Journal of Computer Science 5 (2): 146-153, 2009
ISSN 1549-3636
© 2009 Science Publications

Incremental DataGrid Mining Algorithm for Mobility Prediction of Mobile Users

[1]U. Sakthi and [2]R.S. Bhuvaneswaran
[1]Department of Computer Science, Research Scholar, Anna University, Chennai, India
[2]Ramanujan Computing Center, Anna University, Chennai, India

Abstract: Problem statement: Mobility prediction is the important issue in Personal Communication Systems (PCS). Mobile users moving logs are stored in data grid located in different locations. Distributed data mining algorithm is applied on this moving logs to generate the mobility pattern of mobile users. As new moving logs are added to the data grid, existing mobility pattern becomes invalid and it should be updated. One of the existing work to derive the new mobility pattern is re-executing the algorithm from scratch results in excessive computation. **Approach:** We had designed new incremental algorithm by maintaining infrequent mobility patterns, which avoids unnecessary scan of full database. Incremental data mining algorithm taken lesser time to compute new mobility patterns. The discovered location patterns can be used to provide various location based services to the mobile user by the application server in mobile computing environment. Data grid provided geographically distributed database for computational grid which implements incremental data mining algorithm. We built data grid system on a cluster of workstation using open source globus toolkit 4.0 and Message Passing Interface extended with Grid Services (MPICH-G2). **Results:** The experiments were conducted on original data sets and data were added incrementally and the computation time was recorded for each data sets. The performance improvement for increment size of 100 K was about 55% for 0.20% support count and it is increased to 60% for 0.25% support count. The performance is increased about 65% for the support count 0.30%. **Conclusion:** We analyzed our results with various sizes of data sets and the proof shows the time taken to generate mobility pattern by incremental mining algorithm is less than re-computing approach. In future the execution time can further be reduced by balancing the workload of grid nodes.

Key words: Incremental data grid mining, mobility pattern, knowledge grid, mobility rules, parallel mining

INTRODUCTION

Data grid is designed to allow large moving logs to be stored in repositories. In business area it is necessary to develop environment for analysis, inference and discovery over the data grid. Therefore, the evolution of the data grid is represented by knowledge grid offering high level services for distributed mining and extraction of knowledge from data repositories available on data grid[1]. The Knowledge Grid (KG) is a parallel and distributed architecture that integrates data mining techniques and grid technologies[4]. The knowledge grid is exploited to perform distributed data mining on very large data sets available over grids to find hidden valuable information, process models to make decisions and results to make business decisions[5,8]. In present study knowledge grid is developed to predict the next location of mobile user in mobile environment. By using the predicted location, the system effectively allocate resources to mobile users in the neighbor location and it is possible to answer the queries that refer to the future position of mobile users.

A Personal Communication System (PCS) allows mobile users to move from one location to another location since these systems are based on the notion of wireless access. In mobile system each mobile user is associated with Home Location Register (HLR) which stores up-to-date location of the mobile users. These logs accumulate as large database, in which data mining technique is applied to find the frequently followed location. Each Base Station (BS) in PCS is connected with separate home location register led to the geographically distributed data grid node. Grid network was built with cluster of grid node contains moving logs of mobile users. Existing research work applied data mining technique on mobile data for path mining in a single database server[2]. The sequential apriori algorithm implemented on single grid was proposed

Corresponding Author: U. Sakthi, Department of Computer Science, Research Scholar, Anna University, Chennai, India

in[13]. If size of the moving logs is very large, the overhead in integrating the data source will be too high. To overcome this problem data mining algorithm is executed on conventional distributed environment[6,14]. During the mining process size of the data set transferred between nodes was reduced by local and global mining[15]. This method is not efficient with respect to resource sharing and process co-ordination.

