Hybird Sorted Hash Join

Hyun-Kwang Shin
Information Communication Engineering
Yeungnam University
Gyungbuk Gyungsan, Korea
shg3786@ynu.ac.kr

Gyu-Sang Choi
Information Communication Engineering
Yeungnam University
Gyungbuk Gyungsan, Korea
castchoi@ynu.ac.kr

Abstract— In this paper, we propose a novel hash-join scheme with a name Hybrid Sorted Hash-join which reduces probing time due to the sorted records only within a bucket, whereas the hybrid hash join scheme sequentially scans records in a hash table until the corresponding record is matched. Our proposed scheme shows the significant performance improvement compared to the hybrid hash join scheme and will conduct in-depth performance results between our proposed scheme and the prior scheme.

Keywords-component; Database, Join, Hybrid Hash-Join, Grace Hash-Join

I. INTRODUCTION

In this paper, we propose a new hash-join scheme, called hybrid sorted hash-join, to sort records, while the hybrid hash-join scheme sequentially scans records in hash table until the corresponding records is matched[1][2][3][4].

To insert a new record into a certain bucket in the proposed scheme, it first checks whether a bucket is full or not. If the bucket is full, it moves to the next bucket. If the bucket is not full, it compares the key of the record to insert with the key of the last record in the bucket. If the key of the record to insert is smaller than the key of the last record in the bucket, it moves the last record to the right and then compares the new record with preceding record of the last record. Otherwise, it simply insert the new record next the last record. It means that the new record will be inserted the proper slot in the bucket, by keeping the ascending order among records within the bucket. In our proposed scheme, we use binary search algorithm to find the record with in a bucket, instead of scanning records. To probe the record, it visits the first bucket and find the corresponding record using binary search algorithm. If the record is not found, it moves to the next bucket to search, and keep doing this until the record is found.

Now, we will explain the proposed scheme with a simple example, and Figure 1 depicts how the hybrid sorted hash-join works. In the second phase in our proposed scheme, the directory entry 0 initially has one bucket to hold 4 records, 8, 16, 24 and 40, as shown in Figure 1 (a). Within the bucket, 4 records are stored as ascending order, and a new record 80 will be inserted to this directory entry. Since the bucket has no space to store it, the new bucket is first allocated and then the record is inserted to the first slot in the second bucket. Figure 1 (b) shows the status of the hash table after the new record 80 is added. Now, we will add the new record 32 to the first directory entry in the hash table. Since the first bucket is full, it checks whether the next bucket has available space to store the new record. The new record will be added to the first slot and the record in the second bucket is moved to the second slot in the second bucket, in order to keep ascending order among records within the second bucket.

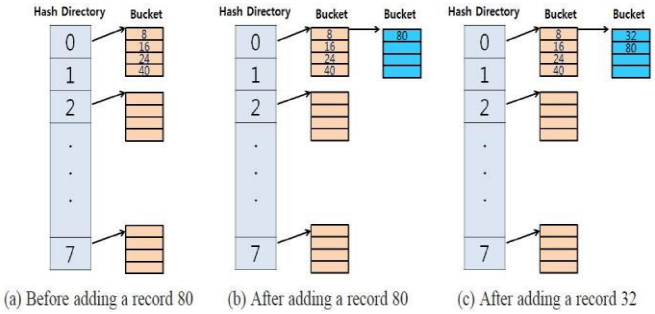


Figure 1. Hybrid Sort Hash Join

II. EXPERIMENT VALIDATION

In this section, we will explain the experiment set-up and then show performance evaluation.

A. *Experiment Set-up*

In our experiments, we synthetically make two tables R and S, as shown in Figure 3. Table R has the primary key Attr_A and the other attribute Attr_C, and S has two attributes Attr_B and Attr_C which are both foreign keys. It means that the combination of the attributes Attr_B and Attr_C is the primary key of table S. In our experiments, all attributes value of tables R and S are randomly generated using normal distribution in most experiments, including the primary key. In addition, Table 1 shows the configurations of our simulation test-bed. In this paper, HHJ and HSJ stands for Hybrid Hash Join and Hybrid Sorted Hash Join (i.e., the proposed scheme) schemes, respectively.

