拡張SEQUOIA2000ベンチマークを使った分散並列オブジェクトデータベースシステム「出世魚」の並列処理性能評価

田村，慶一
九州大学大学院システム情報科学府知能システム学専攻 ： 博士後期課程

中野，裕也
九州大学大学院システム情報科学府知能システム学専攻 ： 修士課程

金子，邦彦
九州大学大学院システム情報科学研究院知能システム学部門

牧之内，顕文
九州大学大学院システム情報科学研究院知能システム学部門

https://doi.org/10.15017/1515721

出版情報：九州大学大学院システム情報科学紀要. 6 (1), pp.83-88, 2001-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第6巻　第1号　平成13年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.6, No.1, March 2001

拡張SEQUOIA2000ベンチマークを使った分散並列オブジェクトデータベースシステム「出世魚」の並列処理性能評価

田村慶一*・中野裕也**・金子邦彦***・牧之内顕文***

Evaluation of Parallel Processing of Distributed Parallel Object Database System ShusseUo using Extended Sequoia2000 Benchmark

Keiichi TAMURA, Yuya NAKANO, Kunihiko KANEKO and Akifumi MAKINOUCHI

(Received December 15, 2000)

Abstract: ShusseUo is a parallel object database system developed by us. It enables to distribute statically and/or dynamically a database on computers(sites) connected by a commodity network with Internet interface. It allows the sites to process queries in parallel. The data of the database, either statically or dynamically distributed, can be shared by any site on ShusseUo. Performance of speedup is a major issue in the acceptance of parallel database system. The paper discusses speedup performance of a very large geo-spatial database Extended Sequoia 2000 on ShusseUo. The experiments show good speedup.

Keywords: Very large geo-spatial database, Parallel processing, Extended sequoia2000 benchmark, Speedup

1. はじめに

我々は，NOW (Network Of Workstations) [1]上で稼働する分散並列オブジェクトデータベース「出世魚」[2]の開発を行ってきた．NOWとは，ネットワークによりつながれたワークステーションのことであり，Cluster Computingともいう．「出世魚」の開発では，1テラバイトを超える大規模な地球観測データベースや空間データベースの問合せを，NOWを構成する複数のサイトを使って並列処理し，高い処理性能を得ることを目的としてきた．

並列処理の性能を評価する指標に，スケールアップとスピードアップとの2つがある[3]．1台で1つの問合せ処理を行う場合と，N台でN倍に拡張したデータベースで同じ問合せ処理を行う場合では，応答時間の比は理想的なシステムでは1:1になることをスケールアップという．1台で1つの問合せ処理を行う場合と，N台で同じ1つの問合せ処理を行う場合では，応答時間の比は理想的なシステムでは1:1/Nになることをスピードアップという．

「出世魚」の並列処理の性能評価の一環として，拡張Sequoia2000ベンチマーク[4]を「出世魚」上に実装し，並列処理の性能測定[5][6]を行ってきた．測定の結果，良いスケールアップを得られていることを示すことができたが，スピードアップの性能評価が課題として残っていた．本研究の目的は，拡張Sequoia2000ベンチマークを「出世魚」上で実施し，スピードアップを得ることである．

本論文では，(1) スピードアップの測定のために，「出世魚」のバージョンアップ，データベース設計の修正と問合せの並列処理アルゴリズムの修正を行ったこと，(2) スピードアップを得るために，インデックスの実装を改良したこと，(3) 16倍に拡張した拡張Sequoia2000ベンチマークデータベースを用い，スピードアップの測定を行ったことを報告する．

本論文の構成は以下の通りである．2.で拡張Sequoia2000ベンチマークと「出世魚」について述べる．3.ではスピードアップの測定を行うために行ったデータベース設計と，並列処理アルゴリズムとの修正を説明し，4.ではスピードアップを得るために行ったインデックスの実装の改良を説明する．5.でスピードアップの測定の結果を報告し，6.でまとめる．

