ハッシュを用いたOSIディレクトリの高速名前解析処理方式

6F－2

西山 智　横田 英俊　堀内 浩規　小花 貞夫　浅見 徹　鈴木 健二
国際電信電話株式会社 研究所

1. はじめに

OSIディレクトリ[1]の高速化には，複数のディレクトリシステムエージェント（DSA）に分散格納されたディレクトリ情報ベース（DIB）上で，ディレクトリ操作の対象となるエントリを確定する処理（名前解析）の高速化が重要となる．筆者らは単一のDSAに格納されたDIBに対してはハッシュにより名前解析の高速化が実現できることを示した[2]．本稿では，複数のDSA間に分散格納されたDIBに対するハッシュを用いた名前解析処理方式について報告する．

2. 名前解析の概要

OSIディレクトリの標準/勧告[1]ではエントリに対して検索，更新等の操作を行なうために提示名から操作の対象となるエントリを確定する処理（名前解析）を行なう．これには別名 (Alias) の展開やDIBが複数のDSAに分散格納された場合の操作転送先DSAの決定が含まれる．名前解析のためにDSAは表1に示す知識を持つ．これらの知識はディレクトリ情報木（DIT）に対応する知識木（KT）にモデル化される．標準/勧告では

(1) まず最大限一致するKTの部分木（名前コンテクスト）を検索する，

(2) 部分木内の名前解析を行なう，

というKTをたどる論理的な名前解析手順（図1）を示しているが，具体的な実現方式は実装に任されている．

表 1: DSAが保持する知識

内部参照 (INTR)	自DSAに格納するエントリへのポインタ
上位参照 (SUPR)	自DSAの知識では名前解析できない場合の転送先DSAに関する情報.DSA内に1つのみ存在
下位参照 (SUBR)	自DSA内に格納されているエントリの特定の直接下位エントリを格納するDSAに関する情報
不特定下位参照 (NSSR)	自DSA内に格納されているエントリの不特定の直接下位エントリを格納するDSAに関する情報
クロス参照 (CR)	自DSAが格納するエントリとは直接上位・下位関係にないエントリを格納するDSAに関する情報

3. ハッシュによる名前解析処理方式

3.1 基本方針

- INTRとその他の知識を分離して検索
  INTRは格納するエントリへのポインタであり，転送先DSAを示すその他の知識と性格が異なる．そこでINTRの検索とその他の知識の検索を分離する．
- ハッシュによるエントリの直接検索
  INTRを保持せず，ハッシュによりエントリを直接検索する．
- ハッシュによるその他の知識の検索
  INTRを除く知識もハッシュを用いて検索する．多数のエントリを格納するDSAでも（INTRを除く）これらの知識の件数は多くはならないため，主記憶上のハッシュ表に格納する．

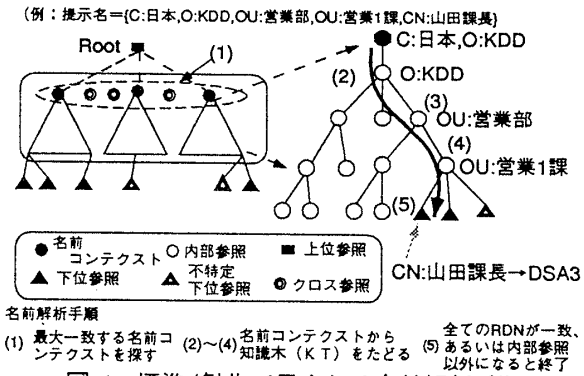

図 1: 標準/勧告で示される名前解析手順

3.2 名前解析の処理手順

基本方針から名前解析の処理手順は以下の通りとなる．

(1) ローカル名前解析
提示名で指定されるエントリを検索する．エントリが存在する場合は名前解析終了とする．

(2) 分散名前解析
ローカル名前解析でエントリが存在しない場合，DSAが保持する（INTRを除く）知識から最大一致する知識を検索し，名前誤りか他のDSAに転送するかを判断する．

3.3 ローカル名前解析

ローカル名前解析については，単一DSAに格納されたDIBに対する解析処理[2]がそのまま適用できる．

(1) 格納方式

- 識別名のハッシュ値をキーとしてB-木にエントリを格納する．他のエントリとハッシュ値が衝突した場合もそのまま格納する．
- 想定される最大格納エントリ数より十分大きな値域を持つハッシュ関数（ここでは,DES(Data En-

"Hash-Based Name Resolution Method for OSI Directory"
Satoshi NISHIYAMA, Hidetoshi YOKOTA, Hiroki HORIUCHI, Sadao OBANA, Tohru ASAMI and Kenji SUZUKI
KDD R & D Laboratories

cryption Standard) ベースの関数）を使用し, ハッシュ値の衝突を減少させる.

(2) 処理手順

- 提示名のハッシュ値をキーとして B-木からエントリを検索する. そのハッシュ値を持つ全てのエントリを検索し, 提示名と格納される識別名の比較によりエントリを確定する.

