Scalable Data Mining with Log Based Consistency DSM for High Performance Distributed Computing

Hideaki Hirayama
Graduate School of Information Systems
The University of Electro-Communications
hirayama@yuba.is.uec.ac.jp

Hiroki Honda
Graduate School of Information Systems
The University of Electro-Communications
honda@is.uec.ac.jp

Toshitsugu Yuba
Graduate School of Information Systems
The University of Electro-Communications
yuba@is.uec.ac.jp

Abstract

Mining the large web based online distributed databases to discover new knowledge and financial gain is an important research problem. These computations require high performance distributed and parallel computing environments. Traditional data mining techniques such as classification, association, clustering can be extended to find new efficient solutions.

This paper presents the scalable data mining problem, proposes the use of software DSM (Distributed Shared Memory) with a new mechanism as an effective solution and discusses both the implementation and performance evaluation results.

It is observed that the overhead of a software DSM is very large for scalable data mining programs. A new Log Based Consistency (LBC) mechanism, especially designed for scalable data mining on the software DSM is proposed to overcome this overhead. Traditional association rule based data mining programs frequently modify the same fields by count-up operations. In contrast, the LBC mechanism keeps up the consistency by broadcasting the count-up operation logs among the multiple nodes.

1. Introduction

Discovering knowledge from large databases is well known as data mining. The basic functions of data mining include detecting, predicting and interpreting patterns in on-line databases. Data mining algorithms employ techniques from wide variety of fields such as statistics, artificial intelligence. A popular example of data mining is seen in the retail organization. Such organizations collect massive amounts of data and create large databases with the progress in bar-code technology. One of the primary techniques used in retail organizations to discover the knowledge from the databases is Association rule based mining [1][2]. For example, the knowledge for a store layout, "40 percent of customers who purchase baby's diapers also purchase canbeers", can be discovered by data mining. This discovered knowledge can be used to increase the profit by changing the store layout.

Knowledge need to be extracted from data bases vary from industry to industry. Many industries are in the process of developing their own specialized data mining techniques to satisfy their own requirements. In general, data mining programs must process massive amounts of data available in primary, secondary and distributed storages of SMP(Symmetric Multi-Processor)-type high performance parallel computers and/or cluster-type distributed computing environments.

Developing algorithms and programs that can be executed both in SMP-type parallel computers and cluster-type distributed systems is an important problem in data mining. Programs and algorithms exhibiting such capability are scalable data mining programs. We propose the use of software DSM (Distributed Shared Memory) [3][4] to develop scalable data mining programs for high performance parallel and distributed computing systems. By using the DSM, we can develop programs with shared memory programming model instead of message passing programming model even in

0-7695-0583-X/00 $10.00 © 2000 IEEE

cluster-type distributed systems.

Typical data mining programs based on association rule mining frequently modify the same data fields by count-up operations. When those programs are executed in software DSM, many data in the fields are frequently passed among the plural nodes to keep up the consistency. Thus the overhead of a software DSM is very large for data mining programs. To improve the performance we propose a new Log Based Consistency (LBC) mechanism, on the software DSM. This mechanism is specially designed for scalable data mining. The LBC mechanism keeps up the consistency by broadcasting the count-up operation logs among the plural nodes.

To evaluate the effectiveness of the LBC mechanism, we have developed a scalable data mining system, VISIONA (Virtual Shared Memory Environment for Scalable Data Mining Applications). VISIONA is implemented as a software DSM with LBC and running in distributed systems such as PC clusters or UNIX-computer clusters. Two programs for mining, an association rule based program and a bayesian network program, have been implemented in VISIONA. Performance of the software DSM with LBC was evaluated by executing the programs in VISIONA. We selected association rule and bayesian network mining because of their popularity among users.

In this paper, we present the mechanism of the software DSM with LBC, the implementation of the LBC mechanism and the performance evaluation results of LBC mechanism for association rule based mining and bayesian network mining.

2. Large Scale Data Mining

This section describes the two large scale data mining programs. First one is a mining program based on association rule [1][2] for retail organizations. Second mining program is the bayesian network generation [5] for the manufacturing industry.

2.1. Association Rule Mining

Association rule mining is very popular among the retail organizations. For association rule mining it is necessary to define two measures, a support value and a confidence value. The support value is a measure based on the percentage of the transaction record including an item X. The confidence value is a measure based on the percentage of the transaction record including an item X is also including an item Y. Users specify the constraints, the minimum support value and the minimum confidence value, for mining association rules.