2. これまでの経緯

2.1 拡張Sequoia2000ベンチマーク

Sequoia2000は，1990年に始まった米国における地球科学の研究プロジェクト[7][8]である．地球科学や地理情報を扱うエンジニアや科学者が対象とする大規模なデータを扱うデータベースシステムの性能を測定するためのベンチマークとして，Sequoia2000ベンチマーク[9]が提案された．Sequoia2000ベンチマークを，人工的にデータ精度と数とを拡張して行うベンチマークを拡張Sequoia2000ベ

平成12年12月15日受付
* 知能システム学専攻博士後期課程
** 知能システム学専攻修士課程
*** 知能システム学部門

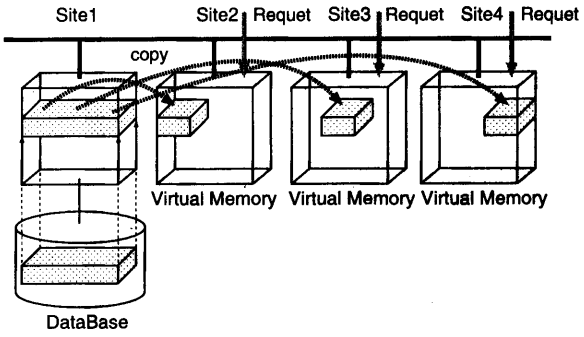

Fig.1　Dynamic Distribution

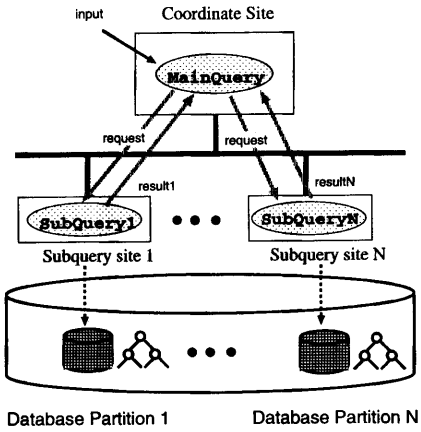

Fig.2　Parallel Processing

ンチマーク[4]と呼ぶ．拡張Sequoia2000ベンチマークは，大規模な地理情報データベースの並列処理を行うデータベースシステムのスケールアップとスピードアップの性能評価を行うために使用されている．

拡張Sequoia2000ベンチマークには11個の問合せ（但し，Query1はデータベース生成）があり，我々は，Non Spatial QueryであるQuery5，Range QueryであるQuery6，Spatial Join QueryであるQuery8とRecursive QueryであるQuery11の4つの実装を終えている．

2.2　分散並列ODBMS「出世魚」

「出世魚」は，3層から成っている[2]．一番最下層は「わかし」であり，ページ単位に分かれたデータ格納空間を提供する．この空間は，永続空間であり，そこに格納されたデータは永続性を持つ（即ち，ディスクに格納される）．一方，データ格納空間はネットワークでつながれた計算機間で「共有」される．2番目の層は「いなだ」である．「いなだ」では，ODMG2.0で定義されているオブジェクト（クラス定義を含む）類とその操作インタフェースを提供している．最上部の層は「わらさ」であり，OQLとVisual Interfaceを提供している．

「出世魚」では，データベースは分散してもいいし，集中していてもよい．各サイトから，他のサイトにあるデータベースを自由にアクセスできる．データベース中のいかなるオブジェクトも，どのサイトからも，読み書きすることが可能である．他のサイトに存在するデータベース中のオブジェクトをアクセスするサイトには，そのオブジェクトのコピーが初めてアクセスするときにコピーが作られる（**Fig.1**）．これを動的分散（dynamic distribution）という．コピーされたデータは各サイトにキャッシュして残り続ける．一方，各サイトがアクセスする予定のデータにより，データベースを分割し，分割したデータベースを各サイトに予め配置することを静的分散（static distribution）という．動的分散は静的分散と比べて，データをコピーするオーバヘッドが生じる．

