Performance Analysis of Distributed Association Rule Mining with Apriori Algorithm

M. A. Mottalib, Kazi Shamsul Arefin, Mohammad Majharul Islam, Md. Arif Rahman, and Sabbeer Ahmed Abeer

Abstract—**One of the most crucial problem in data mining is association rule mining. It requires large computation and I/O traffic capacity. One approach to resolve this problem is the use of distributed data mining algorithms in grid. It offers an effective way to mine for large data sets. Therefore, we implemented distributed data mining with Apriori algorithm in grid environment. However, usage of grid environment raises some issues about the optimization of the Apriori algorithm, especially the cost of the node to node communication and data distribution. In this paper, an Optimized Distributed Association rule mining approach for geographically distributed data is introduced in parallel and distributed environment; therefore, it reduces communication costs.**

Keywords-Data Mining; Apriori Algorithm; Grid Environment; Distributed Computing.

I. Introduction

Data mining is the process of extracting hidden patterns from data [5]. As more data is gathered, with the amount of data doubling every three years [1-2], data mining is becoming an increasingly important tool to transform this data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery [4].

While data mining can be used to uncover hidden patterns in data samples that have been "mined", it is important to be aware that the use of a sample of the data may produce results that are not indicative of the domain [3]. Data mining will not uncover patterns that are present in the domain, but not in the sample. There is a tendency for insufficiently knowledgeable "consumers" of the results to treat the technique as a sort of crystal ball and attribute "magical thinking" to it [7]. Like any other tool, it only functions in conjunction with the appropriate raw material: in this case, indicative and representative data that the user must first collect. Furthermore, the discovery of a particular pattern in a particular set of data does not necessarily mean that pattern is representative of the whole population from which that data was drawn [6]. Hence, an important part of the process is the verification and validation of patterns on other samples of data.

Data mining identifies trends within data that go beyond simple data analysis [8]. Through the use of sophisticated algorithms, non-statistician users have the opportunity to identify key attributes of processes and target opportunities. However, abdicating control and understanding of processes from statisticians to poorly informed or uninformed users can result in false-positives, no useful results, and worst of all, results that are misleading and/or misinterpreted [9].

According to [11-13], data mining commonly involves four classes of task:

- Classification - Arranges the data into predefined groups. For example an email program might attempt to classify an email as legitimate or spam.
- Clustering - Is like classification but the groups are not predefined, so the algorithm will try to group similar items together.
- Regression - Attempts to find a function which models the data with the least error.
  Association rule learning - Searches for relationships between variables. For example a supermarket might gather data of what each customer buys. Using association rule learning, the supermarket can work out what products are frequently bought together, which is useful for marketing purposes. This is sometimes referred to as "market basket analysis".

In this paper, we described Apriori algorithm in section II which has been used in grid. In chapter III and IV, we explained the configuration of grid in Linux operating system and described the proposed method accordingly. Moreover, in chapter V and VI, we described the implementation of data mining in grid and analyzed the performance.

II. Apriori Algorithm

A. Pseudo code: Apriori Algorithm

- Join Step: Ck is generated by joining Lk-1with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code:

```
Ck: Candidate itemset of size k
Lk : frequent itemset of size k
L1 = {frequent items};
for (k = 1; Lk !=∅; k++) do begin
   Ck+1 = candidates generated from Lk;
   for each transaction t in database do
   increment the count of all candidates in Ck+1
   that are contained in t
   Lk+1 = candidates in Ck+1 with min support
   end
return ∪k Lk;
```

Manuscript received December 2, 2010; revised May 5, 2011.
M. A. Mottalib, M. M. Islam, Md. A. Rahman, and S. A. Abeer are with Dept. of Computing and Information Technology, Islamic University of Technology, Gazipur, Bangladesh

K. S. Arefin is with Dept. of Computer Science and Engineering, University of Asia Pacific, Dhaka, Bangladesh

e-mail: mottalib@iut-dhaka.edu, {arefin, majhar999}@uap-bd.edu, {arif.rah, sabbeer.iut}@gmail.com

B. Examples:

TABLE I. LIST OF ITEMS

TID	List of Items
T100	I1, I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

- Consider a database, D, consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. min_sup = 2/9 =22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Step 1: Generating 1-itemset Frequent Pattern

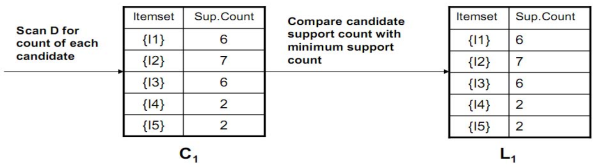

Fig. 1. Itemset Frequent Pattern.

