

分散共有永続ストレージシステム WAKASHI

2 G-6

白 光一、寺本 圭一、天野 浩文、牧之内 顕文

九州大学工学部

1 まえがき

WAKASHIは「出世魚」[1]の一番低いレベルのサブシステムで、既存のオペレーティングシステム上でCプログラマーが分散共有された永続データと揮発データを共に操作するための基本機能を提供する。WAKASHIでは、二次記憶上のファイルをユーザの仮想メモリに写像する仮想メモリ方式を採用することによって、Cプログラマーが永続データを通常の揮発データと機能的にも効率的にも同様に扱えるようにした。さらに、WAKASHIの分散共有メモリ方式により、永続データ及び揮発データを分散共有することが可能であり、ユーザに効率的で使いやすい分散永続プログラミング環境を提供する。

WAKASHIの目標は、分散環境上で効率良くマルチメディアデータを格納・検索・処理を行なうためのストレージシステムを提供することである。従って、WAKASHIの性能評価はマルチメディアデータベースを使ったものでなくてはならない。そのために、我々は既存のオブジェクト指向データベースベンチマーク OO1[3]を拡張し、マルチメディアデータベース操作性能の測定ができるようにした。

本稿では、WAKASHIの実現法と拡張したOO1ベンチマークによるWAKASHIの分散環境上での性能評価について述べる。

2 WAKASHIの実現法

WAKASHIはクライアント/サーバアーキテクチャで実現されている。図1に示すように、複数のクライアントとサーバは高速ローカルエリアネットワーク(LAN)を通してつながっており、WAKASHIサーバは全てのクライアントにアクセスされるデータの永続性と一貫性を保持する。このように、WAKASHIサーバは仮想メモリ方式によりファイルと永続ヒープとの間の写像、各クライアント間の永続ヒープ間の写像を実現する。これにより、クライアントは分散共有永続ヒープと分散共有揮発ヒープとを自分の仮想メモリに一元的に割り付け、両者内のデータを通常のプログラム内の揮発データと同様に扱うことが可能となる。

仮想メモリ方式はデータベースファイルをユーザの仮想メモリに写像し、そのファイルを仮想メモリのページング領域として利用する。従って、この方式は従来のバッファ方式で問題となったデータ形式変換のオーバーヘッドやバッファサイズより大きなマルチメディアデータの処理の問題、及び二重バッファ、ポインタ変換等問題点を解消できる。

分散共有(仮想)メモリはローカルメモリを持つプロセッサ群をネットワークで相互接続したシステムの上で、各プロセッサ間が共有のメモリをあたかも持つようにしようとするものである。分散共有メモリの基本的な実現方式は、各プロセッサのローカルメモリを共有の仮想メモリに写像するものである。そのためには二種類のページングが必要となる。一つはプロセッサのローカル実メモリと二次

記憶上のページング領域間のページングである。これによりそのプロセッサのための仮想メモリを実現する。もう一つは各プロセッサの実メモリ間のページングで、これによりメモリの分散共有を実現する。即ち、データベースファイルを分散共有メモリの二次記憶上のページング領域として利用することによって分散共有永続ヒープを実現する。

WAKASHIはMach OS上で実現した。Machでは、ユーザレベルで仮想メモリ管理を行なうことができるように、EMMI(External Memory Management Interface)を提供している。ユーザレベルの仮想メモリ管理プログラム(即ち、外部ページャ)は一つのサーバであり、それがクライアントプログラムが写像した仮想メモリをメモリオブジェクトとして管理する。メモリオブジェクトとは、実メモリのページと二次記憶上のページとを対応づけ、両者へのアクセス機構を概念化し、仮想メモリを一般化したメカニズムである。WAKASHIサーバは外部ページャとして実現された。WAKASHIサーバは、分散共有永続ヒープ或いは分散共有揮発ヒープをメモリオブジェクトとしてとらえ、二次記憶上のファイルと実メモリとの間、及び各クライアントの実メモリ間のページングを行い、データの永続性と一貫性を保持する。

