システムソフトウェアと
オペレーティング・システム 77－39
マルチメディア通信と分散処理 87－39
（1998．2．27）

基幹システム向けディレクトリレプリケーションサーバの開発

平島 陽子† 菊地 聡† 由井 仁‡
(株)日立製作所
†システム開発研究所、‡ソフトウェア開発本部

企業情報システムを構成する多様なリソースを一元管理し、ユーザに透過的な利用環境を提供するディレクトリ・サービスでは、システム広域化に対応してディレクトリデータのレプリカを分散配置するレプリケーション機能へのニーズが高まっている。一方、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、障害からの迅速な復旧等、システムへの高信頼化が不可欠となっている。そこで、筆者らは障害等で失われたレプリカ・ディレクトリデータの復旧時間を短縮する動的差分レプリケーション方式を開発し、可用性および運用性に優れたレプリケーション・サービスを実現した。

Directory Replication Server for mission critical systems

Yoko Hirashima†, Satoshi Kikuchi†, Hitoshi Yui‡
†System Development Laboratory, ‡Software Development Center
Hitachi, Ltd

Directory service is a open and hi-performance data repository which can manage various resources in a distributed computing environment. And directory replication function realizes deployment of replica servers. As enterprise information systems are getting large, the needs for replication services are increasing. We expect that the directory service will be deployed in the various mission critical systems and will require high reliability such as 24x7 operation, fast recovery from failures of the systems. Therefore we have developed the dynamic-incremental replication technique in order to increase the availability for the services and reduce the management costs.

1. はじめに

ディレクトリ・サービスは、企業情報システムを構成するサーバやプリンタ等の物理的資源、ユーザやアクセスコントロール等の論理的資源を一元的に管理するサービスを提供する。このディレクトリ・サービスが資源に関する情報をユーザやアプリケーションに提供することで、物理的な位置や環境に依存しない資源の利用が可能になる。

ディレクトリ・サービスの標準仕様としてはCCITT（International Telegraph and Telephone Consultative Committee）の勧告 X.500[ref.1]がある。しかし、X.500 規定のディレクトリ・アクセスプロトコル DAP(Directory Access Protocol)は多機能であり、OSI スタックを必要とする負荷の高いプロトコルであるため X.500 を完全ににサポートする製品は少ない。そこで、ミシガン大学は DAP の機能を制限し、TCP/IP 上のアプリケーション（以下 AP と略）としたLDAP(Light Weight Directory Access Protocol)[ref.2]を開発した。今日では、このLDAP がディレクトリ・アクセスの業界標準プロトコルとなっている。

近年、オープン化の流れによりミッションクリティカルな基幹業務系も汎用のミドルウェア・コンポーネントの組み合わせで構築可能になり低価格化、相互接続性の向上をもたらした。また、インターネット技術を利用した社内ネットワークであるイントラネットの構築も盛んである。これらの企業情報システムは多数の AP から構成されるが、各 AP は独自にユーザやネットワーク資源に関する情報を管理しており、システム全体として見た場合同一データが多重管理され管理負荷を増大させている。そこで、ディレクトリ・サービスを各 AP 共通のデータベー

スとして用い、基幹システムの中核に位置づけようという動きが広まっている。しかし、基幹システムの共通データベースとして用いるには、軽量でオープンであることに加え、高い可用性や運用性を備えたディレクトリ・サービスが必要である。

一方、企業情報システムが大規模・広域化するに伴いディレクトリ・データの複製をシステムの各所に配置するレプリケーション機能が期待されている。このレプリケーション機能にも同様に高い可用性、運用性が必要である。

そこで筆者らは、基幹システム対応の高信頼ディレクトリ・サーバおよびレプリケーション・サーバの開発を行った。本稿はディレクトリ・システムにおけるレプリケーションの運用を容易化し、可用性を高める動的差分レプリケーション方式について報告する。

2. 基幹システムにおける要件

基幹システムにおけるディレクトリ・レプリケーション・サービスの要件を以下に挙げる。

(1)一貫性

不正データによる事故を防止するため、システムの障害時にもサーバ間のデータ一貫性を維持する必要がある。

(2)可用性・運用性

ディレクトリ・サーバの停止はシステム全体の停止を招く可能性があるため、データ破壊からの迅速な復旧手段が提供されなければならない。

(3)スケーラビリティ

システムが大規模化した場合にも、サーバ間同期は許容可能な時間内に実行され、且つレプリケーションによるネットワークトラフィックの増大は可能な限り抑止しなければならない。

