基幹システムにおける高信頼ディレクトリ・サーバ

菊地聡†，松岡祐介‡，小林 敦¶，小瀧伯泰¶
(株)日立製作所
†システム開発研究所，‡半導体事業部，¶ソフトウェア開発本部

情報システムの大規模分散化に伴い，分散システムの各種資源を一元的に管理する基盤技術としてディレクトリサービスが運用され始めている。今後はミッション・クリティカルな基幹システムへのディレクトリサービス適用も進展すると予想され，24時間運転，障害からの復旧等，システムの高信頼化が不可欠となってきている。そこで筆者らは，LDAPディレクトリ・サーバにおける高信頼化技術として，RDBMSによるディレクトリ・データの管理方式，論理的に一連の要求を単一の作業単位として処理するトランザクション機能を開発し，アーキテクチャ及び方式の実現性，有効性を確認した。

High reliable directory server for mission critical systems

Satoshi Kikuchi†, Yuusuke Matsuoka‡, Atsushi Kobayashi¶, Michiyasu Odaki¶
†Systems Development Laboratory, ‡Semiconductor & Integrated Circuits Division, ¶Software Development Center
Hitachi, Ltd.

With the spread of large scale distributed environments, it begins to deploy the directory service as a single repository that is managed the various resources in distributed environments. We expected that the directory service was deployed in the various mission critical systems and needed high reliability such as 24x7 operation, recovery from troubles. Then, we have developed high reliable technologies for the LDAP directory server, the method of managing directory data by RDBMS, the transaction function that processed a series of requests as a single work unit logically, and confirmed feasibility and validity of our designs.

1. はじめに

ディレクトリ・サービスは，分散システムにおける物理資源及び論理資源を一元的に管理する事で，システム管理者の運用負荷を大幅に削減するサービスとして期待されている。ディレクトリ・サービスの世界標準としては，CCITT勧告であるX.500[1)]が著名である。X.500の特徴は，各種資源の階層管理，すなわち階層型のネーミング・モデルであり，分散システムにおける各種資源をユニークに識別可能である。ところがユーザは，OSI7レイヤ構造による処理負荷の重さからX.500を敬遠しがちであった。この課題を解決するため，USミシガン大学は，クライアントからX.500準拠のディレクトリ・サービスを簡易にアクセス可能なLDAP（Lightweight Directory Access Protocol）[2)~3)]を開発した。LDAPは，分散システムの普及に伴うディレクトリ・サービスへのニーズ増大を契機に，インターネット標準を事実上決定するIETF（Internet Engineering Task Force）に取り上げられ，インターネット標準のディレクトリ・アクセス・プロトコルとして世界に認知されつつある。

一方，インターネットの浸透，ネットワーク・セキュリティ技術の進歩に伴い，各企業内の情報システムにインターネット技術を活用したイントラネットが定着しつつある。さらに今日では，市場動向，環境変化に応じて柔軟な仮想企業（VE：Virtual Enterprise）の構築に向け，各企業のイントラネット同士を接続

し，企業間のEC（Electronic Commerce），情報共有を実現するエクストラネットも脚光を浴び始めており，イントラネット及びエクストラネットにおける各種ミドルウェアのデータベース（リポジトリ）として，ディレクトリ・サービスに対する期待が高い。

そこで筆者らは，大規模かつミッション・クリティカルな基幹システムに適用可能な高信頼LDAPディレクトリ・サーバを開発した。

本稿では，基幹システムにおけるディレクトリ・サーバの要件，RDBMSによるディレクトリ情報ベース DIB（Directory Information Base）の管理方式，論理的に一連のディレクトリ・アクセス要求を単一の作業単位として処理するトランザクショナル LDAP 機能について述べる。

2. 基幹システムにおける要件

基幹システムにおいては，ディレクトリ・サーバの障害がシステム全体の停止に直結するため，以下の要件を満足する必要がある。

- ディレクトリ・データの破壊抑止，及びデータ破壊からの回復
- 24×7 運転を可能とするオンライン・バックアップ機能
- 複数のディレクトリ・アクセス要求に対する一貫性の確保

