論文 / 著書情報
Article / Book Information

論題(和文)	柔軟なコンテンツ管理のためのルール処理への弁別ネットワークの適用
Title(English)	Application of Discrimination Networks to Rule Processing in Flexible Contents Management
著者(和文)	太田健介, 小林 大, 小林 隆志, 田口亮, 横田 治夫
Authors(English)	Kensuke OTA, Dai Kobayashi, Takashi Kobayashi, Ryo Taguchi, Haruo YOKOTA
出典(和文)	DBSJ Letters, Vol. 5, No. 1, pp. 1-4
Citation(English)	DBSJ Letters, Vol. 5, No. 1, pp. 1-4
発行日 / Pub. date	2006, 6
権利情報 / Copyright	本著作物の著作権は日本データベース学会に帰属します。 Copyright (c) 2006 Database Society of Japan, DBSJ.

柔軟なコンテンツ管理のためのルール処理への弁別ネットワークの適用

Application of Discrimination Networks to Rule Processing in Flexible Contents Management

太田 健介♡ 小林 大♣ 小林 隆志♠
田口 亮♢ 横田 治夫♠

Kensuke OHTA Dai KOBAYASHI
Takashi KOBAYASHI Ryo TAGUCHI and
Haruo YOKOTA

近年，扱われる情報量が増大し，ストレージ上にも扱いの異なる大量の様々なコンテンツが存在するようになった結果，個々のコンテンツの特徴を反映させた細粒度のストレージ管理が求められるようになっている．我々は，各コンテンツのメタデータへの ECA ルールによる処理記述付加による細粒度自律管理を提案している．しかし，格納されるコンテンツ数とそれに伴うルール数の増加により，イベント発生時のルール評価時間が問題となる．本稿では，ルールの発火条件部分を 2 種類に分け，事前評価可能な部分から弁別ネットワークを構成することで実行候補ルールの絞込みを行う手法を提案する．また，高機能ストレージとして提案している自律ディスクへ実装し，その有効性を示す．

Various types of data have been stored in a storage system followed by the rapid increase of information treated by a system. It is highly required to handle each stored content in different manners to reflect its properties. We have proposed an autonomous management with ECA rules as metadate for each stored content. However, evaluating the large number of rule conditions becomes dominant to costs to handle rules for an event. In this paper, we propose a method dividing the rule conditions into two types and constructing a discrimination network from previously-evaluable conditions to filter executable rules. We implement the methods in the autonomous disk system, a high functional storage system we proposed, and evaluate the effect of theproposed method.

1. はじめに

近年，ストレージ上のデータ量の爆発的な増加に伴い，効率的なデータ管理に対する要求が高まっている．同時に，多様な大量のデータが格納されたストレージシステム上には扱いの異なる様々なコンテンツが存在するため，それぞれのコンテンツの特徴を反映させた個別の管理が求めれるようになってきた．

コンテンツの特徴を活かした既存のストレージ管理として，情報ライフサイクルマネジメント（Information Lifecycle Management: ILM）[1] を挙げることができる． [1] ではシステム側に管理のためのポリシーを記述し，管理を行っている．しかし，データ量の増加に伴いデータの多様性も増大してきているため，ストレージ資源を効率よく扱うためにシステム単位のポリシー規定だけではなく，コンテンツ毎に特化した処理や制御ポリシーを詳細に規定することが望まれる．

我々は，より細かい粒度での管理を実現するためのコンテンツごとに制御内容を指定する手法として，各コンテンツに付与されたメタデータの一部として制御ルールを記述し，その制御ルールに従ってコンテンツの管理を行う方法を提案している [2]．制御ルールはアクティブデータベース [3, 4] において用いられる ECA (Event-Condition-Aciton) ルールを用いる．ECA ルールは，あるイベントに対して，アクションの発火条件と，条件が満たされたときに実行されるアクションを宣言的に記述する．発火条件に，コンテンツ等の状況に関する条件の設定をすることで，特定コンテンツの状態が変化するタイミングでルールを発火させることができるため，柔軟なコンテンツ管理が実現できる．

