4K−5

並列 DBMS: DBKernel における動的負荷分散機構の評価

安井 隆宏†　田村 孝之†‡　小口 正人†　喜連川 優†
{yasui, tamura, oguchi, kitsure}@tkl.iis.u-tokyo.ac.jp
†東京大学 生産技術研究所　‡三菱電機(株) 情報通信システム開発センター

1　はじめに

近年, データベースサイズが大規模化する一方で, 意思決定支援システム等に見られるような, 複雑な問合せ, 特に大規模リレーションへの問合せに対する高速な応答への要求が高まっている. しかし, 分散メモリ並列計算機環境においては, ノード間の負荷の偏りのために充分な並列効果を得られないことがあり, タスクスケジューリング等の研究が盛んに行われて来た. 並列データベースシステムにおいても, データの偏り (skew) のためにノード間の負荷にばらつきが生じる事が多い. これに対し, 我々は, 関係データベースにおける Right-deep 方式の多重結合演算処理についてハッシュラインマイグレーションという動的負荷分散アルゴリズムを提案すると共に, 本研究室で構築した並列 DBMS: DBKernel への実装を行った. 本稿では 100 ノード規模の環境で提案するアルゴリズムの有効性を示す.

2　Right-deep 結合演算の実行モデル

N ステージで構成された Right-deep 結合演算 を P ノードで処理した際の実行モデルを図1に示す. まず, リレーション $R_1, R_2, \cdots, R_N$ をハッシュ値により処理すべきノードに送出し, 各ノードでハッシュテーブルを作成する. この各ノードのハッシュテーブルをステージフラグメントと呼ぶ. 全ステージのハッシュテーブルの作成が完了すると, リレーション R_P を ディスクから読み出し, ハッシュ値により処理すべきノードへ送出し, プローブを行い, 結合属性が条件に一致した場合には, タプルの結合を行い結果タプルを生成する. さらに, その結果タプルにより次ステージのプローブを行う.

3　動的負荷分散機構

負荷の偏りを解消するため, 結合演算実行中に同一ステージのステージフラグメント間でハッシュラインをマイグレートさせることにより動的に負荷分散を行う (ハッシュラインマイグレーション). 結合演算処理を行うノードの負荷を監視するため, フォアマンと呼ぶ別の制御ノードを一台用意し, これにより集中管理する. 各ノードでは, ステージフラグメント及びハッシュラインに対し, 入力タプル数 N_{recv}, 比較回数 N_{comp}, 出力タプル数 N_{send} の統計を取り, また, 各ノードはハッシュラインのマイグレート先を登録するマイグレーションテーブルを保持する. フォアマンは, 一定のインターバルで各ノードからステージフラグメントの統計情報を収集し, ノード i, ステージ j の負荷 $SF(i,j)$ を

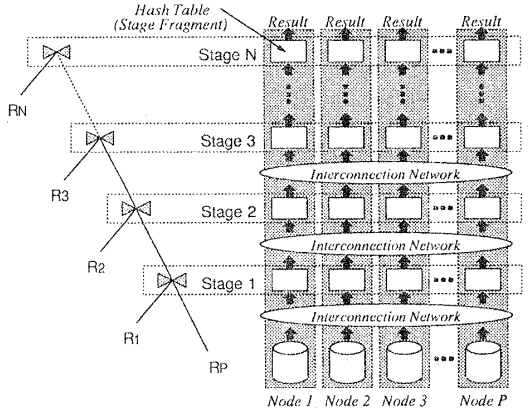


図 1: Right-deep 結合演算実行モデル

ノード数	96 [node]
ステージ数	2 [stage]
タプルサイズ	32 [byte/tuple]
プローブタプル数	3,000,000 [tuples/node]
ハッシュライン数	10,000 [lines/node]
負荷情報収集インターバル	28 [sec]
問合せの結合率	100 [%]

表 1: 実験環境

$$SF(i,j) = N_{recv}(i,j) \times C_{recv} + N_{comp}(i,j) \times C_{comp} + N_{send}(i,j) \times C_{send}$$