- The set of frequent 1-itemsets, L1, consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the set of candidate.

Step 2: Generating 2-itemset Frequent Pattern

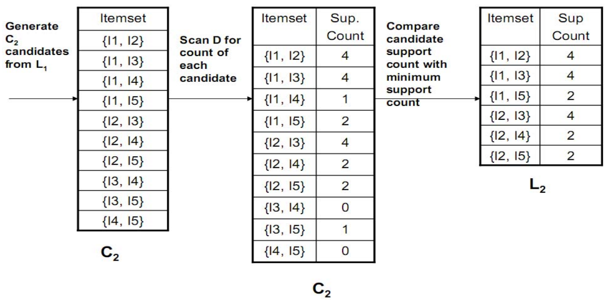

Fig. 2. Itemset Frequent Pattern.

Step 3: Generating 3-itemset Frequent Pattern

- The generation of the set of candidate 3-itemsets, C3, involves use of the Apriori Property.
- In order to find C3, we compute L2 Join L2.
- C3 = L2 Join L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.
- Now, Join step is complete and Prune step will be used to reduce the size of C3. Prune step helps to avoid heavy computation due to large Ck.
- Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent. For example, let us take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L2, We will keep {I1, I2, I3} in C3.
- Let us take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3, I5}.
- However, {I3, I5} is not a member of L2 and hence it is not frequent violating Apriori Property. Thus we will have to remove {I2, I3, I5} from C3.
- Therefore, C3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L3, consisting of those candidates 3-itemsets in C3 having minimum support.

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L3 Join L3 to generate a candidate set of 4-itemsets, C4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.
- Thus, C4 = φ, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.

These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent itemsets

- For each frequent itemset "l", generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "s Æ (l-s)" if support_count(l) /support_count(s) >= min_conf where min_conf is minimum confidence threshold.
- Back to Example:
  We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.
  – Let's take l = {I1, I2, I5}.
  – Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.
- Let minimum confidence threshold is, say 70%
- The resulting association rules are shown below each listed with its confidence.
  – R1: I1 ^ I2 -> I5
- Confidence = sc {I1, I2, I5}/sc {I1, I2} = 2/4 = 50%
- R1 is Rejected.
  – R2: I1 ^ I5-> I2
- Confidence = sc {I1, I2, I5}/sc {I1, I5} = 2/2 = 100%
- R2 is Selected.
  – R3: I2 ^ I5-> I1
- Confidence = sc {I1, I2, I5}/sc {I2, I5} = 2/2 = 100%
- R3 is Selected.
  – R4: I1 -> I2 ^ I5
- Confidence = sc {I1, I2, I5}/sc {I1} = 2/6 = 33%
- R4 is Rejected.
  –R5: I2 -> I1 ^ I5
- Confidence = sc {I1, I2, I5}/ {I2} = 2/7 = 29%
- R5 is Rejected.
  – R6: I5 -> I1 ^ I2
- Confidence = sc {I1, I2, I5}/ {I5} = 2/2 = 100%

- R6 is Selected.

In this way, we found three strong association rules.

C. Methods to Improve Apriori's Efficiency

- *Hash-based itemset counting:* A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- *Transaction reduction:* a transaction that does not contain any frequent k-itemset is useless in subsequent scans.
- *Partitioning:* any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.
- *Sampling:* mining on a subset of given data, lower support threshold + a method to determine the completeness.
- *Dynamic itemset counting:* add new candidate itemsets only when all of their subsets are estimated to be frequent.

