論文 / 著書情報
Article / Book Information

Title	Treatment of Rules in Individual Metadata of Flexible Contents Management
Author	Kensuke OTA, Dai Kobayashi, Takashi Kobayashi, Ryo Taguchi, Haruo Yokota
Journal/Book name	Proc. of International Special Workshop on Databases For Next Generation Researchers (SWOD 2006), Vol. , No. , pp. 77-82
Issue date	2006, 4
DOI	10.1109/ICDEW.2006.153
URL	http://www.ieee.org/index.html
Copyright	(c)2006 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

Treatment of Rules in Individual Metadata of Flexible Contents Management

Kensuke Ohta[1], Dai Kobayashi[2], Takashi Kobayashi[3], Ryo Taguchi[4], and Haruo Yokota[3,2]

[1] Faculty of Engineering
Tokyo Institute of Technology

[2] Grad. School of Info. Sci. and Eng.
Tokyo Institute of Technology

[3] Global Scientific Info. and Comp. Center
Tokyo Institute of Technology

[4] Science and Technical Research Laboratories
Japan Broadcasting Corporation

{ohta@de.cs,daik@de.cs,tkobaya@gsic}.titech.ac.jp, taguchi.r-cs@nhk.or.jp, yokota@cs.titech.ac.jp

Abstract

The properties of contents stored in a computer system are very wide while the data volume treated in the system becomes very large. It is important to treat each stored object in different manners to reflect its properties in the data management for the large amount of stored data. To satisfy the requirement, we propose a method for the autonomous management based on ECA rules stored in metadata of the contents. We study the feasibility of treating a large number of ECA rules corresponding to the number of stored objects. Because the cost for evaluating conditions in the rules becomes dominant to the system performance when the number of objects increases, we divide the conditions into two types, previously evaluable conditions and runtime evaluable conditions, and construct a discrimination network for the previously evaluable conditions of each event to reduce the cost for processing the rules. We implement the methods in the autonomous disk system, a high functional storage system we proposed, and evaluate the efficiency of them.

1 Introduction

The efficient management for large amount of data has been strongly required because the data volume stored in a computer system increases explosively. At the same time, the variety of stored contents has also been very wide. To realize the efficient data management, these various stored objects should be treated in different manners to reflect their properties.

As an approach for reflecting the property of stored contents in data management, the concept of Information Lifecycle Management (ILM) [1, 2, 9] has been proposed. The basic mechanism of the ILM is to migrate the stored objects between storage devices having different cost-performance features, based on given system-wide policies for access frequencies and time information. However, the strategy determined by the system-wide policies is less flexible for the diversity of actual contents. The finer grained control considering each content is required.

To satisfy the requirement, we propose a method that is to specify the control strategy for each stored object in its metadata. We use the ECA (Event-Condition-Action) rules to express the control strategy in the metadata. The ECA architecture was originally proposed for the active databases to move the reactive behavior from database applications into the DBMS using declarative rule definitions [4, 8]. The framework of declarative specification to invoke actions under predefined conditions for a certain event is also useful for managing stored contents. The desired actions for managing them can be invoked if the state of storages or stored objects satisfies given conditions when a specified event occurs. Our methods enable the fine-grain flexible control corresponding to each content.

To implement the ECA-rule based contents management, we have to consider the cost of processing the rules. The cost for evaluating conditions in the rules becomes dominant to the system performance when the number of stored objects increases. It is important to reduce the cost of condition evaluation.

To make the rule processing for stored contents efficient, we propose a method of dividing the conditions of an ECA rule into two types, *previously evaluable conditions (PEC)* and *runtime evaluable conditions (REC)*, and construct a discrimination network for the PEC of each event. It omits the redundant condition evaluation, and reduces the number of rules to be evaluated when the event occurs. We also

propose a method that is to prepare *true rule tables* and *false rule tables* to reduce the cost for updating the discrimination network.

We implement the proposed methods on the autonomous disk system, a high functional storage system we proposed [10], and show efficiency of the proposed methods by experiments on the autonomous disk system.

