Egyptian Informatics Journal (2014) **15**, 51–58

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

Evaluation of DFTDS algorithm for distributed data warehouse

S. Krishnaveni, M. Hemalatha *

Department of Computer Science, Karpagam University, Coimbatore, India

Received 10 April 2013; revised 27 August 2013; accepted 27 October 2013
Available online 28 November 2013

KEYWORDS

Data warehouse;
DTDS algorithm;
VMFTRS algorithm;
WINE algorithm;
DFTDS algorithm

Abstract The distributed data warehouse is mainly based on how the data are being processed and distinguish between dynamic and physically distributed systems. Finding the relevant information from a huge database is a very laborious process and consumes more time. This conflict is addressed using a query scheduling process in the distributed data warehouse is very compact to accomplish these tasks within a few seconds. In this paper, Dynamic Fault Tolerant Dependency Scheduling (DFTDS) algorithm has been proposed to schedule the queries based on their dependency and it automatically allocates the resources by checking the status of the virtual machine based on its acknowledgment. That is, reply to client/user queries in distributed data warehouse systems. Demonstration of the proposed DFTDS algorithm shows a significant reduction in query processing time and memory utilization compared to existing algorithms.

1. Introduction

A distributed system is an interconnected collection of heterogeneous computers connected by a local or wide area network. As the amount of data and number of sites grows, often distributed system becomes crucial for scheduling the queries. Hence it is essential to adopt gird based task and resource scheduling algorithms to resolve these issues in distributed data warehouses. Computing the huge volume of data is a significant task now a day. The existing scheduling algorithms are not possible to retrieve the exact information from the single repositories. In order to get the exact information from a distributed data warehouse and share the large number of data from various repositories, a new algorithm based on grid scheduling has been proposed.

In Local Area Networks (LAN), computers are fully connected with the various physical machines (PM) or host above the PM, virtual machine (VM) comes into distributed resources. Most of the VM's are developed and configured with different properties like RAM, storage, CPU, etc. In some set of circumstances one finds oneself. Several resources remain unused, because most of the system resources are available to the system owners only. An unused or untapped computing power of the distributed system can be utilized to execute various tasks assigned by the users without using any additional

* Corresponding author. Tel.: +91 9659079670.
E-mail addresses: sss.veni@gmail.com (S. Krishnaveni), csresearchhema@gmail.com (M. Hemalatha).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

1110-8665 © 2013 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University.
http://dx.doi.org/10.1016/j.eij.2013.10.002

infrastructure devices. In a LAN, a virtual framework comes into distributed resources. For example, a user can provide certain techniques to access the resources, without compromising the system security and without disturbing operations of the actual owner of the system. Most of the broad distributed and autonomous resources could also be utilized within the execution of a selected query. PM or VM failures may not be likely but also pricey. It is then higher to tolerate the fault instead of discarding the work done already unless the resources needed for completion don't seem to be offered.

Rollback-recovery mechanism [1], which is added in the fault-tolerance in distributed query processing. For read-only queries the perceived knowledge has been that support for fault-tolerance is just too high-ticket to be worthy. The high level expression of user requests during a physical pure mathematics others opportunity for standardization the fault-tolerance provision therefore on scale back the price, and provides higher performance than the employment of generic fault-tolerance mechanisms at very cheap level of question process.

In this paper, a new Dynamic Fault Tolerant Dependency Scheduling (DFTDS) algorithm has been proposed to solve the issues in the distributed data warehouse system. The proposed system integrates our previously proposed Dynamic Task Dependency Scheduling (DTDS) and Virtual Machine Fault Tolerant Scheduling (VMFTRS) algorithms [2–4] for scheduling queries based on their dependency as well as recycling resources without human intervention in case of any fault occurred in the virtual machine. This proposed DFTDS algorithm is an online, non preemptive type. Meaning that, the user given queries' estimated execution time does not know in advance and we do not suspend any queries when they are in the process of being executed. The fortifications are then evaluated through six parameters. A comparison of the proposed algorithm with WINE, DTDS and VMFTRS algorithms has been performed. Workload balancing by election (WINE) algorithm works well on clients' demands and provides better quality of service and data [5]. The Comparison results show that significant performance improvements can be gained through recovering and continuing after failure of virtual machines.

