ハイパフォーマンス
コンピューティング 65－10
（１９９７．３．６）

WS クラスタにおけるデータベース並列検索の評価分析

相場 雄一　青木 久幸

Email: {aiba, aoki}@csl.cl.nec.co.jp

NEC　C&C 研究所

概要

データベースが巨大化し続けた一方で、データベースシステムを単なるデータの蓄積のためだけでなく、蓄積されたデータから様々な情報を引き出す活用法が注目されている。そこでは、高速に検索する性能が要求されるが、データ量が増えると検索の処理量は急激に増えるため、検索の並列処理が必須の技術となる。そこで、複数のワークステーションを用いた並列検索を実現し、評価実験を行なった。その結果、並列検索ではデータの格納方法によっては著しく性能が劣化することが分った。分析を行なったところ、ディスク I/O の競合以外に、プロセス間同期による CPU アイドルの多発が原因であることが分った。本稿では、並列検索の実験とその評価分析について報告する。

Performance Evaluation and Analysis of Parallel Database Scan in a Workstation Cluster

Yuichi Aiba　Hisayuki Aoki

C&C Laboratories, NEC

Abstract

Information retrieval and data analysis of large databases are of concern. As high speed retrieval from large databases requires heavy processing, parallel processing is important technology. We implemented parallel scan on a workstation cluster and experimented with some simple queries by varying some database storing patterns. Initially, the performance turned out quite poor in some cases of patterns. By analysing results, we found that the cause of poor performance is mainly due to frequent CPU idling for inter-process synchronization. This paper describes over evaluation of the parallel scan.

1 はじめに

今日のデータベース(DB)の主流となるRDB(リレーショナルDB)は、1970年代に提唱されて以来、基幹業務等で普及し、蓄積データ量も増え続けた。基幹業務では巨大なDBに対し多数のトランザクションを高速に処理する性能が要求されるが、近年、蓄積されたDBに対する検索によって数値情報を引き出し、意志決定に利用する要求が高まっている。このような利用では、非定型な問い合わせを処理するため、対象となるテーブルを多様に検索する性能が要求される。RDBの検索時に行なわれる関係演算は負荷が重く、高速化に関して数多くの研究がなされてきた。関係演算の並列処理は検索の高速化技術として重要視され、大規模化に欠かせない技術となっている。

以上のような背景から、我々は、RDBの並列検索に着目し、複数のワークステーション(WS)をネットワークで接続したWSクラスタ上で実験的に並列検索を行ない、並列性の評価を行なった。本稿の2では、実験的な並列検索システムの構成概要について説明する。3では、逐次／並列検索による並列性の評価結果とその分析を示す。更に、4では、並列検索における検索パターンを変化させた際の評価結果、及びシミュレーションを含む分析について説明する。

2 WSクラスタによる並列検索実験

図1に示す構成で並列検索の実験を行なう。

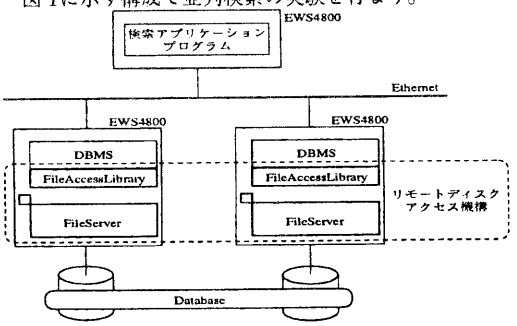

図 1: 並列検索実験システムの構成概要

WSクラスタ

ハードウェアとしては、複数のWSを通信網で接続した構成である。各WSには基本的には独立したOS(UNIX)が載っており、全体を統合管理するためのミドルウェア等が実装されていることもある。

本実験では、WSにNEC製EWS4800を3台使用し、Ethernetで接続した。内2台のWSをサーバ用としDB管理システムDBMSを動作させ、DB格納用のディスクを1台ずつ接続する。もう1台のWSはクライアント専用とし、ここから検索アプリケーションプログラム(AP)を投入する。

DB管理システム(DBMS)

DBMSはDBの構築、構成管理、検索などを行なう。図のように複数のWSのディスクに跨ってDBを構築し、これを1つのDBとして管理する。更に、このDBに対し、複数のWSから並列に検索を行なう。各WS上で動作するDBMSは、DBの構成情報などを共有するため、共有ディスク機構を必要とする。

