分散OS XEROにおける協調処理のためのトランザクション機構の検討

2P−7　坂田尚也　加藤和彦　猪原茂和　益田隆司
東京大学　理学部　情報科学科

1　はじめに

分散オペレーティングシステムXEROの分散共有格納庫(DSR)は，LANのようなネットワーク環境において，複数のファイルサーバを用いたファイルの位置独立な単一の名前空間及びファイルに対する統一的なアクセス機構を実現し，ユーザに情報の分散と永続のための言語独立な機構を提供する[1].

DSRのこの分散性と永続性を利用したアプリケーションの一つに，複数のユーザによる共同設計作業やソフトウェアの共同開発作業を支援するシステムが考えられる．そのようなシステムでは，ユーザは会社の組織のように開発モジュール毎にグループ分けされ，各グループで比較的独立にモジュールの開発を行ない，完成品を他のグループに公開するという形をとることが多い．また各グループにおけるモジュール開発においてもユーザはさらに小グループに細分化され，同様に小モジュールの開発を行なう．つまりモジュールの階層性に対応してユーザは階層的にグループ分けされ，グループ内のユーザ同士で協調してモジュールの共同開発を行い，そのモジュールが完成した段階でもう一層上のグループのユーザに公開する．このときに，ユーザがどのグループに属していようとも同一の名前空間が提供され，且つ所属するグループに応じて適当なファイルの実体にアクセスする機能が望まれる．

本稿では，DSRの一拡張としてExperimental DSR (EDSR)と名付けられた永続情報の管理空間を提案する．EDSRはDSRまたはEDSRの部分空間として階層的に定義され，部分空間内のファイルの名前と実体のマッピングを変更する機能を持つ．よってEDSRを用いることで，グループにおいて開発途中をモジュールの保存，共有する場所は自然に表現できる．

このように状況に応じてマッピングを変更する機能は，多数の異機種コンピュータが接続されたネットワーク環境で複数のユーザがDSRを利用する，という通常の利用環境においても非常に有用である．例えば多数の異機種コンピュータのネットワークにおいては，異機種間で共有できないファイルを各機種毎にEDSRとして定義すれば，同じ名前でマシン毎に適切なファイルにアクセスすることができるし，EDSRを用いた複製を作ることで速度及び可用性を向上できる．また，複数のユーザがDSRを利用をする場合において，ユーザ毎にEDSRを用意すれば，各ユーザは自分用にカスタマイズしたファイルに対しても元と同じ名前でアクセスすることができる．

以下，2章では分散OS XEROのDSRを用いた計算モデルを述べ，3,4章でその拡張としてのEDSRの定義とその実現方法をそれぞれ述べる．

2　DSRに基づいた計算モデル

DSRには情報はコンテキストという単位で格納され，通常のデータとプログラムだけでなく，実行状態のプログラムも統一して扱われる[1]．このDSRを用いたXEROにおける計算モデルは，タスク，スレッド，コンテキストの3つの言葉を使って表現できる(図1参照)．タスクは線形な仮想空間とTask Supervisor(TSV)と呼ばれるスレッドのスケジューリングやカーネル，DSRサーバとの通信などを行なう特別なライブラリとから成る．スレッドはコンテキスト上を走る実行主体である．そして計算は，タスクがコンテキストをDSRから自分の仮想空間にロード／アンロードすることによって行なわれる．この時にコンテキスト中のアドレス情報の再配置が行なわれるので，永続データの直接操作やパラメータとしてコンテキストをタスク間で受け渡すことができる．またDSRでは実行状態のプログラムもコンテキストとして扱われるので，DSRを介した形でタスク間でのコンテキスト移送や，実行中のプログラムのチェックポイントも可能である．