2. Preliminary diagnosis

To improve the performance of task scheduling in a distributed data warehouse system three existing grid based algorithms have been used namely, Optimal Resource Constraint (ORC), Grouping-based Fine-grained Job Scheduling (GFJS) and Heuristic Algorithm (HA). ORC algorithm [6] applies Round Robin (RR) and Best Fit algorithms to distribute the tasks for available processors. GFJS algorithm [7,8] is based on resource characteristics also integrated with Greedy and FCFS algorithms to improve the fine-grained jobs. Then group of fine-grained jobs are used to develop the coarse-grained tasks. The grouping strategy proposed by [9], provides a real grid computing environment and reduces the waiting time of the grouped tasks. Grace et al., used a HA in their proposed framework for calculating and minimizing the processing time [10]. HA shows better output for large size problems.

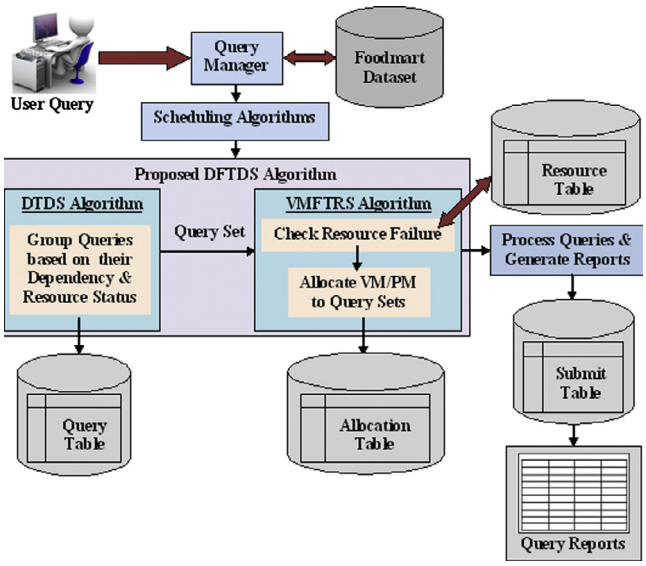

Figure 1 Architecture of proposed DFTDS algorithm.

All these existing algorithms have been implemented in a distributed environment and their processing time, memory utilization and few drawbacks have been identified by the given set of 150 queries in 70 resources (systems or machines) are shown in Table 1. These drawbacks have been addressed in our proposed three new algorithms namely DTDS, VMFTRS and DFTDS.

DTDS algorithm is proposed to address the issues based on query dependency. In DTDS algorithm, initially queries are scheduled and mapped to resources based on queries' dependency and resources' status. The resources are mapped to the queries by the earliest free time of the resource r_i and the earliest start time of the query q_i. If the resource is in running state just skip and search the other resource. If the resource is free

Table 1 Demerits of existing algorithms.

Algorithm	Disadvantage	Processing time (s)	Memory utilization (MB)
Job Scheduling			
ORC	High communication overhead	362	12.583
GFJS	High preprocessing time and memory size constraint inconsiderable	376	12.339
HA	Hinder a way to assess the quality of redesigns	353	12.139
Resource Scheduling			
RNDRM	The resource allocation period is high	333	11.997
NRMNS	Resource provider agents' decline at the time of minimization of resource's cost	325	11.908
VCGRP	Does not show any error information at the time of any failures occurred in queries	308	11.894

check whether all dependent queries are executed, then insert the independent queries based on sorting order until all the independent queries are executed.

To increase the efficiency of resource management the principles obtained from three existing grid based algorithms have been combined and used namely, Research on Novel Dynamic Resource Management (RNDRM), New Resource Mechanism with Negotiate Solution (NRMNS) and Virtual Computing Grid using Resource Pooling (VCGRP). RNDRM algorithm is an exemplar, to integrate agent technology with grid computing based applications [11]. NRMNS algorithm is an agent based resource management system model [12]. It provided a surrogate solution at the time of a resource discovery glitch. VCGRP algorithm [13] is used to exploit the ideal computing power with less effort.

