並列 SQL サーバ SDC-II の TPC-D ベンチマークによる性能評価

1R－9

田村孝之　喜連川 優　高木幹雄
東京大学 生産技術研究所

1 はじめに

関係データベースは現在さまざまな分野で広く用いられているが，近年ではトランザクション性能だけでなく問い合わせ処理の性能も重視されるようになった．データベースシステムの標準ベンチマーク策定委員会であるTPC もこれまでのトランザクション処理に対するベンチマークに加え，意思決定支援システムにおける非定型問い合わせ処理を対象としたTPC Benchmark D を昨年発表した．以来，いくつかのDBMS ベンダがTPC-D ベンチマークの測定結果を公表しているが，データベースが非常に大規模なためそのほとんどが並列機上での実行結果である．しかし，複雑な多重結合演算を含む問い合わせにおいては単純な並列化だけでは不十分であり，高度なアルゴリズムを用いることが重要である．高並列関係データベースサーバ SDC-II [1] では，このような非定型問い合わせ処理の高速化を目指し，I/O の効率化と併せて right-deep tree に基づく多重結合演算の支援を行っている．本論文では，SDC-II 上で TPC-D の問い合わせを実行した場合の性能評価を行い，実行方式が性能に与える影響について考察する．

2 SDC-II における TPC-D 問い合わせの実行方式

SDC-II は，オメガネットワークで結合された最大 8 台のデータ処理モジュール (DPM) から構成され，各 DPM はプロセッサ (MC68040 25MHz) 最大 7 台，4 台の SCSI ディスク装置，および 32MB のメモリとを結合した共有バスクラスタである．SDC-II が目的とするのは大規模関係データベースに対する非定型問合せ処理の高速化であり，I/O 効率が高く，負荷の偏りに強いハッシュ結合アルゴリズムを採用することに加え，インテリジェントな I/O プロセッサとデータ駆動型のプロセス実行モデルを用いることでハードウェアとソフトウェアの両面から I/O 性能自体の向上を図っている．

TPC-D ベンチマーク [2] は，大量データをアクセスし，複雑度の高い問い合わせが実行される意志決定支援システムを対象としたベンチマークであり，8 つのリレーションに対して 17 の問い合わせと 2 つの更新が発行されるようになっている．これらのうち最も単純な Query 1 と 6 は単一リレーションに対する集計問い合わせであるが，最も複雑な Query 5, 8, および 9 では 6 つのリレーションに対する結合演算が行われるため多重結合演算の効率が要求される．

表 1: TPC-D 問い合わせの実行環境

System Configuration	SDC-II		IBM RS/6000 SP Model 302 (1995/12/18)
CPU	MC68040 25 MHz		Power2 66 MHz
#CPUs	16 (4 × 4)		32 (1 × 32)
Memory Size	32MB × 4		256MB × 32
Disk Capacity	4GB × 16		2.2GB × 192
Database Size	1GB	10GB	100GB
DB / Memory	7.8	78	12.2

TPC-D による SDC-II の性能評価には，スケールファクタ 1 および 10 (それぞれ総容量 1GB および 10GB) のデータベースを用いた．表1 に問い合わせの実行に用いた環境を IBM RS/6000 SP 302 のものと併せて示す．SDC-II の DPM 数は 4 台に固定し，NATION および REGION 以外の各リレーションは各 DPM 間にデクラスタリングされて格納される．NATION, REGION は小さいので特定の DPM に集中させた．なお，各 DPM に対するデクラスタリングの方法は範囲分割としたが，問い合わせ実行の際にはこの知識を用いた最適化は行っていない．これは，SDC-II の目的が ad-hoc な問い合わせに対するワーストケースでの性能を向上させることだからである．また，同様の理由によりインデックスも全く用いていないため，全てのファイルアクセスはフルスキャンであり，選択条件により不要なタプルを捨てている．このようなバースト転送時の I/O デバイスの性能は，各ディスクの平均読み出し速度が 2.6 MB/sec，データネットワークの平均スループットが約 10 MB/sec である．

また，問い合わせのコンパイル，実行プランの生成は人手で行っているため，フロントエンドのオーバヘッドは実行時間には含まれていない．問い合わせの結合演算および集計演算にはハッシュ分割アルゴリズムを用い，多重結合演算においては可能な限り right-deep tree[3] を用いるようにしている．1 GB のデータベースでは LINEITEM 以外のリレーションはそれほど大きくなく，同時にハッシュテーブルを作ることが可能なため，中間結果の書き出しは全く不要である．一方，10 GB のデータベースに対しては，中間結果がメモリに収まらなくなるため，より複雑な実行方式を採る必要がある．例とし

Performance Evaluation of the Parallel SQL Server SDC-II with the TPC-D Benchmark
T. Tamura, M. Kitsuregawa, and M. Takagi
Institute of Industrial Science, The University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, Japan.