III. Implementation of Grid

A. Proposed Grid Infrastructure

We set up our grid then it appeared like a grid client (nodeA) connecting to a grid that appears as a "cluster" of Torque (PBS) job managed machines represented by nodeB and a "cluster" of Sun Grid Engine (SGE) job managed machines represented by nodeC. The Globus toolkit needs to configure to use either of these "clusters" to offload process intensive jobs to expedite completion of a task and show a direct benefit of a compute grid.

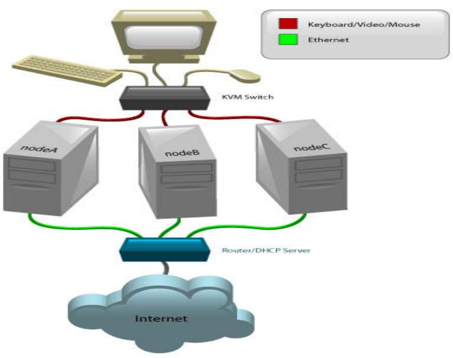

Fig. 3. Grid Experiment-Architecture for 3 (three) Nodes.

Later on we created a fully functional Grid environment consists with:

- A "cluster" of Torque (PBS) job managed machines represented by nodeF & nodeG.
- A "cluster" of Sun Grid Engine (SGE) job managed machines represented by Node C & node H.
- A certificate authority represented by nodeB.
- A client represented by nodeA.

IV. Proposed Method

Our plan is to improve the efficiency of data mining in case of huge amount of data. For this reason, we proposed the distributed mining in grid environment. Our goal is to distribute data among the nodes and use the Apriori algorithm to find the frequent item sets. We described these processes by following steps:

Step 1: Getting Request and Allocating Resource.

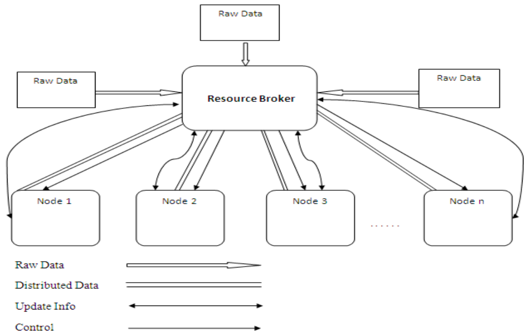

Fig. 4. Architecture of Distributed Mining.

Step 2: Distributing transaction id into different nodes.

Huge amount of data are distributed into different nodes.

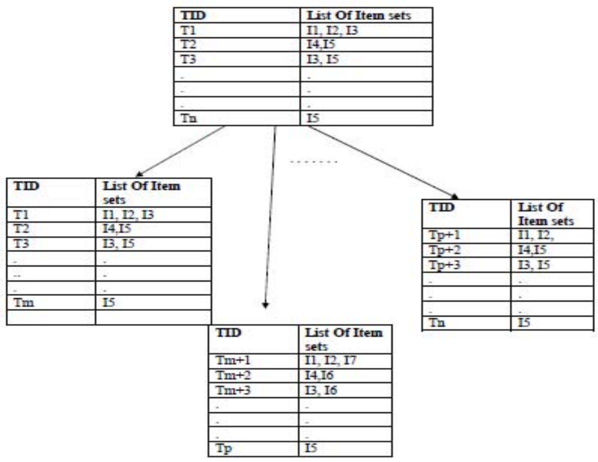

Fig. 5. Distributing Data to different nodes.

Step 3: Generating the list of Item set and distribute the list among nodes.

Suppose I1, I2, I3, I4, I5 are the five items for which we have to find out the frequent item set are distributed among three nodes n1, n2 and n3.

Step 4: Finding the number of Occurrences of item set in each node

	I1	I2	I3	I4	I5
N1	2	2	1	2	4
N2	3	2	3	2	2
N3	1	0	2	3	3

Step 5: Sending an array consist of number of occurrence of itemset to resource broker (The arrays showed in previous step are sent to the resource broker).