The rest of the paper is organized as follows. At first, in Section 2, we describe assumptions related to the storage system, metadata and the ECA architecture we use. Then, we explain the ECA rules for managing stored contents in Section 3. Section 4 describes a naïve approach for processing rules and the proposed methods. Then, the effectiveness of the proposed method is considered through the experiment on the autonomous disk system in Section 5. The related works is described in Section 6, and the conclusions and future work are described in Section 7.

2 Assumptions

At first, in this section, we briefly describe assumptions of a storage system, metadata and ECA architecture to prepare the consideration of ECA rules in metadata to express the fine-grain control strategy.

2.1 Storage System

We assume a high functional storage system to handle the ECA rules in metadata. We have proposed the autonomous disk system as a high functional storage system [10]. It has an instruction set for handling the internal rule. Flexible descriptions become possible for content managements by combining those instructions in ECA rules. However, the proposed method can be also applied to other type of high functional storage systems.

2.2 Metadata

The metadata is an attribute information for a stored object. In other words, each stored object has its own metadata. We assume that the metadata contains the file attribute supported by standard file systems, such as POSIX1003.1 specification and NFS version 3 [3]. It may also contains information defined by users, such as copyright expiration date and relationship to other objects. In this paper, we do not describe about the details of metadata, but assume that ECA rules can also be stored in metadata of each object.

2.3 ECA Rule

The ECA architecture was originally proposed for the active databases [4]. It has a number of variants [8]. Here, we assume that an ECA rule has three components: an event, a condition, and an action. The *event* part of the rule describes a happening to which the rule may be able to respond. The *condition* part of the rule examines the context in which the event has taken place. The *action* describes the task to be carried out by the rule if the relevant event has taken place and the condition has been evaluated to *true*. We assume that users know the commands of the high functional storage system to be written in the ECA rules.

3 ECA Rule for Contents Management

To apply the ECA rules to management of stored contents, we have to consider what kind of information should be specified in the rules.

- In *event*, basic operations for stored contents, such as insertion and deletion, the timer interrupt and the events defined by applications are written.
- In *condition*, the restrictions concerning the metadata of contents, presence of contents, and condition of parameter decided when an event occurs are written.
- In *action*, the tasks using the internal command offered by the high functional storage system are written. If the internal commands that touch the system directly are being offered enough, the range of describable processing will become wide.

The following are example rules for handling stored contents.

Example 1 This is a rule for changing the permission of opening web contents to the public. When the image used by Web contents is open to the public at limited period by the reason of copyright, this rule is used to release the inspection limitation on the day of opening to the public:

Event At 2006/12/31 24:00

Condition If the permission of inspection from the outside is denied

Action Change the permission of the content to allow access from outside

Example 2 This is a rule to delete contents that relates to the deleted image content when the image content is deleted. If there are one or more voice contents and title text whose access frequencies are below the threshold related to the deleted image content, they will be deleted when the image content is deleted:

Event When image content is deleted

Condition If there are one or more voice contents and title text whose access frequencies are below the threshold related to the deleted image content

Action Delete the concerned contents

Thus, the flexible contents management is expressed by the rules in the metadata of each stored object.

4 Rule Processing

In this section, we propose methods to reduce the time for evaluating the condition in a large amount of rules. Before explaining the proposed method, we describe a naïve approach to process rules for comparison. Then, we describe the methods to shorten the evaluation and update operations.

4.1 Naïve Approach

A rule manager is required to register and execute rules to implement a system. When a rule is added to the system, the rule manager registers the rule. At this time, this rule is mapped to the event in the rule manager. When the event occurs, this rule becomes one of the candidate rules to be invoked for the event.

When an event occurs, we get the invocation candidate rules related to this event. The condition of each rule is evaluated. If the evaluation result of the condition is *true*, the tasks described in the action are executed. In this approach, the cost to evaluate the condition in a large amount of rules is expensive. We should evaluate the condition for all rules related to the event whenever the event occurs. However, scanning all rules may waste of many resources.

