データベースシステム 109－21
（1996．7．25）

# 並列 SQL サーバ SDC-II の TPC-D ベンチマークを用いた性能評価

田村孝之 喜連川 優 高木幹雄
東京大学 生産技術研究所
〒106 東京都港区六本木 7-22-1

小川義久
富士通株式会社
〒211 川崎市中原区上小田中 1015

**概要**

SDC-II は関係データベースシステムにおける問い合わせ処理の高速実行を目的とした高並列 SQL サーバである．SDC-II では大量のデータに対して ad-hoc に複雑な問い合わせが発行される環境を想定しているが，このようなシステムの性能を評価する標準ベンチマークとして TPC-D が昨年策定され，数社の並列 DBMS ベンダが結果を公表し始めている．本論文では，SDC-II における並列関係データベース処理の効率的な実行方法を述べた後に，TPC-D ベンチマーク実行時の測定結果を示し，小規模なハードウェア資源で優れた性能を引き出していることを明らかにする．

# Performance Evaluation of the Parallel SQL Server SDC-II Using the TPC-D Benchmark

Takayuki Tamura Masaru Kitsuregawa Mikio Takagi
Institute of Industrial Science, The University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, Japan.

Yoshihisa Ogawa
Fujitsu Limited
1015, Kamikodanaka, Nakahara-ku, Kawasaki 211, Japan.

**Abstract**

The SDC-II is a highly parallel SQL server aiming to speed up query processing in the relational database systems. The SDC-II is intended to operate in such environments where ad-hoc queries with high dgrees of complexity are issued to very large relations. To evaluate performances of such decision support systems, TPC announced TPC Benchmark D, and several parallel DBMS vendors begin to report the results of the benchmark. This paper reveals that the SDC-II shows very good performance in spite of limited hardware resources, by comparing its benchmark results with that of the other commercial systems.

## 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プラットフォーム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．以下第2節では SDC-II のアーキテクチャおよびシステムソフトウェアの構成について概観し，第3節では TPC-D ベンチマークの概要を述べ，第4節で TPC-D ベンチマークを用いた実行結果を示し評価を行なう．

## 2 SDC-II における並列データベース処理

### 2.1 SDC-II のアーキテクチャ

SDC-II は，図1 に示すように，ネットワークで疎結合された最大 8 台のデータ処理モジュール (Data Processing Module, DPM) から構成され，各データ処理モジュールは，プロセッサ (MC68040 25MHz) 最大 7 台，SCSI I/F 4 本，およびステージングバッファ用のメモリとを共有バスで結合した密結合型のアーキテクチャを持つ．このような構成により密結合の利点である軽い通信コストによる高速性とモジュール数の増減によるスケーラビリティを同時に得ることができる．

また，並列データベース処理におけるデータパスの実効バンド幅を確保するために，モジュール内の共有バスとモジュール間のネットワークのそれぞれをデータ系およびコントロール系とに分離している．共有バスについては，処理データ転送用のバスを HBus (High Speed Data Bus)，コントロール用のバスを