The most prominent example of distributed environment is grid, where a large number of computing and storage units are interconnected over a high speed network. Data mining is inductive, iterative process that extracts information or frequent patterns from large volume of data[3]. Most of the applications have created large volume of data sets which are constantly increasing and stored in geographically distributed locations. A parallel and distributed algorithm is implemented on computational grid to mine data stored in data grids. As new moving data is added to the original data base, the existing mobility pattern becomes invalid. Instead of executing the algorithm again we propose new method called incremental mining algorithm. The knowledge grid is a parallel and distributed software architecture that integrates grid technologies and data mining applications.

MATERIALS AND METHODS

The mobile users move from one location to another location in a wireless PCS network. The coverage area of the network is divided into number of location areas. Each mobile device is linked with the Base Station (BS). Each Base station contains Home Location Register (HLR) which stores permanent details of the mobile users and Visiting Location Register (VLR) which stores temporary details of the mobile users. These register includes attributes like user ID, user location, call time and call duration. User ID acts as a key for the mobile user records. The movement history of a mobile user is extracted from log files and it is stored in grid node for mining mobility patterns. The movement of mobile user is called as User Actual Path (UAP) which have the form UAP = $\langle l_1, l_2,\ldots,l_n\rangle$, where n is the number of locations followed by the mobile user and l_k represents k^{th} location in the movement path.

In our grid network, mobile user movement history is geographically distributed in different data grid nodes. Knowledge grid based mobility pattern mining algorithm is executed on computational grid over the data grid to generate the trajectories that are frequently used by mobile users. The frequently followed trajectory is named as User Mobility Pattern (UMP) which is used to generate mobility rules. The communication of candidate item sets between grid nodes is achieved by MPICH-G2 and reduces the computation overhead. Our proposed algorithm is executed on knowledge grid to generate mobility patterns from the database distributed on the data grid node. The execution time of our parallel and distributed algorithm is relatively small when compared to sequential algorithm.

Let UAP = $\langle l_1, l_2,\ldots,l_n\rangle$ be the set of locations. A database DB is a set of mobile user records, where each record contains set of locations. Let DB = $\{DB^1, DB^2,\ldots,DB^m\}$ where m is number of data grid nodes in grid network and DB^k is the database stored in kth data grid node. Local frequent locations are mined and it is communicated to all other grid node to find the global frequent locations called as mobility patterns. Mobility rules are generated from the mobility patterns which is in the form of $A\rightarrow B$ having global confidence c if c% of records in DB that contain A followed by B. Mobility rule $A\rightarrow B$ represents the mobile users current location is A then he/she will move to location B. The mobility rule $A\rightarrow B$ has global support s if s% of records contains $A\cup B$. In general, mobility prediction is performed in two steps. 1. Find all mobility patterns: By definition, each of these location sets will occur at least as frequently as a predetermined global minimum support count, min_sup. 2. Mobility rules are generated from the mobility patterns which satisfy global minimum support and minimum confidence. Mobility Rule is the important knowledge derived from database on a data grid. Moving logs are added incrementally to the data base. It makes existing mobility rules invalidate. Re-execution of mining algorithm is not an efficient process, since it ignores previously discovered rules and repeats the work that has already been done. Incremental mining algorithm is a useful technique for mining mobility rules when logs are added to the data base continuously.

The centralized mobility pattern algorithm was discussed in[4]. In mobile environment, moving log size is very large. It will increase the overhead to integrate all the moving logs into one database server. The algorithm proposed in[4] cannot be efficient for large data size. Many parallel and distributed variants of sequential apriori algorithm have been discussed in other resources[6,7]. In[5], grid implementation of frequent item sets in a grid environment dealt with sales transaction of a company. These algorithms[5-7] cannot be used directly in our domain, because this algorithm does not take into account the network topology while generating the candidate patterns. The services and methods for distributed data mining algorithm were discussed in[5]. The generation of candidate pattern in[5] is not same as the candidate pattern in mobile environment. In PCS, only the sequence of neighboring location of the network can be considered as the mobility pattern. We propose incremental parallel and distributed knowledge grid mining approach based on apriori algorithm for mining mobility patterns in mobile environment.