4.2 Proposed Method

4.2.1 Division of Condition

A conditions of a rule can be divided into two parts: *previously evaluable condition (PEC)* and *runtime evaluable condition (REC)*. PEC can be evaluated regardless of the event occurrence. REC should be evaluated when the event occurs. PEC can exist as a condition of contents that have already been stored. REC can be generated because of the parameter decided for the first time when the event occurs.

For example, the condition "If the permission of inspection from the outside is denied" in the rule of Example 1 is a PEC. The condition "If the voice contents or the title text whose access frequency is below the threshold exists" in a part of the rule in Example 2 is a REC.

PEC may be composed of multiple conditional expressions. It can be divided into partial conditions, *sPEC(sub-PEC)*s including a single metadata item. We evaluate sPECs when the metadata of contents is updated, and evaluate REC of the rule whose evaluation of PEC is *true* when an event occurs. Because the invocation candidate rules are reduced, the rule invocation time is shortened when an event occurs.

4.2.2 Outline of Execution

Discrimination network constructed by registered rules decides invocation candidate rules and evaluates RECs of those rules when an event occurs. Figure 1 is the outline of the process using the discrimination network. The discrimination network is composed of one root node, table nodes, join nodes, firing nodes and arcs that connect nodes.

In the root node, the condition of the registered rule is divided. The table node of each kind of the metadata is

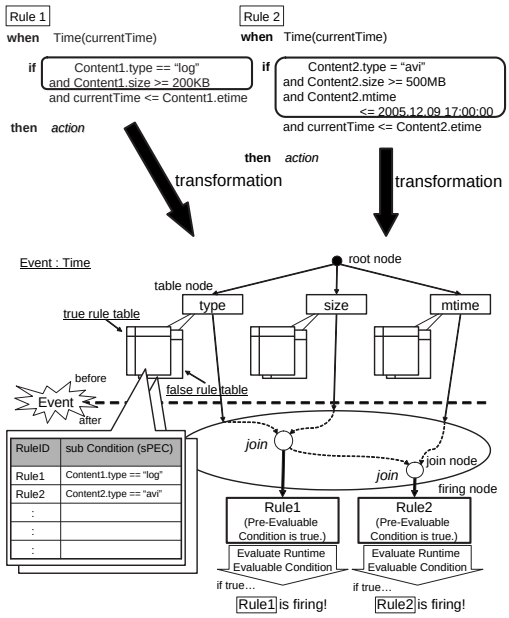

Figure 1. Execution Outline of proposed method

generated, and it has a *true rule table* on which rules are registered rules when the sPEC evaluation is *true*, and a *false rule table* on which rules are registered rules when the sPEC evaluation is *false*. In the join node, two *true rule set*s passed from table nodes or other join nodes are joined. When registering or deleting a rule, the join node is generated or deleted. In the firing node, RECs of rules passed from the join nodes are evaluated, and the *action*s of rules will be executed when their REC evaluations are *true*.

4.2.3 Operation on Event Occurrence

The discrimination network is generated according to the number of event types. In the discrimination network corresponding to the occurred event, the following operations are executed.

1. Each table node passes a *true rule set* to the join node. This *true rule set* is a rule group registered in the *true rule table*.
2. In the join node, rules included in two rule sets are extracted.
3. These rules are passed to the firing node.
4. In the firing node, RECs of the passed rules are evaluated. If the results are *true*, the *action*s are executed.

For example, in Figure 1, the *true rule set*s of the table nodes "type" and "size" are passed to the join nodes when the event is generated. Rules included in those *true rule set*s are extracted and passed to the firing nodes. In the firing nodes, rules passed from the join node are invoked if

their REC evaluation are *true*. If Rule 1 is included in those rules, the action of Rule 1 will be executed. Similarly, Rule 2 can be executed in the same way.

4.2.4 Operation for Registered or Deleted Rules

When a rule is registered:

1. In the root node, the condition is divided into PECs and RECs, and a PEC is divided to sPECs. The sPECs are mapped to the rule including themselves, and they are passed to the corresponding table nodes depended on metadata item included in their sPECs.

2. In the table node, the passed sPEC is evaluated, and if its result is *true*, it will be registered to the *true rule table*; otherwise, it will be registered to the *false rule table*.