により算出する. $C_{recv}, C_{comp}, C_{send}$ は, それぞれ,1 タプルあたりの受信コスト, 比較コスト, 送信コストを表す. 同一ステージ内のステージフラグメント間で負荷の比較を行い, 負荷の偏りを検出した際には, 更に, 偏りの検出されたステージの負荷の高いステージフラグメントから負荷の高いハッシュラインの統計情報を収集する. そして, ハッシュライン k の負荷 $HL(i,j,k)$ を

$$HL(i,j,k) = N_{recv}(i,j,k) \times C_{recv} + N_{comp}(i,j,k) \times C_{comp} + N_{send}(i,j,k) \times C_{send}$$

により算出し, ステージ内のステージフラグメントの負荷が均等になるように, ハッシュラインのマイグレーションプランを作成する. ハッシュラインのマイグレーションが完了したあとでマイグレーションテーブルの更新を行う.

4　PC クラスタへの実装と性能評価

PC クラスタ は, 200MHz Pentium Pro マイクロプロセッサを搭載する PC 100 台を 155Mbps ATM および 10Mbps イーサネットの 2 系統のネットワークで相互結合したシステムである. 各ノードには, 本研究室で開発した並列データベースサーバ DBKernel が実装されている. DBKernel は前述した動的負荷分散機構を有し, 各

Performance Evaluation of Dynamic Load Balancing in Parallel DBMS: DBKernel
T.Yasui†, T.Tamura†‡, M.Oguchi†, M.Kitsuregawa†
†Institute of Industrial Science, University of Tokyo
‡Information & Communication Systems Development Center, Mitsubishi Electric Corporation

		skew1	*skew2*	*skew3*	*skew4*	*skew5*
プローブ数の比	Stage 1	2 : 1 : ... : 1	3 : 1 : ... : 1	2 : 2 : 1 : ... : 1	2 : 1 : 1 : ... : 1	2 : 1 : 1 : ... : 1
	Stage 2	1 : 1 : ... : 1	1 : 1 : ... : 1	1 : 1 : 1 : ... : 1	2 : 1 : 1 : ... : 1	1 : 2 : 1 : ... : 1
結合率の比	Stage 1	2 : 1 : ... : 1	3 : 1 : ... : 1	2 : 2 : 1 : ... : 1	2 : 1 : 1 : ... : 1	2 : 1 : 1 : ... : 1
	Stage 2	1 : 1 : ... : 1	1 : 1 : ... : 1	1 : 1 : 1 : ... : 1	2 : 1 : 1 : ... : 1	1 : 2 : 1 : ... : 1
実行時間 [sec]	負荷分散無し	246	358	264	344	261
	負荷分散有り	170	178	186	175	166

表 2: 実験結果

問合せプロセスを監視し, 問合せプロセス毎に負荷分散が行われる.

我々はこれまで, 30 ノード規模での環境において本負荷分散手法の評価を行って来た [1] が, この度, 主記憶領域の利用効率を高めるため, ハッシュラインのマイグレート先を登録するマイグレーションテーブル及びハッシュテーブルの実装面での改良を行い, 100 ノード規模への適用を可能とした.

性能評価に用いた問合せは $R_2 \bowtie (R_1 \bowtie R_P)$ と表記される 2 重結合演算とし, これを 96 ノードを用いて処理を行った. 問合せで用いるデータについては 表 1 に示すようにステージフラグメントあたりのハッシュライン数が 10,000, プローブタプル数が 3,000,000 である. また, このデータには入力タプル数および結合率をノード間で偏らせることによる人為的なスキューを導入し, ここでは表 2 に示すような 5 種類のスキューを用意した. *skew1* では, ノード 0 のプローブ数 (= 入力タプル数) が他のノードに比べ 2 倍であり, 結合率も 2 倍となっていることを示している. この場合, 出力タプル数は他のノードの 4 倍となることになる. 表 2 からは 5 種類のスキューのいずれにおいても本手法を用いた負荷分散を行う事により, 実行時間が大幅に短縮されていることがわかる.