Association rule mining is done by the following Apriori algorithm. In this algorithm k is an integer which starts from one. The following steps process the item-sets (sets of items) having k items, which are called pass-k. They are iterated after k is incremented.

1. Read all transaction records and count-up the occurrences of the item-sets having k items. These are the candidate item-sets of k.
2. Select the item-sets from the candidate item-sets of k, which support values which are greater than the minimum support value. They are called the large item-sets of k.
3. Make the candidate item-sets of k+1 out of the large item-sets of k. Repeat the above steps until the large item-sets will be null.

It takes very long time to execute the Apriori algorithm, since it reads a large number of transaction records and counts-up the occurrences of a large number of item-sets.

2.2. Bayesian Network Generation

Bayesian network, a new technique of data mining, received much attention in the manufacturing industry. A bayesian network is a directed acyclic graph. The network graphically represents the characteristics and the relations of knowledge with its nodes and arcs. This graphical representation helps to understand the knowledge. Moreover, it has formal probabilistic semantics for statistical manipulation.

When generating a bayesian network, many event data sampled from the environment such as a manufacturing process are read and the occurrences of the events are counted-up. It takes massive amount of time to generate a bayesian network, because it reads a large number of event data and counts-up the occurrences of a large number of events.

3. Log Based Consistency Mechanism

There are two types of DSMs (Distributed Shared Memory). One is a hardware DSM such as "DASH" [6] or "Alewife" [7]. Hardware DSM is one of the hardware architectures for scalable platforms. The

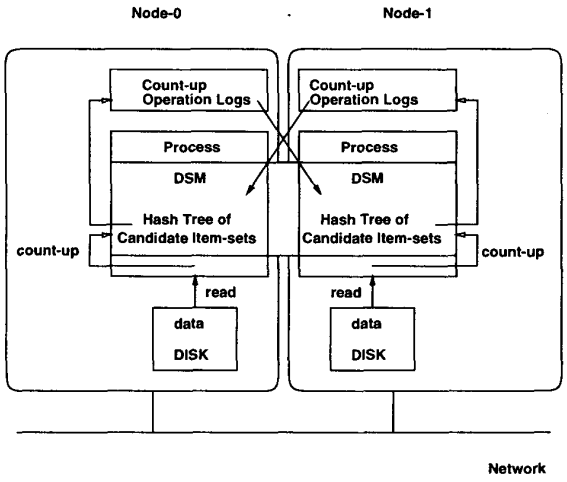

Figure 1. The DSM with LBC.

other is a software DSM such as "TreadMarks" [3] or "CVM" [4]. Software DSM is a technique to make a virtual scalable platform from various other platforms which may use a hardware DSM. Our proposal is to use a software DSM. We do not consider whether a hardware DSM is used in the platform or not. From now on, we refer the software DSM as a DSM.

3.1. LBC Mechanism

The DSM provides a virtual shared memory and enables users to develop programs with shared memory programming model in cluster-type distributed systems. The traditional DSM such as "TreadMarks" has the lazy release consistency [3] and multiple writer protocol [8]. They increase the performance compared to the former sequential consistency protocol by alleviating the problem caused by mis-matches between page size and application granularity which is known as false sharing. That is, they are effective even if the fields in the same page are frequently updated in the plural nodes.

Lazy release consistency and the multiple writer protocol are not sufficient for some data mining programs such as association rule mining or bayesian network generation. Those programs frequently update the same fields in the plural nodes. Therefore, if those programs are executed in the traditional DSM, large number of data items in the fields are frequently passed for keeping the consistency and they can not be executed simultaneouly. Such a problem is hardly alleviated by the lazy release consistency or the multiple writer protocol.

Most of the processing time of those programs is spent by the count-up operations. If we pay attention to the characteristics of the count-up operations, we notice that the operations follow the commutative law and the associative law. By taking advantage of the commutative and the associative features of the count-up operations, we propose the LBC (Log Based Consistency) mechanism. The LBC mechanism can execute the count-up operations simultaneously in the plural nodes while keeping the consistency.