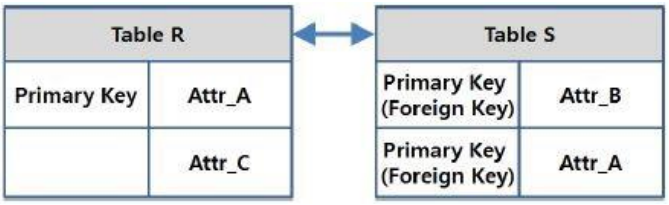


Figure 2. The schema of tables R and S

Table 1. Simulation test-bed Configuration

CPU	Intel(R) Core(TM) i7-4790 Quad-Core, 3.6 GHz
Front-side Bus	1.6 GHz
Main Memory	16 Gbytes
Storage Interface	Serial ATA3
HDD	Seagate Barracuda ST1000DM003 7200 RPM, 1 Tbytes
OS	Cent OS 6.5 (Linux 2.6.34)
File System	ext4

B. Performance Comparison

We first experiment the join operation between R and S tables. In this experiment, the table S has only 100K records but the table R has the varied number of records from 100K to 1 million. In the first phase, it reads and hashes tuples of the tables R into buckets using a certain hash function, and then the hashed tuples are written back to disk. It does the same thing to tuples in table S. After the first phase, the tables S and R will be used to build and probe a hash table, respectively, during the second phase, because the number of records in table S is smaller than table R.

Figure 3 shows the completion times of hybrid hash join scheme and the proposed scheme by varying the number of records of table R from 100K to 1 million. We broke down the total elapsed time to hashing and disk I/O times in the first phase, and building, probing and disk I/O times in the second phase. As we expected, the probing time in hybrid hash join scheme is dominant to scan the records sequentially since the records in the buckets are not sorted. Compared to the hybrid hash join scheme, the proposed scheme shows the significantly reduced probing time of the second phase whereas the building time is deteriorated little. In the proposed scheme, it spends more time to make records sorted within only buckets, but much less time to probe within a bucket by the binary search scheme. In this experiment, the execution times of the first phase between these two schemes are exactly the same since the operations of the first phases are the same.

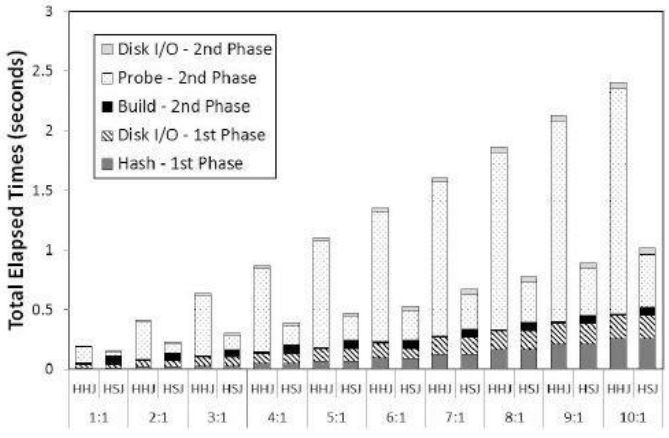
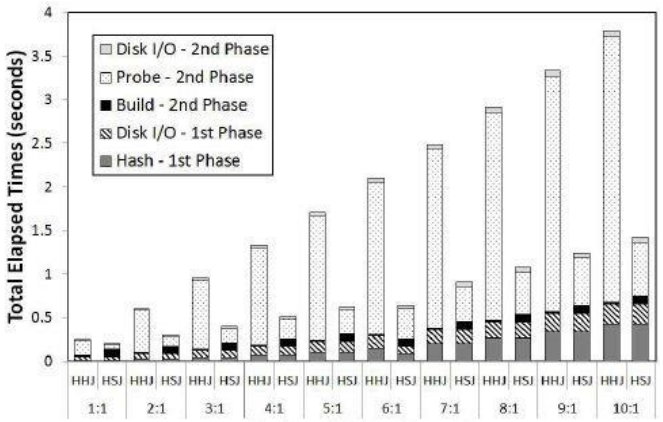
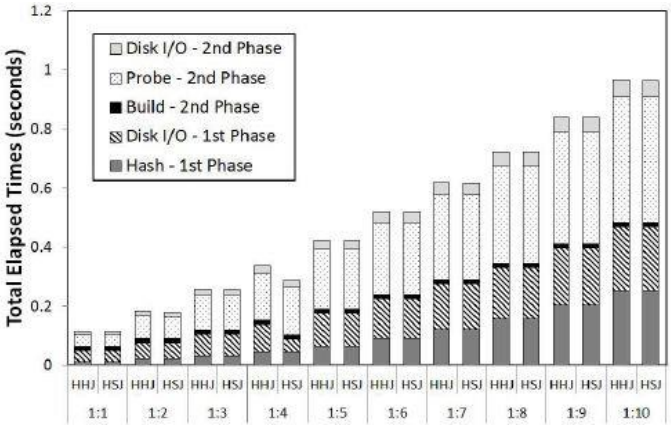


