九州大学学術情報リポジトリ
Kyushu University Institutional Repository

# ラウンドロビン分割法による分散地理データベースにおける並列Spatial Joinの高速化

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

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/10.15017/1515737

出版情報：九州大学大学院システム情報科学紀要. 6 (2), pp.203-208, 2001-09-26. 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2号　平成13年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.6, No.2, September 2001

# ラウンドロビン分割法による分散地理データベースにおける並列Spatial Joinの高速化

田村慶一* ・金子邦彦** ・牧之内顕文**

# Speed-up of Parallel Spatial Join on Distributed Geographical Databases using Round Robin Partition

Keiichi TAMURA, Kunihiko KANEKO and Akifumi MAKINOUCHI

(Received June 15, 2001)

**Abstract:** The combination of spatial data sets based on spatial relationships between objects is a common operation in geographical databases. This operation is called spatial join operation. The spatial join operation, as with all join operations of relational databases has high computing time cost. It is natural to apply parallelism to the spatial join operation. In this paper, we focus on the parallel processing of spatial join when a geographical database is partitioned by round-robin algorithm. We propose two algorithms of the Filter Step in parallel join processing based on a round-robin algorithm. One of the algorithm uses a R*-tree of all data sets, the other uses a set of the R*-tree each partition owns. We implement proposed algorithm on Query10 of Extended Sequoia 2000 benchmark and compare the Filter Step performance of two algorithms.

**Keywords:** Parallel processing, Spatial join, Distributed geographical database, Round robin

## 1. はじめに

地理データベースでは，2つの空間データの集合を空間的な関係で結び付ける演算を扱う．この演算をSpatial Joinという[1]．「交差する川と道路のすべてのペアを求める」や「森林地帯に存在するすべての降雨観測地点の降水量を求める」といったものがSpatial Joinの例として挙げられる．「交差する川と道路のすべてのペアを求める」では，川と道路の空間データの中で「交差する」という空間的な関係で結び付く川と道路のすべてのペアを求める．本論文では，並列Spatail Joinの高速化について扱う．

Spatial Joinはインデックスの有無で次の3つに分類できる．Spatial Joinの2つの入力について，(1)2つの入力のいずれにも空間インデックスが存在する場合，(2)2つの入力のうちどちらか一方に空間インデックスが存在する場合，(3)2つの入力のいずれにも空間インデックスが存在しない場合の3つである．(2)には，空間データを何らかの条件でselectionした結果を，他の空間データとSpatial Joinするような場合が当てはまる．「ある土地利用番号で分類されるpolygonの内部に存在するすべてのpointを求める」といった問い合わせが例としてあげられる．本研究では，(2)の場合のSpatial Joinの並列処理を扱う．従って以下では，単にSpatial Joinと書いて(2)の場合を意味する．

一般に(本研究でも)空間インデックスとしてR*-tree[2]が使われている．R*-treeでは，空間データは近似形状MBR[2]で表現される．MBRとは空間データを囲む最小の長方形である．MBRを使用したSpatial Joinは次の2つのステップで構成される．

- 空間データのMBRが空間的な関係を満たすすべての空間データのペアを結果の候補として取り出す．この処理ステップをFilter Stepと呼ぶ．
- Fileter Stepで取り出した結果の候補のそれぞれについて，実際の形状を取り出して，空間的な関係を実際に満たさないものを結果の候補から取り除く．この処理ステップをRefinemet Stepと呼ぶ．

ここ数年，地理データベースでは十年前に比べて数百倍の空間データを扱う[3]ようになり，並列Spatial Joinの研究が幾つか[3]行われてきた．最も有名なものがParadiseプロジェクト[3)4)]である．Paradiseでは，partition parallelism[5]を用いてSpatial Joinの並列処理を行う．partition parallelismではまず，データベースを複数に分割する．分割したデータベースをパーティションと呼ぶ．パーティション1つを並列処理を行う計算機(以下サイトと呼ぶ)に割り振る．各サイトは割り振られたパーティションに対して問い合わせを実行する．各サイトの結果を集めて問い合わせの結果とする．