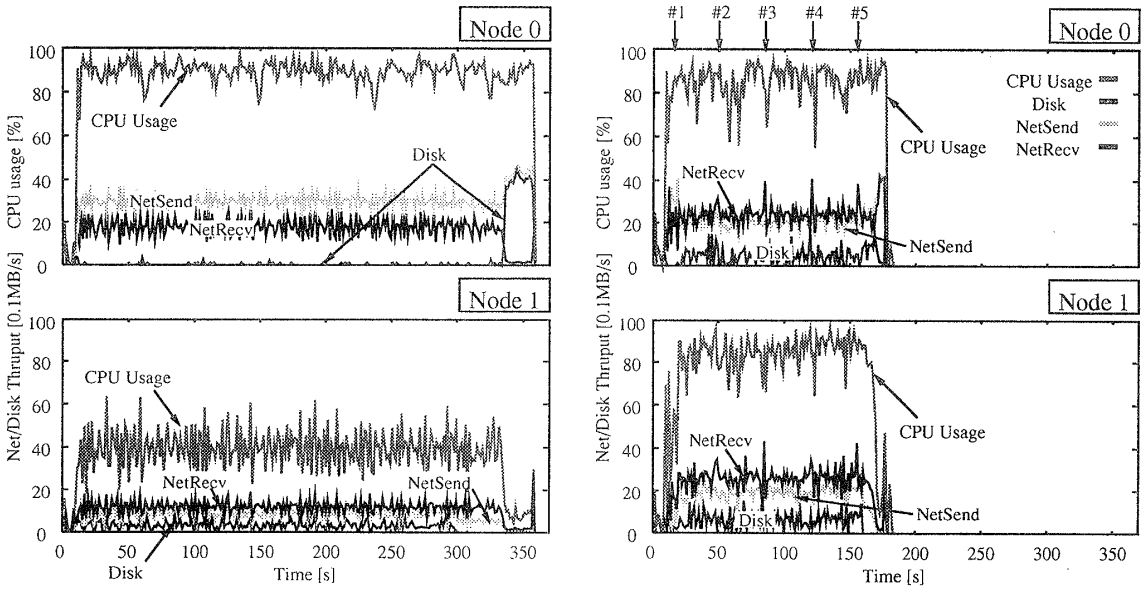
skew2 の実行トレースとして, 負荷の高いノード 0 とその他のノードの代表としてノード 1 を示した (図 2). 負荷分散を行わない場合には, 終始, ノード 0 の負荷が高く, ノード 0 のデータをディスクから読み出せない状態であり, 他のノードのデータの処理を終った後に (335 [*sec*]), やっと読み出す事ができるようになる. 一方, 負荷分散を行った場合, 1 回目のマイグレーション (#1) 後には, 負荷が均等となり, 処理がスムーズに行えるようになっていることが分かる. 今回の実験では, 負荷情報収集インターバルを 28 [*sec*] したため, 5 回の負荷情報収集が行われ, その各々においてハッシュラインマイグレーションが行われた. 1 回目の負荷情報収集時にはスキューは 4.51 であったが, 2 回目以降は, それぞれ 0.45, 0.5, 0.43, 0.28 となり偏りが少なくなったことも確認された.

5 まとめ

ハッシュラインマイグレーション技法を利用した動的負荷分散方式を並列 DBMS: DBKernel に実装し, 100 ノード規模の環境を用いて評価を行った所, 大幅な性能向上が確認され, 本手法が有効であることが確認できた. 今後, 多数段の結合演算を用いた測定や複雑なスキュー分布に対する測定を行い, より詳細な評価を行う予定である.

参考文献

[1] 安井, 田村, 小口, 喜連川. 並列関係データベース処理システムに於ける動的負荷分散機構に関する一考察. 電子情報通信学会 第 9 回データ工学ワークショップ, 1998.



[負荷分散無し] [負荷分散有り]

図 2: 実行トレース