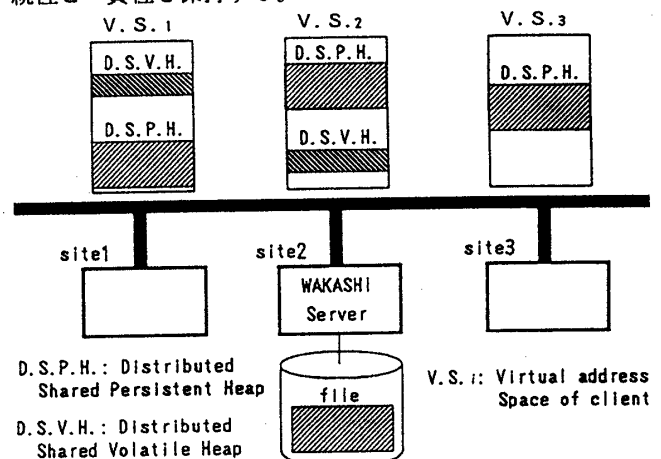


図1: WAKASHIの分散共有永続(揮発)ヒープ

3 性能評価

3.1 マルチメディアデータベース・ベンチマーク

マルチメディアデータの操作はアプリケーションによって異なり、その操作も非常に複雑である。そのため、マルチメディアデータベースの性能測定は非常に困難であると見られている。しかし、一般的に、マルチメディアデータは複合オブジェクトと長いオブジェクトとして表現できる。また、マルチメディアデータ操作をデータのアクセスレベルから見ると、どんなアプリケーションもほぼ同じだと考えられる。従って、マルチメディアデータ操作は、データアクセスレベルでは、複合オブジェクトのアクセスと長いオブジェクトのアクセスとの二種類のアクセスを含んでいるものと考えて良い。データアクセスレベルでは、複合オブジェクト操作の性能測定用ベンチマークとしては、OO1ベンチマークがある。しかし、このベンチマークは短いオブジェクト型(即ち、部品 Part)しか考えて

ないため、長いオブジェクト処理が特徴とされているマルチメディアデータベースシステムの性能評価には適しない。そのため、我々は一つの長いオブジェクト型を OO1 ベンチマークに追加することによって、マルチメディアデータベースのベンチマークをした。従って、次の様に二つのオブジェクト型でデータベースを定義する。

```
Part: {
  id: INT,
  type: STRING[10],
  x, y: INT,
  build: DATE,
  to: LIST OF {p: Part, type: STRING[10],
               length: INT}

  from: LIST OF Part,
  refer: Image}
Image: {
  id: INT,
  data: Bitmap}
```

このデータベースの部品 (Part) は OO1 データベースの部品と同様に、ユニーク *id* を持ち、全部で 2 万個ある。それぞれの部品は他のランダムに選ばれた三つの部品を参照している。しかし、OO1 データベースの部品と違うことは、我々のデータベースの部品には一つの *refer* 属性が追加されており、全体の 1% の部品がランダムに選ばれた一つイメージを参照することにする。

イメージデータはユニーク *id* を持つ。そのイメージデータは *bitmap* 型で定義する。一般的に、イメージはアクセスレベルでは配列として表現される。問題を簡単にするため、ここでは白黒ビットマップイメージだけ考えて、[512, 512] の二次元配列で *Bitmap* 型を表現する。

このデータベースの操作は OO1 ベンチマークと同じように、*Lookup*、*Traversal* 及び *Insert* との三つを行なう。しかし、*Lookup* 操作に長いオブジェクト操作を追加する。即ち、もし検索された部品がイメージを参照していたら、さらにこのイメージを操作する。イメージ操作は複雑で、またアルゴリズムによって異なるが、データアクセスレベルではイメージを表現する配列データのスキャンがイメージ操作の基本的なパターンだと見て良い。そのため、このベンチマークでのイメージ操作は *Bitmap* データを順に一回スキャンする操作として定義する。