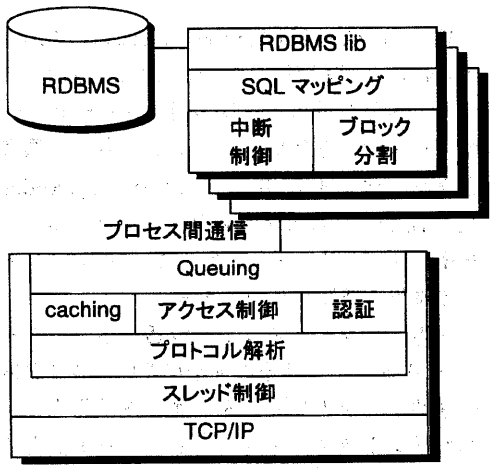

図 1 機能構造

上記要件を満足するため，図 1 に示すように，DIBを保持するDBとして，信頼性の面で実績の高い汎用RDBMSを採用した。

3. DIB 構造

3.1 設計方針

RDBMS による DIB 管理の実現方式検討において，以下の設計方針を掲げた。

(a) 高性能化
- RDBMS 検索機能・性能の最大限活用
- ジョイン，BLOB によるオーバヘッドの削減
- 抽出データ量の最少化
- 検索レコード（行）数の最少化
- 更新時排他期間の最短化

(b) 所要リソースの削減
- 冗長カラム（列）の最少化

従来，RDBMSを用いたX.500のDIB管理方式[4)~5)]が提案されている。本稿では，RDBMSによるLDAPのDIB管理方式，特に，ディレクトリ属性及びクラスの管理方式，及びスコープ検索制御方式について述べる。

3.2 属性管理

最初に，LDAPサーバの属性管理方式について述べる。

LDAPは，ユーザや組織，サービス等のオブジェクトを木構造（ディレクトリ・ツリー）で階層管理可能な X.500 データモデルを採用している。各オブジェクトの実体はディレクトリ・エントリと呼ばれ，ユーザのメールアドレス，姓名，電話番号，FAX番号，写真等，様々な情報を属性として登録可能である。各エントリは，階層情報を含む名称（DN：Distinguished Name）で一意に識別される。

表1にDIBの属性管理方式比較を示す。

ミシガン大学が開発したLDAPサーバは，1エントリの全属性を結合してDBで管理し，検索する属性についてはインデックスを付加する。各インデックスとエントリは，エントリ番号により関連付けられる。このDB構造を応用したものが方式1であり，RDBMSをデータ・ストアとして活用し，LDAPサーバが検索候補

の絞り込みを制御する。しかしこの方式によると，ジョインや全属性の抽出による性能の劣化が見込まれる。

そこで属性管理方式としては，RDBMS を検索エンジンとして最大限活用すべく，エントリ中の各属性を表（エントリ･テーブル）のカラムとして管理する方式 2 を採用した（図 2）。

表 1 属性管理方式

方式 / 項目	1. データ･ストア 全ての属性を結合し 1 カラムで管理	2. 検索エンジン 属性毎にカラムとして管理
検索	検索インデックス及びジョイン要	RDBMS の検索機能を利用可
部分一致検索	検索インデックス及びジョイン要	サポートしている RDBMS 少
データ抽出	全属性の抽出要	必要属性のみを抽出
スキーマ拡張性	高	低
採否	×	○

またエントリ･テーブルの各々のレコードには DN を格納するカラムを付加した。これにより，階層検索と属性検索を単一の SQL で同時処理する。

3.3 クラス管理

次に，LDAP サーバのクラス管理方式について述べる。

ディレクトリ･サービスの各エントリは，複数種の属性を管理でき，管理可能な属性は，オブジェクト･クラスにより決まる。つまりクラスにより各エントリのデータ構造が異なる。このようなディレクトリ･データ全てを単一表で管理すると，以下の問題が発生する。