For example, in the Apriori algorithm for mining association rules, a hash tree of the candidate item-sets is created in the DSM. After the hash tree is created in the DSM, transaction records are read and the occurrences of the item-sets are counted by all nodes. The operations have a large overhead in the traditional DSM, because it is necessary to pass the data in the fields at every count-up operation to keep up the consistency. Hence, the count-up operations can not be executed simultaneously. However, the count-up operations with commutative and the associative laws, it is possible to keep consistency without passing data at every count-up operation.

In the DSM with LBC, when the count-up operations are executed in a node, the count-up operation logs are recorded in a log buffer of the node, as shown in figure 1. These count-up operations can be executed simultaneously in the plural nodes without passing data at every count-up operation.

The count-up operation log has an address of the field which is counted-up in the node and to be counted-up in all other nodes. When the log buffer reaches full, the logs are sent to all other nodes by broadcasting. In other nodes, when they receive the logs, the logs are processed or the count-up operations are executed.

The LBC mechanism makes it possible to count-up the same fields simultaneously in the plural nodes without passing data at every count-up operation. Therefore, the LBC mechanism is effective for the count-up operations by taking advantage of the commutative law and the associative law.

3.2. Implementation of LBC Mechanism

Figure 2 shows the system structure of VISIONA which is the prototype of the DSM with LBC. The Resource Manager is a daemon process to manage shared memory spaces and semaphores which can be used in cluster-type distributed systems. The Space Managers are daemon processes to manage data in shared mem-

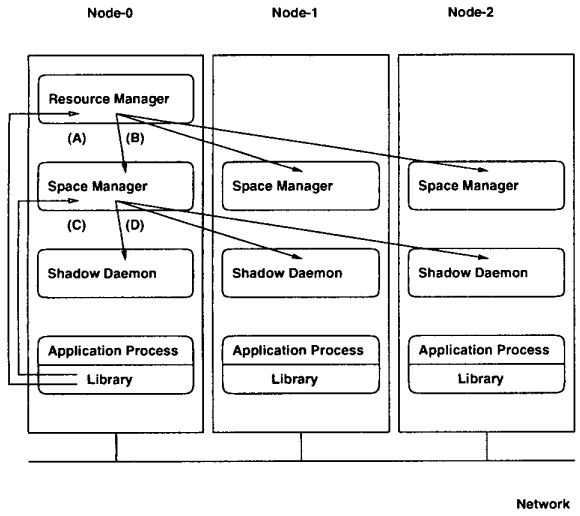

Figure 2. System structure of VISIONA.

ory spaces and send the count-up operation logs to all other nodes by broadcasting. The Shadow Daemons are daemon processes to receive the count-up operation logs sent from other nodes and to reflect them into the shared memory spaces in their own nodes.

When an application process creates a shared memory space or a semaphore, it sends the request to the Resource Manager (A). When the Resource Manager is requested to create a shared memory space, it requests the Space Managers and the Shadow Daemons in all nodes to create the shared memory space and map them (B). Semaphores are controlled by the Resource Manager.

When an application process attaches a shared memory space into its own address space, it sends the request to the Space Manager in its own node to query the address of the shared memory space. When an application process counts-up the occurrences of the candidate item-sets in the shared memory space, it sends the request to the Space Manager in its own node (C). The Space Manager saves the count-up request as a log in a log buffer. When the log buffer reachs full, the Space Manager sends them to the Shadow Daemons in all other nodes by broadcasting (D). When the Shadow Daemons receive the count-up operation logs, they reflect them into their own shared memory spaces.

3.3. Programming with LBC Mechanism

```
if (master_process) {                                                    (step-1)
    VISIONACreateSpace(key1, size);
    VISIONACreateSemaphore(key2);
}
spacep = VISIONAAttachSpace(key1);                                       (step-2)
/* the Apriori algorithm */                                              (step-3)
k = 1;
while (the candidate item-sets of k are not empty) {
    /* pass-k */
    while ((read a transaction record) != empty) {                       (step-4)
        while (an item-set of k can be selected from the transaction record) { (step-5)
            search the item-set in the hash tree;                        (step-6)
            /* address of the count-up field of the item-set: fieldp */
            VISIONACountup(fieldp);                                      (step-7)
        }
    }
    VISIONAFlushCountupBuffer(key1);                                     (step-8)
    VISIONAWaitSemaphore(key2, number_of_process);                       (step-9)
    make the large item-sets of k                                        (step-1C
    by comparing the support values of the candidate item-sets
    with the minimum support value;
    make the candidate item-sets of k+1                                  (step-11
    out of the large item-sets of k;
    k++;                                                                 (step-12
}
VISIONADetachSpace(key1);                                                (step-13
if (master_process) {                                                    (step-14
    VISIONADeleteSpace(key1);
    VISIONADeleteSemaphore(key2);
}
```

Figure 3. Pseudo code for association rule mining executed in VISIONA.