Globus toolkit is the widely used middleware environment for implementing grid systems. The toolkit addresses information sharing, security, virtualization, resource and data management, communication and fault detection.

Knowledge grid: Grid computing is the most emerged area for high performance distributed applications like knowledge discovery process. Knowledge Gird (KG) is an integration of basic grid services, data grid services and computational grid services for distributed data mining and knowledge discovery[10,12]. We built Knowledge Grid using the open-source globus toolkit[9].

Globus toolkit services: The basic grid components provided by the globus toolkit are:

Grid Security Infrastructure (GSI): Provides authentication (identity of the users and services) based on certificate produced by certificate authority and standard X.509. The mutual authentication is provided by Secure Socket Layer (SSL) protocol.

Monitoring and Data Service (MDS): MDS is an information service component provides information about available grid resources and periodically collects their status. It provides static information like hostname, operating system and dynamic information like CPU workload, memory status.

Globus Resource Allocation and Management (GRAM): responsible for resource allocation, job creation, monitoring management, job control and provides interface for job submission on remote machine.

GridFTP: It is an extension of FTP for parallel data transfer, file transfer based on GSI authentication mechanisms.

HeartBeat Monitor (HBM): Responsible for identifying globus process failures and application process failures and immediately recovery action can be taken.

Global Access to Secondary Storage (GASS): Exposes interfaces that help the clients to access the data in a uniform manner from heterogeneous data sources. Data caching service is utilized to improve data access performance. Metadata catalog and services allows us to search for a multitude of services, based upon the object metadata attributes.

Dynamically-Updated Resource Online Co-allocator (DUROC): It acts as a coordinator between sub jobs running on different computational grid.

Knowledge grid services: The knowledge grid contains Core K-Grid layer and High level K-Grid layer[11]. The Core K-Grid layer of knowledge grid performs two main services. (1): Knowledge Directory Service (KDS) manages metadata about data source, algorithm used for data mining, mining results and visualization tool. All metadata are represented in eXtensible Markup Language (XML) documents stored in Knowledge Metadata Repository (KMR). The discovered mobility patterns after the execution of distributed mining process is stored in Knowledge Base Repository (KBR); (2): The Resource Allocation and Execution Management Service (RAEMS) generates data mining process execution plan and it will be stored in a Knowledge Execution Plan Repository (KEPR). The execution plan will generate resources request expressed using the Resource Specification Language (RSL) for GRAM. The high level K-Grid layer supports the following services Data Access Service (DAS) is responsible for accessing data for data mining. Tools and algorithm access Service (TAAS) is responsible for loading tools and algorithm defined in KDS. Execution Plan management Service (EPMS) enables users to create execution plan by assigning programs to data resources. On multiple execution of program, different execution plans are created. Result Presentation Service (RPS) is responsible for presenting mobility patterns to the users stored in Knowledge Base Repository (KBR).

Inter-process communication using MPICH-G2: MPICH-G2 is a grid enabled open source library for implementing Message Passing Interface (MPI). It supports parallel and distributed data mining MPI application to run on cluster of machines of different architecture. MPICH-G2 uses TCP for inter-machine communication and vendor API for intra-machine communication. Our distributed Knowledge grid based Mobility Pattern Mining (KMPM) algorithm is executed on several computational grids using MPICH-G2 component. MPICH-G2 uses RSL script for sub jobs execution. The design of our parallel and distributed mining of mobility pattern application on knowledge grid is shown in Fig. 1. Initially Grid Security Infrastructure (GSI) generates certificates for the user authentication to sign on other site. The user can use Monitoring and Discovery Service (MDS) to select grid node based on memory, CPU load and network topology. The globus-run command is executed to submit job on multiple machines by creating MPI computation. MPICH-G2 uses RSL to specify the URL of the computational and grid resources. The RSL script defines the job in the following way:

```
  +
  (&(resourceManagerContact= "kalannia/jobmanager-pbs")
  (count = 10)
  (label = "subjob 0")
  (environment=
(GLOBUS_DUROC_SUBJOB_INDEX 0)
  (directory = "/home/sakthi/gridmining")
  (executable = "/home/sakthi/gridmining/KMPM")
  )
```