3.4 分散名前解析

(1) 格納方式

- INTR を除く知識（知識の種類, 識別名, 転送先 DSA の名前/アドレス）を, 識別名のハッシュ値をキーとしてハッシュ表に格納する. また, 別名（Alias）および自 DSA に格納する DIT 部分木の頂点エントリの識別名（CP:名前コンテクスト）もこのハッシュ表に加えて知識として扱う.
- 識別名/提示名の比較は処理時間を要するので, ローカル名前解析で用いたハッシュ関数による識別名のハッシュ値も比較用に格納し検索の高速化を図る.

(2) 処理手順

- 提示名（またはその一部）のハッシュ値をキーとしてハッシュ表から知識を検索する. 提示名から検索を開始し, 知識が見つかるまで RDN を 1 段ずつ減じつつ検索を繰り返す（図 2）. 他の DSA で既に名前解析が開始されていた場合は, RDN 段数が他の DSA で解析済みの段数になった時点で名前解析失敗とし, DSA 間の名前解析のループを防ぐ.
- 同一識別名を持つ知識が複数存在した場合, 優先度は Alias > SUBR,CR > NSSR > CP とする.
- Alias が存在する場合は, 別名展開を行ないローカル名前解析に戻る.
- SUBR,NSSR,CR が存在する場合はその知識の示す DSA に転送する.
- CP が見つかった場合は名前解析失敗とする.

4. 評価と考察

本稿で示した名前解析処理方式を, 筆者らが先に開発した高速 OSI ディレクトリ用 DBMS: ASSIST/D[3]に実装し, SUN SPARC2 670MP 上で評価を行なった.

格納エントリ件数を変化させた場合の Read 操作（提示名 4 段）の成功時並びに名前誤り時の操作応答時間を図 3に示す. 操作成功時はローカル名前解析, エントリ読み出しと結果返送が実行されており, 名前誤り時はローカル名前解析, 分散名前解析とエラー返送が実行されている. 図 3には操作成功時の応答時間について, 名前解析のみを勧告/標準に示す手順に従った従来の処理方式[3]に変更した結果も比較のために示した. 図 3から名前解析の処理性能は従来の処理方式と比較して高速であり, また操作成功時, 名前誤り時とも処理時間が $O(log($格納エントリ件数$))$ であることから多数のエントリを格納する DSA での名前解析に特に有効である.

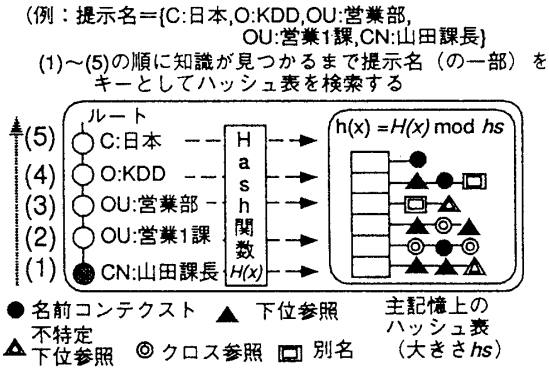

図 2: 分散名前解析の処理手順

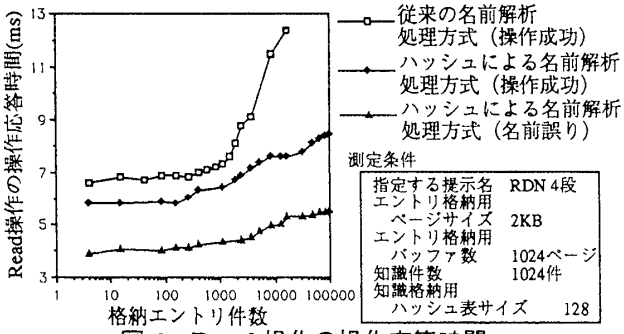

図 3: Read 操作の操作応答時間

また, 分散名前解析に要する時間はハッシュ表サイズ 128, 知識件数 1024 件の場合で 4 段の RDN を持つ提示名の名前誤り検出に約 0.12 ミリ秒, また一致知識の検出に最悪 0.14 ミリ秒程度であり, 全体の処理時間と比較して殆ど無視できるまで高速化できた.

5. おわりに

本稿では, 複数の DSA に分散格納された DIB の上でのハッシュを用いた名前解析処理方式について報告した. 本方式では名前解析を INTR 検索に相当するローカル名前解析とその他の知識を検索する分散名前解析に分けた. ローカル名前解析は提示名のハッシュ値をキーとしてエントリを格納する B-木を検索することで, また分散名前解析では主記憶上のハッシュ表に格納された知識を検索することで名前解析の高速化を図った. 評価の結果, $O(log($格納エントリ件数$))$ で名前解析が実現できることが分かった. 最後に日頃御指導頂く KDD 研究所 浦野所長, 眞家次長に感謝します.

参考文献

[1] ISO/IEC 9594 1-8: The Directory (1988).
[2] 西山 他: OSI ディレクトリ情報ベース（DIB）のための高速名前解析処理方式, 情処研究会報告 MDP 61-29,(1993).
[3] 西山 他: 拡張可能 DBMS 構築技法に基づく高速 OSI ディレクトリ専用 DBMS の設計と評価, 情処論文誌 Vol.34, No.6,(1993).