Partition Based Spatial-Merge Join[a]

Jignesh M. Patel and Davis J. DeWitt

[a] In *Proceedings of the SIGMOD International Conference on Management of Data, page259-270, 1996.*

- Partition Based Spatial-Merge Join(PBSM) algorithm
- Performance study

Introduction

- Spatial Database System
- *Spatial Join* Operation
  - Filter Step
    * MBR(Minimum Bounding Rectangle)
    * produces *candidates* of the actual result
    * I/O bound
  - Refinement Step
    * checks the candidates
    * CPU bound

Introduction (Cont.)

- Numerous algorithms focus on filter step
  - many of the earlier algorithms are based on transforming an approximation of a spatial object into another domain.
  - most of the newer algorithms require spatial index.
- PBSM Join does not need indices on either of its inputs

Introduction (Cont.)

- Perfomance study on three spatial join algorithms
  - a simple indexed nested loops based join algorithm(NLJ)
  - an R-tree based join algorithm(RBJ)
  - the PBSM algorithm

PBSM Join

- filter step
  - using MBR
  - R and S denote two inputs to join and are a sequence of tuples
  - each tuple has a spatial attribute to be used in join condidtion and an identifer, OID.

PBSM Join(Cont.)

- filter step(cont.)
  - *plane-sweeping* technique
  - outputs, $< OID_R, OID_S >$
  - spatial partitioning function

PBSM Join(Cont.)

- refinement step
  - sorts the result in filter step by using OID_R as the primary sort key and OID_S as the second sort key
  - reads the OID_R part into memory, the read OID_S part sequnetially.

Spatial Partitioning Function

- *universe*
- number of partitions $P = \lceil \frac{(||R||+||S||)\times Size}{M} \rceil$, where M is the memory size.
- spatial partitioning skew
- *tiles*

Spatial Partitioning Function(Cont.)

- two issues obout choosing the spatial partitioning function:
  - number of tiles: The effect is small(5%)
  - scheme for tile-partition mapping: Round Robin

Performance Evaluation

- Indexed Nested Loops Join(NLJ)
  - R and S are two inputs with $||R|| < ||S||$
  - an index on R will be built if neither input has an index on the joining attribute.
  - each tuple of S is used to probe the index on R to get the result
  - using "bulk loading" [b] to build an index.

[b] D.J. DeWitt, N. Kabra, J. Luo, J.M. Patel, and J.Yu. "Client-Server Paradise", In *Proceedings of the 20th VLDB conf.*, September 1994.

Performance Evaluation(Cont.)

- R-tree Based Join Algorithm(RBJ)[c]
  - using bulk loading to build indices on the two inputs, R and S.
  - traverse starts from the roots of the two R-trees and move down in tandem until the leaf nodes are reached (depth-first fashion).

[c] T. Brinkhoff, H.P. Kriegel, and B. Seeger. "Efficient Processing of Spatial Joins Using R-trees", In *Proceedings of the 1993 ACM-SIGMOD Conf.*, May 1993

Performance Evaluation(Cont.)

- Methodology
  - implementing three algorithms in Paradise [b]
  - Paradise is a database system that handle GIS type of applications

[b] D.J. DeWitt, N. Kabra, J. Luo, J.M. Patel, and J.Yu. "Client-Server Paradise", In *Proceedings of the 20th VLDB conf.*, September 1994.

- Methodology(Cont.)
  - Sun SPARC-10/51 with 64MBytes of memory.
  - running SunOS Release 4.1.3.
  - One Seagate 2GBytes disk was used as a row device to hols database.
  - the log for the system was kept on a second 2GByte Seagate disk.

- Methodology(Cont.)
  - two main parts:
    - Join with no indices
    - Join with indices
  - three collections of data sets:
    - TIGER/Line [d] file for the State of Wisconsin.
    - sorting spatially the objects in TIGER data
    - Sequoia 2000 Storage Benchmark [e] .

---

[d] U.S. Bureau of the Census, Washington DC. *"TIGER/Line File(TM), 1992 Technical Documentation"*.

[e] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith, "The SEQUOIA 2000 Storage Benchmark". In *Proceedings of the 1993 ACM-SIGMOD Conf.*, May 1993

Performance Evaluation(Cont.)

- Joins with no pre-existence of the indeices
  - NLJ algorithm builds an index on the smaller input
  - RBJ algorithm builds the indices for both inputs
  - Summmary:
    * overall, PBSM has better performance than the other two
    * NLJ performs better than RBJ when the difference of the sizes of the two inputs is significant.
    * perfomances improve when the inputs are clustered.

Performance Evaluation(Cont.)

- Joins with pre-existence of the indices
  - If one of inputs has an index:
    * NLJ probes that index
    * RBJ builds an index on the other input
  - If both inputs have indices:
    * NLJ probe the smaller index
    * RBJ skips building any index
  - Summary: PBSM has a better performance if an index exists only on the smaller input. Otherwise, RBJ is better.

Performance Evaluation(Cont.)

- CPU costs
  - CPU cost dominates the I/O cost
  - The reasons:
    - performing spatial operations like probing an R-tree index, joining partitions using a plane-sweeping strategy, and spatial sorting during bulk loading an index are computational intensive.
    - the storage manages works hard at minimizing the I/O costs.

Conclusions

- providing an algorithm, PBSM, for spatial joins which does not need index in advance.
- not only on filter part, but also on refinement part.
- choosing partitioning function is heuristic
- handling the partition skew – not well done