

[Egyptian Informatics Journal (2014) 15, 51–58](http://dx.doi.org/10.1016/j.eij.2013.10.002)

Cairo University

Egyptian Informatics Journal

[www.elsevier.com/locate/eij](http://www.elsevier.com/locate/eij) [www.sciencedirect.com](http://www.sciencedirect.com/science/journal/11108665)

ORIGINAL ARTICLE

Evaluation of DFTDS algorithm for distributed data warehouse

S. Krishnaveni, M. Hemalatha [\*](#_bookmark0)

*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

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. Dem- onstration of the proposed DFTDS algorithm shows a significant reduction in query processing time and memory utilization compared to existing algorithms.

© 2013 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information,

KEYWORDS

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

Cairo University.

1. Introduction

A distributed system is an interconnected collection of hetero- geneous computers connected by a local or wide area network. As the amount of data and number of sites grows, often dis- tributed 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

\* Corresponding author. Tel.: +91 9659079670.

E-mail addresses: [sss.veni@gmail.com](mailto:sss.veni@gmail.com) (S. Krishnaveni), [csresearchhema@gmail.com](mailto:csresearchhema@gmail.com) (M. Hemalatha).

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

**Production and hosting by Elsevier**

data warehouses. Computing the huge volume of data is a sig- nificant 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 dis- tributed 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 con- nected with the various physical machines (PM) or host above the PM, virtual machine (VM) comes into distributed re- sources. 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 var- ious tasks assigned by the users without using any additional

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>

52 S. Krishnaveni, M. Hemalatha

infrastructure devices. In a LAN, a virtual framework comes into distributed resources. For example, a user can provide cer- tain 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 re- sources needed for completion don’t seem to be offered.

Rollback-recovery mechanism [[1]](#_bookmark9), 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 pro- posed system integrates our previously proposed Dynamic Task Dependency Scheduling (DTDS) and Virtual Machine Fault Tolerant Scheduling (VMFTRS) algorithms [[2–4]](#_bookmark10) for scheduling queries based on their dependency as well as recy- cling resources without human intervention in case of any fault occurred in the virtual machine. This proposed DFTDS algo- rithm is an online, non preemptive type. Meaning that, the user given queries’ estimated execution time does not know in ad- vance and we do not suspend any queries when they are in the process of being executed. The fortifications are then eval- uated 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]](#_bookmark12). The Comparison results show that significant performance improvements can be gained through recovering and continuing after failure of virtual machines.

1. Preliminary diagnosis

To improve the performance of task scheduling in a distrib- uted data warehouse system three existing grid based algorithms have been used namely, Optimal Resource Con- straint (ORC), Grouping-based Fine-grained Job Scheduling