しかし，ストレージ中に格納されるコンテンツ数の増加により，各コンテンツに対応した制御ルール数も増大する．それに伴い，各ルールの発火条件評価にかかる時間の増加が無視できない問題となる．大量の ECA ルールの発火条件評価は非常にコストが高いので，コンテンツ毎の細粒度のルール制御を実現するために，この発火条件評価を高速化をする必要がある．

本稿では，イベント発生時のルール発火作業の短縮を目的とし，各イベントに対して，部分的な条件評価を実行し，発火候補ルールを絞込みを行う手法を提案する．提案手法では条件評価済みのルール集合をノードとした弁別ネットワークを構成することで，不必要な条件評価とイベント発生時の条件評価処理自体の時間の短縮を行う．高機能ストレージとして我々が提案している自律ディスク [5] を用いた実験により提案手法の有効性を示す．

以下に本稿の構成を述べる．まず 2. でコンテンツへのルールの記述方法について述べる．次に 3. で，まず本稿で考察するルール処理方法について述べた後，提案するルール処理の効率化手法について述べる．4. では，提案手法の有効性を自律ディスク上の実験を通して考察を行う．5. で関連する研究について述べ，6. で全体のまとめと今後の課題について述べる．

2. コンテンツへの処理記述

コンテンツの状態やアクセス傾向に応じた処理を実現するために，格納されている全てのコンテンツを考慮したシステム単位のポリシーを規定することはその多種性から困難である．細粒度での異種コンテンツに対する処理の差別化を実現するため，我々は，メタデータへの ECA ルールの付加を考える．

2.1 メタデータ

コンテンツ毎に記述するメタデータにおいて，そのデータに関する情報は，POSIX1003.1 仕様や NFS [6] などの標準的なファイルシステムがサポートするメタデータ属性に加え，ユーザ等が拡張的に定義するコンテンツをより特徴化させる項目を追加することも考慮する．さらに，我々は付加情報と同様に制御処理をメタデータとして格納する．ルールを記述する際は，定義されているメタデータ属性やシステムが実行できる 管理コマンドについて既知であることを前提とする．

2.2 ECA ルール

ECA ルールは，あるイベントに対して，アクションの発火条件とそのアクションの内容を宣言的に記述するものであり，Event・Condition・Action の 3 要素から構成される．各要素には，以下に示すような記述がなされる．**Event** には，トリガされるルールの原因となるイベントを記述する．**Condition** へは，ルールがトリガされるたびにチェックされる条件を記述する．条件評価はアクションが実行される前に行われる．トリガされたルールのうち，条件評価の結果が真であるときに **Action** に記述された内容が実行（発火）される．

♡ 学生会員 東京工業大学 大学院総合理工学研究科 物理情報システム専攻 ota@alab.ip.titech.ac.jp
♣ 学生会員 東京工業大学 大学院情報理工学研究科 計算工学専攻・日本学術振興会特別研究員 DC daik@de.cs.titech.ac.jp
♢ NHK 放送技術研究所 taguchi.r-cs@nhk.or.jp
♠ 正会員 東京工業大学 学術国際情報センター {tkobaya@gsic, yokota@cs}.titech.ac.jp

2.3 ECA ルールのコンテンツ管理への適用

ECA ルールを用いて記述した制御ルール群をコンテンツのメタデータの一部として付与する．以下に ECA ルールをストレージ管理に適用した場合に，各要素で記述できる項目について述べ，コンテンツへの処理記述例を示す．