今回は，動的分散した時のスピードアップの測定のみを行う．スケールアップの性能評価の結果[5)6)]，Hot（詳細は5.で説明）時，動的分散の場合と静的分散の場合では，性能はほぼ同じであるということが示され，「出世魚」の特徴である動的分散の有効性が示されている．

3.　実　　装

3.1　データベースの再設計

「出世魚」での拡張Sequoia2000ベンチマークの実装では，パーティションパラレリズム[3]に従い問合せの並列処理を行う[5)6)]．ラウンドロビン法[3]を用いてデータを分割する．分割したデータへの問合せを部分問合せと呼ぶ．データをN個に分割し，N個のサイトで並列処理を行う場合，N個のサイトにそれぞれ分割したデータを1づつ割り振る（**Fig.2**）．各サイトが同時に，割り振られた分割したデータに対して部分問合せを行う．分割したデータ毎にインデックスを用意し，1個のデータベースパーティション（Database Partition）を作成する．インデックスとして，B+ツリーとR*ツリー[10]を用いる．

スケールアップは，変化するパラメータが使用するサイト数とデータベースの規模であるため，使用するサイト数に合わせデータベースパーティションを作成していた（サイト数が4の時は，パーティションは4個，サイト数が8 であればパーティションは8個作成）．スピードアップは，変化させるパラメータがサイト数のみであるため，並列処理を行うサイト数が様々変わったとしても同じデータベースを使用する必要がある．よって，使用可能な最大サイト数でデータを分割したデータベースパーティションを作成するように，データベースの再設計を行った．

3.2　並列処理アルゴリズムの修正

並列処理を行うモジュールとしてMainQueryとSubQueryを作成した[5)6)]．SubQueryが各サイトで部分問合せを行うモジュールである．SubQueryがアクセスす

るパーティションはある1つに固定されていた．データベースの再設計に伴い，SubQueryが自由にまた複数のパーティションにアクセスできるように並列処理アルゴリズムを修正した．

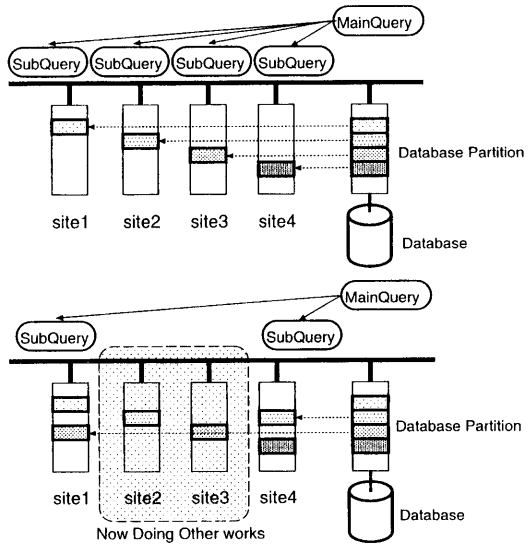

Fig.3 Database Partition

MainQueryは入力として，Query番号と問合せの条件，使用するサイト数を受け取る．使用するサイトに存在するSubQueryに部分問合せの要求を出す．すべてのSubQueryから返事を受け取ると，部分問合せ実行の命令をSubQueryに出す．MainQueryは，SubQueryから部分問合せの結果を受け取りまとめる．SubQueryは，MainQueryから部分問合せの要求を受け取る．受け取るメッセージの内容は，Query番号と問合せの条件と担当するパーティション番号である．部分問合せ実行の命令を受け取ると，部分問合せを行い，結果をMainQueryに返す．MainQuery とSubQueryとの間の通信は，TPC/IPプロトコルで行う．また，MainQueryはOSのマルチスレッドを使って，SubQueryへの要求，結果の受け取りを並行処理している．

使用できる最大のサイト数を4とすると，パーティションは4個用意されている．使用するサイト数が4であれば，各サイトが1つのパーティションを担当する．使用できるサイト数が2であれば，各サイトが2つのパーティションを担当する（**Fig.3**）．SubQueryは複数のパーティションに対する結果をまとめてMainQueryに返す．各パーティションは，各サイトでアクセスがあると動的分散され，各サイトにコピーが作成される．