3. The join node is generated if it does not exist.

For instance, in Figure 1 , sPEC "The file type of Content 1 is log" is registered in the table node "type", and sPEC "The size of Content 1 is 200KB or more" is registered in the table node "size" when Rule 1 is registered. The join node will be newly generated, if there is no join node that joins "type" and "size".

When the rule is deleted, the corresponding sPECs will be deleted from the table node, and the corresponding join node will be deleted.

4.2.5 Operation for Updated Content

When a sPEC of the condition concerning updated contents exists, in the table nodes including the sPEC, it is re-evaluated and registered to the *true rule table* or *false rule table* according to the result of the re-evaluation. The rule does not need to be restructured at the update by doing management of sPEC with two kinds of tables. Thus, the update processing time can be reduced.

5 Experiment

The proposed method is implemented on the autonomous disk system to show their effectiveness.

5.1 Environment

The experiment is conducted on the autonomous disk system as a decentralized storage technology proposed. An autonomous disk is imitated by using Java on Linux cluster. The experimental environment is composed of a network switch having enough backbone performance and eight Linux boxes shown in Table 1.

5.2 Scenario

We measured the rule processing time when the event occurs by the proposed method and the delay time when contents are updated by using the proposed method.

Table 1. Experimental Environment

CPU	AMD Athlon XP-M1800+ (1.53GHz)
Memory	PC2100 DDR SDRAM 1GB
Disk	TOSHIBA MK3019GAX (30GB, 5400rpm, 2.5inch)
OS	Linux 2.4.20
Java VM	Sun J2SE SDK 1.5.0_03 Server VM

```
when   time(currentTime)

  if       this.item == "video"
     and   o1.item == "audio"
     and   this.title == o1.title
     and   currentTime <= this.etime
     and   currentTime <= o1.etime

then   action
```

Figure 2. Rule used in the experiment

To set the rule used in the experiment, the following situations are assumed. When various stream data are stored, we would like to store those data in one container format as much as possible. If there are video data and voice data of the same title and the expiration date of each data is not due when the event occurs, the system will move those data to other storages and edit them there automatically.

Figure 2 shows the rule used in the experiment. Each one of the first three conditions represents a sPEC, and the whole three conditions in Figure 2 corresponds to a PEC, and the last two ones denote RECs. We describe a rule in Figure 2 to each content. To avoid the fluctuation of processing time by the difference of the *action*, the *action* is not executed in the experiment.

5.3 Comparison of Rule Processing Time

We compare the rule processing time of our method with that of the processing method by scanning all rules. The rule processing time when an event occurs is measured with changing the ratio of the rule when its PEC evaluation is *true*.

We provide a rule to each content, and show the effect of narrowing by evaluation of PEC. All the REC evaluations are assumed to be *true*. We measure the rule processing time when the number of registration rules is 100, 1000, 3000, 5000, 8000 and 10000.

Figure 3 and Figure 4 show the rule processing time when the probability of the its PEC evaluation to be *true* is 10%, and 5% among rules with the possibility of invocating when the event occurs.

The rule processing time of the proposed method depends on the ratio of PEC and REC in the condition of rule. Therefore, we measure the rule processing time when the ratio of PEC and REC changes.

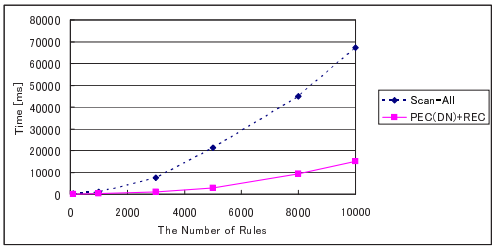

Figure 3. Rule processing time when probability of PEC evaluation to be ture is 10%

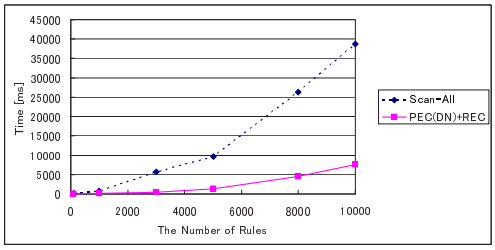

Figure 4. Rule processing time when probability of PEC evaluation to be ture is 5%