リモートディスクアクセス機構

各WS上で動作するDBMSは共有のディスク上にDBを構築するが、今回使用した構成ではハードウェアあるいはOSによるディスク共有のサポートがない。そこで、ディスク共有の機構を実現したリモートディスクアクセス機構なるソフトウェアを開発した[3]。この機構は、以下のモジュールから構成される。

- FileAccessLibrary(FAL) … ディスクアクセスを要求するAPにリンクされるライブラリで、FileServerにディスクアクセスの処理を要求する。
- FileServer … ディスクの接続されているWS上で動作するユーザプロセスで、FALからのディスクアクセス要求を処理する。

検索アプリケーションプログラム(AP)

クライアント専用のWS上で動作し、接続網を介して各WS上で動作するDBMSに、並行処理可能な検索要求を投じる。

3 スケーラビリティ評価

図2に示すように、1台のWSだけで検索を処理する逐次検索の場合と、2台のWSで検索を処理する並列検索の場合を比較する。

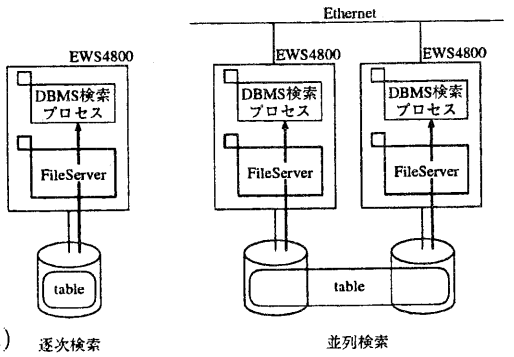

図 2: 逐次検索と並列検索の比較

3.1 評価内容と結果

以下に示すテーブル(平均レコード長50バイト)を作成し、10万件のレコードを挿入した。逐次検索では、2台のディスクの内の一方を使用し、並列検索では、両方のディスクを使用してテーブルを作成した。

属性名	Data Type
key	unique integer(0 ~ 99,999)
num	10進16桁のランダム整数
dt	ランダムな日付
pct	ランダム整数(10,000 ~ 10,099)
name	20文字のランダムな固定長文字列

検索文については以下のような3つの検索文を用意し、全件検索を行なった。3つの検索文は、検索のためのCPU処理の負荷量が異なり、負荷の軽い順にselect1 < select2 < select3となる。検索対象となるテーブルをディスクから読むためのI/Oの負荷は3つの検索文で等しく、抽出される件数も等しい。

- select1: (属性numの値がAより大きいものを抽出)

```
SELECT num    FROM test
  WHERE num < A ;
```

- select2: (属性numの値がAより大きいものを抽出し10乗)

```
SELECT POWER(num, 10)    FROM test
  WHERE num < A ;
```

- select3: (属性numの値の10乗が、A^{10}(Aの10乗を計算して指定)より大きいものを抽出)

```
SELECT num    FROM test
  WHERE POWER(num, 10) < A^10 ;
```

逐次検索および並列検索において検索に掛かった処理時間を測定した。結果を表1に示す。測定値は4回の試行の平均値を示す。

	select1	select2	select3
逐次検索	13.77(秒)	17.91	27.03
並列検索	7.43(秒)	12.01	14.01
並列度	1.85	1.49	1.93

表 1: 検索処理時間の比較

3.2 考察

表1中の並列度は、並列検索が逐次検索と比較して何分の1の処理時間となっているかを示す。2台のWSを使用しているので、並列度は2になることが望ましいが、実際には2に満たない。この原因として以下の2つが挙げられる。

- 検索処理に含まれる逐次部分の影響
  検索処理における逐次部分は、各DBMSへの処理分割、結果収集である。select2の並列度が他に比べて低い値を示すことから、select2の逐次処理の割合が大きいことが予想される。
- 各処理系の動作時間のバラツキの影響
  2分割した各処理が各DBMSにおいて常に一定の時間で終了すれば、並列度は2となるが、実際には処理時間には変動があり、更に、全体の処理時間は、遅い方の処理時間によって決定される。これによって、2並列による処理時間は、逐次の処理時間の半分よりも大きくなり、並列度は2に満たない。