- 検索対象レコード多による性能劣化
- 冗長カラム多によるディスク所要量の増大

上記の問題を解決するため，類似のデータ構造を持つクラスをグルーピングし，個別のエントリ･テーブルとして管理した。

さらにクラスの継承関係を管理するため，使用するオブジェクト･クラス毎に Boolean 型カラムを追加し，対象エントリのクラス継承関係により値を登録する。

サーバは，オブジェクト･クラスが特定された検索要求を受けた場合，アクセス対象が何れのエントリ･テーブルに格納されているか判断し，SQL を発行する。

但しオブジェクト･クラスを指定しない検索要求，及び更新要求に関わる性能向上のため，エントリ･テーブルと別に，全ディレクトリ･エントリに関するディレクトリ識別名（DN）とオブジェクトクラスの関連付けを管理する DN テーブルを設けた。

3.4 スコープ検索

次に，LDAP サーバのスコープ検索制御方式について述べる。

LDAP は，以下の 3 種のスコープ検索機能を備える。

・サーチ：部分木を対象に，条件を満足するエントリを抽出

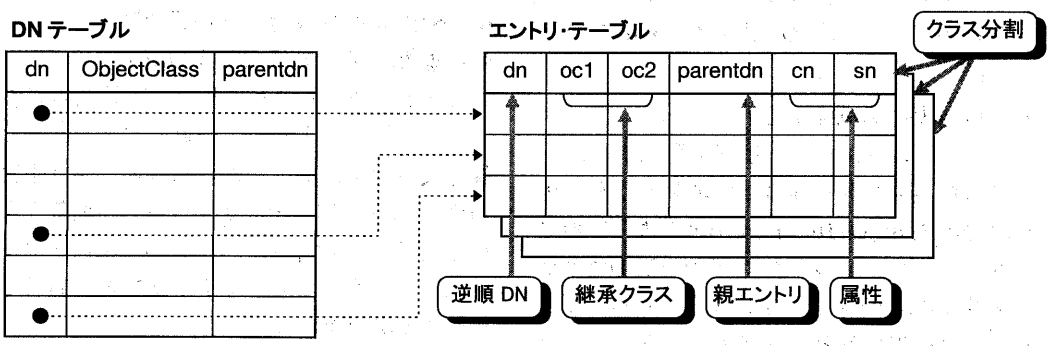

図 2 DIB 管理スキーマ

・リスト：あるエントリの直下に位置するエントリ（子エントリ）を抽出
・リード：特定の1エントリを抽出

以下に，サーチ及びリスト検索における高速化手法を説明する。

(a) サーチ処理の高性能化

LDAPにおけるDNは，"CN=Satoshi Kikuchi, OU=SDL, O=Hitachi, C=JP"のように，ツリーの下位から最上位に向かって記述されるが，表記通りにRDBMSに格納すると，以下の問題が発生する。

クライアントはサーチ検索を要求する際，検索範囲とする部分木の最上位エントリを，例えば"O=Hitachi, C=JP"のように指定する。このときサーバはRDBMSに対して後方部分一致検索のSQLを発行することになる。RDBMSは，検索性能の向上手段としてインデックス機能を備えている。しかし通常のインデックス機能は，全一致及び前方一致の高速な文字列検索を可能とするが，中間一致または後方一致については，性能面での効果を得られにくい。

この問題を解決するため，"C=JP, O=Hitachi, OU=SDL, CN=Satoshi Kikuchi"のように，DNを構成する各要素（RDN）の順序を反転して記憶する逆順DNカラムを設け，RDBMSのインデックスを前方一致化することにより，サーチ検索時の高性能化を図った。LDAPサーバは，クライアントからのサーチ検索要求を受けて，指定された検索範囲を，例えば"C=JP, O=Hitachi"のように逆順化した後，SQLを発行する。

(b) リスト処理の高性能化

クライアントはリスト検索を要求する際，抽出したいエントリ群の上位エントリを，例えば"OU=SDL, O=Hitachi, C=JP"のように指定する。このときサーバが，上述の逆順DNカラムを対象に，指定された上位エントリのDNを部分一致検索すると，直下のエントリだけでなく，更に下位のエントリまで抽出されてしまう。このため，サーバ内での冗長なフィルタリング処理が不可欠となる。