地理データベースをpartition parallelismを用いて並列処理する場合，空間データをどう分割するかというとい

平成13年6月15日受付
* 知能システム学専攻博士後期課程
** 知能システム学部門

うことと，どのような手法で並列処理するかが課題となる．Paradiseでは分割法として空間分割法を提案している．Spatial JoinのFilter StepではR*-treeを用いる．Paradiseでは，パーティション毎にR*-treeを作成し，Spatial JoinのFilter Stepに使用している．空間分割法については2.で説明を行う．空間分割法は，2つの空間データを「交差する」という空間的な関係でSpatial Joinする場合，各サイトでは片方のデータの1つパーティションと，もう一方のデータの1つのパーティションデータ同士を比較してFilter Stepを行う[6]．

我々は，partition parallelismを用い地理データベースの並列処理の研究を行ってきた[7]．我々はデータ分割法としてラウンドロビン法を用いる．ラウンドロビン法でデータを分割すると，2つの空間データを「交差する」という空間的な関係でSpatial Joinする場合，各サイトではそこにある1つのパーティションとの比較を行う．

本研究の目的は，ラウンドロビン法で空間データを分割している場合の並列Spatial JoinのFilter Stepの高速化である．Filter Stepにおいて，各パーティションごとに作成したR*-treeを使用する(方式(1)とする)か，データ全体で作成したR*-treeを使用する(方式(2)とする)かという2つの選択がある．空間データを何らかの条件でselectionした結果を，他の空間データとSpatial Joinするような場合，方式(1)では各サイトはselectionしたすべての結果と，そのサイトが担当するパーティションのR*-treeを比べる．方式(2)では，各サイトはselectionした結果とすべてのデータのR*-treeと比べる．2つの方式のどちらが高速かは，条件や入力の数によって変わるが，よりベストな方法を選択できるように，処理のステップ数で判断したい．

本論文の構成は以下の通りである．2.で関連研究としてParadiseのSpatial Join の並列処理について説明する．3.ではラウンドロビン法を用いたSpatial Join並列処理の処理方式について提案し，4.ではをSequoia2000ベンチマークのQuery10 を用いて詳細処理ステップを説明する．5.では提案した方式について処理ステップ数の比較を行う．6.では実際にSequoia2000ベンチマークのQuery10を用いた性能測定を報告し，7.でまとめる．

## 2. 関 連 研 究

空間分割法でのデータ分割手順の概要は次の通りである．

1. 領域全体を複数のタイル (Tile) に分割する．
2. 空間データを各タイルに分割していく．複数のタイルにまたがる空間データについては，それら複数のタイルに複製 (Replication) を作成する．
3. 各タイルをラウンドロビン法またはhash法で各パーティションに割り振る．

**Fig.1**ではタイル数は4つである．**Fig.1**を2つのパーティションに分割する．polygon1はタイル1とタイル2に渡っているためpolygonのパーティション1とパーティション2に割り振られる (**Fig.2**の(a))．polygon2はタイル3の内部に存在するのでpolygonのパーティション1に割り振られる(**Fig.2**の(a))．同様にpoint1，point3とpoint4はpoint のパーティション1にpoint2はpointのパーティション2に割り振られる．分割したパーティションを各サイトに1づつ割り振る．

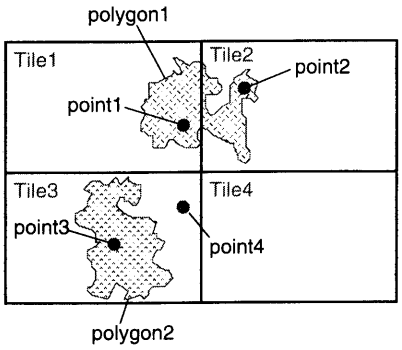


**Fig.1**　Example of spatial data sets.

Sequoia2000ベンチマーク[8]のQuery10では，2つの空間データpolygonとpointを扱う．polygonは属性値として土地利用番号(landuse)を持ち，pointは属性値として名前(name)を持つ．ある土地利用番号のpolygonを効率的に求めるためにpolygonの各パーティションでは，予め土地利用番号をkey値としたB+ツリーが用意されているものとする．また，パーティション毎にR*-treeを作成する．Query10のOQL(Object Query Language)文は以下のようになる．

```
select point.name
from   polygon in Polygons, point in Points
where  polygon.landuse = LANDUSE and
       polygon.include(point)
```

