メディア変換機構を提供する分散永続オブジェクトシステム

高野了成[†] 佐藤元信[†]
早川栄一[††] 高橋延匡[††]

分散環境において協調作業を行う場合，利用環境や共有相手の環境に応じてデータの表現形式を変換したり，データ間の整合性を管理するといった操作を一貫した枠組で行いたいという要求がある．我々は，このような作業を capture-arrange-publish の 3 フェーズから成るモデルとして定義し，メディア変換機構を提供する分散永続オブジェクトシステムを設計した．本システムの特徴は，(1) オブジェクト間の関連性を容易に記述するために，位置透過かつ単一アドレッシング可能なオブジェクト空間を提供すること，(2) メディア変換を効率的に行うためのキャッシュ機構とその一貫性を保証するための履歴管理を提供すること，(3) メディア変換時の対話的処理や定型処理に対する支援としてスクリプト言語の呼出し機構を提供することである．

A Distributed Persistent Object Store with Media Transcoding Mechanism

TAKANO RYOUSEI ,[†] SATO MOTONOBU ,[†] HAYAKAWA EIICHI [††]
and TAKAHASHI NOBUMASA [††]

A distributed computing system requires appropriate operations that depended on user's environment: data convertion, pattern recognition and so on, and consistency control of sharing compound documents in cooperative works. Therefore, We proposed a data managemnet model consist of 3 phases: capture, arrange, publish, and designed a distributed persistent object store system with media transcoding mechanism. The feature of this system is following: (1)presenting a platform and API to easily describe relations between data that is supported by single addressing object space management and location transparency of objects, (2)efficient media transcoding that is supported by improvement of cache utilization and consistency management according to access patterns, and (3)calling mechanism of scripting language to support interaction and routine work.

1. はじめに

近年，計算機を文書作成や，研究開発などの発想活動支援に利用する機会が増加している．個人が PDA などの携帯端末を含めた性質の異なる複数台の計算機を所有し，利用環境に応じて使い分けたり，複数人でデータを共有し，協調作業を行うことも多い．このような場合，データは互いに密な関連性を持ち，オリジナルが同一のデータでも作業環境や共有相手の環境に応じてデータ表現を変換することが必要になってくる．さらに，データの編集段階において，取得したデータに対して認識や抽出などの処理を行ないデータの抽象度を上げていく必要がある．そこで，不定型で断片的なアイデアを逃さず記録し，参照関係，順序関係などの構造を保持したまま共有空間へ格納でき，加工，編集しやすいデータ形式に変換する機構を提供するフレームワークが求められている．

† 東京農工大学工学部
Faculty of Engineering, Tokyo University of Agriculture and Technology
†† 拓殖大学工学部
Faculty of Engineering, Takushoku University

我々はペンインタフェースによるグループウェアである分散手書き KJ 法システム[3] や，複数人によるソフトウェア開発を支援する電子研究ノート[4] などの研究を行ってきた．これらのアプリケーションでは，発想の記録を重視して，まずは，データ自体をビットマップ画像や，筆点列といった一次情報で格納し，後に文字列やリンクなどを持つ構造化されたデータなどへ変換することを想定している．この場合，一つのデータが複数のデータ表現を持つことになり，これらを効率的に管理する機構が必要となってきた．さらに，計算機環境の多様化によって GPS やセンサなどから環境情報を取り込んだり，動画像などマルチメディアデータの編集に対する要求も出てきた．

このような要求に対して，我々はユーザの利用環境に応じて，分散したデータを一貫して管理できるフレームワークを実現することを目的とし，メディア変換機構を特徴とする永続オブジェクトモデルである「意紙」[1] を用いたデータ管理システムを提案した．「意紙」は一つのデータに対する複数の表現形式を抽象化し，分散透過な永続オブジェクトを提供する．

