Kappa-P の並列問い合わせ処理

永沼和智[1]，佐藤裕幸[1]，河村元夫[2]

5R−4

1: 三菱電機(株)情報電子研究所　2: (財)新世代コンピュータ技術開発機構

1 はじめに

第5世代コンピュータプロジェクトでは、大規模知識処理を目的として、並列推論マシンPIM [1]及びその上で動作する並列知識処理システムの研究開発を行っている。Kappa-P [2]は、これらの知識処理システムに大量の複雑なデータの効率的な処理を提供するために開発された、並列非正規関係データベース管理システムである。Kappa-Pは、PIMの各クラスタに配置されたローカルDBMSと呼ばれる複数のデータベース管理システムから構成されており、クラスタ間での並列性を得るために、一つのテーブルを水平に分割し複数のローカルDBMSに配置することができる。そのテーブルへの問い合わせは、インタフェースプロセスにより構成テーブルへの並列な問い合わせに置き換えられ、各ローカルDBMSにより並列に処理される。本論文では、この水平分割テーブル処理、また分割テーブルの欠点である通信コストの増大を防ぐための方法を、実測値と共に報告する。

2 問い合わせ処理概要

2.1 処理の流れ

Kappa-Pでは、原始コマンドと呼ばれるプログラムインタフェースを提供している。原始コマンドとは、一つのテーブルに対して読みや更新の処理を行うコマンドで、インタフェースプロセスがユーザからコマンドを受ける。インタフェースプロセスは、コマンドを受けるとサーバと通信し、指定されたテーブルがどのローカルDBMSに存在するかと言う情報を得る。サーバは、データベースのグローバル情報を管理しているプロセスである。そして、インタフェースプロセスはそのローカルDBMS情報をコマンドに付加し、コーディネータにその情報を送る。その情報によりコーディネータは、各ローカルDBMSへコマンドを送り、各ローカルDBMSでコマンドが処理される。
(図1参照)

2.2 水平分割テーブル

水平分割テーブルとは、1つのテーブルをその属性値などにより複数の構成テーブルにレコード単位で分割し、複数のローカルDBMSに配置したものである。この分割方法には、属性値の範囲による分割、属性値のハッシュ値による分割、ラウンドロビンの3種類がある。属性の範囲による分割はその属性を条件に指定された時に有効であるが、構成テーブルのレコード数がまちまちになってしまうと言う欠点があり、これに対しラウンドロビンは構成テーブルの大きさが均一になるので、各テーブルに対する処理時間が均一になりやすい。これらは、そのデータに対する検索を考えて選択する必要がある。

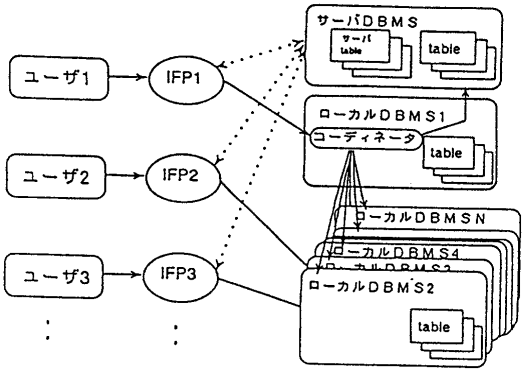

図1: 問い合わせ処理

水平分割テーブルに対する検索は、構成テーブルの存在する各ローカルDBMSで並列に処理されるが、ユーザは分割テーブルであることや構成テーブルの所在などを意識せずに一つのテーブルとしてに扱うことができる。ユーザから水平分割テーブルに対するコマンドが出されると、インタフェースプロセスはサーバとの通信により、そのテーブルの各構成テーブルの名前と存在するローカルDBMSを知る。そして、その情報によりコマンドは各ローカルDBMSに対する複数のコマンドに変換され、コーディネータ経由で各ローカルDBMSで並列に実行される。このときレコードの追加や削除のコマンドは、追加、削除するべき構成テーブルを分割情報から判断しその構成テーブルへのコマンドに変換される。

2.3 フィルタ付きレコード読み

テーブルを分割した場合にローカルDBMS間の並列性は得られるが、その分ローカルDBMS間の通信コストが増大すると言う欠点がある。特にユーザがデータを一度読み込み、そのデータに対してPIMの各クラスタを使って並列に処理を行いたい場合には、一箇所にデータが集められてから再び分散させ、そのデータに対する処理を行った結果をもう一度集めるために、かなりの通信コストがかかる。この通信コストの増大を防ぐために、Kappa-Pではフィルタ付きレコード読みの機能を提供している。このフィルタ付きレコード読みでは、ユーザがその検索の結果に行う自分で定義した処理(フィルタ)を指定することにより、その処理が分割された構成テーブルのあるクラスタで実行され、その処理の結果がユーザのいるクラスタに集められる。このため、一度データを

Parallel query processing on Kappa-P
Kazutomo NAGANUMA[1],Hiroyuki SATO[1], Moto KAWAMURA[2]
1:Mitsubishi Electric Corporation　2:Institute for New Generation Computer Technology.