Query10のFilter Stepの概要は以下のようになる．各サイトでは，polygonのパーティションからある土地利用番号のpolygonのオブジェクト識別子を求める．求めたpolygonのオブジェクト識別子に対して，polygonのMBRがpointのパーティションに格納されているpointのMBRと交差するかどうかpointのR*-treeを使って調べ，交差するpointのオブジェクト識別子を求める．複製を作成しているために，結果の候補に重複が存在する可能性がある．最後に，各サイトの結果を集め重複を取り除く．

空間分割法の特徴は複製である．各サイトに配置されたタイルについて，タイルと重なり合うようなすべての空間データが格納されている．polygonとpointの1つのタイルをタイルごとに比べるだけでSpatial JoinのFilter

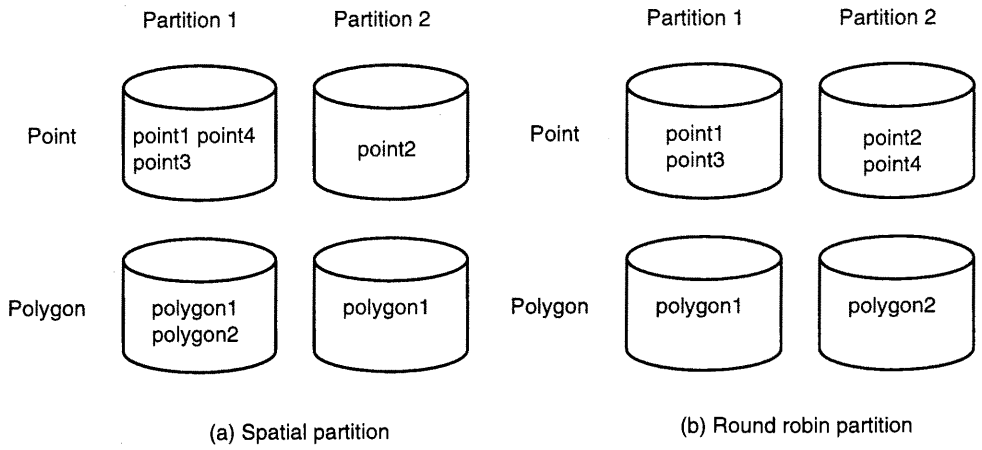


**Fig.2** Example of spatial partition and round robin partition.

Stepを並列に実行できる．但し，交差以外の条件，例えば「間の距離が100km」であるといったSpatial Joinでは，他のサイトに配置されているタイルを必要とする．

## 3. ラウンドロビン法を用いたSpatial Join並列処理

ラウンドロビン法で，**Fig.1**のデータを2つのパーティションに分割すると，polygonのパーティション1にpolygon1，パーティション2にpolygon2が，pointのパーティションにpoint1とpoint3，パーティション2にpoint2とpoint4が割り振られる．交差するpolygonとpointをすべて求めるにはpointのパーティション1はpolygonのパーティション1とpolygonのパーティション2を比べなければすべての結果を取り出すことができない．ラウンドロビン法では，どのような空間的な関係であっても他のサイトに配置されているパーティションを必要とする．

地理データベースにおけるSpatial Joinでは，2つの空間データの集合で「交差するものすべてを取り出す」以外に，「2つの空間データ間の距離がある距離であるもをすべてを取り出す」や「2つの空間データが接するものをすべて取り出す」といった演算を扱う．いずれにしてもラウンドロビン法では，一方の空間データのパーティションと他方の空間データのパーティションすべてとを比較してFilter Stepを行う．従って各パーティションごとのR*-treeとデータ全体のR*-treeが必要になる．

Filter Stepを実装するのに，パーティション毎に作成したR*-treeを使用する(方式(1))かまたは，分割していないデータから作成したR*-treeを使用する(方式(2))かで2つの方式が考えられる．Query10を方式(1)と方式(2)で実装すると以下のようになる(**Fig.3**に違いを示す)．

**方式(1)** 各サイトがB+-treeから取り出したすべてのpolygonを一端集め，各サイトに再分配して，取り出したすべてのpolygonについて，1つのpointのパーティションのR*-treeを使ってpolygonのMBRと交差するpointを取り出す．

