5R−1

Evaluation Results of Multi-Way Joins in Shared-Nothing Database Environment

- The Case for Left-Deep and Right-Deep Execution Query Trees -

Lilian Harada and Naoki Akaboshi
Fujitsu Laboratories*

1. Introduction

In this paper we introduce a performance modeling of multi-way joins represented in the left and right-deep tree structures, in a shared-nothing environment. The structure of the query execution tree and the balance of the utilization of the system resources are important factors in the efficient parallel processing of multi-way joins. In the following we present our analytical model to determine their elapsed time, showing the effects of resource utilization (amount of main memory, disk and interconnection network bandwidth), and identifying resources bottlenecks that limit their performance.

2. Inter-Operator Parallelism in Multi-Way Joins

The following descriptions are concentrated on the inter-operator parallelism of multi-way joins, using the hash-join algorithm for each operator. Although not explicitly referenced, intra-operator parallelism described in [1], [2] is always applied.

The execution of a multi-way join of N relations on different N-1 join attributes, i.e., $R_1 \bowtie_{a_1} R_2 \bowtie_{a_2} \ldots \bowtie_{a_{N-2}} R_{N-1}$, can be denoted by left-deep and right-deep structures of query execution trees, as shown in Fig.1. Each of the query trees follow the convention that the left and right descendants of a join operator represent the building and probing relations, respectively.

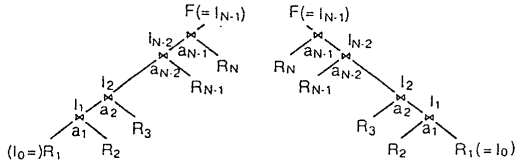

Fig. 1 Left and Right-Deep Trees

Table 1 and 2 resumes the active operators, the operators phases and their processing time for left and right-deep trees illustrated in Fig. 1, respectively, composed of scan/selection operations of the N input relations ((σR_i), for i=1,...,N), N-1 hash join operators ($(I_{i-1} \bowtie_{a_i} R_{i+1})$, for i=1,...,N-1), and a write operator of the final result F. From Table 1, we can see that, concerning the inter-operator parallelism, the total execution of a left-deep query tree is composed of the execution of a first stage of degree 2 and (N-1) stages of degree 3 operators.

From Table 2, we can see that a right-deep tree can support N active operators, but there are potential problems concerning memory utilizations. When the amount of memory is not enough to accommodate the N-1 hash tables of R_i, for i=2,...,N, the right-deep tree can be decomposed into disjoint segments such that the hash tables of the active operators in each segment fit into memory. The segments are executed one by one bottom up with all resources in the system devoting to one segment at a time. The last join operator in each segment have to spool its tuple production to a temporary relation in the disk, which will be the probing relation of the next segment. Therefore, as the number of segments increases, the total intermediate data required to be written and read from the disk increases.

Stage	Operators	Operator phases	Phase time
1	σ R_1	scan R_1 read	$T_{diskR}(R_1)$
	$R_1 \bowtie R_2$	build R_1 split	$T_{net}(R_1)$
		R_1 build	$T_{cpuB}(R_1)$
i ($2 \le i \le N-1$)	σ R_i	scan R_i read	$T_{diskR}(R_i)$
	$I_{i-2} \bowtie R_i$	probe R_i split	$T_{net}(R_i)$
		R_i probe	$T_{cpuP}(R_i)$
	$I_{i-1} \bowtie R_{i+1}$	build R_{i+1} split	$T_{net}(R_{i+1})$
		R_{i+1} build	$T_{cpuB}(R_{i+1})$
N	σ R_N	scan R_N read	$T_{diskR}(R_N)$
	$I_{N-2} \bowtie R_N$	probe R_N split	$T_{net}(R_N)$
		R_N probe	$T_{cpuP}(R_N)$
	Write F	write F write	$T_{diskW}(F)$