The parameter "resourceManagerContact" specifies the URL of the cluster resource and the corresponding jobmanager-pbs. The parameter "count" specifies the number of nodes required for computation, and the "label" represents name of the sub-job. The parameter "environment" specifies the directory in which the globus is installed. The parameter "directory" specifies the working directory, and the "executable" parameter specifies the location of the executable. The mining job is started on the computational resource by using GRAM running on each server. The MPICH-G2 calls Dynamically-Updated Request Online Coallocator (DUROC) library file to start the mining on multiple computational resources specified by the RSL script.

Parallel and distributed algorithm: Let us assume there are n processing nodes P_1, P_2,…,P_n in our distributed system. The data base contains mobile User Actual Path (UAP) is distributed over n processing nodes. In our work, modified form of Count Distribution (CD) algorithm is used to mine User Mobility Pattern (UMP) from User Actual Path (UAP). The locations which are frequently followed by mobile user are called Mobility Patterns. Count Distribution algorithm is previously used in various domains. In our study, CD algorithm is used with new method for calculating support count of subsequence in UAP. Let X.sup and X.sup_i be the global and local support count of subsequence X at a process P_i. X is globally large if

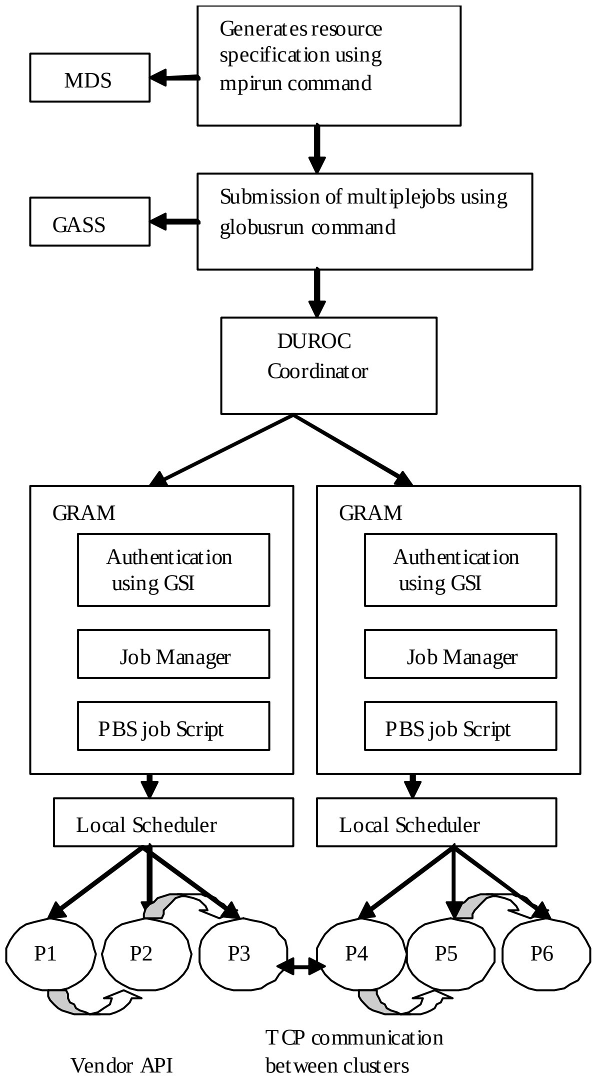

Fig. 1: Inter-process communication

X.sup>S and locally large if X.$sup_i > S_i$, where S is Global minimum support and S_i is local minimum support at P_i. Our KMPM algorithm is executed on n processing nodes in parallel. Intermediate results are transferred to other nodes using send and receive commands of MPICH-G2. The pseudo code for KMPM algorithm is given below:

```
// pseudo code for local mining at a process $P_i$
KMPM( )
input: moving path of mobile users
output: local frequent mobility pattern
$LS_k$ = null //k is the length of subsequence
for each UAP a$\in$ Di
    find the subsequence of UAP and put it in S
    for each subsequence s ∈ S
        //calculate the support count and store it in $LS_k$
        s.count = s.count+s.suppInc
    end for
 end for

// pseudocode for global mining at kth pass
globalmining()
input:  local frequent patterns
output: global frequent patterns
k = 1
while ($GS_k \neq$ null)
      for every from 1 to n increment by 1
      node Pi exchange and merge local support counts
      of $LS_k$ with all n nodes and find the global support