Figure 3 shows a pseudo code for association rule mining in the DSM with LBC. The functions with prefix "VISIONA" are provided by VISIONA library. In step-1, a master process creates a shared memory space and a semaphore. All the processes in the plural nodes of the cluster-type distributed system count-up the occurrences of the candidate item-sets in a hash tree on the shared memory space. Synchronization is done with the semaphore.

Steps 2 through 13 are executed by all the processes. In step-2, all the processes attach the shared memory space into their own address spaces. Steps 3 through 12 show the sequence of the Apriori algorithm. This sequence iterates with incrementing k until the candidate

item-sets of k become empty.

Steps 4 through 7 iterate for all the transaction records. In step-4, a transaction record is read. In step-5, an item-set of k is selected from the transaction record. In step-6, the item-set is searched in the hash tree. In step-7, the occurrence of the item-set is count-upped.

In step-8, the count-up operation logs remained in the log buffer are flushed to all other nodes. In step-9, all the processes synchronize with the semaphore. In step-10, after completion of the pass-k, the large item-sets of k are made. In step-11, the candidate item-sets of k+1 are made out of the large item-sets of k. In step-12, the variable "k" is incremented and the sequence iterates from the step-4.

When the candidate item-sets of k reach empty, the Apriori algorithm terminates. In step-13, all the processes detach the shared memory space. Finally, in step-14, the master process deletes the shared memory space and the semaphore.

This is a pseudo code for association rule mining, and it can be executed simultaneously in cluster-type distributed systems. But the code is not only for cluster-type distributed systems but also for SMP-type parallel computers.

4. Evaluation

We have implemented a program of association rule mining in VISIONA to evaluate the performance of the DSM with LBC. We synthesized the data for evaluation according to the way specified in [2] under the following conditions.

- total number of items: 400
- mean record length: 100B
- mean number of items in a record: 10
- total number of records: 600,000 (approximately 60MB)
- minimum support value: 0.4

To evaluate the effect of the DSM with LBC, we compared its performance with SMP-type parallel computer performance. Four uni-processor computers (represented as UPs) and an SMP-type parallel computer (represented as SMP) with the following specifications are used for this evaluation.

Table 1. Performance of association rule mining.

type	number of processors	pass-1 [sec.]	pass-2 [sec.]	pass-3 [sec.]	pass-4 [sec.]	total [sec.]
SMP	1	44	192	84	38	358
	2	29	99	44	25	197
	3	27	70	31	24	152
	4	25	61	29	23	138
DSM	1	73	331	133	65	602
	2	45	172	68	33	318
	3	34	119	46	23	222
	4	29	91	36	17	173

Table 2. Performance of association rule mining (normalized).

type	number of processors	pass-1	pass-2	pass-3	pass-4	total
SMP	1	1.00	1.00	1.00	1.00	1.00
	2	0.66	0.52	0.52	0.66	0.55
	3	0.61	0.36	0.37	0.63	0.42
	4	0.57	0.32	0.35	0.60	0.39
DSM	1	1.00	1.00	1.00	1.00	1.00
	2	0.62	0.52	0.51	0.51	0.53
	3	0.47	0.36	0.35	0.35	0.37
	4	0.40	0.27	0.27	0.26	0.29

- UP:
  a uni-processor computer (processor: 70MHz MB86904(SPARC), memory: 32MB)
- SMP:
  a four-way SMP computer (processor: 100MHz RT625(HyperSPARC), memory: 128MB)

Four UPs are inter-connected via a 10Mbps ethernet. And the SMP is also inter-connected with the UPs via the same ethernet. A data file is stored in a local DISK of each computer in each case. The DISK performance of the UP and the SMP is different.

Figure 4 shows a pseudo code for association rule mining in the SMP. This is basically same as the code shown in figure 3. The main difference is step-7. In step-7, the occurence of the item-set is count-upped by incrementing the shared variable in a critical section with mutex_lock and mutex_unlock.