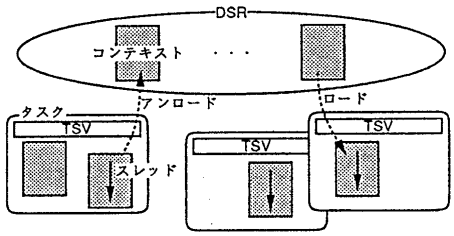

図1: DSRに基づいた計算モデル

3　Experimental DSR

Experimental DSR (EDSR)は，DSRと同様に分散環境上でコンテキストを格納するための永続空間であるが，EDSRはDSRまたはEDSRの子として階層的に定義され，親EDSRの名前空間の部分集合から成る名前空間を持つ．そして子EDSRでは，親EDSRと同じ名前に対して親EDSRとは異なるコンテキストにマップできるだけでなく，親EDSRにおいてその名前にマップされているコンテキストに対して，copy-on-writeまたはcopy-on-registrationのマッピングのリンクを張ることができる．copy-on-writeとは初めて書き込みを行なう時に子EDSRにコピーを作り，copy-on-registrationは親EDSRのコンテキストをそのまま子EDSRにコピーする．このとき各タスクはどこか一つのEDSRに所属することで，名前空間は元々のDSRと同じものが提供されることに変わりはないが，実際にアクセスされるコンテキストは所属するEDSRから祖先方向にEDSRをマージしたものであり，各名前に対して最も子孫のEDSRにおけるマッピングが適用され，結果として各タスクが所属するEDSRに応じて異なる．

このような階層的なEDSRを導入した結果，全体の名前空間は，各ユーザから見える単一の名前空間，管理者から見たコンテキストの論理的な違いを表す名前空間，物理的な違いを表す名前空間，の3階層できたことになり，以下のような名前の変換が行なわれる．ただし，ユーザレベルと論理レベルのパス名は同じものでなければいけないが，物理レベルのパス名はそれと異なるパス名でも構わない．そこで以下，前者を論理パス名，後者を物理パス名と呼んで区別する．

ユーザレベル	pathname	
	↓	コンテキスト単位で共有可
論理レベル	EDSR name : pathname	
	↓	ディレクトリ単位で共有可
物理レベル	hostname : pathname′	

Cooperating Transactions for the XERO Distributed Operating System by Naoya SAKATA, Kazuhiko KATO, Shigekazu INOHARA, and Takashi MASUDA (Dept. of Information Science, Univ. of Tokyo)

またEDSR間では物理レベルを通じてのディレクトリ単位でのコンテキストの共有，親子EDSR間ではコンテキスト単位での共有が可能であり，このEDSRを用いることで，異機種ネットワーク環境でのソフトウェアの共同開発作業の階層性は，図2のように直接表現できる．

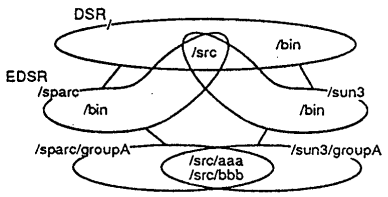

図 2: EDSRを用いた共同開発作業支援

4 実現方法

EDSRは，複数のディスクがあるネットワーク環境において利用されることを想定している．各ホストにはDSRサーバがあり，ローカルなディスク中のコンテキストの論理レベル名を管理する．一方DSRやEDSRは論理的なものであるので，(E)DSR内のコンテキストは一般に分散した複数のサーバにまたがって存在する．よってユーザレベル名からそのコンテキストを管理するサーバの名前解決(name resolution)には効率的な実現方法が必要になる．本稿ではこの名前解決にSpriteのprefix表のアルゴリズム[2]を拡張したものを用いる．

コンテキストの論理レベル名は，次の2つの表1,2によってそのコンテキストが存在するディスクを管理するDSRサーバに登録される．ただし，サーバが直接管理するディスク上のコンテキストにはUNIXのような階層的な名前が付けられているとする(物理パス名)．

EDSR name	論理パス名 prefix	物理パス名
/	/src	/src
/sparc	/src	/src
/sun3	/src	/src
/sparc	/bin	/sparc/bin

表 1: ホストhancockのコンテキスト登録テーブル

名前	リンク (*remote*,*cp-on-wr*,*i-number*)
aaa	*i-number*
bbb	*i-number*
ccc	*remote*