      count of all subsequence and store it in $GS_k$
 end for
for each subsequence in $GS_k$
       if the global support count of subsequence is
      above the minimum global support count then put
      the subsequence in  global mobility pattern $GMP_k$
    end for
    increment k by 1
end while
```

The above algorithm is an adaptation apriori algorithm in a distributed environment. Every process generates subsequence of length k called as Local Subsequence (LS_k) and then calculates local support count for each LS_k. These subsequence local support count is exchanged with all other process to generate Global Subsequence (GS_k) and then calculates global support count for each GS_k. The subsequences which have a support count greater than the threshold global support count are selected as Global Mobility Pattern (GMP_k). For instance, consider UAPs ⟨4, 6, 8, 0, 5⟩, ⟨2, 4, 8, 0, 6⟩ and ⟨1, 2, 4, 6⟩ where the number 4 represents location of mobile user. The support count of the subsequence ⟨4, 6⟩ can be calculated as follows. s.count = s.count+suppInc and suppInc= $\frac{1}{1+\text{totdis}}$ where totdis is the number of location between 4 and 6. s.count value is 2 because it appears in 1st and 3rd UAP. In 2nd UAP there are two locations between 4 and 6. Therefore the support value for 4 and 6 is ⟨4,6⟩.count = 2+ $\frac{1}{1+2}$ = 2.33. It will increase the accuracy of the support counting. This algorithm will generate more accurate Global Mobility Patterns (GMP).

Mobility rule generation: In our Knowledge grid, after the execution of parallel and distributed mining algorithm, Mobility Patterns (MP) are stored in Knowledge Base Repository (KBR). It can be used to generate mobility rules. For example, Mobility pattern is ⟨4, 6, 8⟩.

Mobility rules are as follows:

⟨4⟩→⟨ 6, 8⟩
⟨4, 6⟩→⟨8⟩

From the UMPs, all possible mobility rules are generated and their confidence value is calculated. In general, mobility rule R is represented as $\langle t_1, t_2,..,t_i\rangle \rightarrow \langle t_{i+1}, t_{i+2},..,t_p\rangle$. Confidence value for the rule R is calculated using the following formula:

$$\text{Confidence (R)} = \frac{\langle t_1, t_2, ..., t_i\rangle.\text{count}}{\langle t_{i+1}, t_{i+2}, ...t_p\rangle.\text{count}}$$

Then the mobility rules which have a confidence c higher than a predefined confidence threshold ($\text{conf}_{\min}$) are selected. These mobility rules can be used in next phase for mobility prediction. The mined mobility rule is compared with the current location of mobile user to predict the next possible locations.