本稿では，まずターゲットである発想活動支援における計算機利用の作業モデルを定義し，「意紙」に基づく永

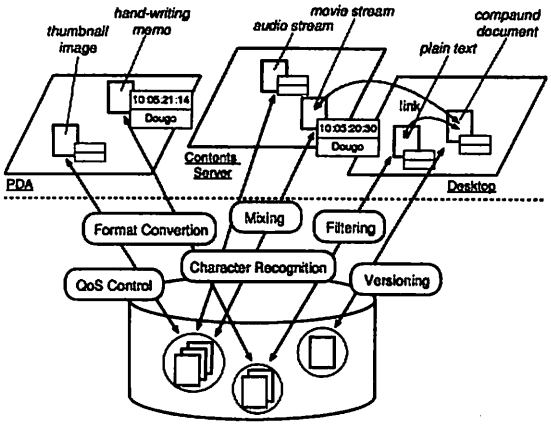

図 1 想定する作業シナリオ

続オブジェクトシステムの設計，OS/omicron 第 4 版上への実装方式について述べる．

2. 要求分析

2.1 想定する作業シナリオ

図 1 では，我々が計算機に蓄積されたデータをいかに加工し，編集することを考えているか示す．計算機環境は PDA，デスクトップ PC，大容量ストレージや高速入出力機器を持つコンテンツサーバから構成され，各データは生成時の時間や位置といった環境情報をもっている．

デスクトップ上から位置情報によるフィルタリングを実行した場合，動画ストリーム，手書きメモがあることがわかった．計算機環境に合わせて動画の QoS を変換したり，動画にインデックスタグを付け，そのサムネイル画像を生成したり，手書きメモから文字認識によってテキストを生成するといったデータ加工を行なう．そして，これらの素材を組み合わせて，複合文書を作成するため，さまざまなデータ間のリンクを行なうことになるが，データがどの計算機上に存在するかを透過に操作することができる．また，編集中のデータはいつでも後戻りできるよう履歴が残されている．

2.2 作業モデル

このような作業に対して，計算機を利用して知識を効率的に蓄積，編集，再利用することが重要であるという視点から，我々は capture-arrange-publish[2] の 3 フェーズから成る作業モデルを採用した．

この作業モデルでは，発想過程を重視するため，さまざまなフォーマットの一次情報や，データ取得に伴う時間情報，位置情報などの属性情報を格納する (capture)．そして，取得したデータに対して，属性情報を利用した検索やフィルタリングを行ったり，データ間のリンクを操作して，抽象度の高いデータへ加工，編集する (arrange)．最後に，WWW や紙への印刷物 (出力) として，または他のモジュールへの入力フォーマットとして出力する (publish)．これらのフェーズがサイクルとなりループすることで，さらに知識が蓄積，利用されていく．

2.3 メディア型とメディア変換

分散環境ではデータに対して (1) 個人での共有性と (2) 他者との共有性が生じる．(1) は PDA とデスクトップ PC など性質の異なる複数の計算機で同一のデータを扱う点，(2) はデータフォーマットの可搬性が必要になる点が問題になる．そこで，各フェーズで適切な操作が行えるように，データ共有に伴う変換処理を一貫して扱う枠組みをシステムが提供することが要求される．なお，データの表現形式を変換することをメディア変換と呼び，各表現形式をメディア型と呼ぶ．メディア変換の例として，画像，音声のフォーマット変換，文字認識，OCR(Optical Character Reader) などのパターン認識，仮名漢字変換，文書整形などが挙げられる．

2.4 システムへの要求

上記の要求分析をまとめると，capture-arrange-publish という作業サイクルにおいて，ユーザの利用環境に応じた一貫した処理のフレームワークが要求されていると言える．データを公開する場合は，標準化されたフォーマット形式を利用する方が便利であるが，オリジナルデータや様々な属性情報が付随している編集中のデータも発想過程の記録として利用価値が残っている．このように細粒度のデータが複雑な関連性を持つデータ構造を効率的に管理，操作できることが要求される．さらに，複数の計算機間で協調作業を行う場合は，複製の一貫性管理が必要になる．