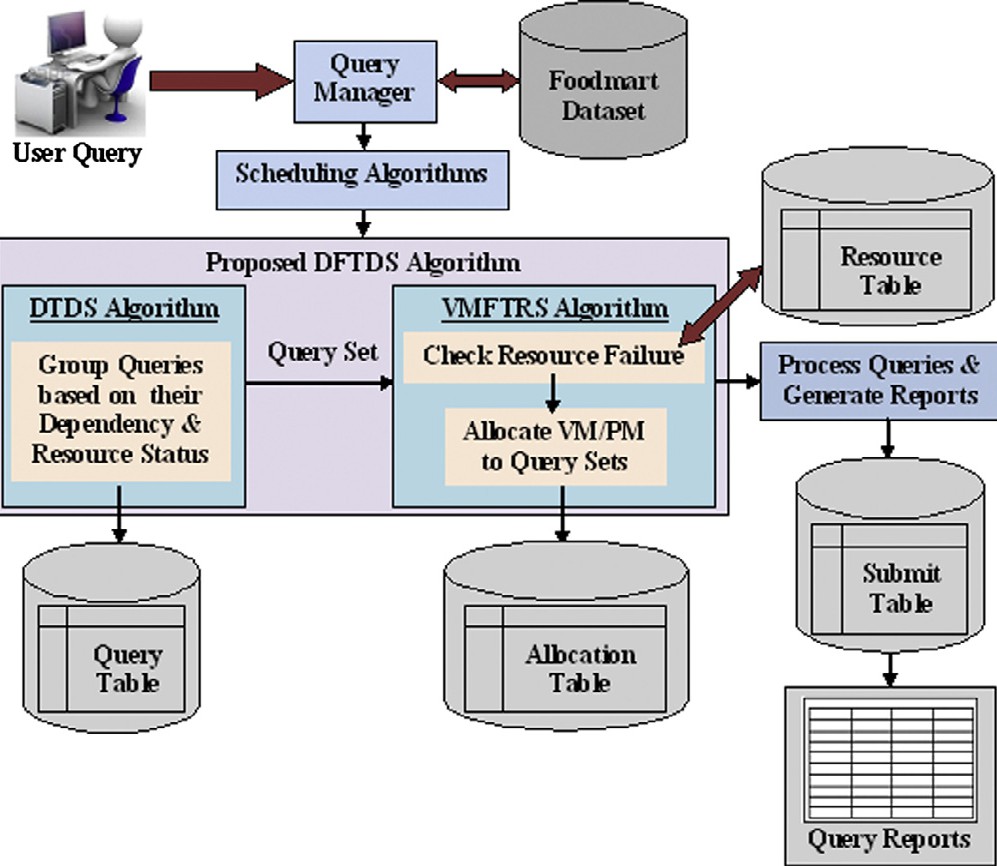


Figure 1 Architecture of proposed DFTDS algorithm.

(GFJS) and Heuristic Algorithm (HA). ORC algorithm [[6]](#_bookmark12) applies Round Robin (RR) and Best Fit algorithms to dis- tribute the tasks for available processors. GFJS algorithm [[7,8]](#_bookmark12) 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]](#_bookmark12), provides a real grid computing environment and re- duces 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]](#_bookmark12). HA shows better output for large size problems.

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 gi- ven set of 150 queries in 70 resources (systems or machines) are shown in [Table 1](#_bookmark2). 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’ depen- dency and resources’ status. The resources are mapped to the queries by the earliest free time of the resource ri and the ear- liest start time of the query *qi*. If the resource is in running state just skip and search the other resource. If the resource is free

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

|  |  |  |
| --- | --- | --- |
| 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* | | |

Evaluation of DFTDS algorithm for distributed data warehouse 53

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 prin- ciples obtained from three existing grid based algorithms have been combined and used namely, Research on Novel Dynamic Resource Management (RNDRM), New Resource Mecha- nism with Negotiate Solution (NRMNS) and Virtual Comput- ing Grid using Resource Pooling (VCGRP). RNDRM algorithm is an exemplar, to integrate agent technology with grid computing based applications [[11]](#_bookmark13). NRMNS algorithm is an agent based resource management system model [[12]](#_bookmark13). It provided a surrogate solution at the time of a resource discov- ery glitch. VCGRP algorithm [[13]](#_bookmark13) is used to exploit the ideal computing power with less effort.

When these algorithms are used few drawbacks are listed in the [Table 1](#_bookmark2) 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 combi- nations of physical resources’ MIPS with query size until all the queries are allocated.

1. Proposed DFTDS algorithm
   1. *Design details*

The proposed DFTDS algorithm has been designed in such a way that first queries are grouped based on their depen- dency. Then grouped queries are allocated to resources based on VMFTRS algorithm followed by VCGRP techniques as shown in [Fig. 1](#_bookmark1). The work flow of the proposed DFTDS algorithm has been illustrated using pseudocode in [Fig. 2](#_bookmark3). 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.

* 1. *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.