3. ディレクトリ・レプリケーション

3.1. 目的

レプリケーションとは、あるディレクトリ・サーバが持つデータの複製を他のサーバに持たせることである。元本ディレクトリ・データを持つサーバをマスタ、複製を持つサーバをレプリカと呼ぶ。サーバ間でデータの一貫性を保つため、マスタ・データが更新された時にはマスタからレプリカへ更新情報が送られる。この更新情報の送受に着目した場合、送信側をサプライヤ、受信側をコンシューマと呼ぶ。レプリケーションの代表的な目的は以下２点である。

(1) サーバ多重化により、単一個所の障害によるサービス中断を防ぐ。

(2) 物理的に利用者に近い場所に情報を置くこと、及び複数のレプリカを作り負荷を分散することでアクセス速度を向上させる。

3.2. レプリケーション方式の分類

レプリケーション方式の分類を以下にまとめる。

(1) トポロジー

レプリケーションのトポロジーは、ハブ&スポークとメッシュに大別できる。

ハブ&スポークでは、ハブとなるサーバがディレクトリ・データの変更情報を集め、他のサーバへ伝播する。メッシュでは、変更が発生したサーバがそれぞれ他のサーバに変更情報を伝播する。

(2) 管理主体

データの管理元に着目すると、レプリケーションは single-master (master-slave) モデルと multi-master モデルに大別できる。

single-master モデルでは、単一のマスタが元本データを管理する。ディレクトリ・データの更新はマスタ以外では実行不可であるため、更新要求は一つのマスタに集中する。multi-master モデルはマスタが複数存在し、複数のマスタが各自レプリカへ更新情報を伝播する。multi-master モデルでは、複数箇所でディレクトリ・データの更新が可能であるが、矛盾する更新が同時に実行される可能性があるため、矛盾解決アルゴリズムが必要になる[ref.3]。

(3) 起動主体

起動主体に着目すると、レプリケーションは、push 型と pull 型に大別できる。

push 型は、サプライヤが能動的に更新情報をコンシューマに送信する方法である。サプライヤはディレクトリ・データの更新の有無を監視可能なので、リアルタイムに更新情報を伝播できる。また、複数のコンシューマへのレプリケーション要求送信をス

ケジューリング可能である。一方 pull 型は、コンシューマが能動的に更新情報を取得する方法である。コンシューマは定期的にサプライヤへデータ変更の有無を問い合わせなければならないが、負荷が一定値以下の場合に限りレプリケーションを実行する等、コンシューマの稼動状態に応じた制御が可能である。

(4) 転送データ

転送データに着目すると、レプリケーションは complete レプリケーションと incremental レプリケーションに大別できる。

complete レプリケーションは、サプライヤが自身の持つ全データをコンシューマに送る方法である。一方、incremental レプリケーションは、前回の変更情報送信時と現在のディレクトリ・データの差分を送る方法であり、complete レプリケーションに比べ転送量が少ないが差分情報を管理する必要がある。また、新規レプリカを構築する際にマスタ・データの複製である初期データを別途作成しなければならない。

上記分類の他に、ディレクトリ・データの一部分を選択的にレプリケーションする selective レプリケーション、コンシューマがレプリケーションにより得たデータを更に第三のサーバへ伝播する cascade レプリケーション等がある。

4. レプリケーション・サーバ

筆者らが開発したレプリケーション・サーバについて説明する。

4.1. 前提レプリケーション方式

基幹システム向けレプリケーションでは、スケーラビリティ維持のためにデータ転送量がより少ない pull 型、incremental レプリケーションを採用した。また、multi-master は十分な信頼性を持つ矛盾解決プロトコルが未確立であるので、single-master を採用した。single-master の場合、トポロジーはハブ&スポークとなる。