Event には， insert，delete などの基本的な操作をはじめ，タイマ割り込みやアプリケーションで定義したイベントの記述が可能である．Condition には，コンテンツのメタデータとして記述された情報に対する制約条件や，検索結果の有無などを記述できるほか，イベント発生時に決定するパラメータについての条件も記述可能である．Action には，高機能ストレージが提供している insert，delete，update といった基本的なコマンドを記述可能である．ルールの下に，システムが備えるストレージ管理コマンド群が想定でき，記述可能な処理の範囲は広い．

ルール記述例 コンテンツ削除時に関連するコンテンツも同時に削除するルールである．例えば，映像コンテンツが削除された場合に，それに関連する複数の音声コンテンツや字幕テキストなどでアクセス頻度が閾値以下のものも同時に削除する場合などに利用される：

Event コンテンツが削除される時

Condition 削除対象のコンテンツに関連する音声コンテンツ，または字幕テキストでアクセス頻度が閾値以下のものが存在する場合

Action 関連コンテンツを削除する

このように，ユーザによるメタデータの定義とコンテンツの制御ルールの記述により，柔軟なコンテンツ管理が可能である．

3. ルール処理

3.1 単純なルール処理法

ECA ルールで記述されたルールを処理するために，システム側にルールの登録と実行を行うルールマネージャが必要とされる．

ルールの登録時には，ルールマネージャによりルールの登録作業が行われる．このとき，イベントとそのイベントに関連し発火する可能性のあるルールとのマッピングをあらかじめ行っておく．イベント発生時には，ルール登録時に行われたマッピングを用いて発生したイベントに関連して発火する可能性のあるルール群を取得し，それぞれのルールについて発火条件である Condition の評価を行う． Condition の評価結果が真となれば，ルールが発火され，Action に記述された処理を実行する．

しかし，ストレージ内のコンテンツの最新の状態を知るために，イベント発生の度に全てのルールを走査することは，発火しない多くのルールについての冗長な Condition 評価処理が含まれておりコストが高い．

3.2 提案手法

本稿では，大量のルールに対する Condition 評価の高速化のために，ルールを弁別ネットワークに構築することで事前に評価可能なコンディションについての評価をイベント発生前に実行し，発火候補ルールの絞込みによってイベント発生時のルール処理時間を短縮する手法を提案する．

3.2.1 コンディションの分割

コンディションを，イベント発生とは関係なく評価可能な部分（PEC：Previously Evaluable Condition）と，イベント発生時に渡されるパラメータに対する条件で構成されイベント発生時に評価すべき部分（REC：Runtime Evaluable Condition）に分割する．ルール記述例における「削除対象のコンテンツに関連する〜」という条件は，PEC であり，「アクセス頻度が閾値以下〜」という条件は REC である．

また，PEC は，1 つまたは複数の条件式で構成されることから，PEC をさらに，最終更新時刻 (図 1 における mtime) 等の単一のメタデータ項目を含む部分コンディション sPEC（sub-PEC）に分割することができる．

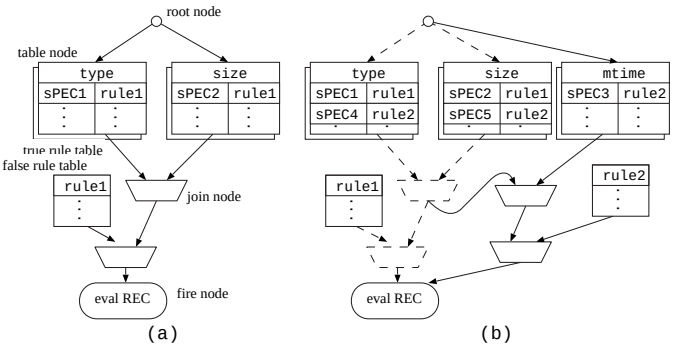

図 1: Condition 分割を用いた条件評価のための弁別ネットワーク (DN) (a)rule1 による DN の構築． (b) rule2 登録により再構成された DN

Fig.1 The discrimination network (DN) for evaluating divided conditions. (a) DN composition with rule1. (b) Reconstructed DN by registering rule2