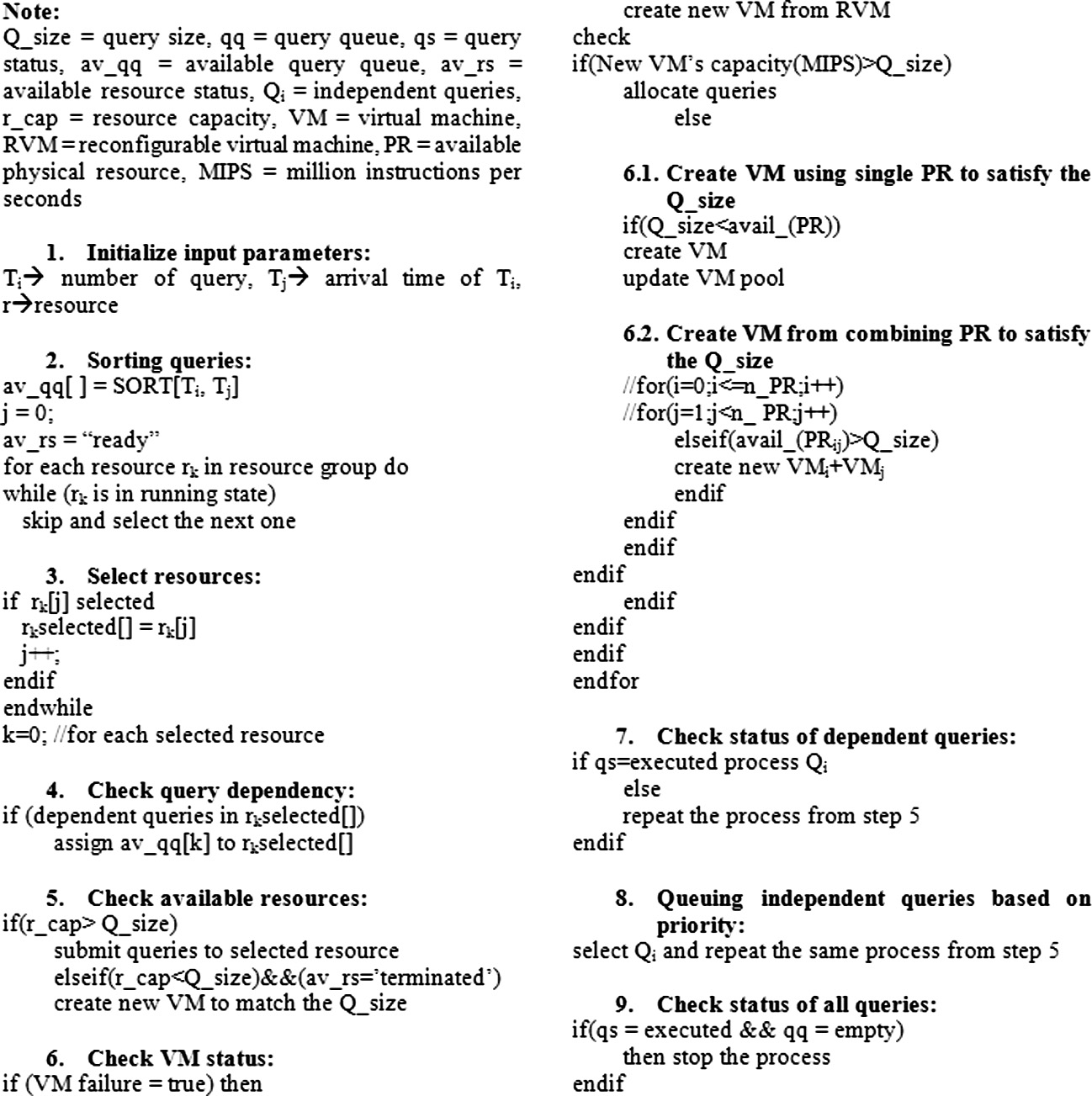


Figure 2 Pseudocode for proposed DFTDS algorithm.

54 S. Krishnaveni, M. Hemalatha

*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).

Abbreviation

*q* = number of queries given by the user *i* =

*i*th query

*MUqi* = Memory utilization for qith query

*k* = number of cycles executed the same set of queries

*Tq* = total number of queries given by the user Aq = executed queries

IEq = iterated error queries

dj = data occurred in given queries

Qi = i number of queries Rj = j number of resources

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.

( [ (*Qi* ∈ *Rj*)])] \* 100

*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.