Figure 3. The Analysis of Completion Times between Hybrid Hash Join and Hybrid Hash Sort Join by Varying the Number of Records in Table R

Next, we set the table R to only 100K records and the table S to the varied number of records from 100K to 1 million in this experiment. After the first phase, the tables R and S will be used to build and probe a hash table during the second phase, respectively, since the number of records of table R is smaller than table S in this experiment. In Figure 4 (a), the proposed scheme shows much shorter completion times compared to the hybrid hash join scheme as the number of records in table S increases 100K to 1 million. In this experiment, it needs to scan all records across buckets in hybrid hash join scheme since we assume that records in table R are not sorted. In contrast, the hybrid sorted hash join can significantly reduce the scanning time using binary search method within a bucket because the records within only a bucket are sorted.



(a) Table R is not sort



(b) Table R is not sort

Figure 4. The Analysis of Completion Times between Hybrid Hash Join and Hybrid Hash Sort Join by Varying the Number of Records in Table S

In Figure 4 (b), the hybrid hash join and the proposed schemes show similar completion times. In this experiment, the binary search algorithm can be applied for both the hybrid hash join and proposed schemes since we assume that all records in table R are sorted according to the primary key.

Next, we additionally use uniform and zipf distributions to generate records in tables R and S randomly, in addition to normal distribution in previous experiments. In this experiment, the number of records in table R is varied from 100K to 1 million, while the number of records in Table S is 100K. During the second phase the tables S and R will be used to build and probe a hash table, respectively, because the number of records in table S is smaller than table R. Figure 5

depicts that the proposed hybrid sorted hash join improves completion times up to 170 % across different distributions compared to the hybrid hash join scheme. The reason why the uniform distribution show short completion times compared to other distributions is that it can evenly divide the records across hash directory entries.

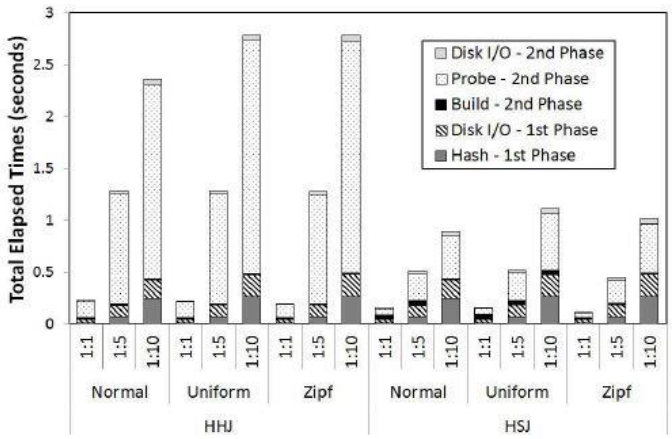


Figure 5. The Analysis of Completion Times between Hybrid Hash Join and Hybrid Hash Sort Join by Varying the Distributions

Now, we varied bucket sizes, record sizes, and the number of records. Figure 6 show the completion times of these two schemes as the bucket size increases from 512 Bytes to 16 Kbytes. The completion times of two schemes show the largest at the bucket size 512 byte but the smallest the at the bucket size 16 Kbytes. As the bucket size increases, it shows shorter completion times because the bigger bucket size can hold more records and each directory entry in a hash table has less the number of buckets. It means that the time to scan can significantly be reduced by the binary search algorithm if each bucket has more records. In Figure 7, the completion times of these two schemes are decreased as the record size increases from 8 Bytes to 128 bytes. A bucket can store less records at the bigger record sizes compared to smaller record sizes. A large record size needs more the number of buckets, and it incurs more searching time. In our experiment, the record sizes as 8 and 128 bytes show the shortest and longest completion times, respectively.