Mobility prediction: In mobile web environment, next location of mobile users is predicted using mobility rule and current location of the mobile user. Mobility rule contains two parts namely, head-the part before the arrow and tail- the part after the arrow. Our process generates set of rules whose head matches with the current location of the mobile user. These rules are called as matching rules. The first location in the tail of the matching rule and match value is stored in the resultant array. Match value is calculated by summing up the support value of the UMP and confidence of the rule. The matching rules in the array are sorted in descending order with respect to match value. This process generates most confident and frequent rules. The parameter m defines number of predictions required. It selects only first m locations from the resultant array. In Fig. 2, there are three locations 4, 6 and 8. For example, currently the mobile user is in location 4. Our algorithm generates matching rules ⟨4⟩→⟨6⟩ and ⟨4⟩→⟨8⟩. The match value is calculated for each predicted location and stored in resultant array. Resultant array contains two values [(6, 78.56), (8, 68.56)]. If m = 1, then the location 6 will be predicted as next location. If m = 2, then the locations 6 and 8 are the predicted as next locations. The predicted location can be used by expert system to provide location based service to the mobile user.

Incremental mobility rule mining: The parallel and distributed algorithm is executed on data grid to find the mobility pattern when moving logs are added to and

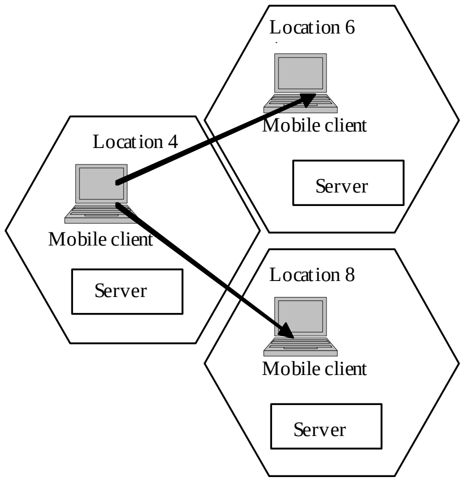

Fig. 2: Movement of mobile user in GSM network

Table 1: Parameters used in experiment

Sr. No.	Notation	Meaning
1	DB	Transactions in original database
2	db	Transactions that are newly added
3	DB^+	Transactions in the updated database DB $\cup$db
4	GMP^+,GMP^{DB},GMP^{db}	Mobility pattern the respective database
5	NMP(GMP^+), NMP (GMP^{DB}) and NMP(GMP^{db}) the respective database	Negative Border in
6	NMP	infrequent pattern sequences which did not satisfy the minimum support
7	ML^{db}	new moving log transactions added to the database

removed from the database without re-executing the algorithm. The algorithm uses the concept of negative border for data mining by maintaining the infrequent mobility pattern. Infrequent sequence represents the set of sequences, which did not satisfy the minimum support. The various parameters used in our experiment is shown in Table 1. During each pass of the data mining algorithm, the set of sequences (GS_k) are computed from the previous mobility patterns (GMP_{k-1}). The negative border with infrequent sequence is found by $NMP_k = GS_k - GMP_k$. where NMP_k represents the infrequent sequence in kth pass. The algorithm for updating the mobility pattern as follows:

```
Function updatemobilitypattern ($GMP^{DB}$, NMP($GMP^{DB}$), $ML^{db}$)
compute $GMP^{db}$
for each sequence s $\in$ $GMP^{DB}$ $\cup$ NMP($GMP^{DB}$) do
    $t_{db}(s)$ = number of transaction in db containing s
    $GMP^+ = \phi$
for each sequence s $\in$ $GMP^{DB}$ do
  if ($t_{DB}$ (s)+ tdb(s)) >minsup
  then $GMP^+ = GMP^+ \cup s$
for each sequence s $\in GMP^{db}$ do
  if s $\cup GMP^{DB}$ and s $\in NMP^{db}$ and
  ($t_{DB}$ (s)+ tdb(s))> minsup then
    $GMP^{db+} = GMP^{db+} \cup s$
   if $GMP^{DB} \neq GMP^{db+}$ then
    NMP($GMP^+$) = negativebordergen($GMP^+$)
    else NMP($GMP^{DB+}$) = NMP($GMP^{DB}$)
if $GMP^{DB}$ $\cup$ NMP($GMP^{DB}$) $\neq$ NMP($GMP^{DB+}$) $\cup$ NMP($GMP^{DB}$) then
 s = $GMP^{DB+}$
repeat
   compute s = s $\cup$NMP (s)
until s does not grow
$GMP^{DB+}$ = {$\times$ s | support(x) > = minsup}
```