When these algorithms are used few drawbacks are listed in the Table 1 have been identified. To overcome these conflicts previously proposed VMFTRS algorithm is used. The main aim of VMFTRS algorithm is recycling the virtual machines. Primarily allocate queries to resources and check the status of virtual machines. If any virtual machine failed, then sort all other virtual machines based on MIPS (Million Instructions Per Second) availability. If the query size is less than the virtual machine's availability in an array, then queries are allocated to that particular virtual machine. Else evaluate any other virtual machines' availability with query size. If the state is unsatisfied check all possible combinations of virtual machines' or combinations of physical resources' MIPS with query size until all the queries are allocated.

3. Proposed DFTDS algorithm

3.1. Design details

The proposed DFTDS algorithm has been designed in such a way that first queries are grouped based on their dependency. Then grouped queries are allocated to resources based on VMFTRS algorithm followed by VCGRP techniques as shown in Fig. 1. The work flow of the proposed DFTDS algorithm has been illustrated using pseudocode in Fig. 2. Initially user queries are stored in query table. Physical machines (PM) and virtual machines (VM) with their bandwidth details are saved in the resource table. In allocation table, details of allocated machines are stored and resource daemon reads this table to update the resource table. Whenever queries are submitted to PM or VM an entry is made in submit table and query daemon reads the entries from this table to update the query table. All these four tables are maintained in resource distributing monitor.

3.2. Algorithm steps

Step 1: Collect all the queries and sort them according to the size, arrival time and dependency.

Step 2: Initialize the number of VM reconfigurable, number of physical machines and query size.

```
Note:
Q_size = query size, qq = query queue, qs = query status, av_qq = available query queue, av_rs = available resource status, Qi = independent queries, r_cap = resource capacity, VM = virtual machine, RVM = reconfigurable virtual machine, PR = available physical resource, MIPS = million instructions per seconds

    1. Initialize input parameters:
Ti→ number of query, Tj→ arrival time of Ti, r→resource

    2. Sorting queries:
av_qq[ ] = SORT[Ti, Tj]
j = 0;
av_rs = "ready"
for each resource rk in resource group do
while (rk is in running state)
  skip and select the next one

    3. Select resources:
if rk[j] selected
  rkselected[] = rk[j]
  j++;
endif
endwhile
k=0; //for each selected resource

    4. Check query dependency:
if (dependent queries in rkselected[])
      assign av_qq[k] to rkselected[]

    5. Check available resources:
if(r_cap> Q_size)
      submit queries to selected resource
      elseif(r_cap<Q_size)&&(av_rs='terminated')
      create new VM to match the Q_size

    6. Check VM status:
if (VM failure = true) then
      create new VM from RVM
check
if(New VM's capacity(MIPS)>Q_size)
      allocate queries
          else

      6.1. Create VM using single PR to satisfy the
          Q_size
      if(Q_size<avail_(PR))
      create VM
      update VM pool

      6.2. Create VM from combining PR to satisfy
          the Q_size
      //for(i=0;i<=n_PR;i++)
      //for(j=1;j<n_ PR;j++)
            elseif(avail_(PRij)>Q_size)
            create new VMi+VMj
            endif
      endif
      endif
endif
      endif
endif
endif
endfor

    7. Check status of dependent queries:
if qs=executed process Qi
      else
      repeat the process from step 5
endif

    8. Queuing independent queries based on
        priority:
select Qi and repeat the same process from step 5

    9. Check status of all queries:
if(qs = executed && qq = empty)
      then stop the process
endif
```

Figure 2 Pseudocode for proposed DFTDS algorithm.

Table 2 Evaluation metrics.

