B-treeの分散配置とその性能評価

2G－7

小幡 康　高須 淳宏　安達 淳
東京大学大学院工学系研究科　学術情報センター研究開発部

1 はじめに

現在、B-treeはデータベースのアクセス法として最も広く用いられている。そして、更新処理環境に対応できるように様々な並行処理制御アルゴリズムが考案、検討されてきた[2]。本研究ではB-treeを実装するにあたって、マルチプロセッサ、ディスクといった伝統的なハードウェア構成、木の配置の工夫による性能向上をシミュレーション、実システムでの実験を通じて確かめる。なお、本稿では更新処理として挿入処理についてのみ考察する。

2 B-treeの構成法

まず並列プロセッサ環境でのシステム構成について述べる。ここでは各ディスクに付属したプロセッサが他に対して独立して木のオペレーションを行うものである。本稿では、B-treeの構成法として多重化法と分散配置法について考察する。

多重化法とは複数のディスク上に同一のB-treeを格納するものである。書き込み操作に対しては、処理の分散化を図るためにまず１つのディスクにアクセスしてB-treeを下降させていき、更新すべきエントリ・ブロックが判明した時点でその内容を他のディスクにブロードキャストさせるという方法をとる。この場合、全てのディスクを同時に書き換える訳ではないのでデータの一貫性を保証するために並行処理制御が必要となる[1]。

もう一つのB-treeの配置法として、木を縦方向に分割した方式(分散配置法)が挙げられる。この方法では更新処理において結果のブロードキャストの必要が無い点で多重化法より有利である。

以上は並列プロセッサ環境での話であったが、二次記憶の処理に大きな比重の置かれるシステムにおいて、プロセッサをディスク数だけ設けるのは効率的であるとは言えない。そこで本研究では単一のプロセッサにバスを通じて複数のディスクをつなげたハードウェア構成の場合も考慮する。各ディスクへのデータ配置は並列プロセッサ環境と同じく多重化法と分散配置法を考える。

Distributed B-tree and Its Performance Evaluation
Yasushi Obata[1], Atsuhiro Takasu[2], Jun Adachi[2]
[1]University of Tokyo, Graduate School of Engineering
[2]National Center for Science Information Systems

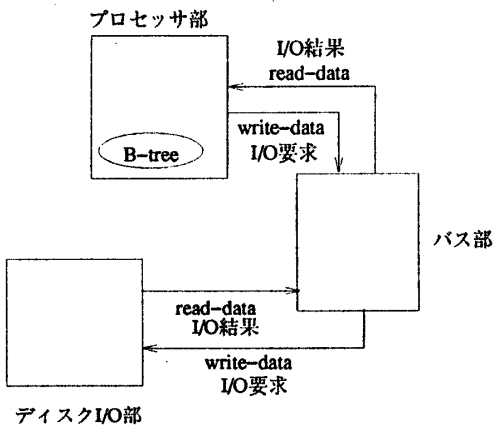

図 1: 各プロセッサ間の通信

3 実験方法

本章ではB-tree構成法を比較するためのシミュレータについて説明する。このシミュレータはnCUBE社製の並列コンピュータ、nCUBE2(16台のプロセッサによる疎結合ハイパキューブ構造)上に実装されている。

3.1 並列プロセッサ環境

本研究で想定したシステム構成と実験環境との違いは、nCUBE2では各プロセッサにディスクが装備されていないという点である。そこで本実験ではファイル全体に該当するものとして各プロセッサに起動時にメモリを割り当て読み書き毎に計算によるアクセス遅延を設定することによって、ディスク入出力をエミュレートした。

3.2 単一プロセッサ環境

単一プロセッサ環境についてはプロセッサ、SCSIバス、ディスクといった各部品にnCUBE2のプロセッサを割り当てるリアルタイム・シミュレーターを作成した。各部品間の通信形態を図1に示す。

4 実験内容とその結果

100000個のキーが入ったB-treeに500回のオペレーションを行わせ、ディスクの数を変えてスループットの変化を調べた。オペレーションについては挿入処理、もしくは検索処理でその比率は検索処理のみ、更新処理を10%含む場合の２通りについて実験した。以下にその結果を示す。