また，メディア変換には変換の可逆性，データ品質の制御を考慮する必要があるが，メディア型が多様になり，単純な QoS ポリシによってこれらを宣言的に制御するのは困難である．そこで，変換によってデータの欠損が起きることを通知したり，変換に伴うユーザとの対話処理を支援するためのプログラミングインタフェースの提供が必要である．

3. 設計方針

3.1 目　　標

本研究の全体目標を次に示す．

- データの共有が容易なオブジェクト管理
  データ間に密な関連性を持つようなデータ構造を扱うため，関連性をプログラミング言語から容易に操作できるようにする．さらに，分散環境に透過なオブジェクト操作を可能にする．
- 拡張可能なメディア変換機構
  メディア型やその操作の種類はシステムがあらかじめ決定できないので，動的な定義，拡張を可能にする．また，メディア型のアクセスパターンに応じた一貫性ポリシの選択を可能にする．
- 環境情報を考慮した履歴管理
  発想過程の記録を残すため，データを時空間情報などの環境情報と結びつけた履歴管理を提供する．
- スクリプト言語の呼出し機構

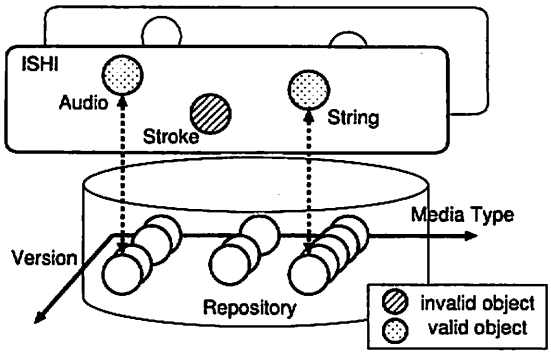

図 2 「意紙」モデル

メディア変換の結果を静的に記述することが困難な対話処理や定型処理は，システムを変更することなくユーザが記述したスクリプトを呼び出すことで処理できるようにする．

3.2 「意紙」の概念

マルチメディアデータなど多態的な性質を持つデータは，一つの実体に対して認識処理やフォーマットの違いから複数の表現形式を持っていると言えるが，このような複数のメディア型のオブジェクトを一つの集合として扱えるようにする．このような多態的オブジェクトを「意紙」と呼び，「意紙」に対してメディア変換を行うことで，要求するメディア型のオブジェクトを得ることができる．「意紙」のモデルを図 2 に示す．

さらに，「意紙」は各メディア型のオブジェクトに対する履歴の集合でもある．あるメディア型のオブジェクトを変更した場合，他のメディア型のオブジェクトとの一貫性が崩れる．このような影響の波及はアプリケーション依存であり，すべてのオブジェクトの状態を最新に合わせるように自動化することは困難である．そこで，変更によって無効化されたオブジェクトは消去するのではなく，履歴として保存しておく．このように履歴を保存することでメディア型間でのオブジェクトの一貫性や複数人での協調作業の整合性を保証すると共に，作業過程を残すことにもなる．

図 2 では，「意紙」は三つのメディア型のオブジェクトから構成されているが，最新の状態として整合性が取れているのは音声型と文字列型だけであり，筆点列型は無効オブジェクトとなっている．筆点型のオブジェクトを操作したい場合は，他のメディア型からメディア変換するか，履歴操作によって過去の状態を見る必要がある．

3.3 メディア変換機構

メディア変換の様子を図 3 に示す．メディア変換機構は変換元，変換先のメディア型，変換メソッドや，変換メソッドに対する可逆性などのヒント情報を管理する．オブジェクトを参照する際，指定したメディア型のオブジェクトが存在すればアクセスは成功するが，オブジェクトが存在しない場合は，メディア変換機構によって現存するオブジェクトから新規にオブジェクトが生成される．