ここで、バラツキのある2つの処理系によって並列処理を行なう際の並列度について考える。各処理系の処理時間$t(0 \leq t)$が同じ確率密度関数$P(t)$に従うとし、両処理系が終了するまでの処理時間T_pの期待値を求める。各処理系の処理時間をt_1,t_2とすると、$T_p = max(t_1, t_2)$より、T_pの期待値E_pは以下のようになる。

$$E_p = \int_{t_1=0}^{\infty}\int_{t_2=0}^{\infty} T_p \cdot P(t_1)P(t_2)dt_1dt_2$$
$$= 2\int_{t_1=0}^{\infty} t_1P(t_1)\left\{\int_{t_2=0}^{t_1} P(t_2)dt_2\right\}dt_1$$

また、2つの処理系を逐次に実行した場合の処理時間T_sの期待値E_sは以下のようになる。

$$E_s = \int_{t_1=0}^{\infty}\int_{t_2=0}^{\infty} (t_1+t_2)\cdot P(t_1)P(t_2)dt_1dt_2$$
$$= 2\int_{t_1=0}^{\infty} t_1P(t_1)dt_1$$

ここで、簡単のため一様分布関数$P(t) = \frac{1}{2W}, (A - W \leq t \leq A + W)$について計算すると、$E_p = A + \frac{W}{3}, E_s = 2A$となる。従って、並列度は$E_s/E_p = \frac{2A}{A+\frac{W}{3}}$となる。これによると、平均値の3分の1のバラツキがあると、並列度が10%低下することになる。どんなに処理を均等に分割しても、バラツキの大きな処理系では、並列度の劣化は著しい。

ちなみに、並列検索で得られた測定値において、Wの目安として標準偏差を計算すると以下のようになる。

	select1	select2	select3
平均値(秒)	7.43(秒)	12.01	14.01
標準偏差(秒)	0.30	0.30	0.39

標準偏差は平均値の2.5 ~ 4.0%と小さい値を示し、この値から、各処理系のバラツキによる並列度劣化は1%程度と計算される。検索処理にはディスクI/Oが含まれるため、大きなバラツキが予想されたが、実測値のバラツキは小さい。ディスクへのアクセスがシーケンシャルで、ディスク内部のキャッシュにヒットしたため、ディスク内部の機械動作が行なわれていないことが原因と考えられる。また、バラツキの影響の小さいことから、select2の並列度の低い結果は、逐次処理の割合の大きいことが原因と考えられる。

4 並列検索評価実験

この実験では、2つのDBMSからのディスクへのアクセス競合の影響を調べた。

4.1 評価内容

スケーラビリティ評価実験で用いたテーブルと検索文を用い、ディスク上への格納パターンを変えることによ

り、図3に示すような3種類のアクセスパターンを実現し、実験を行なった。

- pattern1 … 各WSから、ローカルのディスク上のデータを検索する。スケーラビリティ評価における並列検索と同一である。この検索パターンでは、ディスクへのアクセス競合が発生しない。
- pattern2 … 各WSからの検索は、ローカルのディスク上のテーブル領域から開始し、引き続いてリモートのディスク上のテーブル領域に検索を行なう。両WSからの検索処理が全く同様に進行すれば、ディスク上のアクセス競合は発生しないが、実際には多少の競合があるものと考えられる。
- pattern3 … 両WSからの検索は、同一ディスク上のテーブル領域から開始し、他方のディスク上のテーブル領域に移行する。この場合、両WSからの検索よるディスク上のアクセス競合がかなり発生するものと考えられる。

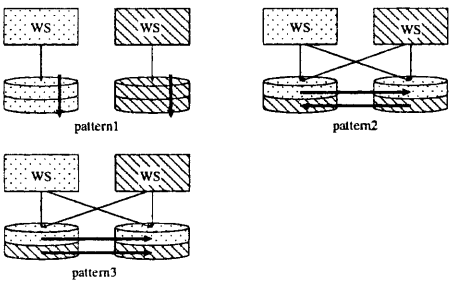

図 3: 3種類の検索パターン

実験環境にTinyTOPAZ[2]と呼ぶトレースデータ採取ツールを接続し、各プロセスの消費したCPU時間、I/Oの発行回数、各ディスクの稼働時間などのデータを詳細に採取する。TinyTOPAZはトレース・ベースの性能測定ツールであり、専用のケーブルを介してWSに接続することにより、僅かなオーバヘッドで詳細な情報を採取することができる。