集めてから分散させると言うことがなくなりまた、一回の通信データもフィルタを通った後のものになるので、通信コストはかなり軽減される。

3 評価

実際に一つのテーブルを分割した場合に、どれだけの速度向上が見られるのか、64 台のプロセサを持つ PIM/m を使って測定した。

3.1 測定方法

測定の対象としたデータは、公共のデータベース PIR の遺伝子データである。このテーブルのデータは約 61 メガあり、3 万 3 千のレコードを持っている。このテーブルを 16,32,64 に分割し、それぞれ次の測定を行った。なお、ディスクの数が少ないため必要なデータはあらかじめ主記憶に読み込んでから測定を行ったが、分割しないテーブルは主記憶の容量が足りないために実行できなかった。

(1) 属性 sequence が CH という文字列を含むレコードの検索。
この検索では、約 4119 件のレコードが結果として得られる。

(2) フィルタ付きレコード読み。
指定したフィルタは、属性 sequence がある一定のパターンの文字列を含むレコードを検索するモチーフサーチである。この検索では、47 件のレコードが結果として得られる。

(3) フィルタ付きレコード読みを使わずに、一度レコードを読み込んでからその結果に対して (2) と同じフィルタを並列に実行する。

3.2 測定結果 / 考察

測定結果を表 1に示す。

(1) の測定結果では、32 分割の実行時間が 16 分割のテーブルの 2 分の 1 に近い数値となっている。これは、テーブルの大きさが 2 分の 1 となりそれがそのまま実行時間に反映している。実際は通信時間、前処理、後処理の時間は 2 分の 1 とはならないので、厳密には 2 分の 1 よりも多少大きな数値となっている。

しかし 64 分割の場合には、32 分割の 2 倍とはならずに約 1.6 倍となっている。これは、一つの分割テーブルに対する処理が非常に軽くなったため、通信にかかる時間が無視できなくなったと言うこと、各ローカル DBMS で行われる検索の前のジョブやトランザクションの生成、テーブルのロック、検索の後の終了処理などにかかる時間は分割テーブルであろうと普通のテーブルであろうと一定であるため、実際の検索時間が短くなるとこれらの時間の占める割合も無視できなくなったと考えられる。

(2) の測定では、32 分割テーブルの実行時間は 16 分割の約 2 分の 1 となっており、64 分割の実行時間は約 4 分の 1 となっている。これは、(1) のレコード検索と比べここで指定したモチーフサーチの負荷がかなり高く実行時間もかかっているため、先に述べたような通信、前処理、後処理にかかる時間が無視できる範囲であったと考えられ、分割の効果が十分表れていると言える。

(3) の測定結果を (2) と比べてみるとそれぞれ約 1.5 倍から、2 倍近い時間がかかっている。フィルタ付きレコード読みを使わずに同じ処理をする場合には、フィルタの処理に入る前のレコード読みの結果が一度ユーザのいるクラスタに集められてから、再び分散することになる。この通信コストがそのまま数字に表れていると考えられ、フィルタ付きレコード読みの有効性が確認された。

テーブル分割数	16	32	64
(1) レコード検索	33,331	18,158	11,575
(2) フィルタ付き レコード検索	132,447	68,892	37,664
(3) レコード検索 + フィルタ	206,549	101,173	66,190

表 1: 各テーブルの実行時間 (msec)

この測定結果から、(1) の前方一致、後方一致と言うような検索に対しては、このテーブルを 64 以上に分割してもそれほどの効果が得られていない。これに対し (2) のモチーフ検索では、64 分割でも十分な効果が得られており、更に分割した場合の効果も期待できる。このように、検索の負荷の大きさにより適した構成テーブルの大きさは異なり、そのデータの使用目的のより分割数を決定することが必要である。

4 おわりに

今回測定に使用した並列推論マシン PIM は 64 台のプロセサを持っているが、現在 ICOT では 256 台版の PIM が動作している。また、Kappa-P は大量で複雑なデータを扱うことを目的としており、今後は今回扱った以上に大量で複雑のデータに対して負荷の大きい処理が要求されると考えられる。そのような要求に対して水平分割テーブルは非常に有効な手段であることが今回の測定により確認できた。ただ、データの複雑さ、データに対する処理の負荷の大きさにより、最適な分割テーブル数が異なるため、今後は分割をする際の目安になるような解析をする必要があると思われる。

参考文献

[1] 瀧: *Parallel Inference Machine PIM*, The International Conference on Fifth Generation Computer Systems'92,1992.

[2] 河村他: 並列データベース管理システム Kappa-P の概要, 第 45 回情処全国大会,5R− 3,1992,10.

[3] 横田他: *Overview of the Knowledge Base Management System(KAPPA)*, The International Conference on Fifth Generation Computer Systems'88,1988.