# Distributed Shared Memory with Log Based Consistency for Scalable Data Mining

Hideaki Hirayama
Graduate School of Information Systems, The University of Electro-Communications and
Computer & Network Development Center, Toshiba Corporation
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

*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

Data mining discovers the knowledge from massive amounts of data. The knowledge required varies from industry to industry. Therefore, many industries are developing specialized data mining programs to satisfy their needs and requirements. A popular example is data mining in retail organization. In general, large number of data mining programs process massive amounts of data available in SMP(Symmetric Multi-Processor)-type parallel computers or cluster-type distributed systems.

A scalable data mining program should be able to run in both SMP-type parallel computers and cluster-type distributed systems. The development of scalable data mining programs will be easier with the use of software DSM (Distributed Shared Memory) such as [2] [3]. DSM enables the development of scalable programs based on the shared memory programming model rather than the message passing programming model even in cluster-type distributed systems.

Association rule [1] is the most popular technique used in data mining. Typical association rule based data mining programs frequently modify the same data fields by count-up operations. Thus the overhead of a software DSM is very large for data mining programs. If those programs are executed in the software DSM, many data items in the fields are frequently passed among the multiple nodes for keeping up the consistency. So we propose a new consistency mechanism, the LBC (Log Based Consistency) mechanism, on the software DSM. It is especially designed for scalable data mining. The LBC mechanism keeps up the consistency by broadcasting the count-up operation logs among the multiple nodes. To evaluate the performance of the LBC mechanism, we have developed a scalable data mining system, VISIONA (Virtual Shared Memory Environment for Scalable Data Mining Applications).

## 2 Large Scale Data Mining

Association rule based mining [1] in the retail organization is an example of data mining programs processing massive amounts of data. The association rule defines two measures, a support value and a confidence value. The support value is a measure based on the ratio of the transaction record including an item X. The



0-7695-0368-3/99 $10.00 © 1999 IEEE

confidence value is a measure based on the ratio of the transaction record including an item X 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. The steps outlined process the item-sets (sets of items) having k integer number of items, which are called pass-k. And they are iterated after k is incremented.

1. Read all transaction records and count-up the occurrences of the item-sets having k items. They are called the candidate item-sets of k.

2. Select the item-sets from the candidate item-sets of k, which support values 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.

## 3 Log Based Consistency Scheme

There are two types of DSMs (Distributed Shared Memory). One is a hardware DSM such as "DASH" [4] and the other is a software DSM such as "TreadMarks" [2] or "CVM" [3]. We propose to use software DSM but not to use hardware DSM.

The software DSM or simply DSM provides a virtual shared memory and enables users to develop programs with shared memory programming model in cluster-type distributed systems. The traditional DSMs such as "TreadMarks" has the lazy release consistency and the multiple writer protocol [2]. 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 the false sharing. That is, they are effective even if the fields in the same page are frequently updated in multiple nodes.

The lazy release consistency and the multiple writer protocol are insufficient and ineffective for association rule based data mining programs and others. Those programs frequently update the same fields in the multiple nodes. Therefore, if those programs are executed in the traditional DSM, many data items in the fields need to be passed frequently for keeping up the consistency. Moreover, they can not be executed simultaneously. Such a problem is hardly alleviated by the lazy release consistency or the multiple writer protocol.

It is observed that most of the processing time of those programs is spent on the count-up operations. These count-up operations follow the commutative law and the associative law. By taking advantage of the features of commutative law and the associative law of the count-up operations, we propose the LBC (Log Based Consistency) scheme. In the LBC scheme count-up operations can be executed simultaneously in the multiple nodes while keeping up the consistency.

In the Apriori algorithm for association rule based mining, 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 items in the fields at every count-up operation for keeping up the consistency. Those count-up operations can not be executed simultaneously. However, performing count-up operations with the commutative law and the associative law, it is possible to keep up the consistency without passing data items at every count-up operation.

In the DSM with LBC implementation, 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. The count-up operations can also be executed simultaneously in the multiple nodes without passing data at every count-up operation.