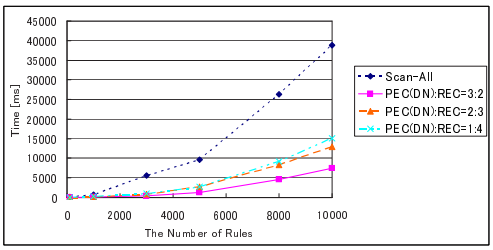

Figure 5. Rule processing time when ratio of PEC and REC are changes

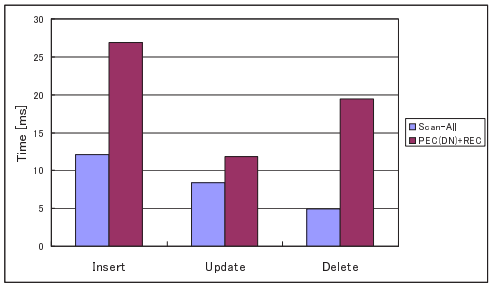

Figure 6. Average processing time of insertion, update, and deletion

Figure 5 shows the rule processing time when the ratio of the rule that the evaluation of the PEC is *true* is 5%. We measure the rule processing time when the ratio of PEC and REC is 3:2, 2:3, and 1:4. We intentionally changed the ratio of PEC and REC included in a rule. The rule of PEC:REC=3:2 is the same as the rule used in the experiment (Figure 2).

5.4 Comparison of Contents Updating Time

We measured the latency delay by implementing of the proposed method when the metadata of contents is updated. In this experiment, we measure three operations (insertion, update, and deletion of contents). This is because the updating of discrimination network is necessary when the metadata of contents is changed by each operation.

Figure 6 shows the result of the average processing time for 1000 insertions, updates, and deletions of contents. These operations are done from one client to an autonomous disk sequentially.

5.5 Discussion

5.5.1 Rule Processing Performance

In the experiment in Section 5.3, our method always reduces the rule processing time when an event occurs.

In the rule processing method scanning all rules, the rule processing time increases as the number of rules increases. This is because the condition evaluation for all rules relative to the event must be executed.

In the proposed method, the processing time also increases as the number of rules increases. This is because the number of rules whose PEC evaluation is *true* increases, and the join processing time increases according to the increase of the number of rules.

In other experiment, the processing time increases as the ratio of REC included in the condition increases. However, because the processing rule number is decreased by narrowing the invocation candidate rules, the rule processing time of our method is less than that of the method scanning all rules.

5.5.2 Contents Updating Performance

In the update operation, the latency delay for implementing of the proposed method is only 3ms, because we only need to re-evaluate the sPECs and register them to the *true rule table* or *false rule table* in the table node including the sPECs that should be changed.

As for the insertion and deletion of contents, the latency delay for implementing of the proposed method is 15ms and 16ms. This is because not only the renewal of the table node but also the registration or deletion of the rule from the discrimination network, and generation or deletion of the join node are needed. However, the delay can be acceptable compared with the imprement of rule processing time.

6 Related Work

The research on content-aware storage management is one of hot topics. OSD (Object-Based Storage Device) is the mechanism that the data under storage is treated in each content [6]. OSD manages the file, not by block unit, in each object with high abstraction level. In the description

of the metadata and the control rule in each content, the compatibility of OSD to our reseach is good.

Policy-based ILM provides the control policy to the entire system, and the storage management has been proposed to the automation of ILM using the centralized management system [2]. It is difficult to do the control reflecting the state and the access tendency of individual contents in detail because each policy targets all contents.

The storage management using the ECA rule has been researched in active databases [8], and the ECA rule itself has been well studied in the active databases [4, 8].

In the rule-based systems such as a production system [5], the Rete algorithm [7] is widely used to evaluate the condition of rules at high speed. An rule-based system is demanded to retrieve a corresponding data set at high speed from a large amount of data. Therefore, the Rete network based on the Rete algorithm has been used in the rule-based system.