Table 1 shows the performance of association rule mining executed by the SMP and the UPs using the DSM with LBC (represented as DSM). Table 2 shows the results normalized by those of the UP. According to the results, speedup by increasing the number of processors of the DSM with LBC is greater than or equal to that of SMP-type parallel computer. Because there are no more UPs, we use four UPs and the SMP together for further evaluation.

Table 3 shows the performance of parallel and distributed system (the UPs and the SMP) using the DSM with LBC. The data file is equally divided without considering the difference of processor speed of each computer. The result of six processors is measured by four

```
main(atgc, argv) {                                              (step-1
  spacep = malloc(size);
  for (i = 0; i < N; i++) {
    if (!fork()) {
      apriori();
      return;
    }
  }
  apriori();
}

apriori() {                                                     (step-2
  /* Apriori algorithm */                                       (step-3
  k = 1;
  while (candidate item sets of k-th pass are not empty) {
    /* step of k-th pass */
    while ((read data) != empty) {                              (step-4
      while (item sets of k items are selectable from the data) {  (step-5
        search the item-set in the hash tree;                   (step-6
        mutex_lock(&mt[fieldp % 400]);                          (step-7
        *fieldp++;
        mutex_unlock(&mt[fieldp % 400]);
      }
    }
    make the large item-sets of k                               (step-8
    by comparing the support values of the candidate item-sets
    with the minimum support value;

    make the candidate item-sets of k+1                         (step-9
    out of the large item-sets of k;

    k++;
  }
}
```

Figure 4. Pseudo code for association rule mining executed in SMP.

UPs and two processors of the SMP. The result of eight processors is measured by four UPs and four processors of the SMP. Table 4 shows the results normalized by the result of the UP.

When the evaluation is done by processors with different speeds, the final result depends on the processor with the lowest speed, if the data file is equally divided for all the processors. Compared to the processor with the lowest speed, sufficient speedup is achieved.

Table 5 shows the count-up log size and the execution time when executing association rule mining. The maximum log size per time is 0.30MB/second. It is small enough compared to the transfer rate of 10Mbps ethernet.

Table 3. Performance of association rule mining in the parallel and distributed systems.

number of processors	pass-1 [sec.]	pass-2 [sec.]	pass-3 [sec.]	pass-4 [sec.]	total [sec.]
6	25	64	26	13	128
8	22	50	19	10	101

Table 4. Performance of association rule mining in the parallel and distributed systems (normalized).

base of normalization	number of processors	pass-1	pass-2	pass-3	pass-4	total
UP	6	0.34	0.19	0.20	0.20	0.21
	8	0.30	0.15	0.14	0.15	0.17

So even if the scale of association rule mining is increased, the log size is small enough compared to the transfer rate of 100Mbps ethernet which is popular in today's computing environment. So we do not consider that the broadcast of the count-up logs creates a bottleneck of the performance.

Finally, we compared the performance of the DSM with LBC to the performance of CVM. CVM is the DSM with lazy release consistency and multiple writer protocol. In this case, total number of records is only 1,000 (approximately 100KB), because of the bad performance of CVM.

Figure 5 shows a pseudo code for association rule mining in CVM. This is basically same as the code shown in figure 3. The main difference is step-7. In step-7, the occurence of the item-set is count-upped by incrementing the shared variable in a critical section with cvm_lock and cvm_unlock.

Table 6 shows the performance of association rule mining executed in CVM and VISIONA. It shows that CVM or the DSM with lazy release consistency and multiple writer protocol cannot get the speedup by increasing the number of processors. On the other hand VISIONA or the DSM with LBC can get it.

5. Conclusion

In this paper, we presented the implementation and the evaluation of the LBC mechanism. The LBC is a

Table 5. Number and size of the count-up operation logs.