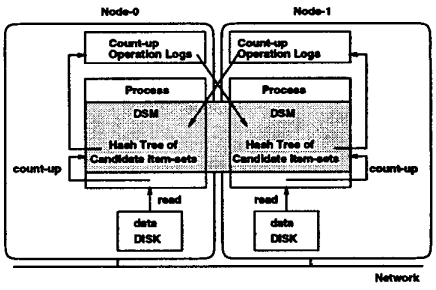


Figure 1. The DSM with the LBC scheme.

The count-up operation log has an address 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. When the other nodes 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 multiple nodes without passing data at every count-up operation. There-

fore, the LBC mechanism is very effective for the count-up operations. The mechanism make use of the characteristics of commutative and associative laws.

## 4 Implementation

Figure 2 shows the system structure of VISIONA which is a prototype of the DSM with the LBC scheme. Each Manager unit is a daemon process. The Resource Manager manages shared memory spaces and semaphores which can be used in cluster-type distributed systems. The Space Managers manage the data in shared memory spaces and send the count-up operation logs to all other nodes by broadcasting. The function of Shadow Daemons is to receive the count-up operation logs sent from other nodes and to reflect them into the shared memory spaces in their own nodes.

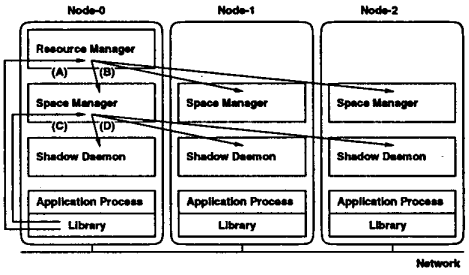


Figure2. System structure of VISIONA.

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 wants 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) accordingly. 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 reaches 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.

## 5 Programming

Figure 3 shows the pseudo code for association rule based mining in the DSM with LBC. The functions with prefix "VISIONA" are provided by the VISIONA library. In step-1, a master process creates a shared memory space and a semaphore. All the processes in the multiple 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.

```
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-10
    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. Association rule mining (pseudo code).

- Step-2 through step-13 are executed by all the processes.
- In step-2, all the processes attach the shared memory space into their own address spaces.
- Step-2 through step-13 are executed by all the processes.
- Step-3 through step-12 show the sequence of the Apriori algorithm, the sequence iterates with incrementing k until the candidate item-sets of k become empty.
- Step-4 through step-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 counted-up. 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 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 the pseudo code for association rule mining. It can be executed simultaneously in cluster-type distributed systems.

## 6 Evaluation

We have implemented a program of association rule mining in VISIONA to compare the performance of the DSM with LBC. We synthesized the data for evaluation according to the specifications given in the [1]. Specifically, 400 total number of items, 100B mean record length, 10 mean number of items in a record, 600,000 total number of records (approximately 60MB) and 0.4% minimum support value.

To evaluate the performance of DSM with LBC, we compared the performance with four uni-processor computers (represented as UPs) and an SMP-type parallel computer (represented as SMP). UP has a 70MHz SPARC processor and 32MB memory. SMP has four 100MHz HyperSPARC processor and 128MB memory. A data file is stored in a local DISK of each computer in every case. The DISK performance of the UP and the SMP is different.

Table 1 shows the performance of association rule based 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, speed up obtained by increasing the number of processors of the DSM with LBC is greater than or equal to that of SMP-type parallel computer.

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. Normalized performance.

| 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 |

## 7 Conclusion

This paper presented the use of DSM as a common tool to develop scalable data mining programs. This is an efficient way of developing scalable data mining programs. To decrease the overhead of the DSM, we proposed the LBC mechanism on the DSM.

To evaluate the effect of the DSM with the LBC, we have implemented VISIONA in UNIX computer clusters. Performance results of the evaluation shows a speed up for increasing number of processors of the DSM with LBC. The speed up is greater than or equal to that of SMP-type parallel computer.

## References


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

[2] 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.

[3] P. Keleher, "The Coherent Virtual Machine," Technical Report Maryland TR93-215, Department of Computer Science, University of Maryland, September 1995.

[4] 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.