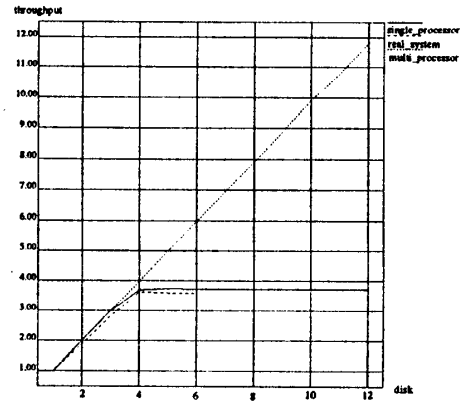

図 2: read100%の時のスループットの比較

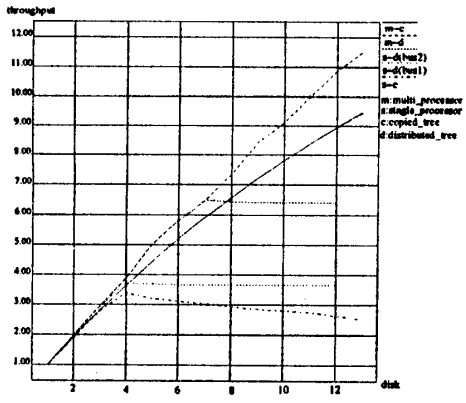

図 3: 更新処理を 10%含む時のスループットの比較

4.1 読みだしのみの場合

ここでは配置法は多重化法のみで、マルチプロセッサと単一プロセッサの比較を行った (図 2)。単一プロセッサの場合はバスがボトルネックとなり途中で飽和してしまっている。この様子を SUN-SS20 に SCSI バスを通じてディスク (Seagate 社製、4GB、アクセス平均時間 8ms) をつなげた実際のシステムと比較した。すると、バスの 1 回当たりの入出力に 2.5ms のオーバヘッドがあることが分かった。以下の実験ではこの実システムに即したパラメータを選択した。

4.2 更新処理を含む場合

ここでは配置法は分散法と多重化法の両方について比較実験を行った。(図 3)。これより次のことが分かる。

- 同一のハードウェア構成ではディスクアクセスの回数から分散法の方が高いスループットを示す。
- 単一プロセッサ環境での多重化法は飽和後、入出力回数の増加によりスループットが落ちていく現象が見られる。
- 単一プロセッサ環境ではバスがボトルネックとなる

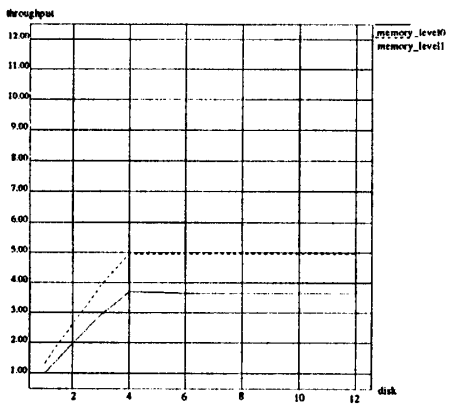

図 4: ルートをメモリに常駐させた場合との比較

が、ディスクの台数=ディスクのアクセス時間／バス上でのオーバヘッド となった時点で頭打ちとなる。性能が勝る分散配置法でバスの数と飽和の関係も調べた。その結果、図 3示される様に以下のことが分かった。

- 単一プロセッサ環境でバスの本数を 2 倍にすると飽和する時のディスクの台数もほぼ倍になる。

4.3 キャッシュを用いたシステムとの比較

最後に分散配置された木全体をディスクに格納したものとルートをメモリに常駐させたものについて比較を行った (図 4)。キャッシュを用いた場合、入出力の回数がその分減るため用いない場合に比べてほぼ一定の割合でスループットが増加している。

5 まとめ

以上の比較実験により、更新処理環境ではコストとの兼ね合いを考えると木を分散配置し、シングルプロセッサに SCSI バス 1 本につき 3 台までを上限とし、ディスクの数を増やしていくことが最も優れた方法であることが分かった。また、キャッシュの有無については図 4においてキャッシュを用いた方をそのコスト分だけ右にずらして比較すればよい。

本研究では伝統的なハードウェア構成で、B-tree を高い並列度をもって処理するための一つの設計指針となるものを示した。今後は分散配置における信頼性の向上が課題となるだろう。

参考文献

[1] 片山, 高須, 安達, 「多重化された B+Tree インデックスに関する考察」, 1993 年 電子情報通信学会秋季大会 D-99.

[2] T. Jhonson, D. Shasha, "The Performance of Current B-Tree Algorithms," ACM Transaction on Database Systems, Vol.18, No.1, March 1993, pp51-101.