4.2 測定結果

結果を図4のグラフに示す。図中のElapseは、検索を投入してから処理終了までに掛かる経過時間を4回計測し、平均値を示す。更に、DBMSの検索プロセスの消費するCPU時間(DBMS)、FileServerの消費するCPU時間(FileServer)、ディスクの稼働時間(Disk)を両WSにおいて採取し、その平均値を示す。

図4のグラフから以下のような特徴が見られる。

経過時間の違い

pattern1とpattern2を比較すると、pattern2の経過時間の増加だけが顕著である。DBMSのCPU時間などの増加は、経過時間の増加と比較すると小さく、他

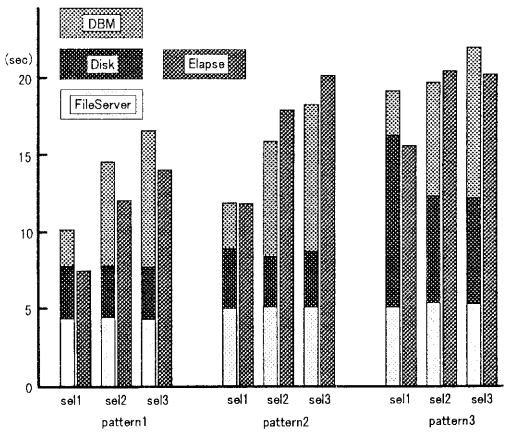

図 4: 並列検索の測定結果

に経過時間を増加させる要因があると考えられる。pattern3では、更に経過時間の増加が著しい。

ディスク稼働時間の違い

ディスク稼働時間が、pattern3ではかなり長くなっている。これは、2つのDBMSから要求されたアクセスが同一ディスクへ集中し、アクセス競合が頻発しているためと考えられる。

pattern3において、ディスク稼働時間が、select1でかなり長くなっている。これは、select1の検索プロセスにおける検索処理が軽いため、ディスクアクセスが短い間隔で発生し、ディスク上でのアクセス競合が更に頻発しているためと考えられる。

4.3 ディスクにおける衝突の頻度

pattern3においては、ディスク稼働時間が長くなっているだけでなく、測定毎の値に大きなバラツキがあった。この原因は、2つのWSからのアクセスが同一ディスクに集中することに関連している。

実験で使用したディスクは、ディスクキャッシュを持ち、シーケンシャルなリードアクセスに対して応答時間を短縮する効果がある。pattern1のようなアクセスでは、各WSから各ディスクに対してシーケンシャルなアクセスを行なうため、ディスクキャッシュが有効に作用する結果となっている。

これに対しpattern3では、各WSからのアクセスが同一ディスクに集中するため、アクセスがシーケンシャルにならず、ディスクキャッシュが有効に作用しないことが多い。これによって、アクセスの応答時間が確率的に伸び、全ディスク稼働時間を増加させている。

シーケンシャルアクセスの場合の1回のI/Oの応答時間は平均2.5msecとなるが、ディスクキャッシュが有

効に作用しない場合、平均20msecとなる。更に、後者の場合、ヘッドシーク等の機械動作があり、±15msec程度のバラツキがある。これらの値を基に、ディスク稼働時間の実測値から、そのディスクに対するアクセスがどの程度ランダムになっているか推測できる。

ちなみに、本実験ではI/Oの回数は各ディスクに対し1430回ずつとなっている。pattern3のselect1では、ディスク稼働時間の実測値が11～12secとなっており、ランダムアクセスの割合は30～36%、pattern3のselect2、及びselect3では、ディスク稼働時間の実測値が6～7secとなっており、ランダムアクセスの割合は10～15%となる。select1が他よりもランダムアクセスの割合が高いのは、ディスクにおける衝突の頻度が高いことを示している。

4.4 経過時間に関する考察

pattern2, 3をpattern1と比較すると、経過時間の増加が著しく、測定毎の変動が大きかった。この変動は、ディスクの稼働時間、DBMSの検索プロセスの消費CPU時間、及びFileServerの消費CPU時間の変動よりもかなり大きく表れた。