そこで，両者をイベント発生時に評価するのではなく，コンテンツ更新時に sPEC の評価を行い，イベント発生時に PEC の評価が真となるルールを高速に選出することで，REC の評価を実行する発火候補ルールを削減し，ルール発火時間を短縮する．

3.2.2 弁別ネットワークの概要

本手法では，イベント発生時に発火候補ルール特定のために，登録されたルールを用いて弁別ネットワークを構築する．図 1(a) はあるルール rule1 が登録された弁別ネットワークの概要を示したものである．また図 1(b) では (a) に対し異なる rule2 の登録が行われた後の拡張を表す．弁別ネットワークは，以下の要素から構成される．ネットワークはひとつのルートノード (root node) を持つ．イベントが発生されるたびに全ルールセットがルートノードからネットワークを辿る過程で発火候補ルールが弁別され，発火ノード (fire node) を抜けたルールが実行される．

テーブルノード (table node) は，メタデータの種類ごとに生成され，暫定発火候補ルールテーブル (true rule table) と発火対象外ルールテーブル (false rule table) を持ち，それぞれには sPEC の評価が真となるルールと，sPEC の評価結果が偽のルールが登録されている．図 1(a) では，type と size に関するテーブルノードが構築され, rule1 上の condition を分解した sPEC が登録されている．

結合ノード (join node) では，テーブルノードまたは他の結合ノードから渡される 2 つの暫定発火候補ルール群の結合を行う．結合ノード群は，ルール登録・削除時にその condition 条件式を満たすように生成・削除される．図 1(b) はある rule2 の登録による結合ノード群の再構成の概念を表す．

発火ノードでは，結合ノードから渡される，PEC を満たしたルールの REC の評価とルールの発火が行われる．

3.2.3 イベント発生時の動作

弁別ネットワークは発生する可能性のあるイベント別に構築されている．あるイベントが発生した場合，対応する弁別ネットワークが選択され，以下の処理が行われる．

1. それぞれのテーブルノード上の暫定発火候補ルールテーブルから発火候補ルールセットを結合ノードに渡す
2. 結合ノードにおいて，2 つのルールセットの結合操作を行い，両セットに含まれるルール群を抽出する
3. 抽出されたルール群を発火ノードに渡す
4. 発火ノードにおいて，渡されたルール群について REC の評価を行い，評価結果が真となればアクションの処理を実行する

3.2.4 ルール登録・削除時の動作

表 1: ストレージノード諸元
Tab.1 Specifications of a storage node

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

ルール登録時にはルール内の Condition を解析し，弁別ネットワークへの登録を行う．

1. 登録対象ルールの Condition を PEC と REC に分割し，PEC をさらに sPEC に分解する．sPEC と登録対象ルールへの参照を該当テーブルノードに渡す
2. テーブルノードにおいて，渡された sPEC の評価を行い，評価結果が真ならば，暫定発火候補ルールテーブルへ，偽ならば発火対象外ルールテーブルへ登録する
3. 登録対象ルールの Condition 内における sPEC 間条件式に従い，結合ノードを生成する

ルール削除時は，登録時の手順 1 と同様にテーブルノードを特定した後，各テーブルから該当ルールに関する sPEC および該当ルールへの参照を削除する．また，対応する結合ノードを再構築する．

3.2.5 コンテンツ更新時の動作

コンテンツの更新等によりメタデータ内容が変化した場合，sPEC の再評価が必要となる．

変化したメタデータに関する条件の sPEC が存在した場合，その sPEC を含むテーブルノード中の暫定発火候補ルールテーブル・発火対象外ルールテーブルのどちらかにおいて，sPEC を再評価し，その結果真偽の変化したレコードについてテーブルへの再登録を行う．