Formula

*PTqi = (etqi* — *tqi)*

(*MUqi*) *= (Tm* — *Fm)*

P P

*n m*

*k*=1 *i*=1 *j*=1

*EFE* = (*Tq*—*lEq*) \* 100

*Eq = (Tq – Aq)*

*IEq* = [(*Eq*)*k*/*N*]

*PTqi*

*k k*=1 *dj*

*Scalability* = *PT*(*Qi*++[*Rj*])

*PT*(*Qi*)[*Rj*]

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

P

*l*

*Tq*

*m*

*k*=1

P

*n*

1. Dataset description

1

*RM* = [

*k*

P

*QC* = 1

The food-mart dataset has been used to evaluate the perfor- mance 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, Cus- tomer, 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.

Description

The difference between the arrival time (*tqi*) and the execution time (*etqi*) of given queries

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

The number of same queries are repeated in various sites or resources

The average rate of error occurred queries executed over different resources

Total processing time taken to execute the queries given by the user

Average processing time (PT) of the different set of queries is evaluated by the same set of resources

1. Performance metrics

To evaluate the performance of the proposed algorithm vari- ous parameters have been used such as processing time, mem- ory utilization, replication metric, error free execution, query cost and scalability. Based on [[14–16]](#_bookmark13) the formulation of met- rics is given as shown in [Table 2](#_bookmark4).

1. Experimental results and discussions

In distributed data warehouse, when the data to be fetched in- crease, more number of resources have to be added to the net- work and many joint and split operations have to be performed. To address this conflict existing grid based schedul- ing 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 concen- trates to allocate queries to appropriate resources based on query dependency. The proposed resource scheduling algo- rithm focuses on checking the status and management of

Table 2 Evaluation metrics.

Performance metric

Processing Time (PT)

Memory utilization (MU)

Replication metric (RM)

Error free execution (EFE)

Query cost (QC)

Scalability

Evaluation of DFTDS algorithm for distributed data warehouse 55

Figure 3 Main interface of proposed DFTDS algorithm screen.

(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 con- current VM’s MIPS not satisfy the condition (Query\_si- ze < VM\_MIPS) then combined the free VMs or PMs until all queries are executed.