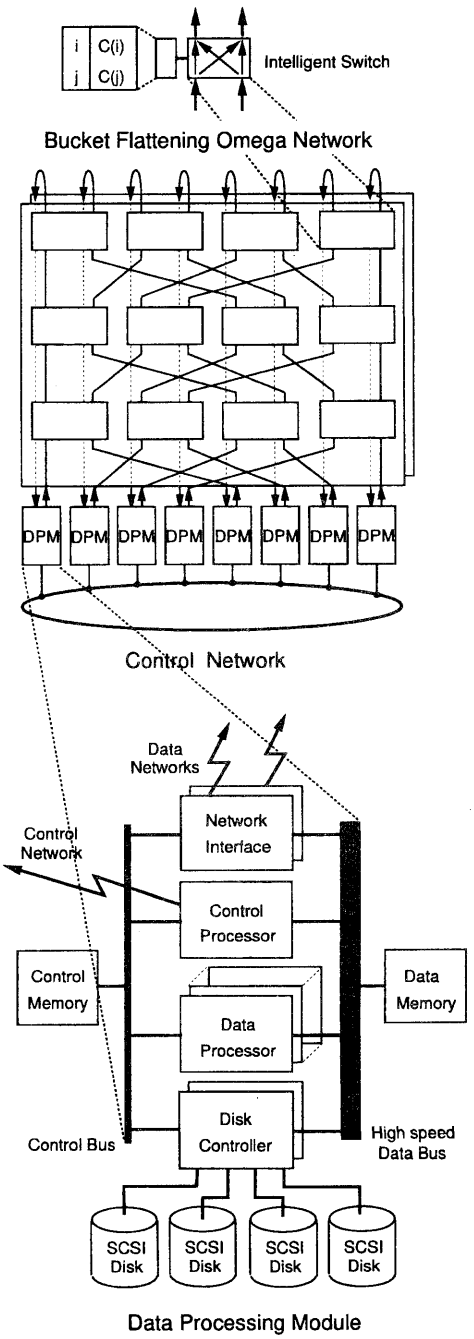



図 1: SDC-II の構成

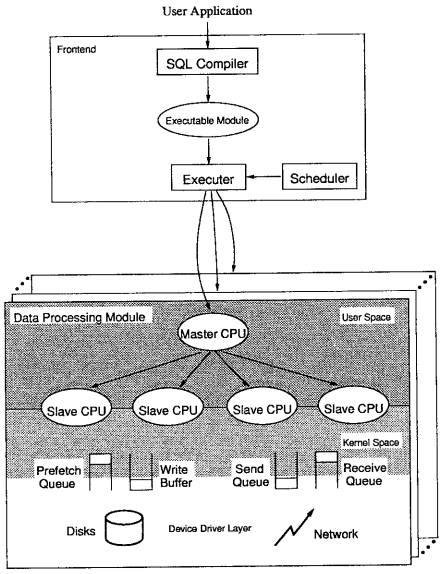



図 2: SDC-II のソフトウェア構成

CBus (Control Bus) と呼んでおり，これに対応して HBus からのみアクセスできる DM (Data Memory, 32MB) と，CBus からのみアクセスできる CM (Control Memory, 2MB) という 2 種類の共有メモリが存在する．一方，ネットワークについては処理データ専用のものを DNet (Data Network)，フロントエンド計算機との通信に用いるものを CNet (Control Network) と呼んでいる．CNet には汎用性を考慮して Ethernet を用いているが，DNet は大量データを転送するための高速性だけでなく，並列データベース処理を効率的にサポートするための機能性 [2] も併せ持つ間接多段網である．

## 2.2 SDC-II のシステムソフトウェア

大規模データベースに対する関係演算処理においては，大量のデータの移動に伴って処理が進むため，ディスクやネットワークに対する入出力操作には高い効率が要求される．SDC-II では，ディスクおよびネットワークに専用の管理 CPU を用い，メインの CPU を割り込み処理や入力バッファの割り当てなどの低レベルの処理から解放しているが，それに加えてシステムソフトウェアにおいても以下に述べる技法を用いることにより，I/O の効率化を図っている．

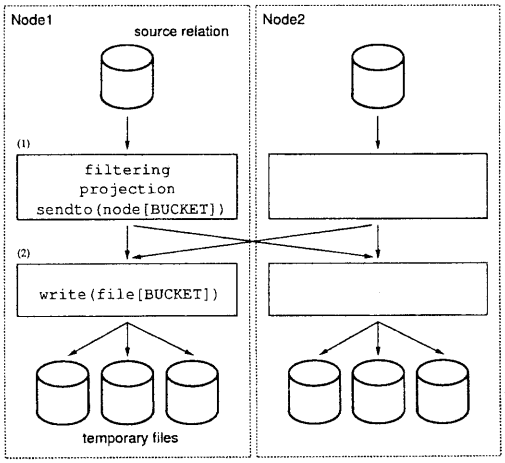



図 3: ハッシュ分割アルゴリズムにおける分割フェーズ

SDC-II のソフトウェア構成を，図 2 に示す．ユーザアプリケーションから発行された SQL 問い合わせ文は，フロントエンドマシン上で実行される SQL コンパイラによりコンパイルされ，実行プランの最適化が施された後に，実行形式のオブジェクトモジュールとして出力される．その後，Scheduler の決定に従って Executer がその問い合わせの実行を開始すると，関係する DPM 上のマスタ CPU (Control Processor) に対して実行要求メッセージが送られ，その結果 マスタ CPU は指定された実行モジュールをロードして新たなプロセスを生成する．DPM 上の各プロセスからの出力は Executer によりまとめられ，ユーザアプリケーションに実行結果として返される．DPM 上で実行される処理の内容は問い合わせにより異なるが，ほとんどの場合入出力デバイスをオープンした後にフォーク操作を行い，複数の CPU 上でプロセスのコピーが並列に実行される．

