2U－8

永続プログラミングシステムP3Lの OO1ベンチマークを用いた性能評価

鈴木 慎司, 喜連川 優, 高木 幹雄
東京大学生産技術研究所

1 はじめに

P3Lは発見時のポインタ書き換えおよびフォールト時におけるOIDの代理OIDへの書換えを採用した永続Cプログラミング環境である[4]。本論文では、P3L処理系の実行性能をOO1 Benchmarkに基づいて報告、考察する。

2 P3Lのポインタ書換え方式

P3Lでは、ポインタの発見時書き換えを採用している。オブジェクトの読み込みには、MMUによるアクセストラップを利用することも考慮しているが、現在の実装ではポインタ書換え時にターゲットオブジェクトを読み込んでいる。その結果、存在検査(アクセス対象のオブジェクトが読み込み済みであるか否かの検査)は省略されている。

ポインタの書き換えは、ObjectStoreやTexas Persistent Store[1]で採用されているページフォールト時にページ内の全ポインタを書き換える方式と異なり、各ポインタがメモリ中から読み出されるまで遅延される。ポインタの書換えのために必要となるアドレス割り付けを遅延させることで、前記のシステムに比べ、仮想記憶の使用量を小さく抑えることができる。P3Lの処理系では、ポインタの読み出しは、コンパイラによって静的に検出される。そのような読み出しに対し、コンパイラは通常のメモリフェッチ命令に加え、ポインタが書換え済みであるか否かを検査し、未書換えならば書換えを行なう命令列を追加する。現在はGnu C/G++コンパイラのコード生成部に変更を加えることで実装を行なっている。

上で述べたように、2次記憶上のオブジェクト識別子(OID)から仮想メモリアドレスへの書き換えは、ポインタの読み出し時まで遅延されるが、オブジェクトの読み込み時に、その内部のOIDは仮想メモリと同一サイズの縮退したOIDに置き換えられる。

コンパイラでポインタ書き換えのコードを生成すること、主記憶上に置かれたOIDのサイズを仮想メモリポインタのサイズに合わせることで、言語に新規の参照の機構を導入することなく、高い効率を維持することが可能となった。

Evaluation of a Persistent Programming System P3L through OO1 Benchmark
S.Suzuki, M.Kitsuregawa, M.Takagi
Institute of Industrial Science, University of Tokyo

3 OO1 Benchmark

OO1 BenchmarkはSun Micro SystemsのR. G. G. Cattelらによって提唱されたエンジニアリング応用向けのベンチマークである[5]。データベースは、数十バイト程度(厳密な構造は規定されておらず、P3Lでは44byte)の部品が互いに参照しあうネットワークからなり、各パーツからは別の部品へ3つのリンクが出ている。このリンクは9割の確立で、1パーセント以内の近さの部品に接続される。「近さ」は各部品に与えられたシリアル番号によって定義される。今回使用したデータベースは2万個の部品からなる。OO1 BenchmarkにはLookup, Traversal, Insertという3つの計測対象の操作が存在するが、今回は巡行の性能を最もよく反映するTraversalテストを行なった。このテストでは、2万個の部品からランダムに一個を抽出し、そのリンクを7段に渡って巡行し、到着可能な3280個の部品に対して空の関数呼びだしを行なうものである。

4 ベンチマーク対象

ポインタに基づいた巡行を繰り返し行なう応用を、主記憶が十分にある環境で動作させた場合、著者らの知り得る限りTexasやObject Storeで用いられている書き換え方式がほぼ性能の上限を規定する。というのは、ページ書換え方式では書き換えの検査、読み込みの検査のために命令を追加する必要がないからである。各種のベンチマークがこの推定を裏付けている。P3Lの実装はTexas等と同様のアプリケーションシナリオを想定しており、Texasとの比較は必然である。一方, Exodus/E[2], 使用した商用のODBMSは巡行主体の応用にはそれほど力を入れておらず、I/Oが主体となるようなプログラムの実行に対して最適化されている。巡行主体のアプリケーションにおける、P3L, Texasの性能的な優位度を確認するためにこれらのシステムを含めた。

5 結果

図1に計測結果を示す。横軸はTraversalを繰り返した回数を、立て軸は各回のTraversalが費やした時間を示している。最初の部品は毎回ランダムに選択している。

ここで注目すべきことは、同一のストレージサーバを用いているP3LとEを比較した場合、繰り返しの初期段階、つまりI/Oが頻繁におこっている段階ではEのほうが高い性能を示していることである。これは、P3Lが読み込み時にポインタフィールドの代理OIDの書換えを行

なっている事とともに、Eがページ単位でのデータ取得を行なうのに対し、P3Lがオブジェクト単位での取得を行なっているせいで応用プログラムとサーバ間の通信回数が増加するためであると考えられる。一方、Texasが特に高い性能を示しているのは、サーバが応用プログラム内に埋め込まれ、データ読み込みのためのコストが抑えられていることが原因であると思われる。