図 3 で示すように，ある計算機 (ホスト A) で入力したストローク型オブジェクトに対して，別の計算機 (ホスト B) が文字列型として読み込む場合，その時点でメディア変換が実行され，文字列型オブジェクトとしてアクセスできる．また，メディア変換自体は言語に非依存であり，ホスト C のようにスクリプト言語からの操作も可能である．

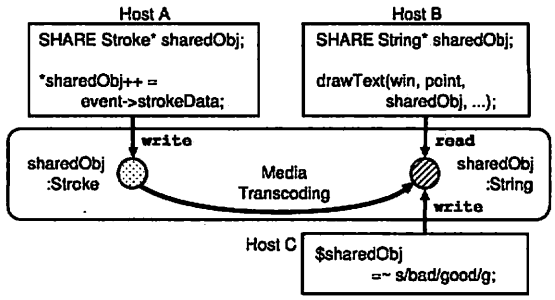

図 3 メディア変換機構

4. 設　　計

4.1 プログラミングモデル

永続オブジェクトを一意の仮想アドレスを割り当て，プログラミング言語からはオブジェクトへのハンドルとして共有変数を利用する．共有変数は名前と型を持ち，型はメディア型と対応する．オブジェクトのハンドルとして変数を利用する以外に，オブジェクト指向データベースにおけるデータベースルートのような open インタフェースを採用する方法も考えられる．本システムで変数によるハンドリングを採用したのは，一時オブジェクトと永続オブジェクトを透過に扱うためである．

システムで一意な共有変数とダイナミックリンク機構を利用し，分散オブジェクトのバインディングを実行時に解決することで，アプリケーションはオブジェクトを位置透過に扱えるようになり，オブジェクトの所有者が変わっても，プログラムを変更する必要はない．なお，所有者とは永続オブジェクトが存在する計算機のことであり，ローカル計算機の仮想アドレス空間へマップされた永続オブジェクトの複製をキャッシュオブジェクトと呼ぶ．

4.2 分散永続オブジェクト管理機構

本システムでは，複数ユーザとの協調作業などにおけるオブジェクト共有を目的としているので，共有したい永続オブジェクトの一貫性だけを管理することで，一貫性保持のコストを軽減することができる．

そこで，オブジェクトを一時オブジェクトと永続オブジェクトに分類する．一時オブジェクトは各計算機ローカルで有効なオブジェクトであり，永続性を持たない．永続オブジェクトは，全システムでの一意性が保証される．永続オブジェクトの一意性はシステムレベルで提供するので，言語非依存であり，永続オブジェクト間の参照をポインタ変数の構文として記述することが可能になる．したがって，オブジェクト間の関連性の記述が容易になる．

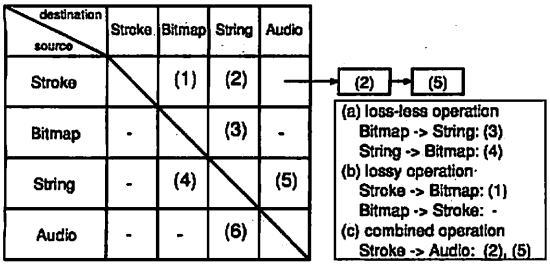

図 4　メディア型と変換メソッドの対応

```
STATUS ishi_register_media_type(CHAR* dtypo);
STATUS ishi_release_media_type(CHAR* dtypo);
STATUS ishi_register_transcoder(CHAR* stypo, CHAR* dtypo
                                CHAR* func, ULONG hint);
STATUS ishi_release_transcoder(CHAR* stypo, CHAR* dtypo,
                               CHAR* func);
```

(a) media transcoding operation

```
STATUS ishi_commit_object(OID id);
STATUS ishi_abort_object(OID id);
STATUS ishi_get_version_info(OID id, VERINFO* ver);
STATUS ishi_checkout_object(VERINFO* ver, OID* id);
STATUS ishi_merge_version(VERINFO* ver, OID id);
STATUS ishi_branch_version(VERINFO* ver, OID* id);
```