Table 1 Stages in Left-Deep Tree Processing

Stage	Operators	Operator phases	Phase time
i ($1 \le i \le N-1$)	σ R_{i+1}	scan R_i read	$T_{diskR}(R_i)$
	$R_{i+1} \bowtie I_{i-1}$	build R_i split	$T_{net}(R_i)$
		R_i build	$T_{cpuB}(R_i)$
N	σ R_1	scan R_1 read	$T_{diskR}(R_1)$
	$R_{j+2} \bowtie I_j$ ($0 \le j \le N-2$)	probe I_j split	$T_{net}(I_j)$
		R_j probe	$T_{cpuP}(I_j)$
	Write F	write F write	$T_{diskW}(F)$

Table 2 Stages in Right-Deep Tree Processing

3. Performance Modeling

Here we present our analytical model of a N-way join on a shared-nothing system composed of PE nodes. Each stage processing time is determined by overlapping the CPU computation, disk transfer, and interconnection network transfer times, shown in Tables 1 and 2. The total time is the summation of all these stage times.

In order to simplify our analysis, we don't consider the CPU computation when overlapped with the disk and network transfer times. We also assume that all source and intermediate relations have the same size $||R||$ bytes,

*e-mail address:{lilian. akaboshi}@flab.fujitsu.co.jp

which are declustered across all PE nodes, and the available aggregate memory $||M||$ is larger than 2 $||R||$. With these simplifications, and using the notation ω_{ioRs}, ω_{ioRr} and ω_{ioWr} for sequential read, random read and random write I/O bandwidths, respectively, we have that :

$$T_{diskR}(R) = \frac{||R||}{PE \cdot \omega_{ioR_r}}$$

$$T_{diskR}(R) + T_{diskW}(R) = \frac{||R||}{PE \cdot \omega_{ioR_r}} + \frac{||R||}{PE \cdot \omega_{ioW_r}}$$

$$T_{net}(R) = \frac{PE-1}{PE} \cdot \frac{||R||}{PE \cdot \omega_{comm}}$$

Therefore, the total elapsed time for the left and right-deep trees becomes :

- Left-Deep Tree

$$T = \frac{||R||}{PE} \cdot \Big\{ max(\frac{1}{\omega_{ioR_s}}, \frac{PE-1}{PE} \cdot \frac{1}{\omega_{comm}}) + (N-2) \cdot max(\frac{1}{\omega_{ioR_s}}, \frac{PE-1}{PE} \cdot \frac{2}{\omega_{comm}}) + max(\frac{1}{\omega_{ioR_r}} + \frac{1}{\omega_{ioW_r}}, \frac{PE-1}{PE} \cdot \frac{1}{\omega_{Comm}}) \Big\}$$

- Right-Deep Tree
  When $||M|| \geq (N-1) \cdot ||R||$, the total elapsed time is simply determined as the summation of the stages elapsed times in Table 2. However, when $2 \cdot ||R|| \leq ||M|| < (N-1) \cdot ||R||$, there are two cases.

case 1

$$T = (N-1) \cdot \frac{||R||}{PE} \cdot max(\frac{1}{\omega_{ioR_s}}, \frac{PE-1}{PE} \cdot \frac{1}{\omega_{comm}}) + \frac{(N-1)}{\lfloor \frac{||M||}{||R||} \rfloor} \cdot \frac{||R||}{PE} \cdot max(\frac{1}{\omega_{ioR_r}} + \frac{1}{\omega_{ioW_r}}, \lfloor \frac{||M||}{||R||} \rfloor \cdot \frac{PE-1}{PE} \cdot \frac{1}{\omega_{comm}})$$