3.3 その他

最新版の「出世魚」と，前回の測定で使用した「出世魚」との違いは以下の通りである．

- 64bit版「いなだ」のバグの修正したことで，2ギガバイト以上のパーティションを作成できるようになった．
- トランザクション開始のチューニングを行った．トランザクション開始がこれまでより高速化され，性能の絶対値が向上する．

スキーマのメソッド定義にいくつかODMG2.0に準拠していない部分があったため更新を行った．また，スキーマに定義している問合せに使用するメソッドにいくつかに不備があったため正しく修正を行った．

4. インデックスの実装の改良

Sequoia2000ベンチマークのデータには，4つのデータタイプ（point, ploygon, graph, raster）が存在する．polygonは，湖，森林，砂漠や市街地などの土地利用の地図上の形状を表すXY座標（単位はメートル）の集合と土地利用番号から構成される．我々は，polygonの1つデータを次のODLで定義するクラスPolygonのオブジェクトとしてデータベースに格納している．

```
struct PointData {
 d_Int x;
 d_Int y;
};
PolygonBase : d_Object {
 private:
 d_Ref<PointData> pointer;
 public:
 Boxdata getBoundingbox();
 d_UInt size();
};
class Polygon:PolygonBase {
 private:
 d_Int landuse;
 public:
 Polygon();
 Polygon(d_Int landuse,
         d_Ref<Pointdata> pointer);
 Polygon()~
 d_Boolean intersectBox(Boxdata rectanble);
};
```

Query6の定義は，“Find all the polygons that intersect a specific rectangle and store them in the DBMS “である．Query6をOQLで書くと次の通りとなる．

```
select *
from POLYGONS p
where p.intersectBOX(RECTANGLE)
```

Polygonは属性値であるdataに格納されているXY座標の集合が表す形状のMBR （Minimum Bounding

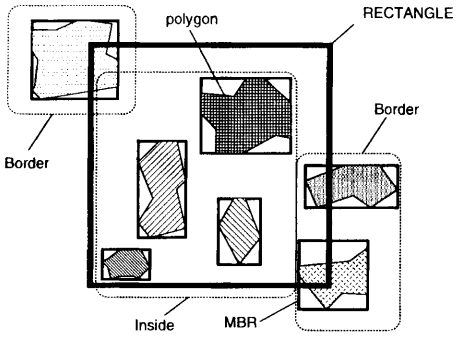

Fig.4　Query6

Rectangle)（**Fig.4**）により，R*ツリーで空間インデックスを作成している．空間インデックスでは，内部のデータ構造の操作を容易にするために，対象のオブジェクトを近似形状で代用する．MBRは近似形状の一つである．Query6の部分問合せは，条件としてRECTANGLEが与えられ，それぞれ担当するパーティションのR*ツリーを走査しRECTANGLEと交差するMBRが示すPolygonへのポインタ（d_Ref）を取り出す（フィルタリング）．RECTANGLEは，正方形であるのでMBRがRECTANGLEの内部に存在するもの（**Fig.4**の**Inside**）は，必ずXY座標の集合が表す形状がRECTANGLEと交差する．RECTANGLEとMBRが交わるようなもの（**Fig.4**の**Border**）は，XY座標の集合が表す形状を調べ（リファイメント），交差するかどうか判定（intersectBoxを使用）する．この時，はじめてデータ部分へのアクセスが行われる．

Query6の部分問合せの大半は，インデックスを使用したフィルタリングとなる．リファイメントが必要な数は，フィルタリングで取り出されたオブジェクトの平均約70分の1であることが分かっている．R*ツリーを使用するQuery8とQuery11にも同様のことがいえる．以上のことをふまえ，R*ツリーの実装を見直した．

