with multiple relationships have been simulated in table (3). There are several copies from each database which have been distributed in different sites and have been shown in table (4). The distance of each host of certain query has been shown in table (5) [17]. In general case, we suppose that query Q include the $R_1, R_2, \dots, R_n$ relationships and the relationship R_i is in a certain database of which there are M_i copies located in the hosts $H_{i1}, H_{i2}, \dots, H_{im}$ .In the above simulation example each query includes five relationship and the number of host are twelve which are indicated with notations as Host1 to Host12 [14].

Table 2: Configurations of the hosts in the simulated grid environment

Host	RAM(MB)	MIPS	Sector Size (Byte)
Host1	512	654	512
Host2	128	409	512
Host3	256	409	512
Host4	1024	837	512
Host5	512	613	512
Host6	1024	613	512
Host7	128	327	512
Host8	1024	1674	512
Host9	2048	1674	512
Host10	1024	736	512
Host11	1024	837	512
Host12	512	613	512

Table 3: Simulated database and relationship

Database	Relation	#_Records	Size(MB)
DB1	R1 R3	550000 3000000	100.2 650
DB2	R2	100000	150.5
DB3	R4	50000	90.7
DB4	R5	300000	120.8

Table 4: Simulated database replicas

Database	Replica Hosts
DB1	Host1, Host2, Host3
DB2	Host4, Host5, Host6, Host7
DB3	Host8, Host9
DB4	Host10, Host11, Host12

Table 5: Distance of each host from certain query

Host	Distance	Host	Distance	Host	Distance	Host	Distance
Host1	5000000	Host4	4000000	Host7	3000000	Host10	5000000
Host2	12000000	Host5	7000000	Host8	13000000	Host11	8000000
Host3	2000000	Host6	6000000	Host9	9000000	Host12	6000000

5-Parameter values and algorithm configuration

We implemented the PSO algorithm by using the VB6.0 and the described parameters for simulation are as Table (6).

Table 6: Experimental parameters for PSO

Number	Parameter	Value
1	ω	[0.1, 0.9]
2	Dimension	5
3	$[X_{min}, X_{max}]$	[1, 100]
4		[1, 10]
5		2
6	θ	[0, 1]
7		0
8	k	100
9	c	4

We conducted our experiments on PSO algorithm with particle numbers of 30, 40, 50, 60, 70 and 80 with the maximum repetition of 100, 200, 300, 400 and 500 in which the value was considered to be 0, 0.5 and 1. it is necessary to mention that each experiment was conducted independently for 20 times and average value obtained for fitness function was considered as the value of fitness function for that experiment.

6. Experiments result

The experimental results are as following:

- the value of fitness function for different particle with different repetition number.

The figures (1) to (5) indicate the experiment results can particles with repetitions of 100, 200, 300, 400 and 500. In these figures the θ value has been considered as 0, 0.5 and 1. As the figure (1), (2) and (3) show in the repetitions of 100, 200 and 300 by increasing the particle numbers to different θ′s ,the fitness function reaches a relative balance but with lower numbers of particles, the fitness function value has higher Fluctuation for different θ's. But as the figures (4) and (5) show for the repetitions of 400 and 500 the fitness function value yields a better balance for different θ′s and the fluctuation decreases Notably. Therefore it can be concluded that there is a direct relationship between the number of particles and the number of different repetition in which by of repetitions, the fitness function value gets lower and this is a desired result for us.