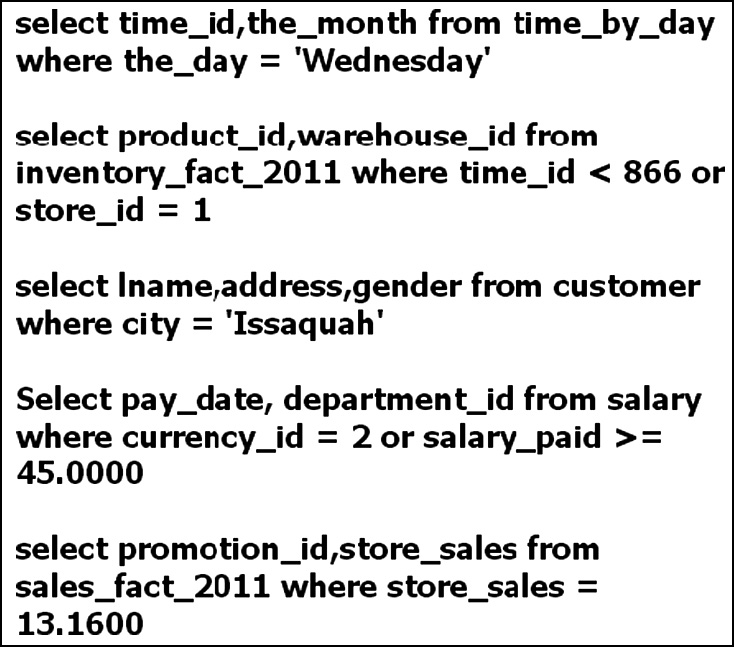
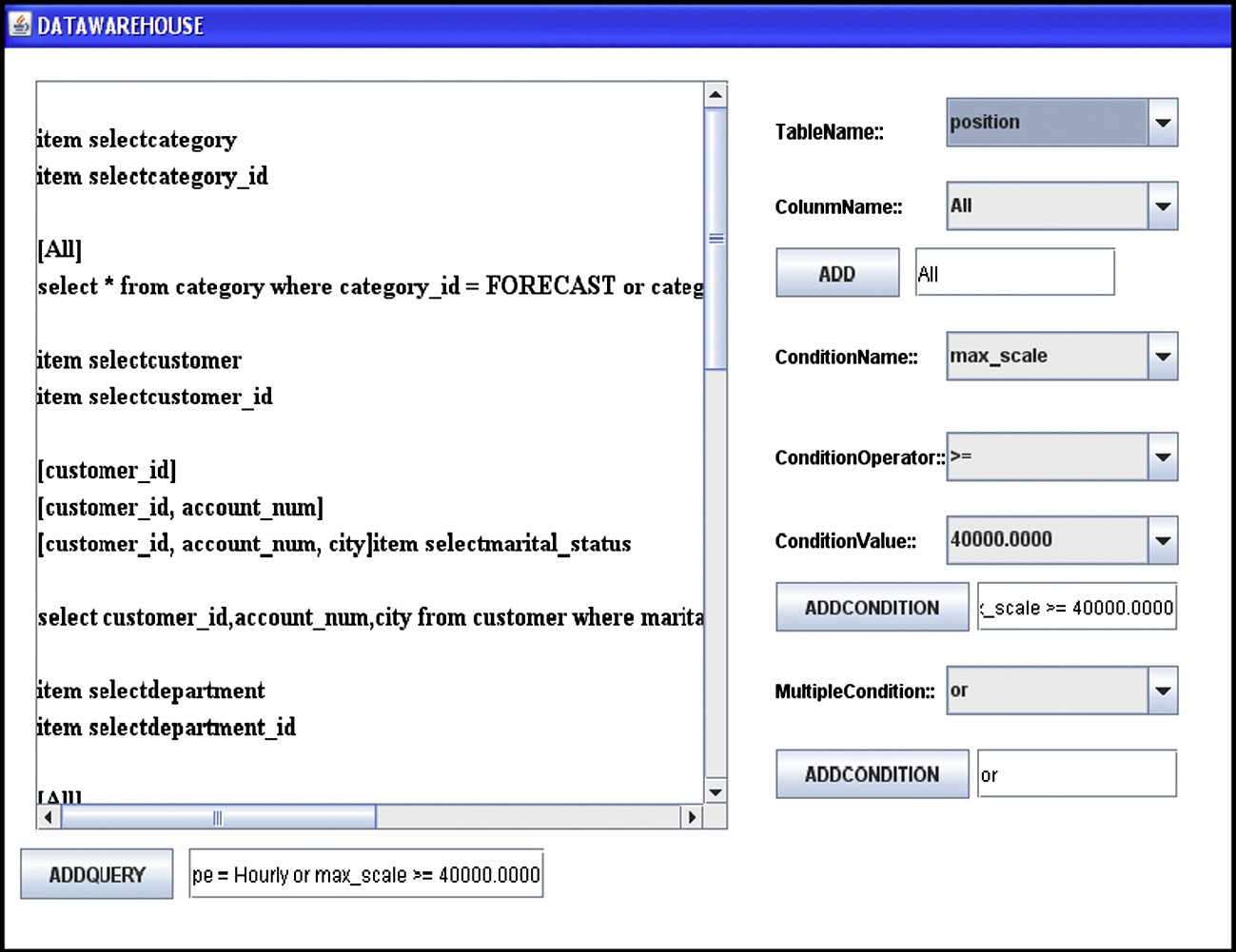


Figure 4 Data warehouse sample queries.

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 config- uration 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 depen- dency and resources’ bandwidth capacity.

Queries are randomly generated by the user in a distributed data warehouse environment. [Fig. 3](#_bookmark5) shows the main interface screen of our proposed DFTDS algorithm where the client en- ters 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](#_bookmark6) 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, replica- tion 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](#_bookmark7) and differentiated them into three groups of resources as well as six sets of queries. By increasing the que- ries as well as resources the processing time is automatically in- creased. 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.