3.2 評価結果

測定プログラムは INADA[2] で開発された。測定環境として、二台の OMRON 製の LUNA-88K ワークステーション (四つの 88100CPU (25MHz)、32 メガ主記憶) を使い、それぞれサーバマシンとクライアントマシンにする。データベースはサーバマシン上に構築し、WAKASHI サーバもこのマシンで動く。クライアント (測定プログラム) をサーバマシン、クライアントマシンで実行させてローカル及び遠隔データベース操作性能を測定する。

図 2 はローカル *Lookup* 測定と遠隔 *Lookup* 測定の結果をそれぞれ示している。この測定は 1000 個の部品と 10 個のイメージを処理する。Cold の場合、すべてのデータがファイル中に存在しており、全メモリ中には存在していないので、永続ヒープ上でアクセスされる全てのページがページフォールトを起こし、WAKASHI サーバはページインのためのファイル I/O 操作を実行する必要がある。遠隔の *Lookup* 測定には、ファイルの I/O 操作だけでなく、サーバマシンからクライアントマシンへの (RPC による) データ転送も行うため、かなりの時間がかかる。しかし、イメージは単に永続ヒープ領域内の一部として操作

されるので、長いオブジェクト処理固有のオーバーヘッドは見られない。特に、Warm の場合、すべてのデータが実メモリに常駐しており、I/O やデータ転送が行われないため、WAKASHI サーバが載ってないクライアントマシン上でのプログラム実行は最もよい。

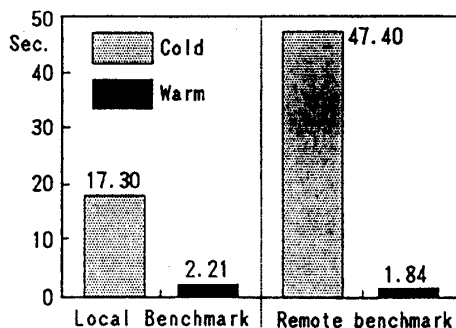


図 2: *Lookup* 操作の測定結果

Traversal 測定には 3280 個の部品を検索する操作が含まれる。従って、この測定には 3280 回のポインタ変換 (*pointerswizzling*) が発生する。ポインタ変換のオーバーヘッドを測定するため、ポインタ変換をしない *Traversal* 操作もシミュレーションし、変換する場合と比較した。WAKASHI は仮想メモリ方式によって、ファイルのデータ形式はメモリ上の形式が同一になるため、図 3 に示しているように、永続ポインタ変換のためのオーバーヘッドがほとんどない。

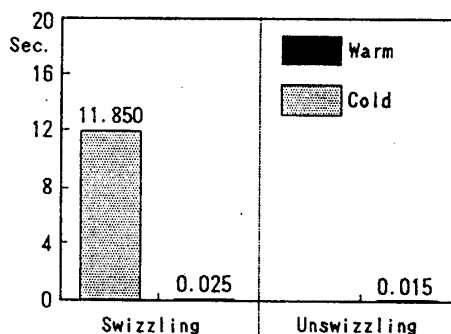


図 3: *Traversal* 操作の測定結果

以上のベンチマーク測定は非常に簡単だが、WAKASHI サーバによる長い永続データへのアクセス、ポインタ変換等は効率的であると確認された。

4 むすび

WAKASHI は仮想メモリ方式と分散共有メモリ方式との結合により、Mach OS 上で実現した。WAKASHI によるマルチメディアデータベース操作性能を測定するための OO1 ベンチマークを拡張し、このベンチマークを使って評価した。その結果、WAKASHI サーバに基づく長い永続データのアクセス、ポインタ変換等は効率的に実現できることを確認した。

参考文献

- [1] 牧之内顕文, "「出世魚」プロジェクト", 情報処理学会第 44 回全国大会, 2H-1.
- [2] 天野浩文 他, "永続オブジェクト指向プログラミング言語 INADA の分散共有永続データ操作機能", 情報処理学会第 46 回全国大会, 2G-7.
- [3] R. G. G. Cattell et al. "Object Operations Benchmark", ACM Trans. on database systems, Vol. 17, No. 1, March 1992, pp.1-31.