Performance metric	Description	Formula	Abbreviation
Processing Time (PT)	The difference between the arrival time (t_{qi}) and the execution time (et_{qi}) of given queries	$PT_{qi} = (et_{qi} - t_{qi})$	q = number of queries given by the user i = ith query
Memory utilization (MU)	Variation between the systems' total free memory (Tm) before user given their queries and free memory (Fm) of the system after the execution of all input queries	$(MU_{qi}) = (Tm - Fm)$	MU_{qi} = Memory utilization for qith query
Replication metric (RM)	The number of same queries are repeated in various sites or resources	$RM = \frac{1}{k}[\sum_{k=1}^{l}(\sum_{i=1}^{n}[\sum_{j=1}^{m}(Qi \in Rj)])] * 100$	k = number of cycles executed the same set of queries
Error free execution (EFE)	The average rate of error occurred queries executed over different resources	$EFE = \frac{(Tq-IEq)}{Tq} * 100$ $Eq = (Tq - Aq)$ $IEq = \sum_{k=1}^{m}[(Eq)k/N]$	Tq = total number of queries given by the user Aq = executed queries IEq = iterated error queries
Query cost (QC)	Total processing time taken to execute the queries given by the user	$QC = \frac{1}{k}\sum_{k=1}^{n}\frac{PTqi}{dj}$	dj = data occurred in given queries
Scalability	Average processing time (PT) of the different set of queries is evaluated by the same set of resources	$Scalability = \frac{PT(Qi++[Rj])}{PT(Qi)[Rj]}$	Qi = i number of queries Rj = j number of resources

Step 3: Initially queries are first scheduled and mapped to the resource based on its query dependency and resource capacity (MIPS).

Step 4: check if (query status = pending and resource status = terminated).

Then create new VM to match its query size.

Step 5: if VM failure = true then create new VM from RVM and check its capacity with query size.

Step 6: if new VM Capacity > query size then allocate the queries.

Else

Create new VM using single or combined PR technique to satisfy the query size.

Step 7: check if all dependent queries are allocated.

- If yes, then go to step8
- If no, then repeat step 4 and continue the process till all queries have been allocated.

Step 8: Process independent queries based on its priority using the same procedure adopted for dependent queries.

Step 9: check if all queries are executed

- If yes , then stop the process.

If no, go to step 4 and repeat the same process.

4. Dataset description

The food-mart dataset has been used to evaluate the performance of our proposed DFTDS algorithm. It contains twenty-four relevant tables. Total number of records are 3,20,835. Table names are Account, Category, Currency, Customer, Days-check, Department, Employee, Expense_fact, Inventory_fact_2010, Inventory_fact_2011, Position, Product, Product_class, Promotion, Region, Reserve_employee, Salary, Sales_fact_2010, Sales_fact_2010, Sales_fact_dec_2011, Store, Time_by_day, Warehouse and Warehouse_class. In this work, these tables are randomly distributed into different sites.

5. Performance metrics

To evaluate the performance of the proposed algorithm various parameters have been used such as processing time, memory utilization, replication metric, error free execution, query cost and scalability. Based on [14–16] the formulation of metrics is given as shown in Table 2.

6. Experimental results and discussions

In distributed data warehouse, when the data to be fetched increase, more number of resources have to be added to the network and many joint and split operations have to be performed. To address this conflict existing grid based scheduling algorithms have been proposed and the results have been compared and analyzed. This helps to generate reports faster and easier. The proposed task scheduling algorithm concentrates to allocate queries to appropriate resources based on query dependency. The proposed resource scheduling algorithm focuses on checking the status and management of