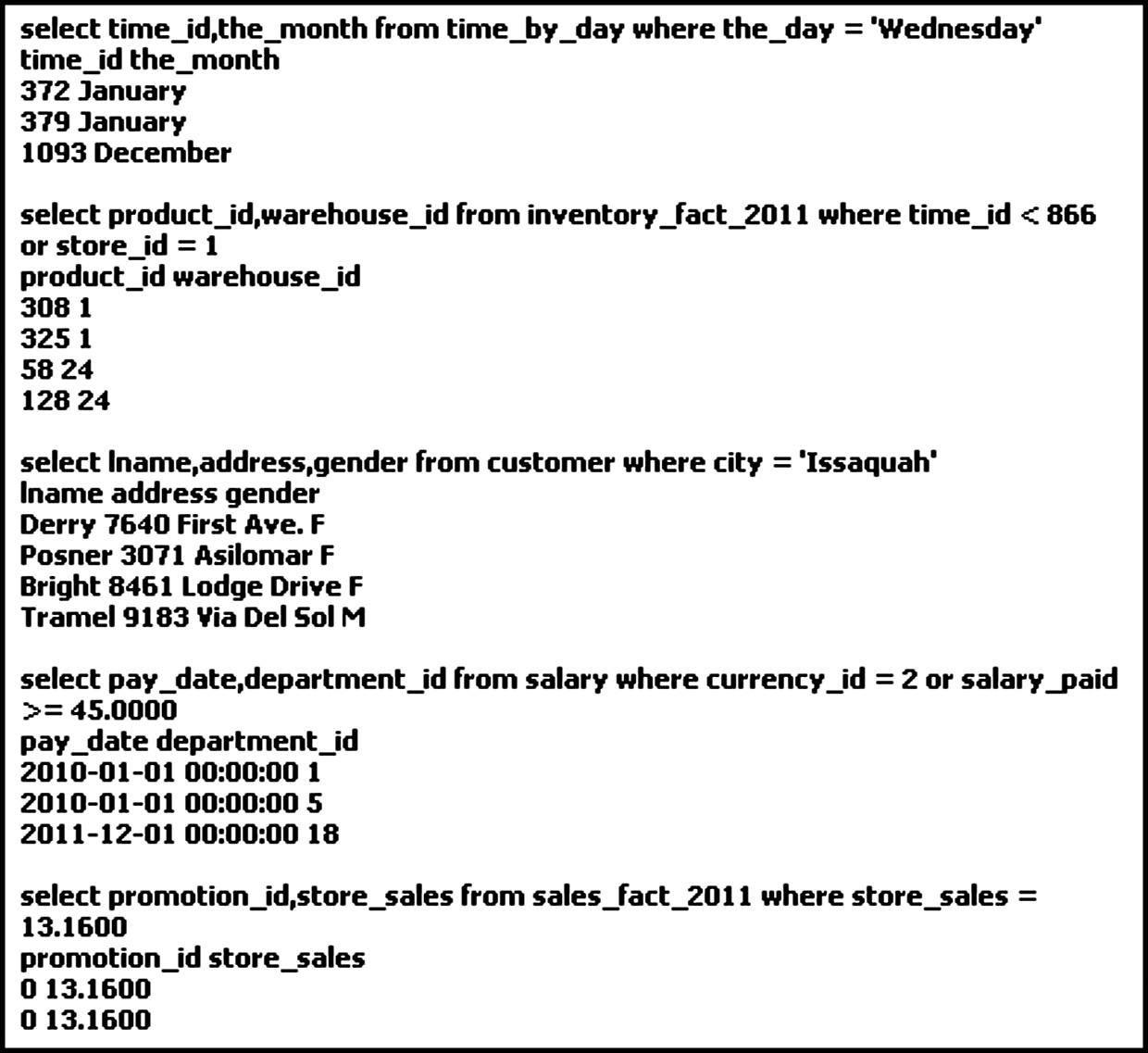
56 S. Krishnaveni, M. Hemalatha

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  R-10 | R-40 | R-70 |  | DTDS  R-10 | R-40 | R-70 |  | VMFTRS  R-10 | R-40 | R-70 |  | Proposed  R-10 | DFTDS  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](#_bookmark8). For 25 queries DFTDS algo- rithm 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.