Initially the original transactions are mined to generate the Global Mobility Pattern (GMP^{DB}) for the specified minimum support. While the Mobility Patterns are generated, their negative border GMP(GMP^{DB}) is also generated and retained. The negative border is used to avoid re-computation when new transactions are added to the database. When new moving log transactions are added to the database (db) the frequent mobility pattern for the new transactions (GMP^{db}) are generated for the user specified minimum support. There are three cases to update the mobility pattern. (1): The support count is calculated for each sequence in the (GMP^{DB}) from the (GMP^{db}) if the minimum support is satisfied and GMP^{DB} is updated. (2): The support count of sequence that is common to both GMP^{db} and NMP^{db} are counted and GMP^{DB} is updated if support count is specified. Some set of sequences in NMP(GMP^{DB}) may not satisfy the support count and it would remain in $\cup$ NMP(GMP^{DB}). (3): The sequences that are in GMP^{db} not in (GMP^{DB}) are counted. The sequences in GMP^{db} would satisfy the support count is added to (GMP^{DB}).

RESULTS AND DISCUSSION

The experiments have been performed on Oracle10G and PostgreSQL installed in globus toolkit 4.0 middleware in Scientific Linux environment. Initially the transactions in the database are considered as original database and datasets are added incrementally for three runs of algorithm to show the performance of Incremental Mining algorithm (IM) over Re-Computing algorithm (RC). The data sets represented in the form of T5I2D1000K, where 5 denotes the average number of locations in the user moving path, 2 denotes support count of locations in the dataset and 1000 K denotes the total number of transactions in K. The experiment is conducted for 0.30, 0.25 and 0.20% support count.

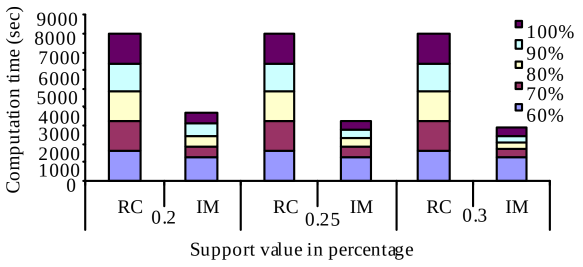

Fig. 3: Dataset T5I2D1400K, Data added increment of 100K from 1000K

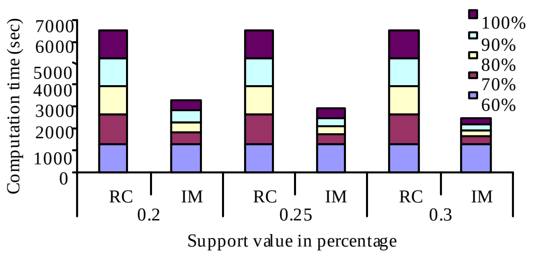

Fig. 4: Dataset T5I2D1400K, Data added increment of 200K from 600K

From the Fig. 3 it is noted that the performance improvement for increment size of 100 K was about 55% for 0.20% support count and it is increased to 60% for 0.25% support count. The performance is increased about 65% for the support count 0.30%. From the Fig. 4 the performance improvement was about 50% for 0.20% and it is increased to 55% for the support count 0.25%. The performance is increased 60% for the support count 0.30%.