(b) versioning operation

```
LOCK   dom_create_lock(VOID);
STATUS dom_delete_lock(LOCK lock);
STATUS dom_lock(LOCK lock, TIME timeout);
STATUS dom_unlock(LOCK lock);
```

(c) lock variables operation

図 5　提供する主な API

4.3　メディア変換機構

メディア変換機構では，メディア型と変換メソッドの対応を図 4 のような 2 次元の表と変換に伴うヒント情報によって重み付けされたリストで表す．リストに対する API を図 5(a) に示す．変換の種類にはビットマップ型と文字列型のような (a) 可逆変換，筆点列とビットマップ型のような (b) 非可逆変換，筆点列型から音声型のような複数の変換を組み合わせることで変換できる (c) 複合変換が存在する．もし変換に対して複数のパスが存在する場合は，利用環境に応じたパスの変換を選択できるようにする．個々のメディア型に関しては QoS パラメータを設定すれば，(b) のような非可逆変換を実現することは比較的簡単だが，(c) の場合，QoS ポリシをルール化することは困難である．そこで，デフォルト変換は示すが，対話処理やスクリプティングによってパスを選択するためのプログラミングインタフェースを提供する．

さらに，メディア変換におけるオブジェクト間の整合性を保証するために，オブジェクトに対するシステムグローバルなロック機構 (図 5) を提供する．

メディア変換時に変換元の永続オブジェクトの所有者がリモート計算機だった場合でも，ローカルにキャッシュオブジェクトが存在する場合はネットワーク通信が発生しない．また，対話処理が必要な場合も暗黙的にキャッシュオブジェクトを利用することで，ネットワーク通信量を削減できる．また，「意紙」内のオブジェクトの変更に伴う他のオブジェクトの無効化は，キャッシュオブジェクトの無効化と同様に扱うことができる．ただし，無効化されたオブジェクトは次節に説明するように履歴リポジトリに保存されるのであって，消去されるわけではない．

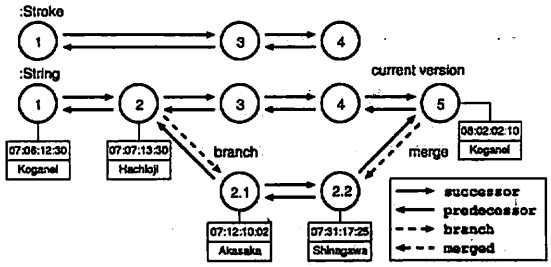

図 6　バージョングラフ

4.4　履歴管理機構

キャッシュオブジェクトに対する操作としてコミットとアボートがある．キャッシュオブジェクトをコミットすると，そのオブジェクトに対する変更は永続化され，バージョン番号が更新される．オブジェクトは「意紙」内でユニークなバージョン番号を持ち，最新バージョン以外のオブジェクトは履歴リポジトリに格納されている．

そこで，履歴管理機構は，図 6 のような各オブジェクトに対するバージョングラフを管理し，図 5(b) のような分岐，マージ，またグラフをたどるためのインタフェースを提供する．バージョングラフは，データ生成時の時間情報，位置情報などの作業環境情報と組で管理される．これによって誰がどんな環境でどんな変更を行なったかという作業履歴を蓄積できることになり，ユーザはこの情報からデータに対するフィルタリングや検索を実行することが可能になる．

図 6 に示した「意紙」は文字列型と筆点列型のオブジェクトを含んでおり，バージョン番号が同じものは，メディア変換で生成されたオブジェクトである．最新バージョンは文字列型にだけ存在するので，筆点列型のオブジェクトをアクセスしたい場合は，履歴操作インタフェースを明示的に指定する必要がある．

また，複数人でオブジェクトに対する協調操作を行なう場合，メディア型の一貫性管理ポリシによって，バージョニング方式を選択できる．バージョン分岐がある並列バージョンの場合，オブジェクトに対して同時に複数の書込みを許すことになる．一方，分岐のない直列バージョンの場合は，同時書込みは許さない．なお，マージはバージョングラフに対する操作であって，実際のオブジェクトに対する作業はユーザの責任である．