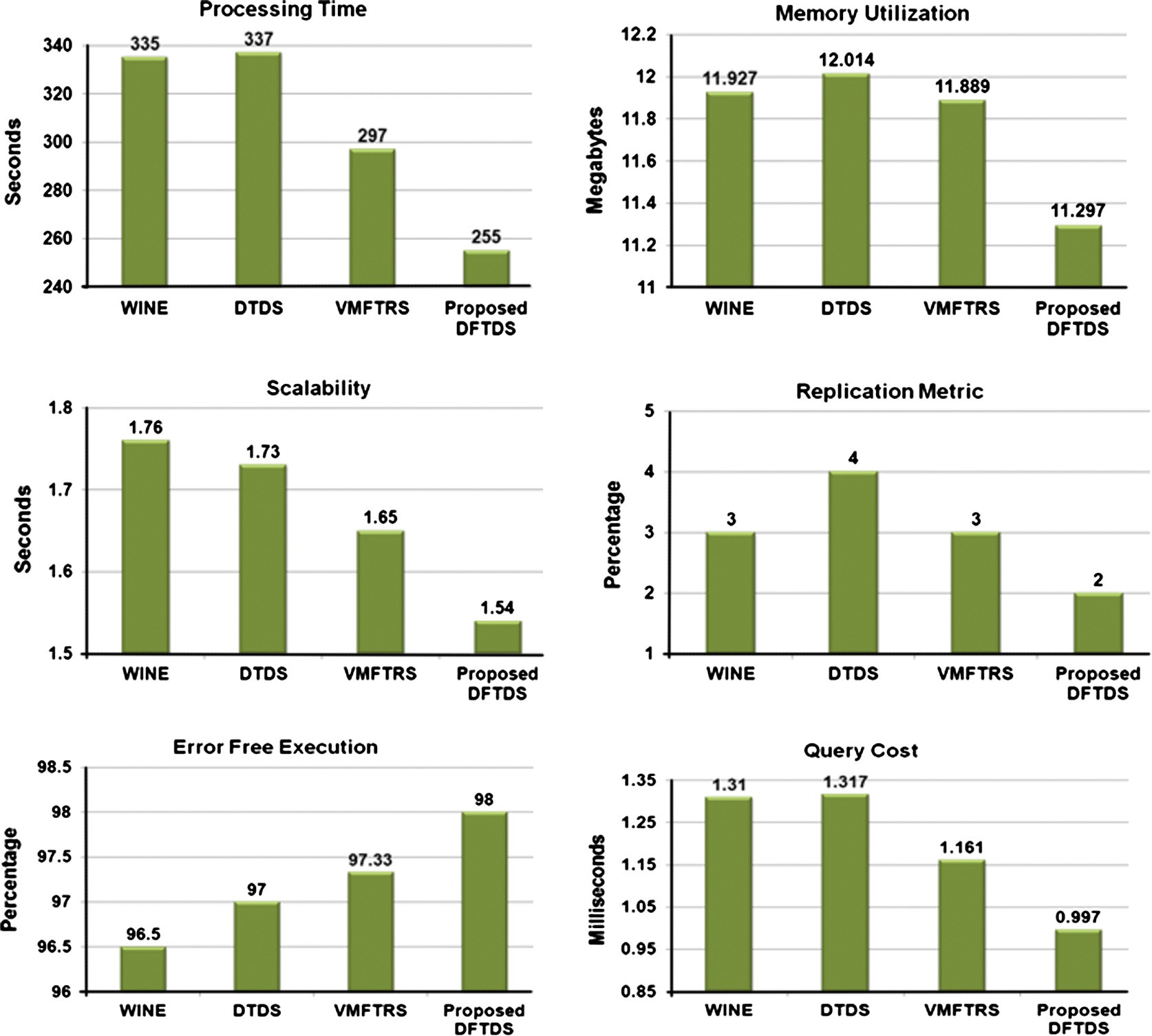
[Fig. 6](#_bookmark11) depicts the four performance metrics values of WINE, DTDS, VMFTRS and proposed DFTDS algorithms. The average performance comparison of the proposed DFTDS algorithm in terms of processing time, memory utilization, sca- lability, error free execution, replication metric and query 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

Evaluation of DFTDS algorithm for distributed data warehouse 57

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 er- ror 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 per- formed within 5 to 10 s depending upon the size of information retrieved.