The Rete algorithm is suitable for executing the evaluation whether the condition of the complex rule is satisfied each contents at high speed in the situation in which a large amount of contents are stored. The proposed method aims to evaluate the condition of rules given to the contents at high speed in the situation when the number of rule is more than that of the contents.

7 Conclusions and Future Work

To realize the fine-grain flexible contents management in high functional storage systems, we propose a method to describe the controle rule in the metadata of individual stored object. The rule specification for each object enlarges the number of rules treated in the system. Because registered rules contain many unsatisfied conditions, the cost of scannnig all rules for an event becomes expensive.

In this paper, we proposed a method to shorten rule invocation time when an event occurs. In the method, we divided the condition of rule into PECs and RECs, and constructed the discrimination network to evaluate the PEC. Because the method reduces the number of invocation candidate rules by the discrimination network when the event occurred, the rule invocation time can be reduced. We also proposed a method to shorten the update of the descrimination network.

We implemented the proposed methods on the autonomous disk system, and compared the execution time of the proposed methods with that of the method scanning all rules by experiments. The results indicate that the rule processing time can be reduced by using our methods. Comparing with the normal systems, although our proposed system has a few milli seconds of performance degradation, it can be hided by implementing an asynchronous descrimination network update function.

As the future work, it is necessary to consider a rule which requires the communication excluding storage beyond the network. We need to reduce the time of REC evaluation. Moreover, the method of describing a large amount of rules automatically is crucial. In addition, We need to introduce a rule verification technique to prevent the disadvantage from happening to the user. For example, we need a technique for discovering rules whose conditions are never satisfied and checking the termination of rules.

Acknowledgements

We thank Mr. Tomohiro Yoshihara for his advice about implementation on the autonomous disks.

This work is partially supported by the JST CREST, SRC, MEXT of Japanese Government via Grant-in-Aid for Scientific Research #16016232 and the MEXT 21st Century COE Program.

References

[1] L. L. Ashton, E. A. Baker, A. J. Bariska, E. M. Dawson, R. L. Ferziger, S. M. Kissinger, T. A. Menendez, S. Shyam, J. P. Strickland, D. K. Thompson, G. R. Wilcock, and M. W. Wood. Two decades of policy-based storage management for the IBM mainframe computer. *IBM Systems Journal*, 42(2):302–321, 2003.

[2] M. Beigi, M. V. Devarakonda, R. Jain, M. Kaplan, D. Pease, J. Rubas, U. Sharma, and A. Verma. Policy-Based Information Lifecycle Management in a Large-Scale File System. In *POLICY*, pages 139–148. IEEE Computer Society, 2005.

[3] B. Callaghan, B. Pawlowski, and P. Staubach. *NFS Version 3 Protocol Specification*. Sun Microsystems, Inc., June 1995. http://ftp.riken.go.jp/pub/internet-doc/rfcs/rfc1813.txt.

[4] U. Dayal. Active database systems. In *Proc. of 3rd Intn'l Conference on Data and Knowledge Bases*, pages 150–169, 1988.

[5] T. Ishida. Parallel rule firing in production systems. *IEEE Trans. Knowl. Data Eng.*, 3(1):11–17, 1991.

[6] V. Kher and Y. Kim. Decentralized Authentication Mechanisms for Object-based Storage Devices. In *IEEE Security in Storage Workshop*, pages 1–10. IEEE Computer Society, 2003.

[7] D. P. Miranker. *TREAT : a new and efficient match algorithm for AI production systems*. Research notes in artificial intelligence. Pitman, Morgan Kaufmann, London, San Mateo, Calif, 1990.

[8] N. W. Paton and O. Díaz. Active database systems. *ACM Comput. Surv.*, 31(1):63–103, 1999.

[9] A. Verma, D. Pease, U. Sharma, M. Kaplan, J. Rubas, R. Jain, M. V. Devarakonda, and M. Beigi. An Architecture for Lifecycle Management in Very Large File Systems. In *MSST*, pages 160–168. IEEE Computer Society, 2005.

[10] H. Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 441–448, Nov. 1999.