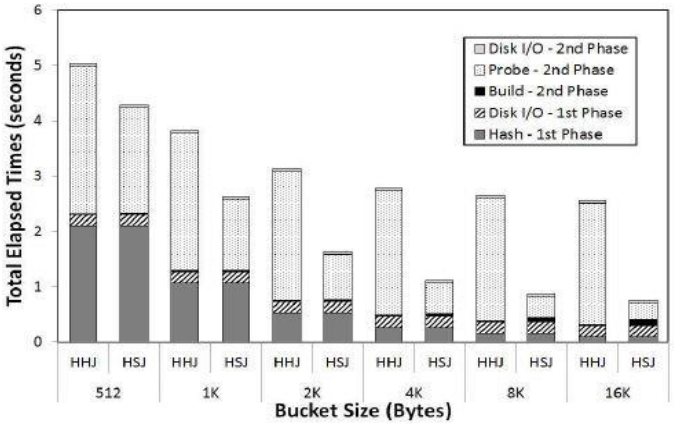


Figure 6. The Analysis of Completion Times between Hybrid Hash Join and Hybrid Hash Sort Join by Varying the Bucket Size

In Figure 8, the proposed scheme improves the completion times up to 400% compared to the hybrid hash join scheme. In this experiment, the more buckets are linked to each hash directory entry as the number of records are increased from 50K to 800K, which means that it needs more probing time to visit more buckets. The hybrid hash join scheme takes negligible building time but much huge probing time since it requires to scan all records across buckets. In contrast, the proposed scheme takes more building time but much shorter probing time because the binary search algorithm is used to search a bucket.

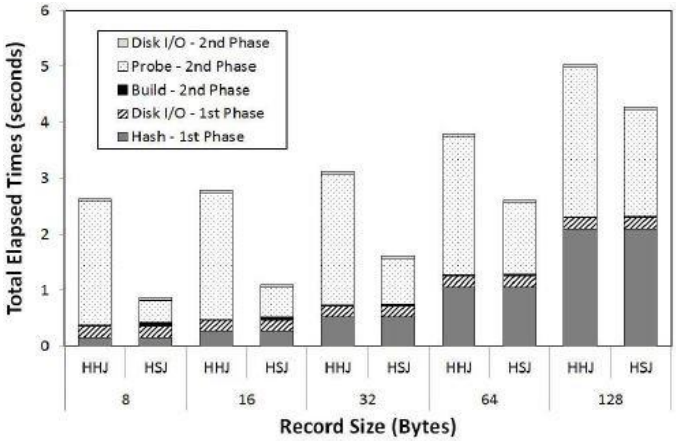


Figure 7. The Analysis of Completion Times between Hybrid Hash Join and Hybrid Hash Sort Join by Varying the Record Size

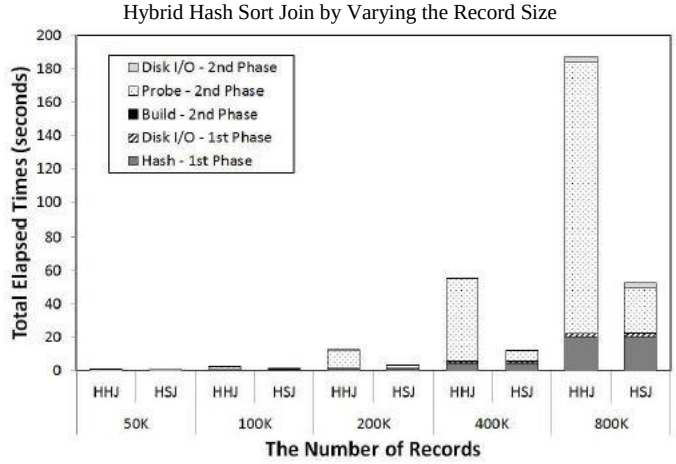


Figure 8. The Analysis of Completion Times between Hybrid Hash Join and Hybrid Hash Sort Join by Varying the Number of Records

III. CONCLUSIONS

In this paper, we proposed a new hash-join scheme, called hybrid sorted hash-join which outperforms the prior scheme, hybrid hash-join scheme, since it sequentially scans records in hash table until the corresponding records is matched. In the future, we will evaluate in-depth performance results between our proposed scheme and the prior scheme.

REFERENCES

[1] J. Do and J. Patel, "Join processing for Flash SSDs:Remembering past lessons", InProceedings of the Fifth International Workshop on Data Management on New Hardware, pp. 1-8, Jun 2009.

[2] J. M. Patel, M. J. Carey, and M. K. Veron, "Accurate modeling of the hybrid hash join algorithm." In ACM SIGMETRICS Performance Evaluation Review, vol. 22, pp. 56-66, May 1994.

[3] K. Bratbergsengen, "Hashing Methods and Relational Algebra Operations.", In proceedings of the 10th International Conference on Very Large Data Bases(VLDB), pp. 323–333, August 1984.

[4] L. Shapiro, "Join Processing in Database Systems with Large Main Memories.", ACM Transactions on Database Systems, vol. 11(3), pp. 239–264, September 1986.