CONCLUSION

In this study, we have proposed incremental parallel and distributed algorithm implemented on Knowledge grid to predict the next location of mobile user in a mobile web computing system. Incremental algorithm performs better when compared to re-computation for larger datasets. In the first step of mining algorithm, mobility patterns are mined from the User Access Path (UAP) and mobility rules are generated using mobility patterns. Finally current location of mobile user is compared with the mobility rule to predict the location of mobile user. By using the predicted movement, the system can effectively allocate resources and provide location based services to the mobile users. Knowledge Grid based Mobility Pattern Mining (KMPM) algorithm for mobility prediction needs less computation time compared to sequential mobility prediction algorithm and it supports scalability. The proposed approach shows how the Knowledge Grid system is used for distributed data analysis. Also compared to other distributed system, grid reduces the message communication overhead using MPICH-G2 technology. The subsequence exchange between processes is effectively achieved by using MPICH-G2. In future the study can be extended by applying the work load balancing concept for distributed data mining.

REFERENCES

1. Luo, C., L. Anil Pereira and M. Soon Chung, 2006. Distributed mining of maximal frequent itemsets on a data grid system. J. Super Comput., 379: 71-90. DOI: 10.1109/71.485502
2. Gokhan Yavas, dimitrios Katsaros, Ozgur Ulssoy and Yannis manolopoulos, 2005. A data mining approach for location prediction in mobile web environments. Data Knowl. Eng., 549: 121-146. DOI: 10.1016/j.datak.2004.09.004
3. Shearer, C., 2000. The CRISP-DM model: The new blueprint for data mining. J. Data Warehous., 5: 13-22. DOI: 10.1136/qshc.2004.012831
4. Wu-Shan Jiang and Ji-Hui Yu, 2005. Distributed data mining on the grid. Proceedings of International Conference on Machine Learning and Cybernetics, Aug. 18-21, IEEE Xplore Press, USA., pp: 2010-2014 DOI: 10.1109/ICMLC.2005.1527275
5. Mario Cannataro, Antonio Congiusta, Andrea Pugliese, Talia and Paolo Trunfio, 2004. Distributed data mining on grids: Services, tools and applications. IEEE Trans. Syst. Man Cybernet., 34: 2451-2465. DOI: 10.1109/TSMCB.2004.836890
6. Cristian Aflori and Mitica Craus, 2007. Grid implementation of apriori algorithm. Adv. Eng. Software, 38: 295-300, http://dx.doi.org/10.1016/j.advengsoft.2006.08.011
7. Thuraisingham, B., 2000. A primer for understanding and applying data mining. IEEE Educ. Activit. Depart., 2: 28-31. http://dx.doi.org/10.1109/6294.819936
8. Agrawal, R., T. Imielinski and A. Swami, 1993. Mining association rules between Sets of items in large databases: Proceedings of ACM SIGMOD International Conference on the Management of Data., May 25-28, Washington, DC., United States, pp: 207-216 http://doi.acm.org/10.1145/170035.170072
9. Oracle, Java Stored Procedures developers Guide., http://otn.oracle.com/doc/oracle8i_816/java/a81353/toc.htm

10. Cannataro, M. and D. Talia, 2003. The knowledge grid. Commun. ACM., 46: 89-93. http://doi.acm.org/10.1145/602421.602425
11. http://www.globus.org/toolkit
12. Foster and C. Kesselman, 2001. The Anatomy of the grid: Enabling scalable virtual organizations: Intl. J. Super Comput. Applied, 2150: 1-4. http://www.springerlink.com/content/flddyth66lrqdk36/
13. Cristian Aflori and Mitica Craus, 2007. Grid implementation of Apriori algorithm. Adv. Eng. Software, 38: 295-300. DOI: 10.1016/j.advengsoft.2006.08.011
14. Agrawal, R. and J. Shafer, 1996. Parallel mining of association rules. IEEE Trans. Knowl. Data Eng., 8: 962-969. DOI: 10.1109/69.553164
15. Assaf Schuster, Ran Wolff and Dan Trock, 2005. A high-performance distributed algorithm for mining association rules. Knowl. Inform. Syst. J., 7: 458-475. http://dx.doi.org/10.1007/s10115-004-0176-3