そこで，各エントリを管理するレコードに，上位エントリのDNを格納する親エントリカラムを追加することにより，サーチ検索時の高性能化を図った。LDAPサーバは，クライアントからのリスト検索要求を受けて，親エントリカラムを参照し，指定された上位エントリと同値を持つエントリを検索する。

3.5 性能

表2に，今回開発したLDAPサーバの性能測定結果を示す。評価システムは，LAN接続された2台のPC（クライアント，LDAPサーバ）と1台のWS（RDBMS）である。サーバには，約10000個のディレクトリ・エントリを予め登録した。

表2 性能評価

項目	検索	応答性能 [s]
組織一覧	list	1.3
証書読込	read	1.0
ユーザ検索	search	1.1
ACL検索	list	1.2

測定結果によると，約1sの検索応答性能であり，実用化の見通しが得られた。測定に用いたディレクトリ・サーバは開発段階のものであり，実用にあたっては，インプリメンテーション，RDBMSのチューニング等により，更なる性能向上が期待できる。

4. トランザクショナルLDAP

4.1 基幹システムにおける問題点

ミッション・クリティカルな基幹業務においては，以下のように，一連のディレクトリ操作実行中に，システムの障害等に伴う不慮の中断，ユーザやAPによる故意の中断等が発生した場合，また複数のAPからの同時アクセスについても，ディレクトリ情報の一貫性を保証する必要がある。

(a) 複数エントリの一括更新ロールバック

APが複数のディレクトリ情報を連続して更新中に，停電やユーザの誤操作，またはマシン本体やプログラムの誤動作等により不意に処理を中断させられた場合，更新中であった一連のディレクトリ情報の内，ディレクトリ・サーバに到達した更新要求だけがDBに反映されてしまう。このような障害が発生した場合，APを再起動し，連続する更新操作の中断点から再試行する必要があるが，不意の中断点をAPが正確に認識する事は困難である。通常はオペレータが介入し，中断時に更新処理していた一連の操作とディレクトリ・サーバの情報とを照合して中断点を発見しなければならず，煩雑な作業を要していた。

(b) アンドゥ機能

ユーザやAPは，複数のディレクトリ情報を連続して更新中に何らかの問題を発見し，それまでに行った更新操作の取り消しを所望するケースがある。しかしLDAPによると，前記更新操作はディレクトリ・サーバのDBに逐次反映されてしまうため，更新済み操作のアンドゥは不可能である。

(c) 更新エントリの排他制御

通常，GUIを用いたユーザ情報の変更は，以下のような一連の操作で行われる。

(c-1) AP：変更前のエントリ情報をサーバから読み出し表示

(c-2) User：表示された情報を修正

(c-3) AP：ユーザが変更した項目について，エントリ情報を更新

LDAPでは，(c-1)以降に他のAPが対象エントリを更新すると，(c-3)の処理が失敗することがある。これを防止するためには，複数のLDAP要求に跨る排他制御が必要である。

4.2 トランザクション制御方式

上記課題を解決するためには，トランザクション機能が不可欠である。しかし元来，LDAPは軽量さを特徴とするプロトコルであり，トランザクション機能を備えていない。

そこで，従来のLDAP操作の軽量さの確保を前提に，LDAPにおけるトランザクション制御機能（トランザクショナルLDAP）の仕様，方式を検討した。

図3に，トランザクショナルLDAPの概念を示す。トランザクショナルLDAPは，ディレクトリ・アクセス期間に，非トランザクション（non-Tx）・フェーズとトランザクション（Tx）・フェーズを有し，LDAPの拡張プロトコルにより両フェーズ間を双方向に移行する。これにより，従来のLDAPを重量化すること無く，かつ意味のある一連のディレクトリ要求を単一の作業単位（トランザクション），つまりアトミック・オペレーションとして処理することが可能となる。

non-Txフェーズは，従来型ディレクトリ・アクセス操作の互換性を確保するフェーズであり，各LDAP要求が1トランザクションとして扱われる。これに対しTxフェーズでは，一連のLDAP要求が単一のトランザクションとして扱われる。