5.　実　　現

PC/AT 互換機で動作する OS/omicron 第 4 版 (以下，V4) における実装について述べる．ネットワーク環境は 10BaseT であり，NIC として 3COM 3C509，3C589 PCMCIA カードを用い，基盤通信プロトコルとして TCP/IP を使用した．

5.1　OS/omicron 第 4 版の概要

V4 は動的な機能拡張，構成変更を提供する OS であり，

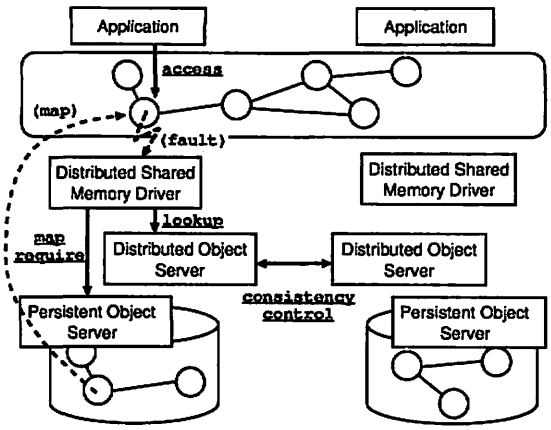

図 7 分散永続オブジェクト機構

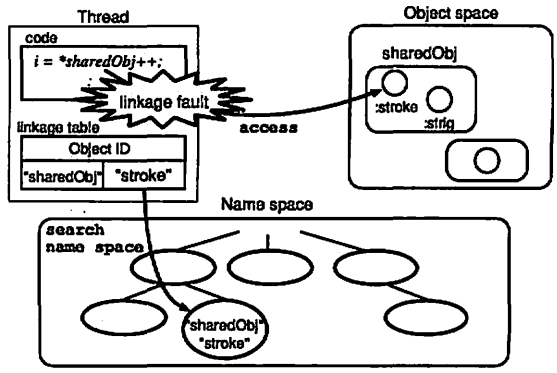

図 8 変数と永続オブジェクトのバインディング

マイクロカーネル，ダイナミックリンカ，メモリ管理システムなどのシステムモジュールと，OS パーソナリティ，ウィンドウシステム，プロトコルスタックなどのサーバ，モジュール群から構成される．

そして，デバイスなど，すべての資源はメモリ管理システムによってさまざまな属性を持つオブジェクトとしてメモリにマッピングされる．オブジェクト間のバインディングは実行時に動的に変更することができる．アプリケーションがこれらのオブジェクトにアクセスした場合，ダイナミックリンカがオブジェクトの属性，識別子名，型情報，実際のアドレスなどの情報を含むリンケージテーブル経由で名前空間を検索し，リンクを確定する．

さらに，OS レベルで共有される 64 ビット単一アドレス空間とセグメント化ページングを提供している．永続オブジェクトをセグメントとして実現することで，単一アドレス空間で問題になるオブジェクトの再配置のオーバヘッドを削減することができる．

5.2 分散永続オブジェクト管理機構

分散永続オブジェクト管理機構はオブジェクトへの (1) 参照の効率的な検出，(2) 参照をポインタと透過に扱う方法，(3) 一貫性管理の三つを考える必要がある．

まず，オブジェクト識別子と仮想アドレスを対応させることで，ハードウェアの MMU 機構を使い，オブジェクト参照を効率的に検出することが可能になる．さらに，システムでフラットな単一アドレス空間を採用することで，オブジェクト参照をポインタと透過に扱うことができる．一貫性プロトコルはメディア型のアクセスパターンの設定によって決定する．例えば，メディア型のバージョニング方式が直列バージョンの場合は write invalidate 方式を使用することで同時書込みが発生しないように制限し，並列バージョニングの場合は write through 方式を利用することで遅延書込みを許す．