4. 実験

本手法の有効性を示すために，高機能ストレージとして提案されている自律ディスク [5] に提案手法を実装し実験を行う．実験では，提案手法によるイベント発生時のルール処理時間の比較と，提案手法を導入したことによるコンテンツ更新時の遅延時間の測定を行う．

4.1 実験環境

実験は，我々の提案する分散ストレージ技術である自律ディスクの模擬実装上で行う．これは Linux クラスタ上に Java を用いて模擬実装されている．今回の実験では表 1 に示す構成の PC8 台と十分なバックボーン性能を持つネットワークスイッチを用いて，実験環境を構成した．

4.2 想定運用シナリオ

実験で用いるルールを設定するために，次のような状況を想定する．ストリーム形式の様々なデータが保存されている場合，できる限りそれらのデータを 1 つのコンテナフォーマットに格納して保存するような状況を考える．それぞれのデータの有効期限が過ぎていない同一タイトルの映像データ，音声データがあった場合，コンテナフォーマットにまとめるために編集用のストレージに移動させるルールを設定する．図 2 にそのルールの例を示す．この例では，発火条件の上 3 つが PEC に相当し，その一つ一つが sPEC であり，下 2 つが REC に相当する部分である．

Condition 評価時間の差異を明確化するため，Action 内には処理を記述のないルールを，1 コンテンツあたり 1 つ付与した．

4.3 ルール処理時間の比較

イベント発生時のルール処理時間を，イベントに関連するして発火する可能性のあるルールのうち PEC の評価が真であるルールの割合を変化させて測定し，ルールの全走査による処理方法と提案手法についての比較を行った．

PEC 評価による絞込みの効果を示すために，REC の評価は，全

```
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
```

図 2: 実験に用いるルール．
Fig.2 Rule description used in the experiment

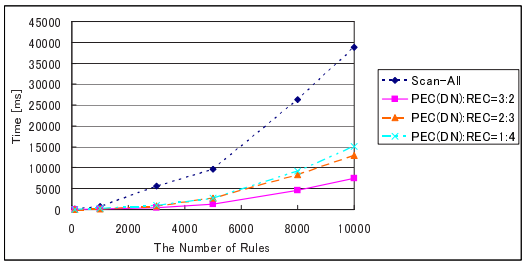

図 3: 発火するルールの割合が 5 %の場合のルール処理時間
Fig.3 Rule-processing duration under the situation that five percent of all rules need to be processed

て真とした．この比較を登録ルール数が 100, 1000, 3000, 5000, 8000，10000 の場合で実行し，登録ルール数によるルール処理時間の変動を測定した．また，提案手法のルール処理時間はルールのコンディション内の PEC，REC の比率に依存するため，コンディション内の PEC，REC の比率を変化させたときのルール処理時間について測定を行った．

図 3 は，PEC の評価が真となるルールの割合が 5 %の場合に，PEC，REC の比率が 3:2，2:3，1:4 のルールについてのルール処理時間を示している．実験で用いたルールは図 2 のルールの PEC と REC の比率を意図的に変えたものを使用した．

4.4 コンテンツ更新処理時間の比較

提案手法を実装したことによるコンテンツ更新時の遅延時間を測定した．今回，測定を行った操作は，コンテンツの挿入・更新・削除の 3 つである．各操作が行われることにより，各コンテンツに付与されているメタデータの情報が変更されるため，弁別ネットワークの更新が必要になる．

図 4 は，1 つのクライアントから自律ディスクに対して逐次的に 1000 回のコンテンツの挿入・更新・削除をそれぞれ行ったときの 1 回当たりの平均処理時間を測定した結果を示している．

4.5 考察

4.5.1 ルール処理時間に関する考察

4.3 の実験では，常に提案手法の方がイベント発生時のルール処理時間が短く，本手法が有効であることが分かった．

また，PEC，REC の比率を変化させた場合の実験では，コンディションに含まれる REC の増加により処理時間は増大する．しかし，処理時間の短縮の要因はルールの絞込みによる処理ルール数の減少によるところが大きいため，全走査によるルール処理時間を本手法のそれが上回ることはない．