1. Conclusion and future work

In this paper, online and non-preemptive DFTDS algorithm has been proposed for the distributed data warehouse environ- ment. The proposed DFTDS algorithm schedules queries by considering their dependency and resource status and then recycles the virtual machines to avoid the failures. Our pro- posed DFTDS is compared with three scheduling algorithms namely, WINE, DTDS and VMFTRS under various parame- ters, 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 aver- age performance evaluation in terms of processing time, mem- ory utilization, scalability, error free execution, replication metric and query cost have been obtained. The results ob- tained 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 sched- uling 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](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0010) [data warehouse using proposed dynamic task dependency sched-](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0010) [uling algorithm. Int J Comput Appl 2012;55(8):17–22](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0010).

58 S. Krishnaveni, M. Hemalatha

1. [Krishnaveni S, Hemalatha M. Query scheduling in distributed](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0015) [data warehouse using DTDS and VMFTRS algorithms. Eur J Sci](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0015) [Res 2012;89(4):612–25](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0015).
2. [Krishnaveni S, Hemalatha M. Query management in data](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0020) [warehouse using virtual machine fault tolerant resource schedul-](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0020) [ing algorithm. Int J Theor Appl Inform Technol](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0020) [2013;47(3):1331–7](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0020).
3. [Thiele Maik, Fischer Ulrike, Lehner Wolfgang. Partition-based](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0025) [workload scheduling in living data warehouse environments.](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0025) [Inform Syst (Elsevier) 2009;34:382–99](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0025).
4. [Somasundaram K, Radhakrishnan S. Node allocation in grid](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0030) [computing using optimal resource constraint (ORC) scheduling.](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0030) [Int J Comput Sci Netw Secur 2008;8(6):309–13](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0030).
5. [Liu Quan, Liao Yeqing. Grouping-based fine-grained job sched-](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0035) [uling in grid computing. 1st IEEE Int Workshop Educ Technol](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0035) [Comput Sci 2009:556–9](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0035).
6. [Liao Yeqing, Liu Quan. Research on fine-grained job scheduling](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0040) [in grid computing. Int J Inform Eng Electron Business](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0040) [2009;1(1):9–16](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0040).
7. [Soni Vishnu Kant, Sharma Raksha, Mishra Manoj Kumar.](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0045) [Grouping-based job scheduling model in grid computing. World](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0045) [Acad Sci Eng Technol 2010;65:781–4](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0045).
8. [Grace Mary Kanaga E, Valarmathi ML, Murali Juliet A. Agent](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0050) [based patient scheduling using heuristic algorithm. Int J Comput](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0050) [Sci Eng 2010;2:69–75](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0050).
9. 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-Sym- posiums on Computer and Computational Sciences, vol. 1, pp. 709–713, 2006.
10. 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.

1. [Rajan Alpana, Rawat Anil, Verma Rajesh Kumar. Virtual](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0065) [computing grid using resource pooling. IEEE-Int Conf Inform](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0065) [Technol 2008:59–64](http://refhub.elsevier.com/S1110-8665(13)00040-6/h0065).
2. ‘‘Calculating Throughput and Response Time’’, available at

<<http://javidjamae.com/2005/06/20/calculating-throughput-and-> response-time/>, 2005.

1. 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.
2. Greg Barish, ‘‘Scalable and High-Performance Web Applica- tions’’, available at <<http://www.informit.com/articles/arti-> cle.aspx?p=26942&seqNum=18>, 2002.