case 2

$$T = (N-1) \cdot \frac{||R||}{PE} \cdot max(\frac{1}{\omega_{ioR_s}}, \frac{PE-1}{PE} \cdot \frac{1}{\omega_{comm}}) + \frac{(N-1)}{\lfloor \frac{||M||}{||R||} \rfloor} \cdot \frac{||R||}{PE} \cdot max(\frac{1}{\omega_{ioR_r}} + \frac{1}{\omega_{ioW_r}}, \lfloor \frac{||M||}{||R||} \rfloor \cdot \frac{PE-1}{PE} \cdot \frac{1}{\omega_{comm}}) + \frac{||R||}{PE} \cdot max(\frac{1}{\omega_{ioR_r}} + \frac{1}{\omega_{ioW_r}}, (N-1) mod \lfloor \frac{||M||}{||R||} \rfloor \cdot \frac{PE-1}{PE} \cdot \frac{1}{\omega_{comm}})$$

4. Evaluation Results

In Fig. 2, we show the performance of a 9-way join for the left and right-deep trees, when varying the memory size. For the curve of network bandwidth 1.72 MB/s, all the segments of both trees are I/O bound and thus, the performance is proportional to the data transferred to/from the disks. We can observe that the left-deep tree is not able to take advantage of memory as it is added. For the right-deep tree, we find that the time is proportional to the number of segments, in this case of constant relation sizes. For this I/O bound case, the results are the same as those presented in [3].

By decreasing the network bandwidth to 0.43 MB/s, the performance for both trees decreases. However, we can see that the degradation is greater for the left-deep tree. For the right-deep tree, when the degree of inter-operator parallelism $\lfloor \frac{||M||}{||R||} \rfloor$ is grater or equal to 3 operators, the processing of the segment is network bound and otherwise, it is I/O bound. We can observe that the points [4,4], [5,3] and [8] have all segments being network bound (≤ 3) and thus, show the same performance; the points [3,3,2] and [6,2] have the same total amount of data transferred through the network in the network bound segments (3+3, and 6, respectively) and the same amount of data spooled into disk in the 1 I/O bound segment and thus, show the same performance; the point [7,1] shows worse performance that [6,2] because it transfers more data through the network in its network bound segment; the point [2,2,2,2] has all segments I/O bound and the performance is determined by its disk time.

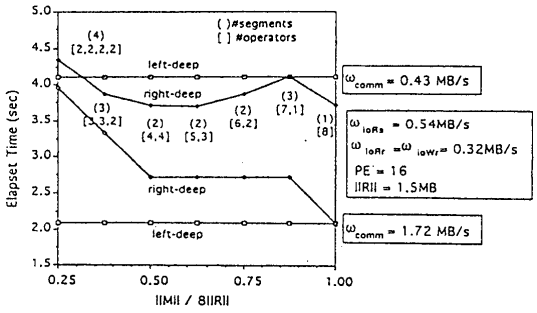

Fig. 2 Performance Results

5. Conclusion

In this paper we present some evaluation results of multi-way join queries in left and right-deep tree structures in a shared-nothing database environment.

From the evaluation results, we could see that the right-deep tree can be processed with different degrees of inter-operator parallelism and utilize the available resources, so that the maximum overlap among their processing can be achieved, by adjusting the number of concurrently active operators. On the other hand, the left-deep tree has not this flexibility, since its inter-operator parallelism is fixed to 3 concurrently active operators.

We are now interested in extending our evaluation to more general queries, and also in including the more complex query structure of bushy trees.

References

[1] M.Kitsuregawa, H.Tanaka, T.Moto-oka, "Application of Hash to Database Machine and Its Architecture", *New Generation Computing*, 1,1, pp.64-74, 1983

[2] D.Schneider and D.J.DeWitt, "A Performance Evaluation of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment," *Proc. of the 1989 SIGMOD Conf.*, pp.110-121, 1989

[3] D.Schneider and D.J.DeWitt, "Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines," *Proc. of the 1990 VLDB Conf.*, pp.469-480, 1990

Acknowledment

We wish to thank Mrs. Miyuki Nakano from Univ. of Tokyo for many helpful suggestions and discussions on this subject.