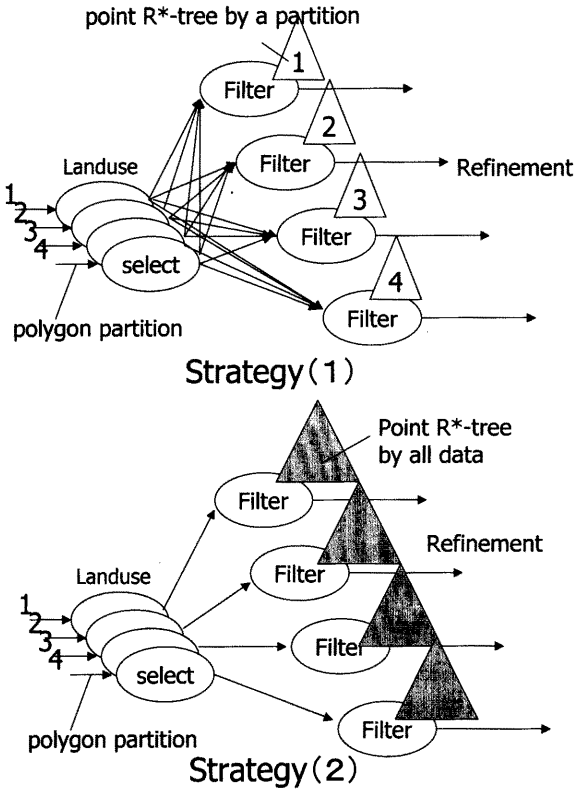


**Fig.3** Strategy(1) and Strategy(2)

**方式(2)** 各サイトB+-treeから取り出したpolygonと，pointデータ全体のR*-treeを使ってpolygonのMBRと交差するpointを取り出す

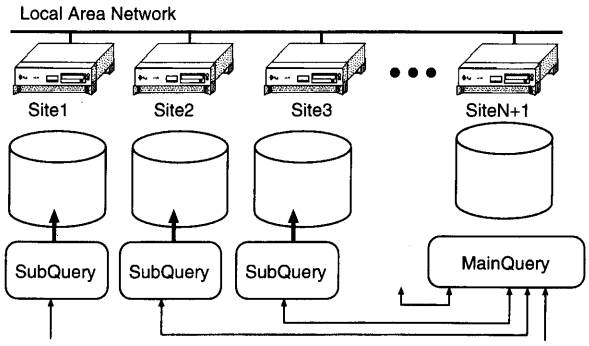


**Fig.4**　MainQuery and SubQuery.

## 4. 並列処理方式

並列処理を行うモジュールはMainQueryとSubQueryである[7]．**Fig.4**のように，N+1台で並列処理をする場合，N台の計算機にSubQueryがそれぞれ配置され，1台にMainQueryが配置される．MainQueryは入力として検索の条件を受け取り，サイト1からサイトNのSubQueryに要求を出す．すべてのMainQueryは，SubQueryから結果を受け取りまとめる．SubQueryは，MainQueryからの要求を受け取ると，自サイトが担当するパーティションに対して処理を行い，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．

Sequoia 2000ベンチマークのQuery10の処理ステップは方式(1)，方式(2)ともに次の7ステップからなる．

- 方式(1)
  - **(1)** MainQueryはSubQueryに土地利用番号を送る．
  - **(2)** 各サイトのSubQueryは，自サイトが担当するpolygonのパーティションのB+-treeを使って，ある土地利用番号のpolygonのオブジェクト識別子を取り出す．
  - **(3)** MainQueryは各サイトが求めたpolygonのオブジェクト識別子を集める．
  - **(4)** MainQueryは集めたpolygonのオブジェクト識別子をすべて各サイトのSubQueryに送る．
  - **(5)** 各サイトのSubQueryは受け取ったすべてのpolygonのオブジェクト識別子に対して次の処理を行う．自サイトが担当するpointのパーティションのR*-treeを使って，polygonのMBRと交差するpointのオブジェクト識別子を求める．
  - **(6)** 各サイトのSubQueryは(5)で求めたpolygonとpointのオブジェクト識別子のペアについて，pointがpolygonの内部に存在するか調べる．もし，polygon の内部にpointが存在するならば，pointの名前を求め，結果に格納する．
  - **(7)** MainQueryは各サイトが求めた結果を集める．
- 方式(2)
  - **(1)** MainQueryはSubQueryに土地利用番号を送る．
  - **(2)** 各サイトのSubQueryは，自サイトが担当するpolygonのパーティションのB+-treeを使って，ある土地利用番号のpolygonのオブジェクト識別子を取り出す．
  - **(3)** 各サイトのSubQueryは(2)で取り出したpolygonのオブジェクト識別子に対して次の処理を行う．point全体のR*-treeを使って，polygonのMBRと交差するpointのオブジェクト識別子を求める．
  - **(4)** MainQueryは各サイトが求めた結果の候補を集める．
  - **(5)** MainQueryは集めた結果の候補をすべて各サイトのSubQueryに送る．
  - **(6)** 各サイトのSubQueryは(5)で受け取ったpolygonとpointのオブジェクト識別子のペアについて，pointがpolygonの内部に存在するか調べる．もし，polygon の内部にpointが存在するならば，pointの名前を求め，結果に格納する．
  - **(7)** MainQueryは各サイトが求めた結果を集める．