Step 6: Calculating the total occurrences of each itemset and checking whether it is frequent or not by applying threshold value.

TABLE II. For Threshold Value 5

	N1	***N2***	***N3***	***Total***	***Checking***
I1	2	3	1	5	=5
I2	2	2	0	4	<5
I3	1	3	2	6	>5
I4	2	2	3	7	>5
I5	4	2	3	9	>5

Step 7: Resource broker send a array consist of 0's and 1's showing which item sets are frequent to all the nodes.

Here, 0 means the corresponding item is not frequent and 1 means the item is frequent. Therefore, the array which will be sent to all the nodes is:

1	0	1	1	1

That means, except I2 all are frequent with respect to the threshold value.

Step 8: Nodes are generating (k+1)-item set by a common sequence.

Now all the nodes will generate the same item sets like: {I1,I3},{I1,I4},{I1,I5},{I3,I4},{I3,I5},{I4,I5}

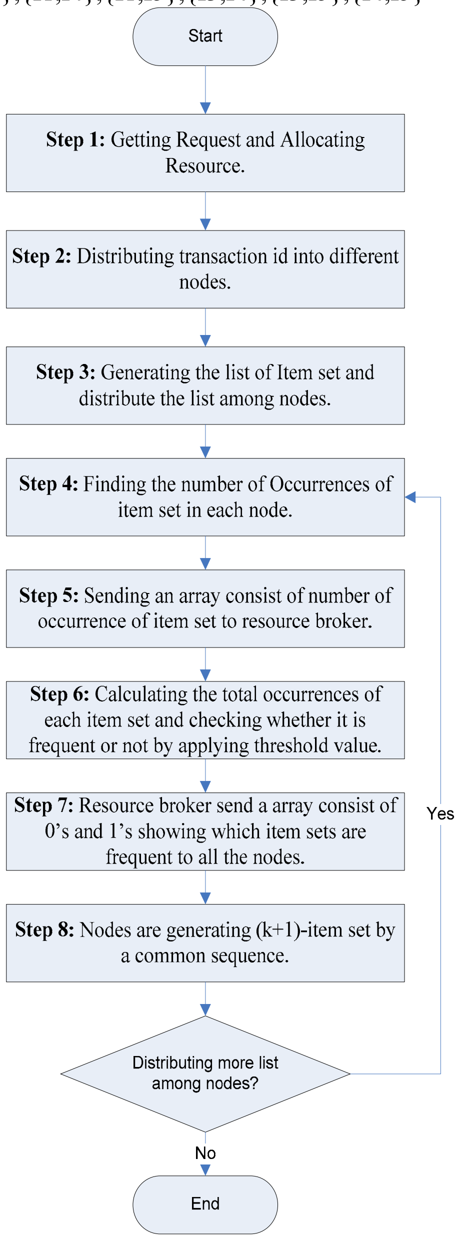

Fig. 6. Flowchart for Proposed Model.

Step 9: Repeat 4 to 8 until it gets the ending criteria. Ending Criteria. When no Ki-item set is frequent, then (Ki-1)-item set is the frequent item sets.

V. Apriori Algorithm in Grid Environment

We used a grid environment to run the program for better performance. Therefore, we used 4 (four) computers in the grid. Besides, we created an execution file and transferred into all nodes with inputs using RSL file. Though for transactions it had some overheads, however, as a whole with larger inputs, the performance has been improved (see figure 7 and 8).

TABLE III. Sample Inputs and Outputs

Input (Transactions)	Single Machine (in Minutes)	Grid Environment (in Minutes)	Performance Improvement (%)
1 Million	3.11	1.47	**47%**
2 Million	8.12	3.91	**48%**
3 Million	22.78	12.65	**56%**
4 Million	31.52	16.29	**52%**
5 Million	43.24	23.56	**54%**

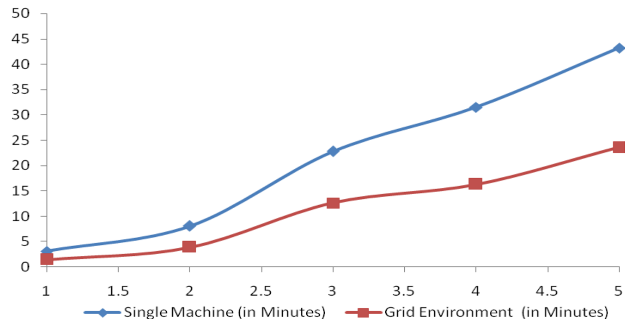

Fig. 7. Performance Measurement in Single and Grid Environment.