4.3 プロトコル仕様

トランザクショナルLDAPの実装には，LDAP V3の拡張メカニズムを使用した。以下に，プロトコル仕様を，ASN.1表記で示す。

以下の仕様において，xxxOidは，X.208で規定されるオブジェクト識別子（OID）であり，

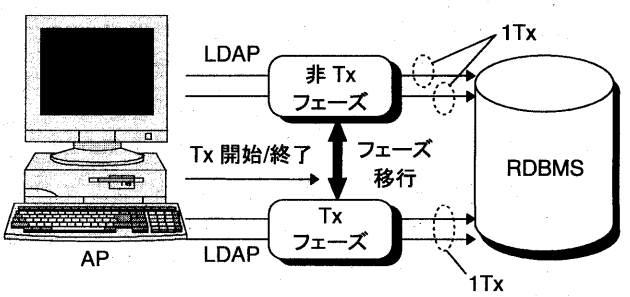

図3 トランザクショナルLDAP

機能を一意に識別するものである。

(a) トランザクション開始（begin）

トランザクション開始を指示するプロトコル要素。requestValue パラメータによりトランザクション中の排他モードを指定。

ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
* requestName [0] <beginOid>,*
* requestValue [1] <transactModeOid> }*
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
* COMPONENTS OF LDAPResult,*
* responseName [10] <beginOid> }*

(b) コミット（commit）

トランザクション開始以降の全操作を確定し，データを反映させるプロトコル要素。

ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
* requestName [0] <commitOid> }*
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
* COMPONENTS OF LDAPResult,*
* responseName [10] <commitOid> }*

(c) ロールバック（rollback）

トランザクション開始以降の全操作を廃棄し，データを元に戻すプロトコル要素。

ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
* requestName [0] <rollbackOid> }*
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
* COMPONENTS OF LDAPResult,*
* responseName [10] <rollbackOid> }*

4.4 シーケンス

図 4 に，トランザクショナル LDAP のシーケンス例を示す。

クライアントが LDAP サーバに接続した直後は non-Tx フェーズにあり，従来と同様，個々の LDAP 操作毎にデータが DB に反映される。クライアントがトランザクションを開始する begin 要求を発行すると，LDAP サーバは Tx フェーズに移行し，以降受付けた一連の要求を単一のトランザクションとして扱う。またクライアントがトランザクションを終了する commit または rollback 要求を発行すると，LDAP サーバは non-Tx フェーズに移行し，従来と同様，個々の LDAP 操作毎にデータを DB に反映する。

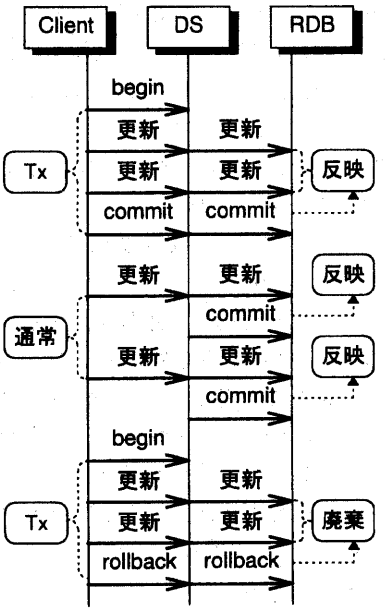

図4 シーケンス

5. おわりに

LDAP ディレクトリ・サーバにおける高信頼化技術として，RDBMS によるディレクトリ・データの管理方式，論理的に一連の要求を単一の作業単位として処理するトランザクション機能を報告した。さらに本方式，機能を適用した LDAP サーバを開発し，アーキテクチャ及び方式の実現性，性能に関するシステム評価を行い，実用化の見通しが得られた。

参考文献

1) ISO/IEC 9594-1～8 / CCITT X.500 シリーズ
2) Yeong, Howes, Kille：Lightweight Directory Access Protocol, RFC1777, 1995
3) Howes, Smith：The LDAP Application Program Interface, RFC1823, 1995
4) 西山，小花：OSI ディレクトリ・システムの実装(2)：情報処理学会第 36 回全国大会, 4F-3, 1988
5) 空，岸本，渡辺，窪田：RDBMS による OSI ディレクトリの実現：情報処理学会データベース研究会資料 95-10, 1993