4.2. 機能の概要

レプリケーション機能をディレクトリ・サーバから独立したレプリケーション・サーバとして開発した。

レプリケーション・データの流れを図 1に示す。サプライヤは LDAP アクセス要求を受けディレクトリ・データを更新すると、コンシューマに更新の内容を伝えるため更新情報をレプリケーション・ログファイルに出力する。レプリケーション・サーバはレプリケーション・ログファイルから更新情報を得て LDAP 更新要求に変換、コンシューマへ発行する。

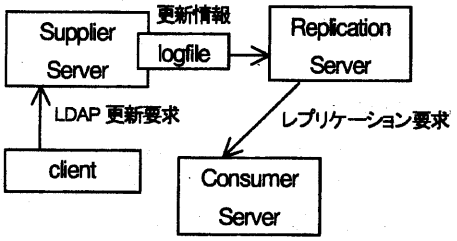

図 1 レプリケーション・データの流れ

LDAP アクセス要求は、ディレクトリ・データを変更する更新系と変更しない検索系に分けられる。検索系は検索要求のみであり、更新系には表 1に示す 4 つの要求がある。LDAP では木構造（ディレクトリ・ツリー）を持つ X.500 データ・モデルを採用しており、ツリーのノードに相当するデータ単位をエントリと呼ぶ。ここで、RDN(Relative Distinguished Name)はエントリの名前であり、同一の親を持つエントリ間で一意でなくてはならない。サプライヤは処理に成功した更新系要求の内容を変更情報として、レプリケーション・ログファイルに記録するものとする。

表 1 LDAP 更新系要求

#	要求
1	エントリ追加
2	エントリ削除
3	RDN の変更
4	属性値の追加/変更/削除

レプリケーション・ログファイルに記録されている個々の要求内容をレプリケーション・レコードと呼ぶ。レプリケーション・レコードには表 2に示す情報が含まれる。DN(Distinguished Name)とは、ディレクトリ・ツリー中でエントリを一意に識別するための名前であり、ツリーのルートから対象エントリ

に至るRDNを並べたものである。

表2 レプリケーション・レコード項目

#	項目
1	コンシューマのホスト名、ポート番号
2	要求受信時刻を示すタイムスタンプ
3	操作対象DN
4	変更操作の種類(表1参照)
5	属性型
6	変更後の属性値

5. 動的差分レプリケーション

5.1. 概要

従来のレプリケーションではレプリカ・データが失われた場合、全マスタ・データのコピーを再取得することで再構築を行う。しかし以下に挙げる点が、基幹システムでは大きな問題となる。(1)マスタ・データをコピーするには一時的にサプライヤの更新実行を禁止しなければならず、サプライヤ側の可用性を低下させる。(2)また、大量のマスタ・データコピー及び転送には長時間を要するため復旧が遅れ、コンシューマ側の可用性が低下する。(3)さらに、マスタ・データコピーにはサプライヤ側管理者作業が必要であり、コンシューマ側のみならずサプライヤ側管理者にも回復作業を強いることになる。

そこで上記問題を解決するため、以下の動的差分レプリケーション方式を開発した。

(1) レプリケーション・レコードにレコードの処理順序を示すレコード識別情報を付与する。

(2) レプリケーション・サーバは、レプリケーション処理に用いたレコードのレコード識別情報をコンシューマのディレクトリ・データ中に登録する。これを最終レコード情報と呼ぶ。

(3) レプリケーション・サーバは、レプリケーション開始前に最終レコード情報をコンシューマから取得し、それが示すレプリケーション・レコード以降のレコードを処理する。

本方式とコンシューマ側での定期的なレプリカ・データのバックアップ、およびサプライヤ側でのバックアップ取得以降のレプリケーション・レコードの保存を組み合わせることにより、以下の手順によるレプリカ・データ復旧を可能にした。

(1) 最終レコード情報を含むバックアップを新たなレプリカ・データとしてリストアする。

(2) リストアしたレプリカ・データに対してレプリケーションを再開する。

以上により、(1) 従来のようにマスタ・データをコピーする必要は無いため、サプライヤを更新禁止にすることが無く、(2)データ転送量はバックアップ取得以降の更新分のみで済むので迅速な復旧が可能になる。また、(3)復旧作業はコンシューマ管理者作業の範囲内であり、サプライヤ管理者作業は不要である。