SDC-II における処理の例として，ハッシュ分割に基づく並列関係演算アルゴリズムにおける分割フェーズの概略を図 3 に示す．この処理は，ディスクから読み込んだタプルから選択条件を満たすものを取り出し，分割キーにハッシュ関数を適用してバケット ID を求め，そのバケットの格納されるノードにネットワークを介してタプルを送出する部分 (図の (1)) と，ネットワークから受け取ったタプルを対応するバケットファイルに書き出す部分 (図の (2)) とから構成さ

れる. (1), (2) のどちらの処理もデバイスからの入力によって実行が起動されるため, 互いに独立した処理として扱う必要があるが, SDC-II ではディスパッチャプロセスが全ての入力をポーリングし, データの到着している入力に対して対応するルーチンを呼び出すという方法を採っている. これは, プロセスやスレッドの切り替えに伴うオーバヘッドをなくし, また各ルーチンの実行順序の制御を容易にするためである. また, ディスパッチャプロセスをカーネル空間で動作させ, コールバックルーチンをカーネル空間で実行することにより, カーネルのディスクバッファやネットワークバッファに直接アクセスすることが可能になり, ユーザ空間とカーネル空間の間でのデータコピーによるオーバヘッドをなくすことができる.

## 3 TPC-D ベンチマークの概要

TPC-D ベンチマーク [3] 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.

表 1: TPC-D ベンチマークのリレーション

| リレーション | 外部参照 | タプル数 |
|---|---|---|
| PART | | 200000 × SF |
| SUPPLIER | NATION | 10000 × SF |
| PARTSUPP | PART<br>SUPPLIER | 800000 × SF |
| CUSTOMER | NATION | 150000 × SF |
| ORDER | CUSTOMER | 1500000 × SF |
| LINEITEM | ORDER<br>PART<br>SUPPLIER | ∼ 6000000 × SF |
| NATION | REGION | 25 |
| REGION | | 5 |

表1に TPC-D で用いられるリレーションを示す. ただし, SF はデータベースサイズのスケールファクタであり, SF = 1 の時の全体のサイズはおよそ 1GB である. また, 表 2には TPC-D の各問い合わせがいくつのリレーションにアクセスするかをまとめた. 多くのリレーションにアクセスする問い合わせでは, それだけ多くの結合演算を実行する必要があり, 多重結合演算の効率が要求される.

表 2: TPC-D ベンチマーク問い合わせの複雑度

| アクセス<br>リレーション数 | 問い合わせ |
|---|---|
| 1 | Query 1, Query 6 |
| 2 | Query 4, Query 12, Query 13,<br>Query 14, Query 15, Query 17 |
| 3 | Query 3, Query 11, Query 16 |
| 4 | Query 10 |
| 5 | Query 2, Query 7 |
| 6 | Query 5, Query 8, Query 9 |

これまでに TPC-D の結果を公表しているのは IBM PC を除くと, IBM RS/6000 SP 302/403, NCR WorldMark 5100M, Tandem NonStop Himalaya K20016, Sun Ultra Enterprise 6000, および HP 9000 Enterprise EPS30 の並列システムであり, いずれも 100GB 以上のテストデータを用いている.

## 4 性能評価

TPC-D による SDC-II の性能評価として, 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て測定を行った. 表 3 に問い合わせ毎の実行時間を IBM RS/6000 SP 302 上の DB2 による結果と併せて示す.

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, データベース生成プログラム DBGEN の性質により各 DPM に対するデクラスタリングの方法は範囲分割になるが, 問い合わせ実行の際にはこの知識は用いていない. SDC-II では, 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を狙いとしているので, 問い合わせに依存した最適化は敢えて行わないことにした. 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. したがって, 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.

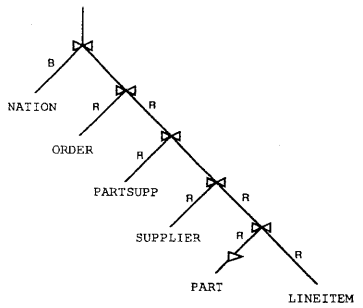
また, 問い合わせのコンパイル, 実行プランの生成は人手で行っているため, フロントエンドのオーバヘッ