4.5.2 コンテンツ更新処理時間に関する考察

更新操作では，提案手法の実装に伴う遅延時間は 3ms 程度と最も小さかった．これは，sPEC を含むテーブルノード中の暫定発火候補ルールテーブル・発火対象外ルールテーブルにおいて，sPEC の再評価とテーブルへの再登録を行うだけでよいためである．

コンテンツの挿入・削除が行われる場合は，処理時間はそれぞれ約 16ms,15ms 増加した．これらは，テーブルノードの更新だけでなく同時にルールの登録・削除，結合ノードの生成・削除も行われ

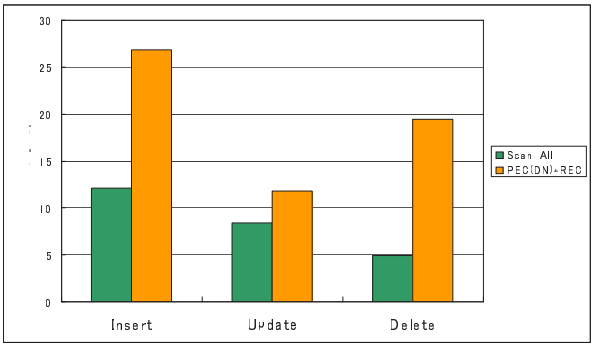

図 4: 挿入・更新・削除操作の平均処理時間
Fig.4 Average time for insert, update or delete a content

るためである．

5. 関連研究

ストレージ中のデータをコンテンツ単位で扱う仕組みとして，ブロック単位で管理されていたファイルをより抽象度の高いオブジェクト単位で管理を行う OSD（Object-Based Storage Device）[7] が挙げられる．メタデータの付与や制御ルールの記述といった点において，本研究が提示するデータの管理方法との親和性がよい．

ポリシーベースの ILM に関する研究として，[1] では格納データの処理についてのストレージ全体に適応されるポリシーを規定し，集中管理型システムによって ILM の自動化を実現させるための機能の提案と実装について述べられている．しかし，格納されている個々のコンテンツに関する状態やアクセス傾向を細かく反映した制御を行うことは難しい．

ECA ルールを用いたデータベース管理の研究は，アクティブデータベースの分野において行われてきた [4]．また，プロダクションシステム [8] などのルールベースシステムなどでは，ルールの条件評価を高速に行うために Rete アルゴリズム [9] が広く用いられている．大量のデータの中からルール条件に一致するデータ集合を高速に検索することが要求される．そのため，ルールベースシステム等に用いられるルールエンジンでは Rete アルゴリズムに基づいた Rete ネットワークを代表とする弁別ネットワークが利用されてきた．

6. まとめと今後の課題

細粒度のコンテンツ管理のために，個々のコンテンツに対するメタデータとしてそのコンテンツに対する制御ルールを記述し，ルールに基づいた柔軟なコンテンツ管理を実現する場合の効率化手法を提案した．

コンテンツ毎にルールを記述した場合，イベント発生の度に全コンテンツに対して記された大量のルールのコンディション評価のためにルールを全ての走査するコストは高い．この問題に対処するため，ルールのコンディションを事前に評価可能な PEC と，実行時に評価する REC に分割し，PEC 評価のための弁別ネットワークを構築し，ルール処理を高速化する手法を提案した．弁別ネットワークによって発火候補ルールの絞込みと評価すべき条件の削減を行うことができ，コンディション評価のコストを大幅に削減することができる．

本手法を我々が提案する高機能ストレージである自律ディスクに実装し，実験により提案手法とルールの全走査による方法との比較を行った．その結果，イベント発生時ルール処理時間が短縮され，提案手法の実装に伴うコンテンツ更新処理の遅延時間が弁別ネットワークの更新を非同期に行うことで隠蔽することが可能であることを確認した．