経過時間以外に測定したデータは各リソースの稼働時間であるが、経過時間にはリソースが稼働していないアイドル時間も含まれる。そこで、pattern2, 3における経過時間の変動の要因として、アイドル時間の影響を考えてみる。すなわち、CPUやディスクの動作にアイドル時間が生じ、アイドル時間が変動することによって経過時間が変動すると仮定してみる。

アイドル時間の発生メカニズムを図5のように単純化して考えてみる。検索においては、DBMS検索プロセスからリード処理をFileServerに依頼し、その完了を待ってからDBMS検索プロセスで検索処理を行なうというように、交互に処理を行なうペアが存在する。

図5 (A)はpattern1に相当する。pattern1では、ペアとなるDBMS検索プロセスとFileServerが同一のWS上で動作している。図5 (A)の場合、各プロセスが干渉なくCPUを交互に使用しているため、アイドル時間が発生していない。

これに対し、図5 (B)は、ペアとなるDBMS検索プロセスとFileServerが別々のWS上に配置された場合を示す。この場合、同じWS上のDBMS検索プロセスとFileServerがCPUを奪い合い(図中の点線で示した部分)、その部分の処理効率が落ち、お互いに処理時間が延びてしまう。この時、もう片方のWS上ではプロセスが待たされ、これがアイドル時間となって現われる。このようなアイドル時間は泡のように細々と発生し、それらが積み重なって、結果的に全体の処理時間を長くする。pattern2、及びpattern3は、この場合に相当すると考えられる。

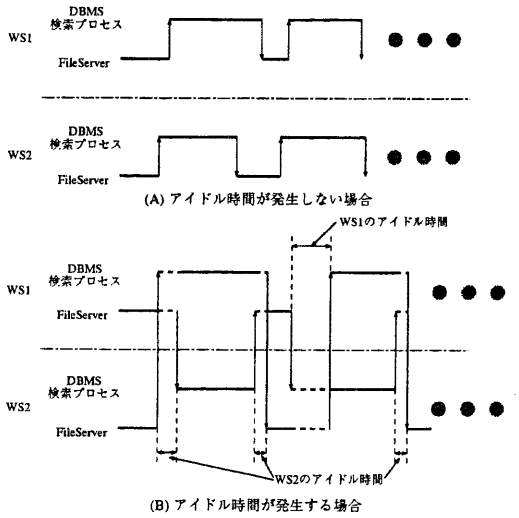

図5: アイドル時間発生の様子

4.5 シミュレーションによる検証

前述の仮定をシミュレーションによって検証する。

シミュレーションモデル

図6のように2つのプロセスproc1, proc2が交互に同期を取りながら処理を進める同期処理系を考える。2つのプロセスが1回ずつ処理を行なう処理単位を基本ルーチンと呼ぶこととする。

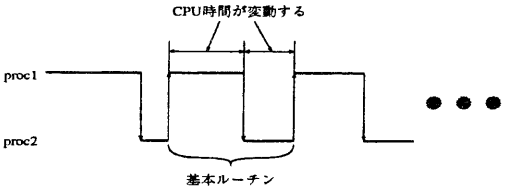

図6: 同期処理系

2つのCPUから構成される1システム内に、2つの同期処理系が同時に動作する環境を想定し、図5に示した2つの動作パターンをシミュレートする。すなわち、各同期処理系が別々のCPU上で動作する動作パターン(A)、各同期処理系を構成する2つのプロセスが別々のCPU上で交差して動作する動作パターン(B)の2つの動作パターンとなる。動作パターン(B)では、別々の同期処理系のプロセスがCPUを奪い合う。

シミュレーション結果とその分析

シミュレーションでは、同期処理系を構成する2つのプロセスのCPU時間の変動範囲を様々に変化させて測定した。本シミュレーションでは、基本ルーチンにおけ

る各プロセスのCPU時間は、平均値A、変動幅Wとし、$A-W \sim A+W$間を一様に変化するものとした。

動作パターン(A)、(B)の各々について、経過時間に対するCPUのアイドル時間の割合を測定した。10回の測定による平均値を表2に示す。

CPU時間			アイドル率	
proc1	proc2	変動率	(A)	(B)
100 ± 0	100 ± 0	0(%)	0.0(%)	0.0(%)
100 ± 10	100 ± 10	10	0.095	3.245
100 ± 20	100 ± 10	15	0.176	4.809
100 ± 30	100 ± 10	20	0.171	6.233
100 ± 10	200 ± 20	10	0.072	3.214
100 ± 20	100 ± 0	10	0.088	3.225
100 ± 20	100 ± 20	20	0.153	6.259