## 5. Filter Stepの処理ステップ数の比較

方式(1)と方式(2)のFilter Stepの処理ステップ数は，Spatial Joinの2つの入力のうちどちらを分割して処理を行っているかと2つの入力の数によって違いが発生する．使用するすべてのデータはメインメモリ上にすでに載っているものとする．例えば，B+-treeから取り出したpolygonの個数をMとする．pointの総数をNとする．サイト数をsとする．1台でFilter Stepを実行したとすると，Filter Stepの処理ステップ数は，

$$Step0 = M \times f_R(N)$$

となる．$f_R(N)$は$N$個のデータを登録しているR*-treeを検索するために必要な平均ステップ数である．方式(1)では各サイトの平均ステップ数は，

$$Step1 = M \times f_R(N/s)$$

となる．方式(2)では各サイトの平均ステップ数は，

$$Step2 = M/s \times f_R(N)$$

となる．$s > 1$で，$f_R(N) > f_R(N/s)$であるので，

$Step0 > Step1$
$Setp0 > Step2$

となり，Filter Stepの処理ステップは方式(1)と方式(2)ともに1サイトで実行するよりも1つのサイトのステップ数が減少する．各サイトのステップ数の合計を取ると，

$$Sum_Step1 = s \times M \times f_R(N/s)$$
$$Sum_Step2 = M \times f_R(N)$$

となる．少なくとも$f_R(N) < N$であるため，$s \times f_R(N/s) > f_R(N)$である．すべてのサイトのステップ数の合計は方式(1)よりも方式(2)の方が少ない．よって，各サイトのステップ数は方式(1)よりも方式(2)の方が少ない．但し，selectionされる数が少ない時，例えばM=1の時，方式(1)の方がステップ数が小さくなる．

空間分割法でのFilter Stepの処理ステップのステップ数は

$$Step3 = \alpha \times M/s \times f_R(\beta \times N/s)$$

となる．但し，各パーティションのデータ数が理想的に均一になっており，複製のために$\alpha$倍と$\beta$倍にデータが増加しているとする．

$$Sum_Step3 = \alpha \times M \times f_R(\beta \times N/s)$$

ここで，$f_R(N)$の平均値は木の高さに比例すると仮定し，$f_R(N)$を$f_R(N) = C \times log_m N$と近似する．mはR*-treeのノードのエントリの最大数であり，Cは定数である．

$$Sum_Step2 = M \times C \times log_m N$$
$$Sum_Step3 = \alpha \times M \times C \times log_m(\beta \times N/s)$$

$m$は約100ぐらいで，$s$は最大でも数百台であると考えると，$\alpha \times log_m(\beta \times N/s) \simeq log_m N$となり，処理ステップ数は空間分割法よりも小さくなる．

## 6. 性 能 測 定

方式(1)と方式(2)の比較を行うために，Filter Stepの処理時間の計測を行う．測定には拡張Sequoia 2000ベンチマークのQuery10を使用し，分散オブジェクトデータベースシステム「出世魚」[9]上に実装した．

### 6.1 測定条件

実験環境を**Table 1**に示す．また，使用するデータは拡張Sequoia2000ベンチマークデータで，polygonとpointのデータ数を**Fig.1**に示す．使用したサイト数は4で，パーティションの数も4つである．データベースは1つのサイトのディスクに格納されている．一度，Queryを実行すると，「出世魚」の動的分散機能により各サイトにデータが分散し，各サイトにデータがキャッシュされる．Queryを一度実行し，もう一度同じパラメータで実行した時のSubQueryでのFilter Stepの処理時間の平均を測定する．参考のためにMainQueryの応答時間とSubQueryでのRefinement Stepの処理時間の平均を測定する．一度Queryを実行すると，データはすでに各サイトに分配され各サイトにキャッシュされている．