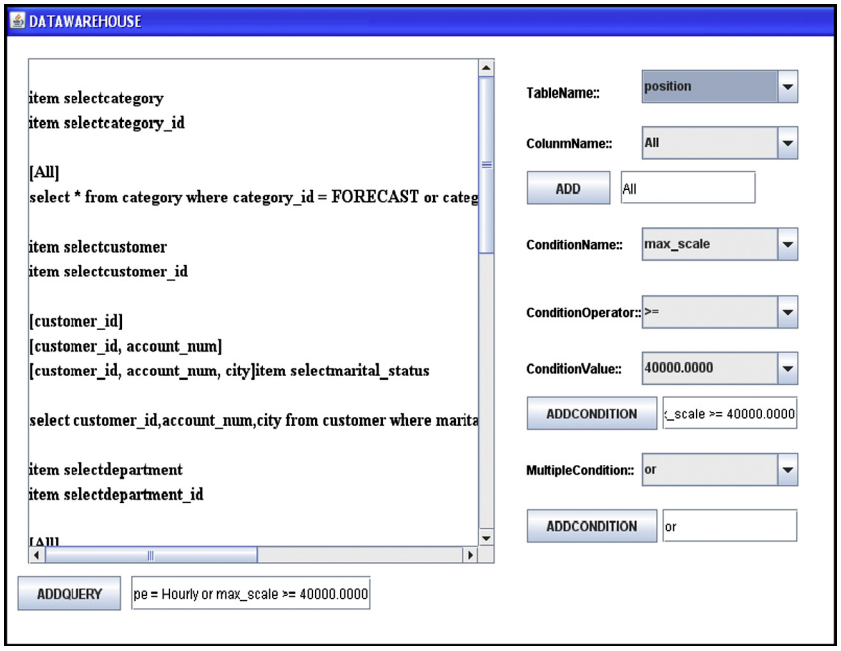

Figure 3 Main interface of proposed DFTDS algorithm screen.

```
select time_id,the_month from time_by_day
where the_day = 'Wednesday'

select product_id,warehouse_id from
inventory_fact_2011 where time_id < 866 or
store_id = 1

select lname,address,gender from customer
where city = 'Issaquah'

Select pay_date, department_id from salary
where currency_id = 2 or salary_paid >=
45.0000

select promotion_id,store_sales from
sales_fact_2011 where store_sales =
13.1600
```

Figure 4 Data warehouse sample queries.

resources as well as tolerate the failure of resources by the use of virtual and physical machines.

Experimental setup consists of 70 machines with the configuration of Intel(R) core(TM)2 DUO CPU 2.80 GHz systems running windows XP and 80 GB SATA hard disks with 2 GB RAM. We implemented the proposed DFTDS as well as all other existing scheduling algorithms using Java 1.7 for various given user queries. WINE, DTDS, VMFTRS and DFTDS are online, non preemptive scheduling algorithms with different inclination.

(1) WINE is the two level scheduling algorithm. (a) Balance over the query and update queues. (b) Both queries and updates are prioritized based on quality of service and quality of data respectively.
(2) In DTDS queries are executed in order of their dependency and resources' bandwidth capacity.
(3) In VMFTRS queries are allocated into resources as per VCGRP algorithm. Inspect the VM status, if any VM as fail, Checkout the availability of other VMs. If the concurrent VM's MIPS not satisfy the condition (Query_size < VM_MIPS) then combined the free VMs or PMs until all queries are executed.

Queries are randomly generated by the user in a distributed data warehouse environment. Fig. 3 shows the main interface screen of our proposed DFTDS algorithm where the client enters the queries for their request. Submitted queries are stored in a text file then allocated into different resources or sites using our proposed DFTDS scheduling algorithm. After query processing is completed the results are collected from resources by the server and sent to the corresponding user. The simplest sample queries and their results are shown in Figs. 4 and 5 respectively.

The above mentioned performance metrics have been used to evaluate the performance of the proposed DFTDS method with other existing methods. In this paper, processing time and memory utilization are evaluated by executing a different set of queries in the interval of 25 (25,50,75,100,125,150) and various group of resources in the interval of 10 (10,20,30,40,50,60,70). The processing time, memory utilization, scalability, replication metric, error free execution and query cost metrics for 25 cycles are measured. Each cycle evaluates the same set of queries and resources. The aggregated results are outlined in the following plot.