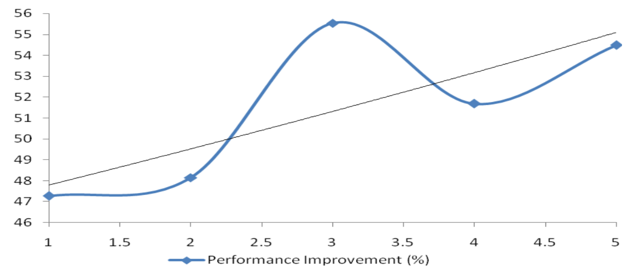

Fig. 8. Percentage of Performance Improvement in Grid Environment.

VI. Performance Analysis

According to step 6 of proposed method, we are getting lists of frequent sets from following figure 9.

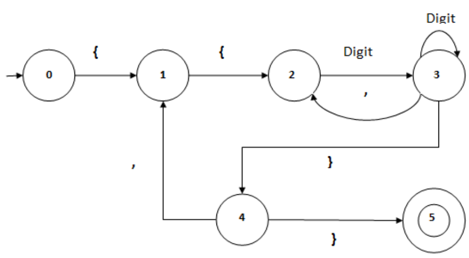

Fig. 9. Merging the output files and Generation of new Input file.

According to step 7 of proposed method, we used binary bit patterns to find frequent itemsets. Where 0 means the item is not frequent and 1 is opposite. After generating the bit stream of 1s and 0s, another input file will be created with same itemsets and then Apriori algorithm will be applied on that input on a single node.

It can be observed that for more frequent itemsets, we are getting better performance in case of checking frequent itemsets (see figure 10), however, binary list is showing better performance in case of less frequent itemsets (see figure 11). Therefore, based on frequent itemsets, step 6 or 7 will be chosen and will secure the best performance transferring fewer bits.

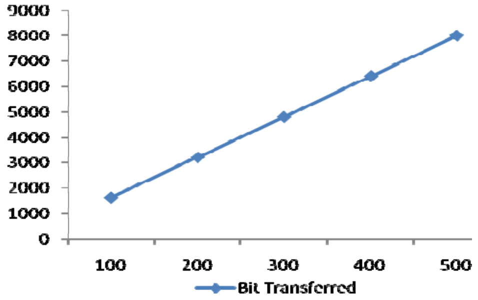

Fig. 10. Performance Measurement from Fequent Lists .

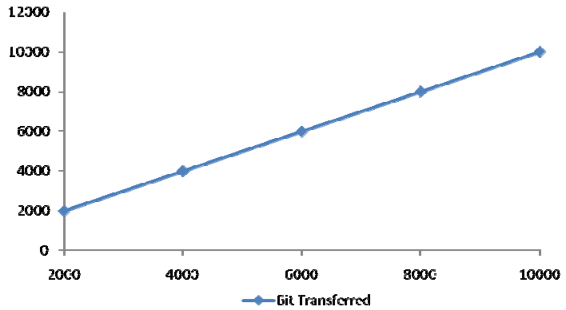

Fig. 11. Performance Measurement from Binary List.

VII. Conclusions

Current data mining tasks can be accomplished successfully only in a distributed setting. The field of distributed data mining has therefore gained increasing importance in the last few decades. In this paper, an optimized distributed version of Apriori algorithm is used for the mining process in a parallel and distributed environment. The response time with the communication and computation factors are considered to achieve an improved response time. The performance analysis is done by increasing the number of processors in a distributed environment.

References

[1] Lon-Mu Liu et al., "Data Mining on Time Series: an Illustration Using Fast-Food Resturant Franchise Data," Computational Statistics & Data Analysis, vol. 37, issue 04, pp. 455-476, (January 2001).