分散オブジェクト機構のシステム構成を図 7 に示し，各要素について次に述べる．

- 分散共有メモリドライバ
  分散オブジェクトへのアクセスに対するセグメンテーションフォールト，ページフォールトなどのバッキングストア操作のポリシを提供する．
- 分散オブジェクトサーバ
  各ホスト上の分散オブジェクトサーバと協調して，共有変数の名前空間の同期，オブジェクトの一貫性管理，ロック変数の提供を行う．
- 永続オブジェクトサーバ
  永続オブジェクトやキャッシュオブジェクトに対するバッキングストアを管理する．また，ポインタ変換☆の管理も行う．

5.3 共有変数と永続オブジェクトの対応

永続オブジェクトに対するアクセスの流れを図 8 に示す．最初に共有変数を参照をした場合，リンケージフォールトが発生し，実行中のスレッドがブロックされる．リンケージフォールトはダイナミックリンカによってフックされ，名前空間から共有変数の識別子名，型名に対応するオブジェクトを検索する．検索が成功すると，オブジェクト識別子である仮想アドレスを得ることができる．

次に得られた仮想アドレスを参照するが，永続オブジェクトがキャッシュされていないので，セグメンテーションフォールトが発生する．セグメンテーションフォールトは，分散共有メモリドライバによってフックされ，分散オブジェクトサーバを介して，オブジェクトの所有者を検索する．そして，キャッシュオブジェクトをローカル計算機の仮想アドレス空間上に作成し，リモート計算機からオブジェクトをコピーし，キャッシュオブジェクトを書込み禁止属性に設定する．メディア型の一貫性ポリシとして直列バージョン方式を採用している場合は，書込み発生時に，リモート計算機に存在するキャッシュオブジェクトに対して無効化要求を出し，永続オブジェクトの所有者を自分に変更する．

このように共有変数のバインディングが解決され，スレッドの実行を再開する．これらの処理はユーザプログラム上では単なる変数への参照であり，システムによって暗黙的に実行される．

☆ x86 プロセッサ上の制限によりセグメントが 8192 個しか利用できない．そこで，Pointer Swizzling を応用することで，セグメント ID として 32 ビットの表現を利用できるようにした．オーバヘッドはフォールト処理時間の約 25% である．

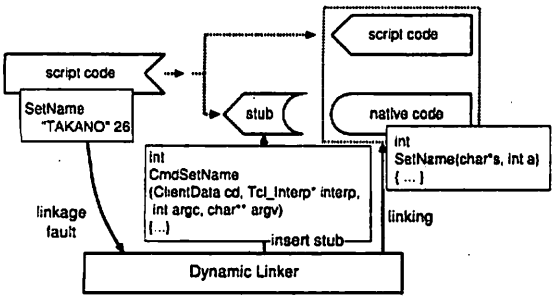

図 9 ネイティブ言語とスクリプト言語のリンク処理

5.4 スクリプト呼出し機構

ネイティブ言語からスクリプト言語，またその逆の呼出しなど，異なる言語で記述されたモジュールをリンクする場合，型変換やスタック操作，エンディアンなど言語間の差異が問題となる．これらの問題を解決する方法として，JNI(Java Native Interface) のような特別な手続きを利用したり，インタフェースを IDL(Interface Definition Language) で記述し，手続き呼出しを抽象化するスタブを自動生成する方法が一般的である．

V4 のダイナミックリンク機構では，システム構成の拡張性，柔軟性を高めるためにモジュールの動的結合に必要な情報をリンケージテーブルという構造に出力し，言語独立にしている．本システムでは，このダイナミックリンク機構を利用することで，動的に関数呼出しのバインディングを変更し，言語間の差異を吸収するためのスタブを挿入する．そして，スクリプト言語の属性が設定されたコード領域にインストラクションポインタが遷したときにフォールトを発生させ，インタプリタの実行，スクリプトの解釈を開始する [5]．さらに，スタブはシステムによって自動生成されるため，プログラマは IDL のような付加的なコードを記述する必要がないという利点がある．