pass	log size [MB]	execution time [sec.]	log size/execution time [MB/sec.]
1	6.7	22	0.30
2	11.7	50	0.23
3	0.7	19	0.04
4	0.0	10	0.00

```
main(atgc, argv) {                                              (step-1)
  cvm_startup(argc, argv);
  spacep = cvm_alloc(size);
  cvm_create_procs(apriori)
  apriori();
  cvm_finish();
}

apriori() {                                                     (step-2)
 /* Apriori algorithm */                                        (step-3)
 k = 1;
 while (candidate item sets of k-th pass are not empty) {
   /* step of k-th pass */
   while ((read data) != empty) {                               (step-4)
     while (item sets of k items are selectable from the data) { (step-5)
       search the item-set in the hash tree;                    (step-6)
       cvm_lock(&mt[fieldp % 400]);                             (step-7)
       *fieldp++;
       cvm_unlock(&mt[fieldp % 400]);
     }
   }
   cvm_barrier(0);                                              (step-8)
   make the large item-sets of k                                (step-9)
   by comparing the support values of the candidate item-sets
   with the minimum support value;
   make the candidate item-sets of k+1                          (step-1C
   out of the large item-sets of k;
   k++;
 }
}
```

Figure 5. Pseudo code for association rule mining executed in CVM.

consistency mechanism on the DSM for scalable data mining in both parallel and distributed computing environments. [1] and [2] show the basic algorithm of association rule mining which is popular among the data mining research community. [9] shows a modification of [1] and [2] to decrease the number of the candidate item-sets to increase the performance. [10] shows the parallel algorithm of [1] and [2] for cluster-type distributed systems with message passing programming model. [5] shows bayesian network generation which is a new technique of data mining.

New techniques for data mining and speedup are important research areas. In the past, new techniques or speedup is performed for each application independently. Our approach is different from the past research. We use DSM as a common tool for scalable data mining programs. The objective is to develop scalable

Table 6. Performance of association rule mining in CVM and VISIONA.

type	number of nodes	pass-1 [sec.]	pass-2 [sec.]	pass-3 [sec.]	pass-4 [sec.]	total [sec.]
CVM	1	1	4	2	1	8
	2	15	21	4	1	41
	4	39	44	9	1	93
VISIONA	1	1	4	2	1	8
	2	1	2	1	1	5
	4	1	2	1	1	5

data mining programs efficiently. To decrease the overhead of the DSM, we proposed the LBC mechanism on the DSM. Also, this paper described VISIONA, a prototype to implement the DSM with LBC.

To evaluate the effectiveness of the DSM with LBC, we have implemented VISIONA both in PC clusters and UNIX computer clusters. Programs of association rule mining and bayesian network generation has been also implemented in VISIONA. According to the results of the evaluation of the DSM with LBC, speedup by increasing the number of processors on the DSM with LBC is greater than or equal to that of SMP-type parallel computer. In the future, we will improve the VISIONA as a DSM system with multiple consistency protocol including the LBC mechanism. The goal is to increase the adaptability to many different data mining programs.

References

[1] R. Agrawal, T. Imielinski, A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proceedings of ACM SIGMOD, pp.207–216, May 1993.

[2] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules," Proceedings of the 20th VLDB Conference, pp.487–499, September 1994.

[3] C. Amza, A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu, W. Zwaenepoel, "TreadMarks: Shared Memory Computing on Networks of Workstations," IEEE COMPUTER, Vol. 29, No. 2, pp.18–28, February 1996.

[4] P. Keleher, "The relative importance of concurrent writers and weak consistency models," Proceedings of the 16th International Conference on Distributed Computing Systems, pp.91–98, May 1996.

[5] D. Heckerman, "Bayesian Networks for Knowledge Discovery," Advances in Knowledge Discov-

ery and Data Mining, AAAI Press/The MIT Press, pp.273–305, 1996.

[6] D. Lenosla, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta, J. Hennessy, "The DASH Prototype: Implementation and Performance," Proceedings of the 19th International Symposium on Computer Architecture, pp.92–103, May 1992.

[7] F. T. Chong, B. Lim, R. Bianchini, J. Kubiatowicz, "Application Performance on the MIT Alewife Machine," IEEE COMPUTER, Vol. 29, No. 12, pp.57–64, December 1996.

[8] P. Keleher, A.L. Cox, S. Dwarkadas, W. Zwaenepoel, "An Evaluation of Software-Based Release Consistent Protocols," Journal of Parallel and Distributed Computing, Vol. 29, pp.126–141, October 1995.

[9] J. S. Park, M. Chen, P. S Yu, "An Effective Hash-Based Algorithm for Mining Association Rules," Proceedings of ACM SIGMOD, pp.175-186, June 1995.

[10] E. Han, G. Karypis, V. Kumar, "Scalable Parallel Data Mining for Association Rules," Proceedings of ACM SIGMOD, pp.277–288, May 1997.