R*ツリーの葉ノードには，オブジェクトを参照するために「いなだ」のオブジェクト識別子（OID）とODMG2.0のd_Ref_Anyの2つの格納している（**Fig.5**の上部）．オブジェクト識別子とd_Ref_Anyのサイズはそれぞれ64ビットである．オブジェクト識別子はSubQueryがMainQueryに結果を返すために使用し，d_Ref_Anyはリファイメントに使用していた．これを変更して，d_Refのみを格納するように変更する（**Fig.5**の下部）．d_RefをSubQueryがMainQueryに返す結果とリファイメントとの両方に使用する．

この変更により，R*ツリーの葉ノードのサイズが今までと比べて2分の1となる．各サイトに転送されるデータ量が最大，半分となることが期待される．また，データに対して行うロック獲得の処理にかかるオーバヘッドも

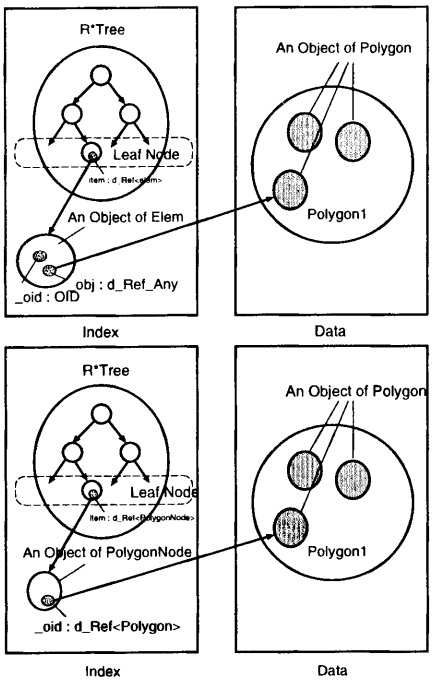

Fig.5　Implementation of R* – Tree

軽減される．スケールアップは応答時間の比が1:1になるかどうかが課題であるため，スケールアップの測定時にはこの違いがサイト数が変わっても影響がなかった．

サイト数を増やした時のスピードアップの効果は，処理を行うCPUの数が増えるだけでなく，扱うデータベースのインデックスやデータがどれだけメインメモリ上に配置できるか，各サイトに転送されるデータ量はどれくらいかということに影響を受ける．扱うインデックスやデータがメインメモリ上にすべて配置できない時，ディスク中のスワップ領域との間にページインとページアウトが起こりディスクI/Oが発生する．データベースシステムにおいてディスクI/Oは性能に関わる．

5. 性　能　評　価

5.1　目的

実装を終えているNon Spatial Query，Range Query，Spatial Join Query，Recursive Queryの4つについて応答時間のスピードアップが得られていることを示す．但し，Non Spatial Queryは，これまでの性能測定において，サイト数が1でも十分高い性能が得られている．問合せ処理以外に必要となるオーバヘッド（通信やスレッドの作成）を考えると，これ以上のスピードアップは期待できない．