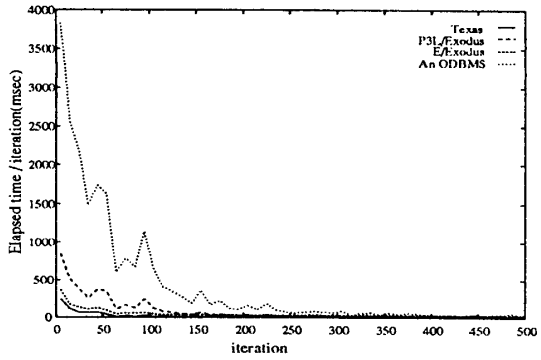

図 1: cold-warm: w/ locality

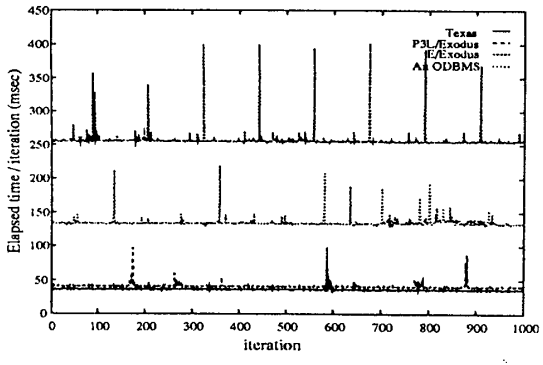

図 2: hot: w/ locality

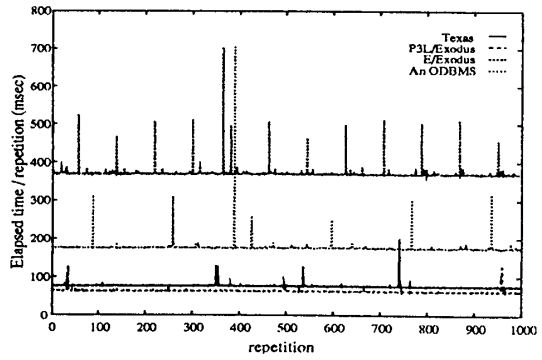

図 3: hot: wo/ locality

上の観測は興味深いものではあるが、今回注目している性能ではない。我々が興味をもっているのは、主記憶中でのデータのキャッシュが十分に進んだ段階での巡行速度である。そこで、上記のTraversalを行なった直後に同一のパターンの Traversalを行なう時間を計測した。（ただし測定精度を増すために各回のTraversalを10回行なった。）その時の結果が図2に示すものである。I/Oは生じないので図1に見られるような起動時の立ち下がりは観測されない。図のようにTexas, P3L, ODBMS, Eの順に高い性能を示しており、TexasとP3Lは他2者に比べ大幅に優っている。

冒頭でP3Lのポインタ書換え方式は、フォールト時書換えに比べ仮想メモリの消費量が少ないと述べた。その影響について考察するために、OO1 Benchmarkの仕様から逸脱するが、部品の接続の局所性を取り払って構築したデータベースを用いて、同一の計測を行なった。ただし、この場合はまったく同じ Traversalを繰り返すのではなく、2度目は1度目の1回目と同一のパターンで繰り返す。これは仮想メモリ中の配置の分断化が最も顕著に現れるのが、1回目の読み込みの時であるからである。この結果を図3に示す。P3Lは、Texasに比べより多くの命令を実行しているにもかかわらず、Texasよりも高い巡行性能を示している。その原因は確認できていないが、TLBミス率である可能性が高い。

6 最後に

P3LのOO1 Benchmarkの hot Traversalに関しての優位性が確認できた。今後は実際のプログラムを用いて性能評価を行なっていく予定である。

参考文献

[1] Paul R.Wilson,'Pointer Swizzling at Page Fault Time: Efficiently Supporting Huge Address Space on Standard Hardware', ACM Computer Architecture News, Vol 19, No.4 June 1991

[2] Seth J. White and David J.Dewitt, 'A Performance Study of Alternative Object Faulting and Pointer Swizzling Strategies', Proc. of Conference on Very Large Data Bases 1992

[3] 鈴木 喜連川 高木,' サロゲートOIDを用いたポインタ書き換え方式' 情報処理学会 第49回, 1994

[4] S. Suzuki, et al. "An Efficient Pointer Swizzling Method for Navigation Intensive Applications", to appear in proc. of Sixth Intl. Workshop on Persistent Object Systems, Trascon France 1994

[5] R.G.G. Cattel and J. Skeen, 'Engineering Database Benchmark', Database Engineering Group, Sun Micro Systems Technical Report 1990