[2] I. Aydin et al., "The Prediction Algorithm Based on Fuzzy Logic Using Time Series Data Mining Method," World Academy of Science, Engineering and Technology 51 (2009).

[3] Dennis Wegener et al., "On Integrating Data Mining into Business Processes," 13th International Conference on Business Information Systems (BIS 2010), Berlin, Germany, (2010).

[4] D. Neef et al., "Making the case for knowledge management: the bigger picture," Management Decision 37(1):72-78 (1999).

[5] C. Bhatt et al., "Knowledge management in organisations: examining the Interaction between technologies, techniques, and people," Journal of Knowledge Management 5, (1): 68-75 (2001).

[6] G. Costello et al., "Knowledge Management in Strategic al-liances: The Role of Information Technology," Templeton Col-lege. Oxford, University of Oxford (1996).

[7] S. Raub and C. C. Ruling: "The knowledge management tussle – speech communities and rhetorical strategies in the development of knowledge management," Journal of Information Technology 16(2): 113-130 (2001).

[8] T. Nguyen Manh et al., "Data warehouse design 2: sense & response service architecture (SARESA): an approach towards a real-time business intelligence solution and its use for a fraud detection application," Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP (DWOLAP), ACM Press, New York (2005).

[9] Valerie Fiolet et al., "Bernard Toursel Optimal Grid Exploitation algorithms for Data Mining," Proceedings of the 5th International Symposium on Parallel and Distributed Computing (ISPDC) (2006).

[10] V. Fiolet et al., "Optimizing Distributed Data Mining Applications Based on Object Clustering Methods," Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (IOCM) (2006).

[11] Shu-Tzu et al., "Decision Tree Construction for Data Mining on Grid Computing," Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (2004).

[12] Wu-Shan Jiang et al., "Distributed Data Mining on the Grid," Proceedings of the 4th International Conference on Machine Learning and Cybernetics, Guangzhou (August 2005).

[13] A. Tjoa et al., "GridMiner: An Infrastructure for Data Mining on Computational Grids," Proceedings of the Australian Partnership for Advanced Computing (APAC), Australia (2003).

[14] Domenico Talia et al., "How Distributed Data Mining Tasks can Thrive as Services on Grids," Proceedings of the Communications of the ACM , Volume 53, Issue 7, Università della Calabria, Italy (2010).

[15] Huaiguo Fu et al., "A new Distributed Data Mining system on Grid," The School of Computer Science and Informatics, University College Dublin, eleld, Dublin 4, Ireland (2005).

[16] Nhien-An Le-Khac et al., "Admire Framework: Distributed Data Mining on Data Grid Platforms," School of Computer Science & Informatics, University College Dublin Belfield, Dublin 4, Ireland (2005).

Prof. Dr. M. A. Mottalib is with the Department of Computing and Information Technology, Islamic University of Technology (www.iut-dhaka.edu), Board Bazaar, Gaspar -1704, Bangladesh. E-mail: mottalib@iut-dhaka.edu.

Kazi Shamsul Arefin is with the Department of Computer Science and Engineering, University of Asia Pacific (www.uap-bd.edu), Hammond, Dhaka-1209, Bangladesh. E-mail: arefin@uap-bd.edu.

Mohammad Majharul Islam is with the Department of Computing and Information Technology, Islamic University of Technology (www.iut-dhaka.edu), Board Bazaar, Gaspar -1704, Bangladesh. E-mail: majhar999@uap-bd.edu.

Md. Arif Rahman is with the Department of Computing and Information Technology, Islamic University of Technology (www.iut-dhaka.edu), Board Bazaar, Gaspar -1704, Bangladesh. E-mail: arif.rah@iut-dhaka.edu.

Sabbeer Ahmed Abeer is with the Department of Computing and Information Technology, Islamic University of Technology (www.iut-dhaka.edu), Board Bazaar, Gaspar -1704, Bangladesh. E-mail: sabbeer.iut@iut-dhaka.edu.