5.2　測定環境

測定環境を**Table-1**に示す．Workstation-Bにデータベースを置く．MainQueryをWorkstation-Bで実行し，

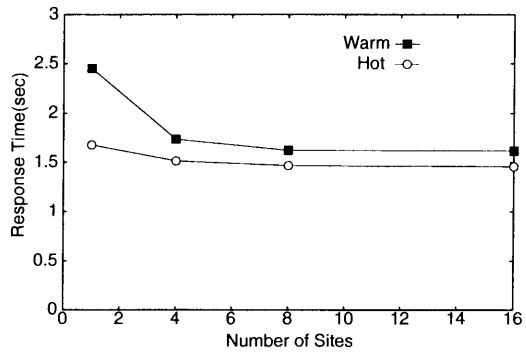

Fig.6 Speedup of Non Spatial Query

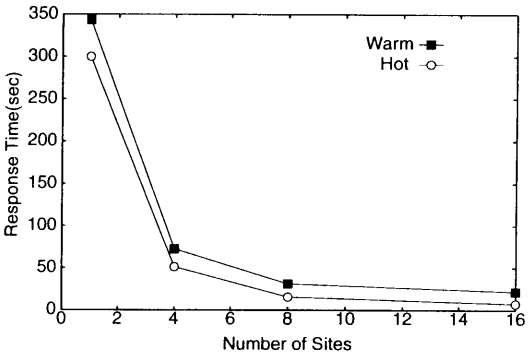

Fig.8 Speedup of Spatial Join Query

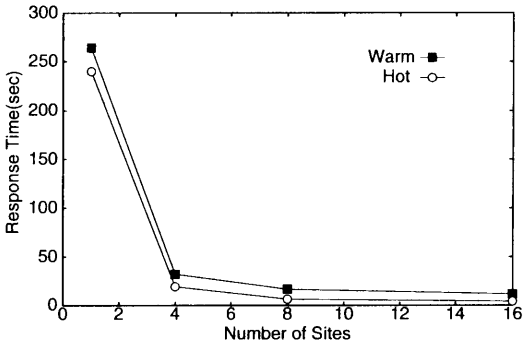

Fig.7 Speedup of Range Query

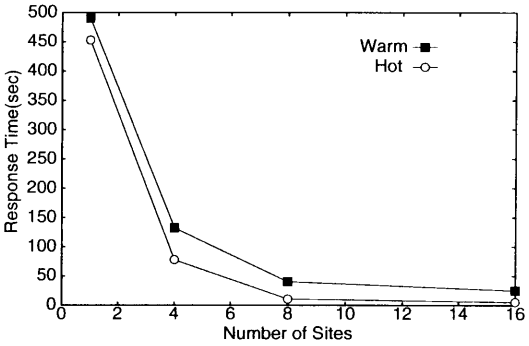

Fig.9 Speedup of Recursive Query

SubQueryをWorkstation-Aを16台使って実行する．使用するデータベースは16倍に拡張した拡張Sequoia2000ベンチマークデータベースで，サイズは約37ギガバイトである．

5.3 内容

Non Spatial Query, Range Query, Spatial Join Query, Recursive Query の 4 つ の 問 合 せ に つ いて応答時間の測定を行う．WarmとHot（Monetプロジェクトチーム[11]によって定義されているもの）の2つの項目で測定を行う．同じ問合せを，問合せの入力値を変えて13回繰り返した時の応答時間の平均時間を取った場合がWarmである．その後，前13回と同じ問合せを行い応答時間の平均時間を取った場合がHotである．Hotの場合，部分問合せを行うサイトに必要なデータがすでに分散配置されているので，データをデータベースが存在するサイトに取りにいく必要がない．サイト数は，1，4，8，16と変化させ測定を行う．

5.4 結果

測定結果を**Table-2**を示す．Non Spatial Query（**Fig.6**），Range Query（**Fig.7**），Spatial Join Query（**Fig.8**），Recursive Query（**Fig.9**）の4つの問い合わせのうち，Range Query，Spatial Join Query，Recursive Queryの3つについて，使用するサイト数が増えるごとに応答時間のスピードアップが得られた．特に，1台と16台を比べると，HotではRange Queryで50倍，Spatial Join Query では40倍，Recursive Queryでは90倍の違いがでている．サイト数が増えた時，応答時間の性能が向上するのは，単にどれだけCPUの数が増えたということだけでなく，どれだけのデータがメインメモリに配置できるかが影響しているしていることを示している．HotとWarmとを比べると，サイト数が多くなるほどHotの方が良い性能を示している．

6. むすびに

今回，新たに拡張Sequoia2000ベンチマークの実装に変更点を加え，16倍に拡張した拡張Sequoia2000ベンチマークを「出世魚」上で実施し，「出世魚」のスピードアップの測定を行った．測定の結果，実装を行っている4つQueryのうち3つのQueryについて，良いスピードアップが得られたという心強い結果が出た．

これからの課題として，未実装のQuery7とQuery10の実装と性能評価，衛星画像であるrasterに関係する4つの問合せ（Query2，3，4，9）の並列処理アルゴリズムの作成，今回の結果を裏付ける様々なデータ（データ転送量，I/Oの数）の収集を行っていきたい．