The processing time for all four scheduling algorithms is shown in Table 3 and differentiated them into three groups of resources as well as six sets of queries. By increasing the queries as well as resources the processing time is automatically increased. For 25 queries DFTDS algorithm takes 45 s. in 10 resources, 91 s. in 40 resources and 139 s. in 70 resources. In 70 resources DFTDS algorithm takes 228 s. for 100 queries, 239 s. for 125 queries and 255 s. for 150 queries.

```
select time_id,the_month from time_by_day where the_day = 'Wednesday'
time_id the_month
372 January
379 January
1093 December

select product_id,warehouse_id from inventory_fact_2011 where time_id < 866
or store_id = 1
product_id warehouse_id
308 1
325 1
58 24
128 24

select lname,address,gender from customer where city = 'Issaquah'
lname address gender
Derry 7640 First Ave. F
Posner 3071 Asilomar F
Bright 8461 Lodge Drive F
Tramel 9183 Via Del Sol M

select pay_date,department_id from salary where currency_id = 2 or salary_paid
>= 45.0000
pay_date department_id
2010-01-01 00:00:00 1
2010-01-01 00:00:00 5
2011-12-01 00:00:00 18

select promotion_id,store_sales from sales_fact_2011 where store_sales =
13.1600
promotion_id store_sales
0 13.1600
0 13.1600
```

Figure 5 Results obtained for sample queries.

Table 3 Comparison table of different algorithms for number of queries with processing time (s).

Number of queries	WINE			DTDS			VMFTRS			Proposed DFTDS		
	R-10	R-40	R-70	R-10	R-40	R-70	R-10	R-40	R-70	R-10	R-40	R-70
25	60	126	166	61	125	161	43	102	157	45	91	139
50	68	130	190	65	132	195	48	112	170	48	108	166
75	174	192	242	177	203	248	120	169	236	119	149	194
100	181	222	278	195	219	272	132	194	249	131	185	228
125	193	246	285	202	255	298	145	221	285	139	198	239
150	224	278	335	213	298	337	154	231	297	147	228	255

Table 4 Comparison table of different algorithms for number of queries with memory utilization (MB).

Number of queries	WINE			DTDS			VMFTRS			Proposed DFTDS		
	R-10	R-40	R-70	R-10	R-40	R-70	R-10	R-40	R-70	R-10	R-40	R-70
25	10.292	10.441	10.833	10.442	10.636	10.748	9.934	10.343	10.703	9.062	9.296	9.683
50	10.338	10.750	10.882	10.551	10.864	10.939	10.002	10.528	10.916	9.443	9.449	10.089
75	10.861	10.976	11.359	10.819	11.179	11.452	10.508	10.837	10.929	9.934	10.137	10.62
100	11.093	11.667	11.582	11.081	11.524	11.636	10.874	11.169	11.464	10.442	10.776	10.909
125	11.264	11.701	11.764	11.657	11.85	11.763	11.063	11.474	11.722	10.954	11.034	11.174
150	11.863	11.985	11.927	11.875	12.136	12.014	11.593	11.806	11.889	11.095	11.265	11.297

The comparison of the proposed DFTDS algorithm with other existing algorithms' performance in terms of memory utilization is shown in Table 4. For 25 queries DFTDS algorithm using 9.062 MB (megabyte) in 10 resources, 9.296 MB in 40 resources and 9.683 MB in 70 resources. In 70 resources DFTDS algorithm uses 10.909 MB for 100 queries, 11.174 MB for 125 queries and 11.297 MB for 150 queries.

Fig. 6 depicts the four performance metrics values of WINE, DTDS, VMFTRS and proposed DFTDS algorithms. The average performance comparison of the proposed DFTDS algorithm in terms of processing time, memory utilization, scalability, error free execution, replication metric and query cost are 20.58%, 1.25%, 13% , 3%, 3% and 28% better than WINE, 50.29%, 1.87%, 11% , 2%, 3% and 27% better than