表 2: ホストhancockのディレクトリ/srcの中身

表1は，コンテキストの論理レベル名をディレクトリ単位で登録するものである．ただしprefixとはコンテキストの絶対パス名の前半部分のことであり，例えば/sparcというEDSRで/binから始まる論理パス名を持つ論理レベル名のコンテキストは，ホストhancockの/sparc/binという物理パス名のディレクトリ以下に存在する可能性がある．そして実際に存在するかどうか，コンテキスト単位で論理レベル名の登録をするのが表2である．これはディスク中の通常のディレクトリの中身であり，名前とともに他のEDSRへのリンク情報が格納されている．この名前に関しては論理パス名と物理パス名で同じでなければならない．

一方，ユーザレベル名から物理レベル名への変換，つまりコンテキストの検索は，効率的な変換を行なうために，表3のような拡張されたprefix表をヒントとして用いる．まずタスクが所属するEDSRにおいてその論理レベル名のコンテキストが存在するか調べる．無かった場合には祖先方向に向かって，各EDSRにおいてその論理レベル名のコンテキストが存在するかどうか，prefix表のアルゴリズムを用いて調べることを繰り返す．最初に見つかったコンテキストが目的のコンテキストである．このprefix表は各DSRサーバにそれぞれ存在し，初期状態は何のエントリも無い状態なので何のヒントにもならないが，キャッシュによって徐々に埋められていくので，2度目からは効率的な検索が可能となる．

EDSR name	論理パス名 prefix	server または EDSR name
/	/src	hancock
/sparc	/bin	hancock
/sun3	/src	/

表 3: ホストevansのprefix表

例えばあるホストevans上の/sun3というEDSRに所属するタスク中のスレッドがコンテキスト/src/aaaを要求した場合，以下のような手順で目的のコンテキストが見つけられる．

1. TSVが自分の所属するEDSR名を付加して，ローカルなDSRサーバに論理レベル名'/sun3:/src/aaa'を要求する．
2. evansのDSRサーバは，自分の管理するディスクに目的のコンテキストがないかどうか，evansにおける表1を調べる．
3. 無い場合は，prefix表3でEDSR名が'/sun3'で且つ論理パス名'/src/aaa'が最長一致するprefix '/src'の最後のフィールドを見て，'/'を得る．
4. '/:/src/aaa'という名前で2から繰り返してhancockを得る．
5. hancockのDSRサーバに'/:/src/aaa'を要求する．
6. hancockのDSRサーバは表1から物理パス名'/src'を得る．
7. 表2から'aaa'を探し，目的のコンテキストを見つける．

このアルゴリズムはSpriteのprefix表のアルゴリズムを各EDSRにおいて繰り返すが，コンテキストの登録情報に変化が起こらない安定した状態では，何度も各EDSRでの検索を繰り返すことなく，物理レベル名への直接の変換が可能である．これは親子EDSR間の登録の関係もprefix表に記録したためであるが，このように沢山ある各EDSRの情報をprefix表に記録するため，prefix表は元のSpriteのものと比べて非常に大きくなる可能性がある．これに対処するためのprefix表の圧縮は今後の課題である

5 おわりに

本稿では，多数の異機種コンピュータが接続されたネットワーク環境において，複数のユーザによって利用されるDSR，特にソフトウェアの共同開発作業等の支援を目的として，階層的なグループ毎に名前と実体のマッピングを変える機構EDSRとその実現方法を述べた．現在はその最初のバージョンをUNIX上に実装中である．

参考文献

[1] K. Kato, A. Narita, S. Inohara, and T. Masuda. Distributed shared repository: a unified approach to distributed and persistent information management. Technical Report 92-1, Department of Information Science, Faculty of Science, University of Tokyo, February 1992.

[2] B. Welch and J. Ousterhout. Prefix tables: a simple mechanism for locating files in a distributed system. In *Proc. IEEE Int. Conf. on Distributed Computing Systems*, pages 184–189, 1986.