Queryは4つパラメータを変えて測定する．使用するパラメータは，土地利用番号11，21，24と91である．それぞれの土地利用番号のpolygonの数を**Table 3**に示す．

**Table 2** Extended sequoia 2000 benchmark polygon and point data.

| Data Type | Number of objects | Average size |
|---|---|---|
| point | 306336 | 24byte |
| polygon | 374428 | 204byte |

**Table 3** Parameter of query.

| Landuse | Number of polygons |
|---|---|
| 11 | 31892 |
| 21 | 17552 |
| 24 | 8108 |
| 91 | 48 |

**Table 4** Result of landuse 11.

| | Filter | Refinement | Response time |
|---|---|---|---|
| (1) | 20.90(sec) | 8.25(sec) | 29.87(sec) |
| (2) | 7.58(sec) | 8.04(sec) | 16.69(sec) |

**Table 5** Result of landuse 21.

| | Filter | Refinement | Response time |
|---|---|---|---|
| (1) | 12.99(sec) | 19.46(sec) | 32.98(sec) |
| (2) | 5.45(sec) | 19.79(sec) | 25.28(sec) |

### 6.2 結果

土地利用番号11の結果を**Table 4**，21の結果を**Table 5**，24の結果を**Table 6**に91の結果を**Table 7**に示す．土地利用番号11，21と24ではpolygonの数が多いため，方式(1)よりも方式(2)の方が各サイトのFilter Stepの処理時間が小さい．処理時間が約半減している．土地利用番号91の数は少ないため，方式(1)と方式(2)との差がほとんどない．

**Table 6** Result of landuse 24.

| | Filter | Refinement | Response time |
|---|---|---|---|
| (1) | 7.57(sec) | 0.80(sec) | 8.80(sec) |
| (2) | 3.05(sec) | 0.93(sec) | 4.31(sec) |

**Table 1** Environment.

| | |
|---|---|
| Workstation A×4 | Sun Microsystems ULTRA/5 (CPU UltraSPARC IIi 270MHz, Me mory 128Mbyte, Disk 22GB Seek Time:9.5ms 7200rpm buffer:512KB) |
| Workstation B×1 | Sun Microsystems ULTRA/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 |
| DBMS | ShusseUo |

**Table 7** Result of landuse 91.

| | Filter | Refinement | Response time |
|---|---|---|---|
| (1) | 0.14(sec) | 0.18(sec) | 0.79(sec) |
| (2) | 0.096(sec) | 0.21(sec) | 0.66(sec) |

## 7. おわりに

本研究では，ラウンドロビン法を用いた時のSpatial Join並列処理のFilter Stepを行うための2つの方式を提案した．方式(2)と方式(1)とをFilter Stepの各サイトの処理時間で比較を行った結果，方式(2)は方式(1)よりも，selectionされる数が多い時，つまりR*-treeが存在しない方のSpatial Joinの入力の数が多い時に，高速化が可能であることを示すことができた．

## 参考文献


1) Ralf Hartmut Güting. An introduction to spatial database systems. *VLDB Journal*, Vol. 3, No. 4, pp. 357–399, 1994.
2) Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pp. 322–331, 1990.
3) 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 *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pp. 336–347, 1997.
4) David J. DeWitt, Navin Kabra, Jun Luo, Jignesh M. Patel, and Jie-Bing Yu. Client-server paradise. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pp. 558–569, 1994.
5) David J. DeWitt and Jim Gray. Parallel database systems: The future of high performance databas e systems. *CACM*, Vol. 35, No. 6, pp. 85–98, 1992.
6) Jignesh M. Patel and David J. DeWitt. Clone join and shadow join : Two parallel spatial join algorithms. In *Proceedings of the ACM GIS 2000 Washington D.C November 2000*, pp. 55–65, 2000.
7) Botao Wang, Taiyong Jin, Keiichi Tamura, kunihiko Kaneko Kenichiro Kimura, and akifumi Makinouchix. 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.
8) Michael Stonebraker, James Frew, Kenn Gardels, and Jeff Meredith. The sequoia 2000 benchmark. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pp. 2–11, 1993.
9) Ge Yu, Kunihiko Kaneko, Guangyi Bai, and Akifumi Makinouchi. Transaction management for a distributed object storage system wakashi - design, implementation and performance. In *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pp. 460–468, 1996.