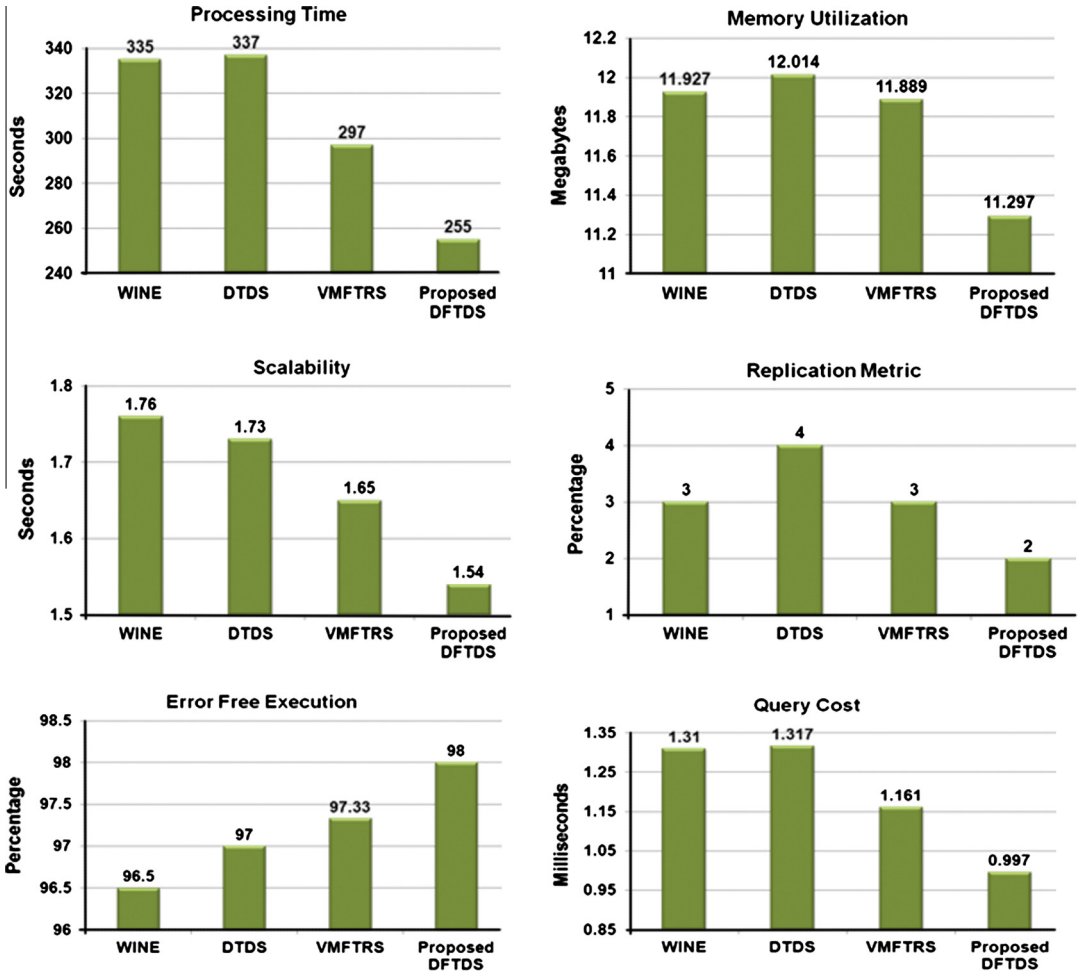

Figure 6 Evaluation plot for various algorithms with different metrics.

DTDS, 12.07%, 1.1%, 7%, 2%, 2% and 13% better than VMFTRS. The results obtained show that when compared with various scheduling algorithms our proposed DFTDS algorithm performs better.

But sometime the existing scheduling algorithms could not produce the complete query result and could not send any error message when 100 queries were sent at a time. Hence our proposed DFTDS algorithm has been executed on different query size ranging between 25 and 150. The lower processing time of DFTDS algorithm is obtained by automatic creation of new VM in case of any failure. Error rectification is performed within 5 to 10 s depending upon the size of information retrieved.