5.2. レコード識別情報の管理

はじめにレプリケーション・レコード識別情報について説明し、次にコンシューマにおける記録形式、登録、参照方法を述べる。

(1) レコード識別情報

レプリケーション・レコードを一意に識別し、その処理順序を明確にするためレプリケーション・レコードにタイムスタンプおよび番号付けを行う。これらをレコード識別情報と呼ぶ。レコード番号は、サプライヤが更新操作を実行した順序にレコードに付与する整数値である。タイムスタンプは、更新操作がレプリケーション・ログファイルに出力される時刻であり、1970年1月1日午前零時からの経過秒数である。タイムスタンプのみでは一秒以内に実行された更新操作を区別できず、番号のみでは番号が一巡し同一番号のレコードが生じた時に区別できないので、タイムスタンプと番号を併用する。

(2) レコード識別情報格納オブジェクトクラス

オブジェクトクラスはエントリの持つ属性を定義するものである。X.500 データモデルでは、エントリは必ず一つ以上のオブジェクトクラスに属し、オブジェクトクラスの定める属性の集合を持つ。

ここで、最終レコード情報は、hitachiDirectoryServer クラスに属するエントリが持つ lastReplRecord 属性とする。以下、hitachiDirectoryServer クラスおよび lastReplRecord 属性を定義する。

hitachiDirectoryServer クラスは、ディレクトリ・サーバを表し、hitachiApplicationProcess クラスはアプ

リケーションを表すオブジェクトクラスである。ISO8824 として規定された ASN.1（Abstract Syntax Notation One）による hitachiDirectoryServer クラス、hitachiApplicationProcess クラス、lastReplRecord 属性型、lastReplRecord 属性値のフォーマット定義を以下に示す。

(a) hitachiApplicationProcess クラス

hitachiApplicationProcess OBJECT-CLASS ::=
{SUBCLASS OF { dSA }
MAST CONTAIN { ipAddress, hostName } }

(b) hitachiDirectoryServer クラス

hitachiDirectoryServer OBJECT-CLASS ::=
{SUBCLASS OF {hitachiApplicationProcess }
MAY CONTAIN
lastReplRecord,userCertificate }}

(c) lastReplRecord 属性型定義

(<oid> NAME 'lastReplRecord'
DESC 'last handled Replication Record identifier'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String')

(d) lastReplRecord 属性値フォーマット

lastReplRecord :=
<hostname>%< record number>%<timestamp>

(3) レコード識別情報の登録

コンシューマは最終レコード情報をコンシューマを表すエントリの持つ lastReplRecord 属性値として保持しているので、レコード識別情報の登録は LDAP 属性値変更要求によって行う。レプリケーション・サーバはレプリケーション要求が成功した後、lastReplRecord 属性値を変更する LDAP 属性値変更要求を発行し、最新の値に書き換えなければならない。

(4) レコード識別情報の参照

レコード識別情報の参照は LDAP 検索要求によって行う。レプリケーション・サーバは、レプリケーション開始時にコンシューマに対して lastReplRecord 属性値を得る LDAP 検索要求を発行し、最終レコード情報を取得する。そして、取得した最終レコード情報が指すレプリケーション・レコード以降レコードをレプリケーション・ログファイルから読み出す。

5.3. シーケンス

動的差分レプリケーションのシーケンスを図 2に示す。

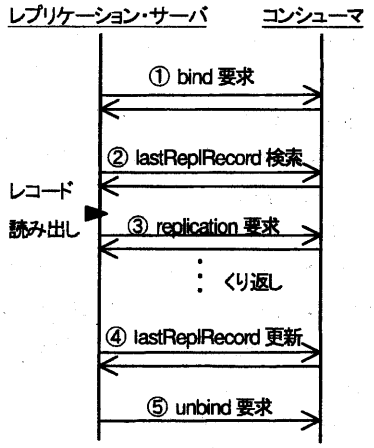

図 2 動的差分レプリケーションシーケンス

① バインド要求を送信し、レプリケーション・サーバとコンシューマ間にコネクションを確立し、セッションを開始

② lastReplRecord 属性値を検索：最終レコード情報より、前回のセッションで最後に処理されたレコードのレコード識別情報を取得

③ レプリケーション要求を送信（複数回でも可）

④ lastReplRecord 属性値を更新:今回のセッションで最後に処理されたレコードのレコード識別情報を新たな最終レコード情報として登録

⑤ アンバインド要求を発行し、セッションを終了

5.4. Controls によるメッセージ削減

LDAPv3 では、アクセス要求に開発者が独自の情報を付加するための Controls フィールドと呼ばれるパラメータが定義されている。あらかじめ、独自の拡張機能をディレクトリ・サーバプログラム内に実装し、Controls によってその拡張機能の実行をディレクトリ・サーバプログラムに指示することで、特定の Controls を持つアクセス要求処理の際に任意の拡張機能を実行させることが可能になる[ref.2]。