本機構をスクリプト言語である Tcl に適用した．Tcl では基本的にデータを文字列として扱うため引数，戻り値の変換が必要であるが，リンケージテーブルの型情報を参照することで，図 9 のように C 言語で記述されたコマンドに対するスタブを自動生成できる．

6. 関 連 研 究

投機的処理を支援する世界 OS[6] は，並列世界モデルに基づいた名前空間を提供しており，各世界の変更に対する伝播として継承 (非対称)，対称的，独立の三つのセマンティクスが存在する．本システムでは，投機的処理もユーザの要求に先だってキャッシュを用意するメディア変換の一例と考えることができる．また，メディア変換に伴う変更のセマンティクスはユーザが決めることになり，システムは履歴管理のみを行う．

Thor[7] はヘテロな分散環境上で安全かつ効率的な永続オブジェクト管理を提供する．本システムがアドレス空間レベルでのオブジェクト共有を提供しているのに対して，Thor のオブジェクトは型安全な言語である Theta で記述されており，各言語ごとのスタブ (Veneer) を介してオブジェクトにアクセスしている．また，細粒度オブジェクトの共有を効率的に行うためにオブジェクトクラスタリングを行っており，本システムでも有効な手法だと考えられる．

7. ま と め

本稿では，メディア変換機構を提供する分散永続オブジェクトシステムの設計と，V4 における実装について述べた．本システムは，ユーザの利用環境に応じて，分散したデータを一貫して処理するためのフレームワークとして，次のような機能を提供している．(1) オブジェクト間の関連性を容易に記述するために，位置透過かつ単一アドレッシング可能なオブジェクト空間を提供する．(2) メディア変換を効率的に行うためのキャッシュ機構とその一貫性を保証するための履歴管理を提供する．(3) メディア変換時の対話的処理や定型処理に対する支援としてスクリプト言語の呼出し機構を提供する．

今後の課題として，メディア間のバージョン同期を支援するライブラリの実現，実用的アプリケーションによる評価が挙げられる．また，V4 以外のシステムが混在する環境での相互運用性を実現することで，さまざまなメディア変換を利用できる環境を構築する予定である．

参 考 文 献

1) 高野了成, 佐藤元信, 早川栄一, 並木美太郎, 高橋延匡: 多態的表現を可能にする永続オブジェクト管理機構, 情報処理学会研究会報告, 99-OS-81, pp.1-6, 1999.
2) Eiichi HAYAKAWA, Tomoyo SATO, Ryousei TAKANO, Motonobu SATO, Nobumasa TAKAHASHI: Flexible, Modular System Architecture for Supporting Creative Work, In Adjunct Conference Proceeedings of HCI International'99, pp.163-164, 1999.
3) 中島一彰, 早川栄一, 並木美太郎, 高橋延匡: 分散環境における発想支援のためのリアルタイム手書き協調作業システムの設計と実現, 情報処理学会論文誌, Vol.38, No.12, 1997.
4) 佐藤友代, 並木美太郎, 早川栄一: ソフトウェア開発過程の記録を支援する電子研究ノートの設計と実現, 情報処理学会研究会報告, 99-HI-83, pp.49-54, 1999.
5) 山本康弘, 早川栄一, 並木美太郎, 高橋延匡: OS/omicron V4 におけるインタプリタ型言語とネイティブコードのリンクの実現, 情報処理学会第 54 回全国大会予稿集, pp.51-52, 1997.
6) 新城靖, 孫軍: 並列世界モデルに基づくオペレーティング・システム, 情報処理学会第 8 回コンピュータシステムシンポジウム論文集, pp.95-100, 1997.
7) B.Liskov, A.Adya, M.Castro, M.Day, S.Ghemawat, R.Gruber, U.Maheshwari, A.C.Myers, and L.Shrira: Safe and Efficient Sharing of Persistent Objects in Thor, In Proceedings of ACM SIGMOD'96, pp.318-329, 1996.