7. Conclusion and future work

In this paper, online and non-preemptive DFTDS algorithm has been proposed for the distributed data warehouse environment. The proposed DFTDS algorithm schedules queries by considering their dependency and resource status and then recycles the virtual machines to avoid the failures. Our proposed DFTDS is compared with three scheduling algorithms namely, WINE, DTDS and VMFTRS under various parameters, using different sizes of query sets and resource groups. The DTDS and VMFTRS proposed by us previously are job and resource scheduling algorithms. WINE is the existing data warehouse algorithm, based on queries and updates. The average performance evaluation in terms of processing time, memory utilization, scalability, error free execution, replication metric and query cost have been obtained. The results obtained show that, our proposed DFTDS algorithm is 20.58%, 1.25%, 13%, 3%, 3% and 28% outperforms than WINE. In future we will model online and preemptive scheduling algorithm for task prediction, resource classification.

Acknowledgment

We thank the Karpagam University for the Motivation and Encouragement to make this work as a successful one.

References

[1] Jim Smith and Paul Watson, ‘‘Fault-Tolerance in Distributed Query Processing’’2005 1–18.

[2] Krishnaveni S, Hemalatha M. Query processing in distributed data warehouse using proposed dynamic task dependency scheduling algorithm. Int J Comput Appl 2012;55(8):17–22.

[3] Krishnaveni S, Hemalatha M. Query scheduling in distributed data warehouse using DTDS and VMFTRS algorithms. Eur J Sci Res 2012;89(4):612–25.
[4] Krishnaveni S, Hemalatha M. Query management in data warehouse using virtual machine fault tolerant resource scheduling algorithm. Int J Theor Appl Inform Technol 2013;47(3):1331–7.
[5] Thiele Maik, Fischer Ulrike, Lehner Wolfgang. Partition-based workload scheduling in living data warehouse environments. Inform Syst (Elsevier) 2009;34:382–99.
[6] Somasundaram K, Radhakrishnan S. Node allocation in grid computing using optimal resource constraint (ORC) scheduling. Int J Comput Sci Netw Secur 2008;8(6):309–13.
[7] Liu Quan, Liao Yeqing. Grouping-based fine-grained job scheduling in grid computing. 1st IEEE Int Workshop Educ Technol Comput Sci 2009:556–9.
[8] Liao Yeqing, Liu Quan. Research on fine-grained job scheduling in grid computing. Int J Inform Eng Electron Business 2009;1(1):9–16.
[9] Soni Vishnu Kant, Sharma Raksha, Mishra Manoj Kumar. Grouping-based job scheduling model in grid computing. World Acad Sci Eng Technol 2010;65:781–4.
[10] Grace Mary Kanaga E, Valarmathi ML, Murali Juliet A. Agent based patient scheduling using heuristic algorithm. Int J Comput Sci Eng 2010;2:69–75.
[11] Fufang Li, Deyu Qi, Limin Zhang, Xianguang Zhang and Zhili Zhang, "Research on Novel Dynamic Resource Management and Job Scheduling in Grid Computing", IEEE-1st Int. Multi-Symposiums on Computer and Computational Sciences, vol. 1, pp. 709–713, 2006.
[12] Junyan Wang, Yuebin Xu, Guanfeng Liu, Zhenkuan Pan and Yongsheng Hao, "New Resource Discovery Mechanism with Negotiate Solution Based on Agent in Grid Environments", IEEE-3rd Int. Conf. Grid and Pervasive Computing Workshops, pp. 23–28, 2008.
[13] Rajan Alpana, Rawat Anil, Verma Rajesh Kumar. Virtual computing grid using resource pooling. IEEE-Int Conf Inform Technol 2008:59–64.
[14] "Calculating Throughput and Response Time", available at <http://javidjamae.com/2005/06/20/calculating-throughput-and-response-time/>, 2005.
[15] Charles B. Weinstock and John B. Goodenough, "On System Scalability", Technical Note, CMU/SEI-2006-TN-012, available at <http://www.sei.cmu.edu/reports/06tn012.pdf>, 2006.
[16] Greg Barish, "Scalable and High-Performance Web Applications", available at <http://www.informit.com/articles/article.aspx?p=26942&seqNum=18>, 2002.