この機能を用いレコード識別情報を各レプリケー

ション要求に付加することで、lastReplRecord 属性値更新のための LDAP アクセス要求発行が不要になり、データ転送量を削減可能である。筆者らが開発したレプリケーション・サーバにおいてこの機能は未実装であるが実現は容易である。以下、拡張機能を用いた動的差分レプリケーションについて述べる。

(1) repControl

lastReplRecord 属性値を更新させる Controls フィールドを repControl と名付ける。ASN.1 による repControl の表記を以下に示す。controlType は、拡張機能を一意に識別するためのパラメータであり、CCITT の X.208 勧告に記述された全世界で一意な識別子（OID：Object Identifier）でなくてはならない。repControl の controlType を表す識別子を仮に <repOid> とする。criticality には、コンシューマが <repOid> に対応する拡張機能の認識または実行ができない場合、エラーとして処理させる場合は TRUE、エラーとせずに repControl を無視させる場合は FALSE を選択する。RepControl では、デフォルトとして TRUE を設定するものとする。controlValue には、拡張された機能が使用する値がオクテット文字列として設定されなければならない。repControl の controlValue には lastReplRecord 属性値を設定する。

repControl ::= {
controlType <repOid>,
criticality TRUE,
controlValue <lastReplRecord> }

(2) サーバ側拡張処理方式

LDAP アクセス要求を処理した後、コンシューマを表すエントリの lastReplRecord 属性値を repControl の controlValue に設定されている値に更新する。

5.5. トランザクション機能による高信頼化

従来のディレクトリ・サーバでは、動的差分レプリケーション方式におけるレプリケーション要求実行とレコード識別情報の更新がディレクトリ・サーバ内部で独立した操作として行われる。このため、二つの操作を実行する間に障害が発生した場合、ディレクトリ・データとレコード識別情報の一貫性を維持できない。この問題は、コンシューマが複数の LDAP アクセス要求または一つの LDAP アクセス要求から生じる複数のデータ・ベースへのアクセスを、単一のトランザクションとして処理する機能を備えることで解決できる。基幹システム対応のディレクトリ・サーバにはトランザクション機能に対するニーズも多く、トランザクション機能の実用化も進められつつある[ref.4]。

コンシューマがレプリケーション要求とレコード識別子の更新要求をトランザクション処理することにより、レプリカ・データと lastReplRecord 属性値との一貫性が保証できる。したがって、電源断等の障害によるレプリケーション中断後も管理者は進行状態を意識せずレプリケーション再開可能であり、管理負荷は大幅に軽減される。

6. おわりに

ディレクトリサービスは基幹業務系を含む企業情報システム構築の基盤技術として期待され、ディレクトリ・データの分散配置を実現するレプリケーションへのニーズも高い。そこで、基幹システム対応のディレクトリ・レプリケーション技術として、障害によるレプリカ・データ破壊からの復旧を容易化し、可用性および運用性を高める動的差分レプリケーション方式を備えたレプリケーション・サーバを開発し、実用化の見通しが得られた。

参考文献

[1] ISO/IEC 9594-1～9/ CCITT X.500 シリーズ

[2] W.Yeong,T.Howes,S.Kille : "Lightweight Dirctory Access Protocol(v3)": RFC2251,(1998)

[3] C. Weider, J. Strassner, "LDAP Multi-master mazu Replication Protocol", INTERNET-DRAFT draft-ietf-asid-ldap-mult-mast-rep-00.txt, Microsoft Corporation, Cisco Systems Corporation, (1997)

[4] 菊地 聡 他:基幹システム高信頼ディレクトリ・サーバ:情報処理学会研究報告書 97-DSM-7,情報処理学会(平9)