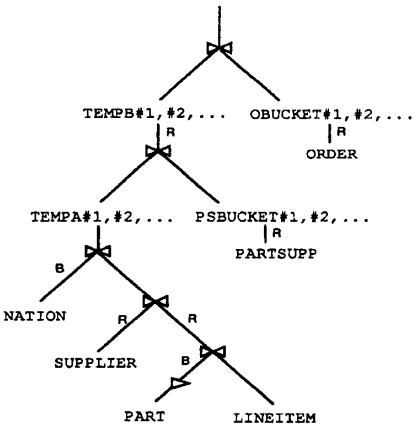
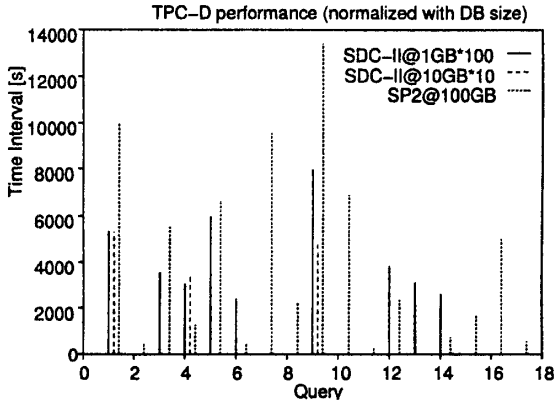


図 1: Query 9 (10GB) の問い合わせ実行木

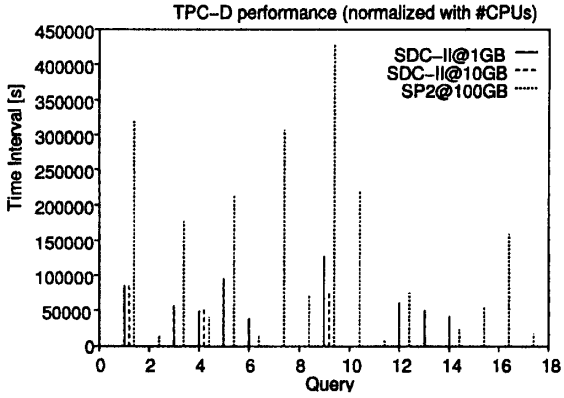
て, Query 9 の実行木を図 1に示す. 初めの 3 段の結合演算の後は, バケット分割を伴う left-deep tree になっている. 木の枝に記した R はタプルをハッシュ値にしたがって DPM 間で再分配することを表し, B はタプルを全 DPM にブロードキャストして各 DPM 毎に複製を持たせることを意味している. この問い合わせでは, ビルドリレーションのハッシュキーはプライマリキーであるため, 最初のデクラスタリング時にハッシュ分割法を用いていればローカルにビルドが行えるが, このような最適化は意図的に避けたため, 全てネットワークを経由する必要がある. また, NATION はサイズが非常に小さいので, ハッシュ分割をせずにブロードキャストを用い, ネットワークをバイパスしている. また, SUPPLIER と PARTSUPP の間では共通のハッシュキーを用いているため再分配の必要がない.

3 TPC-D による性能評価

ここでは, SDC-II における TPC-D 問い合わせの実行時間を商用機と比較して行うが, 表 1からも分かるように共通する環境がほとんど無いので正規化が必要となる. まず, データベースのサイズを共通にした場合の比較を図 2(a) に示す. 全体的に SDC-II の結果の方が問い合わせ毎の変動が少なくなっているが, これは常にファイルの全体をスキャンしているのでインデックスの有無による性能の変化が起こらないためと考えられる. インデックスが有効に働く Query 13 や 14 では当然のことながら不利な結果になっているものの, その他の問い合わせでは処理時間が短くなっている. これは, SDC-II の I/O アクセス性能の高さと多重結合演算の実行効率の高さを示すものと考えられる. 図 2(b) は, データベースのサイズに加えて CPU の台数で正規化した場合の図であるが, SDC-II の方が CPU 数が少ないのでさらに有利な結果になっている. 実際には CPU 当たりの処理能力も異なるのでその差はもっと広がるといえる. データベースのサイズが変わった時に, 性能が線形に変化するとは限らないが, SDC-II 側が初期データ配置や



(a) データベースサイズで正規化



(b) データベースサイズおよび CPU 数で正規化

図 2: TPC-D 問い合わせの実行時間

アクセス手段などの点でワーストケースになっていることを考慮すると, この結果は SDC-II の処理効率の高さを示すものと言える.

4 まとめ

本論文では TPC-D ベンチマークによる SDC-II の性能評価を行い, SDC-II のハードウェアならびにシステムソフトウェアが究めて効率良く稼働していることを明らかにした.

参考文献

[1] 中村, 平野, 田村, 喜連川, 高木. スーパーデータベースコンピュータ SDC-II におけるシステムソフトウエアの設計と実装. 信学論 Vol.J78-D-I, No.2, pp.129–141, 1995.

[2] Transaction Processing Performatce Council. TPC Benchmark^{TM}D (Decision Support) Standard Spec. Rev. 1.1. 1995.

[3] D. Schneider and D. DeWitt. Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines. Proc. of 16th Int. Conf. on VLDB, pp. 469–480, 1990.