Table 1 Environment

Workstation A×16	Sun Microsystems ULTRA5 (CPU UltraSPARC IIi 270MHz, Memory 128Mbyte, Disk 22GB Seek Time:9.5ms 7200rpm buffer:512KB)
Workstation B×1	Sun Microsystems ULTRA5_10 (CPU UltraSPARC IIi 440MHz, Memory 1024Mbyte, Disk 34GB Seek Time:9ms 7200rpm buffer:2048KB × 6 RAID 5)
OS	Solaris 8
Network	100Mbps Switching Hub

Table 2 Result

Number of Sites	1	4	8	16
Query	warm/hot(sec)	warm/hot(sec)	warm/hot(sec)	warm/hot(sec)
Non Spatial Query	2.455/1.675	1.737/1.517	1.624/1.467	1.616/1.457
Range Query	264/240	32.381/19.057	16.754/6.522	12/4.083
Spatial Join Query	343/300	72.291/51.249	31.212/15.329	21.620/7.440
Recursive Query	490.416/452.582	132.494/77.604	40.197/10.350	24.830/5.720

参 考 文 献

1) Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW team. A case for now (networks of workstations). *IEEE Micro*, Vol. 15, No. 1, pp. 54–65, 1995.
2) Ge Yu, Kunihiko Kaneko, Guangyi Bai, and Akifumi Makinouchi. Transaction management for a distributed object storage system wakashi - design, implementation and performance. In Stanley Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pp. 460–468. IEEE Computer Society, 1996.
3) David J. DeWitt and Jim Gray. Parallel database systems: The future of high performance database systems. *CACM*, Vol. 35, No. 6, pp. 85–98, 1992.
4) Jignesh M. Patel, Jie-Bing Yu, Navin Kabra, Kristin Tufte, Biswadeep Nag, Josef Burger, Nancy E. Hall, Karthikeyan Ramasamy, Roger Lueder, Curt Ellman, Jim Kupsch, Shelly Guo, David J. DeWitt, and Jeffrey F. Naughton. Building a scaleable geo-spatial dbms: Technology, implementation, and evaluation. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pp. 336–347. ACM Press, 1997.
5) Botao Wang, Taiyong Jin, Keiichi Tamura, kunihiko Kaneko Kenichiro Kimura, and akifumi Makinouchi. Design and implemntation of extended parallel sequoia 2000 benchmark. *Research Reports on Information Science and Electrical Engineering of Kyushu University*, Vol. 5, No. 1, pp. 1–6, 1999.
6) Taiyong Jin, Kunihiko Kaneko, Keiichi Tamura, Akifumi Makinouchi, and Botao Wang. Scalability of dsvm–based parallel object database system shusseuo for very large spatial database system. In *Proceedings of The 7th Australasian Conference on Parallel and Real-Time Systems*, 2000.
7) Michael Stonebraker. The sequoia 2000 project. *Data Engineering Bulletin*, Vol. 16, No. 1, pp. 24–28, 1993.
8) Michael Stonebraker. Sequoia 2000: A reflection of the first three years. In James C. French and Hans Hinterberger, editors, *Seventh International Working Conference on Scientific and Statistical Database Management, September 28-30, 1994, Charlottesville, Virginia, USA, Proceedings*, pp. 108–116. IEEE Computer Society, 1994.
9) Michael Stonebraker, James Frew, Kenn Gardels, and Jeff Meredith. The sequoia 2000 benchmark. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pp. 2–11. ACM Press, 1993.
10) Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In Hector Garcia-Molina and H. V. Jagadish, editors, *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pp. 322–331. ACM Press, 1990.
11) Peter A. Boncz, Wilko Quak, and Martin L. Kersten. Monet and its geographic extensions: A novel approach to high performance gis processing. In Peter M. G. Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, Vol. 1057 of *Lecture Notes in Computer Science*, pp. 147–166. Springer, 1996.