表 3: TPC-D 問い合わせの実行結果

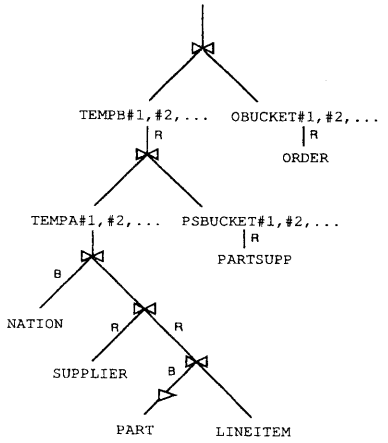
| System Configuration | SDC-II | | IBM RS/6000 SP Model 302 (1995/12/18) |
|---|---|---|---|
| CPU | MC68040 25 MHz | | Power2 66 MHz |
| Number of CPUs | 16 (4 × 4) | | 32 (1 × 32) |
| Memory Size | 32MB × 4 | | 256MB × 32 |
| Disk Capacity | 4GB × 16 | | 2.2GB × 192 |
| Database Size | 1GB | 10GB | 100GB |
| Database Size / Memory Size | 7.8 | 78 | 12.2 |
| Query 1 | 53.2 | 528.5 | 9981.0 |
| Query 3 | 35.6 | | 5534.4 |
| Query 4 | 30.5 | 346.3 | 1252.1 |
| Query 5 | 59.8 | | 6627.8 |
| Query 6 | 24.0 | | 516.9 |
| Query 9 | 74.5 | | 13404.5 |
| (broadcast) | 46.0 | 482.7 | |
| Query 12 | 38.2 | | 2331.5 |
| Query 13 | 31.1 | | 12.6 |
| Query 14 | 25.9 | | 719.9 |

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ている．多重結合演算においては可能な限り right-deep tree[4] を用いるようにしており，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．以下の問い合わせ Query 9 に対しては，その実行木は 1GB および 10 GB のデータベースそれぞれに対して，図 4の (a) および (b) のようになる．

```
select Nation, Year, sum(Amount) as Sum_Profit
from    (select N_Name as Nation,
         extract(year from O_Orderdate) as Year
         L_Extendedprice * (1 - L_Discount)
                - PS_Supplycost * L_Quantity
                        as Amount
         from part, supplier, lineitem, partsupp,
                orderx, nation
         where  S_Suppkey  = L_Suppkey
            and PS_Suppkey = L_Suppkey
            and PS_Partkey = L_Partkey
            and P_Partkey  = L_Partkey
            and O_Orderkey = L_Orderkey
            and S_Nationkey = N_Nationkey
            and P_Name like '%green%'
         )
group by Nation, Year
order by Nation, Year desc;
```



(a) 1GB データベースに対する実行木



(b) 10GB データベースに対する実行木

図 4: Query 9 の問い合わせ実行木

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.

図 4(a) の実行木による結果は, 表 3の Query 9 の項の上段に示したものである. 下段のものは, LINEITEM の再分配の代わりに PART のブロードキャストを用いた場合のもので, ネットワークに対する負荷が軽減されるために性能が向上している. 図 4(b) でも同様の方法を取っている. この場合, 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.

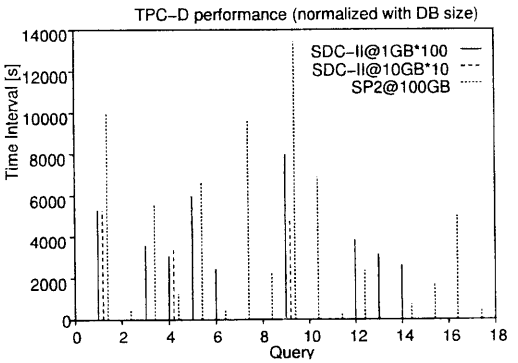



図 5: データベースサイズで正規化した実行時間

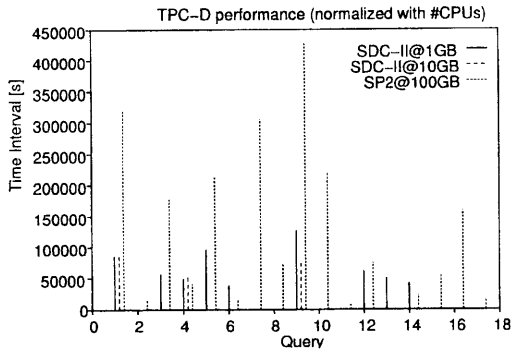



図 6: データベースサイズおよび CPU 数で正規化した実行時間

次に, 商用機の性能と比較を行うが, 表 3からも分かるように共通する環境がほとんど無いので何らかの正規化を行わなければならない. まず, データベースのサイズが共通だった場合についての比較を図 5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 6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.

## 5 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 稔, 平野 聡, 田村孝之, 喜連川 優, 高木幹雄. 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] 田村孝之, 中村 稔, 喜連川 優, 高木幹雄. 並列関係データベース処理を支援する相互結合網: — SDC-II におけるバケット平坦化ネットワークの実装と評価 —. JSPP'95, pp.129–136, 1995.

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

[4] 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.