今後の課題として，REC 評価の高速化や，イベントが発生したストレージ以外との通信が必要なルールについての検討が必要である．また，大量のルールを自動記述する方法や，ルールの停止性問題の解決や，矛盾する発火条件を持つルールの判定のためにルールの検証技術を導入が必要である．

[謝辞]

自律ディスクの実装コードに関して東京工業大学の吉原朋宏氏に助言を頂いた．ここに感謝の意を表します．また本研究の一部は，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST，情報ストレージ研究推進機構 (SRCs)，文部科学省科学研究費補助金特定領域研究 (16016232) および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

[文献]

[1] Mandis Beigi, Murthy V. Devarakonda, Rohit Jain, Marc Kaplan, David Pease, Jim Rubas, Upendra Sharma, and Akshat Verma. Policy-Based Information Lifecycle Management in a Large-Scale File System. In *POLICY 2005*, pp. 139–148, 2005.

[2] 山口宗慶 , 渡邊明嗣 , 小林大 , 田口亮 , 林直人 , 上原年博 , 横田治夫. ルールを含むメタデータによる柔軟なコンテンツ管理. , DEWS2005, 6A-i8. 電子情報通信学会, Mar. 2005.

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

[4] Norman W. Paton and Oscar Díaz. Active database systems. *ACM Comput. Surv.*, Vol. 31, No. 1, pp. 63–103, 1999.

[5] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of DANTE'99*, pp. 441–448, Nov. 1999.

[6] B. Callaghan, B. Pawlowski, and P. Staubach. *NFS Version 3 Protocol Specification*. Sun Microsystems, Inc., June 1995. http://www.ietf.org/rfc/rfc1813.txt.

[7] Vishal Kher and Yongdae Kim. Decentralized Authentication Mechanisms for Object-based Storage Devices. In *IEEE Security in Storage Workshop*, pp. 1–10, 2003.

[8] Toru Ishida. Parallel rule firing in production systems. *IEEE Trans. Knowl. Data Eng.*, Vol. 3, No. 1, pp. 11–17, 1991.

[9] Daniel P. Miranker. *TREAT : a new and efficient match algorithm for AI production systems*. Research notes in artificial intelligence. Pitman, Morgan Kaufmann, 1990.

太田 健介　Kensuke OHTA
平 18 東工大・工・情工卒．同大大学院・総合理工・物理情報システム・修士課程在学中．日本データベース学会学生会員．

小林 大　Dai KOBAYASHI
平 15 東工大・工・情工卒．平 17 同大大学院・情報理工・計算工・修士課程了．同大大学院・情報理工・計算工・博士課程在学中．平 18 日本学術振興会特別研究員 DC. 日本データベース学会学生会員．

小林 隆志　Takashi KOBAYASHI
平 9 東工大・工・情工卒．平 11 同大大学院・情報理工・計算工・修士課程了．平 16 同大大学院・情報理工・計算工・博士課程了．平 14 同大・学術国際情報センター・助手．工博．日本データベース学会，日本ソフトウェア科学会，情報処理学会，ACM 各会員．

田口 亮　Ryo TAGUCHI
平 6 慶應義塾大大学院・理工・計測工・修士課程了．同年より NHK 放送技術研究所．映像情報メディア学会会員．

横田 治夫　Haruo YOKOTA
昭 55 東工大・工・電物卒．昭 57 同大大学院・情報・修士課程了．同年富士通 (株)．同年 6 月 (財) 新世代コンピュータ技術開発機構研究所．昭 61(株) 富士通研究所．平 4 北陸先端大・情報・助教授．平 10 東工大・情報理工・助教授．平 13 東工大・学術国際情報センター・教授．工博．日本データベース学会理事．電子情報通信学会，情報処理学会，人工知能学会，IEEE，ACM 各会員．