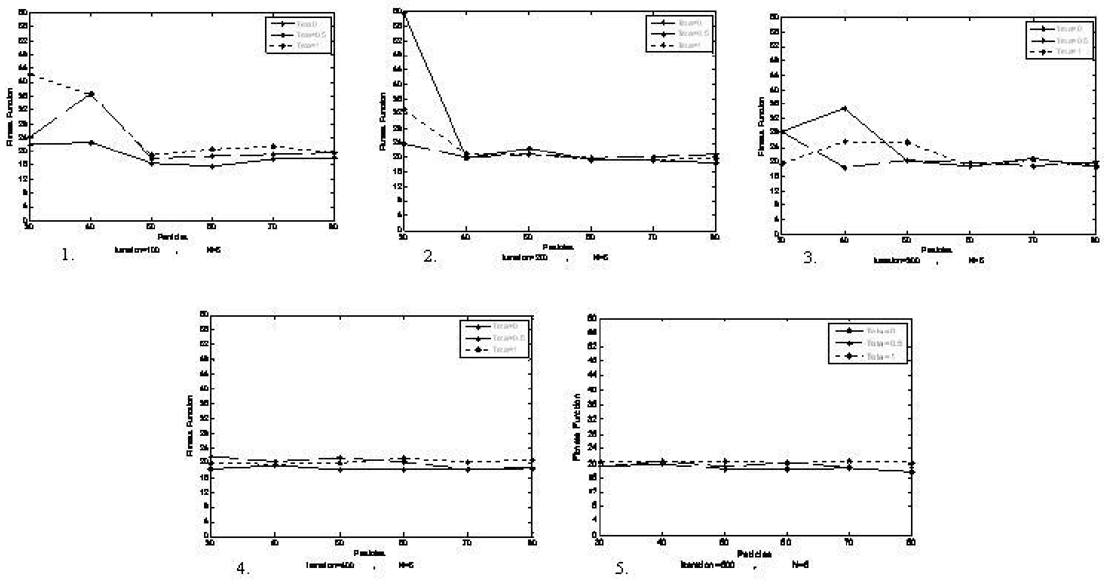

Fig 1 to 5. the value of fitness function for different particle with different repetition number.

• fitness function value for different repetitions with different particles.

The figures (6) to (11) indicate the experiment results for number of repetitions with particle numbers of 30, 40, 50, 60, 70 and 80. In these figures the value of θ has been considered to be 0, 0.5, and 1.

As it has been indicated in figures (6) and (7) the fitness function value for particles 30 and 40 up to repetitions of 400 has more fluctuation but beyond 400 repetition it reaches a good balance. In figure (8) with the particle number of 50 to particles 30 and 40, the fitness function value has improved and show less fluctuation but figures(9), (10) and (11) have good balance for all particles, especially for particles 80, this fluctuation has if the particle number is increased to a certain amount, the fitness function value minimizes. Of course it must be noted that for particles 30 and 40 the increase of the number of repetition pushes the fitness function value to its minimum value.

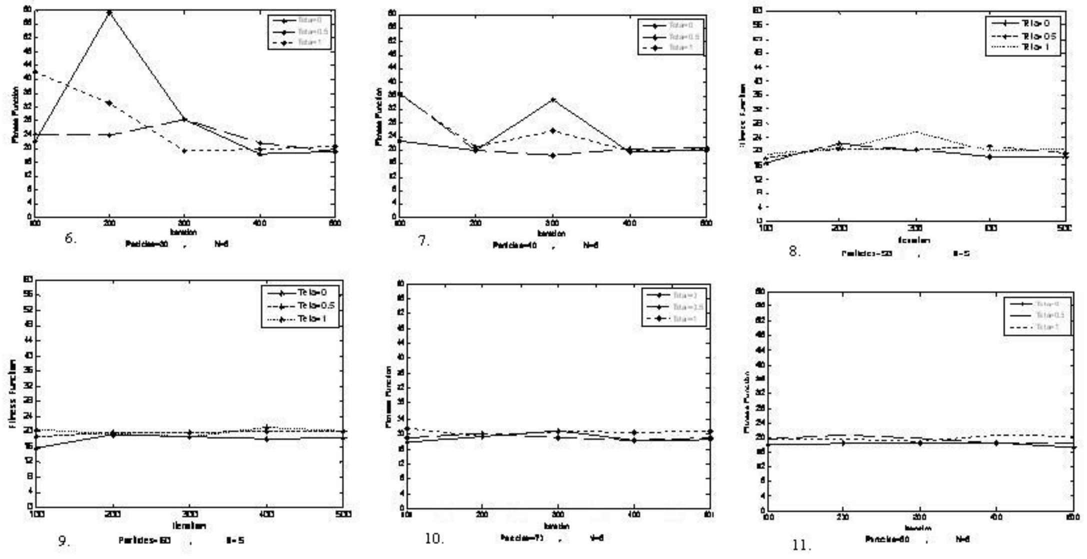

Fig 6 to 11. fitness function value for different repetitions with different particles.

7. conclusion and future works

particle swarm optimization is one of intelligent methods of optimization problems. In this paper, we used the particle swarm optimization algorithm for optimizing the queries for grid databases and the results show that this algorithm yield the best result for the particles 70 and 80 with maximum repetitions of 500.
In future, the processing capacity of queries with larger value will be very important. Therefore, more research about this algorithm and improving them is necessary. Due to new applications for database (inductive database systems, database systems producing front end, and complex views), the research are constantly increasing and harsh activities are done to improve the algorithms of traditional relational database queries. Of course it seems that, in future intelligent agents will be used to optimize the queries.