表 2: シミュレーション結果

結果を見ると、各プロセスの変動が全くない場合には、どちらの動作パターンでもアイドル時間が0になるが、変動幅を広くするとアイドル時間が発生していることが分る。ここで、動作パターン(B)の方が(A)と比較してはるかにアイドル時間の割合が大きくなっている。

シミュレーションによると、動作パターン(A)におけるアイドル時間は、2つの同期処理系を処理するCPUの内、先に終った方で発生する。これは2つの系の最終的な待合わせであり、上で仮定した泡のようなアイドル時間は、動作パターン(A)では発生していない。

これに対し、動作パターン(B)では、各CPUにおいて細かいアイドル時間が泡のようにランダムに発生していた。アイドル時間は、各プロセスの基本ルーチン毎のCPU時間変動によって確率的に発生していた。更に、全体の処理時間も変動していた。

シミュレーションを見る限り、動作パターン(B)のように2つの同期処理系を交差させて動作させた場合に、前記した仮定が適用されると考えられる。

変動幅とアイドル時間の関係

表2において、proc1の変動幅のみを大きくした場合、動作パターン(B)のアイドル時間の割合が増えている。各プロセスの変動幅の大きさがアイドル時間に影響を与えていると考えられる。

アイドル時間は、各プロセスの平均CPU時間と変動幅に依存している。ここで、2つのプロセスのCPU時間の平均値の合計に対する、変動幅の合計の割合を、変動率と呼ぶことにする。

表2において、CPU時間平均値や変動幅を変化させて測定したが、変動率が一定であると、アイドル時間の占める割合はほぼ同等の値を示すことが分る。

アイドル時間発生の一般的条件の考察

同様の現象は、今回の検索実験のケースに限らずに発生すると考えられる。アイドル時間発生の条件を一般化すると、以下のように分解される。

- 一連の処理を順番に複数のプロセスによって進める同期処理系がシステム内に複数動作する。
- 別々の同期処理系に属す複数のプロセスが、同一の資源を同時に使用する。
- 同一資源を複数プロセスで使用した場合、各プロセスに対する処理能力が低下(処理時間延長など)する。

数値計算のように詳細にタスクスケジュールを指示するAPではあまり問題とならないが、DBMSのように多種のプロセスで構成されるAPでは、上記の条件を避けた効率的なスケジューリングを考慮する必要がある。

5 まとめ

本稿では、複数のWSを通信網で接続したWSクラスタにおいて、DB検索の並列処理について評価・分析を行なった。その結果、以下のように、並列処理においては、各資源の処理能力のバラツキが並列度劣化に影響することが分った。

- ある処理を分割して並列に処理する際、各資源の処理能力のバラツキによって並列度が低下する。どんなに処理を均等に分割しても、処理能力のバラツキが大きければ、並列度は劣化する。
- 同期処理系を構成する複数のプロセスの動作パターンと、各プロセスの処理時間のバラツキによって、各資源のアイドル時間が泡のように発生し、全処理時間を長くしてしまう。

謝辞

本研究を進めるにあたり、DBMSのコード変更の環境を提供して頂いたNEC 第三コンピュータソフトウェア事業部の鴨居部長、小林主任、TinyTOPAZ使用に関して、数々の御協力を頂いたNEC C&C研究所の堀川課長、(株)NEC情報システムズの吉田氏、石井氏に深く感謝致します。また、実作業において協力して頂いたBIPシステムズ(株)の内山氏に深く感謝致します。

参考文献

[1] A. L. Cheung and A. P. Reeves, "High Performance Computing on a Cluster of Workstations", *Proc. of 1st Int. Symp. on High-performance Distributed Computing*, Sep. 1992.

[2] 堀川 隆, 紀 一誠: "DBMS動作特性の測定・解析手法", 情報処理学会研究報告 94-ARC-104/94-OS-62, 1994.

[3] 相場 雄一, 青木 久幸: "WSクラスタにおける遠隔ディスクアクセスとその評価分析", 情報処理学会第53回全国大会 講演論文集(1), Sep. 1996.