8. References

[1] A. Silberschatz, H. Korth, and S. Sudarshan, *Database System Concepts*, 5th edition, Mc Graw Hill, 2006.
[2] H. Naijing, W. Yingying, and Z. Liang, “Dynamic Optimization of Subquery Processing in Grid Database”, Third International Conference on Natural Computation, 2007.
[3] D. Taniar, C. H. C. Leung, W. Rahayu, and Goel.S, *High-Performance Parallel Database Processing and Grid Database*, John Wiley & Sons, New Jersy, 2008.
[4] D. Fernandez-Beca, “Allocating Models to Processors in a Distributed System.”, IEEE Trans, Software Eng 15 ,1989, PP.1427-1436.
[5] S. Wang and K. Zhang, “Grid Database System in Grid” www.chinaghrid.net. 2005.
[6] J. Smith, S. Sampaio, and D. Watson, “The Design, Implementation and Evaluation of an Odmg Compliant, Parallel Object Database Server”. Distributed and Parallel Database, Vol.16, No.3, 2004, PP.275-319.
[7] M. N. Alpdemir, A. Mukherjee, and N. W. Paton, “Servicebased Distributed Querying on the Grid”. ICSOC 2003, First International Conference, Trento, Italy, 2003, PP.467-482.
[8] N. Bruno, and S. Chaudhuri, “Exploiting Statistics on Query Expressions for Optimization”. In SIGMOD 2002, Proceedings ACM SIGMOD International Conference on Management of Data, June 3-6 2002.
[9] S. Shivle, and H. J. Siege, “Mapping of Subtasks with Multiple Versions in a Heterogeneous Ad Hoc Grid Environment” Third International Symposium on Parallel and distributed computing, 2004, PP.380-387.
[10] Z. Zhou, “Using Heuristic and Genetic Algorithms for Large-Scale Database Query Optimization.”, Journal of Information and Computing Science, Vol. 2, No. 4, 2007, PP.261-280.
[11] J. Wang, J. Horng, Y. Hsu, and B. Liu, “A Genetic Algorithm for Set Query Optimization in Distributed Database Systems.”, IEEE, 1996, PP.1977-1982.
[12] A. Goyal, L. Vasiliu, and B. Sapkota, “Use of AI Query Optimization of Relational Database.”, in 18th IEEE International Conference on tools with Artificial Intelligence (ICTAI06), 2006.
[13] J. Kennedy, and Eberhart, *Swarm Intelligence*, Morgan Kaufman, 2001.
[14] Z. J. Li, X. D. Liu,X. D. Duan, and C. R. Wang, “Optimal Solution for Grid Resource Allocation Usig Particle Swarm Optimization”, Third International Conference on Multimedia and Ubiquitous Engineering, 2009, PP.339-346.
[15] V. Zadorozhny, A. Gal, and L. Raschid, “AreNA: Adaptive Distributed Catalog Infrastructure Based on Relevence Network.”, In the 31st VLDB Conference, Trondheim, Norway, 2005.
[16] PlanetLab, PlanetLab Platform, http://www.planet-lab.org.
[17] L. Shuo, and H. K. Karimi, “Grid Query Optimizer to Improve Query Processing in Grids”, Future Generation Computer Systems, Vol.24, 2008, PP.342-353.
[18] M. Lovbjerg, “Improving Particle Swarm optimization by Hybridization of Stochastic Search Heuristics and Self-Organized Criticality”, M.S. thesis, supervisor: T.Krink, university of Aarhus, May 2002.
[19] M. Clerc, and J. kennedy, "The Particle Swarm Explosion, Stability and Convergence in A Multidimensional Complex Space", in IEEE Transaction on Evolutionary computation, Vol.6, Feb 2002, PP.58-73.
[20] J. Riget, and S. Vesterstrom, “A Diversity-Guided Particle Swarm Optimizer The ARPSO”, in journal of Evalife Technical Report, Vol.2, 2002.
[21] Y. Shi, and R. Eberhart, “Empirical Study of Particle Swarm Optimization”, In Proceedings of Congress on Evolutionary Computation, 1999, PP.1945-1950.
[22] Z. Jirong, “A Modified Particle Swarm Optimization Algorithm”, Journal of Computers, Vol.4, No.12, 2009, PP.1231-1236.
[23] C. Shu_Chuan, C. Jui_Fang, and P. Jui_fang, “A Parallel Particle Swarm Optimization Algorithm With Communication Strategies”.

[24] L. Chuan, and F. Quanyuan, “A Hierarchical Subpopulation Particle Swarm Optimization Algorithm”.

[25] P. Y. Yin, Y. Shiuh-Sheg, P. P. Wang, and Y. T. Wang, “Task Allocation for Maximizing Reliability of a Distributed System Using Hybrid Particle Swarm Optimization.”, The Journal of System and Software, Vol.8, 2007, PP.724-735.