

Sub-Join: A Query Optimization Algorithm for Flash-based Database

Zhichao Liang, Da Zhou, Xiaofeng Meng

2009-10-11



Outline

- Background
- Motivation
- Related works
- Our method : Sub-Join
- Performance Evaluation
- Summary



Background

- Two important characteristics of flash memory
 - faster data access (random read)
 - worse random write



	Read	Write	Erase
Time	$25 \mu s$	$200 \mu s$	$1.5 ms$



Motivation

➤ Read& Write Throughput (Mb/s)

	HDD	SSD	Ratio
Random Read	3.80	70.76	18.62
Sequential Read	37.45	73.08	1.95
Random Write	3.73	3.71	0.99
Sequential Write	37.69	68.52	1.82

➤ Time of join for Oracle Berkeley DB (s)

	HDD	SSD	Ratio
Time	2.71	1.96	1.38



Motivation (cont.)

- Flash offers little or no benefit when used as a simple drop-in replacement for disk
- Traditional query processing algorithms for data analysis are tuned for disks
- They avoid random I/O operations but exploit sequential I/O operations whenever possible
- So, they fail to take advantage of the fast random reads of flash drives



Related works

➤ PAX Layout& RARE-join by Mehul A. Shah

Fast scans and joins using flash drives [C]// Luo Q, Ross K A. Proceedings of 4th Workshop on Data Management on New Hardware. New York: ACM Press, 2008: 17-24.

➤ Subset by Daniel Myers

MIT CSAIL Research Abstracts [EB/OL].2007[2009-7-19]

On the Use of NAND Flash Memory in High-Performance Relational Databases[D]. Massachusetts:MIT,2008.

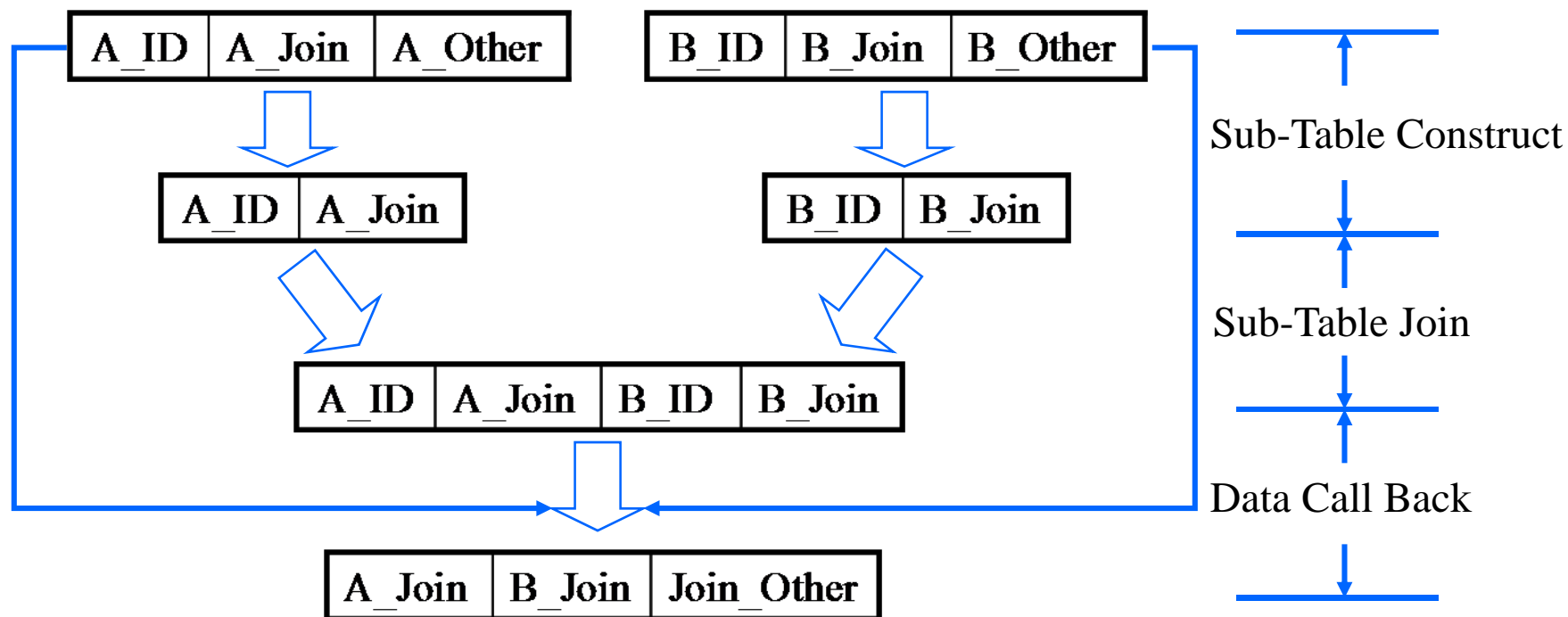
➤ DigestJoin by Yu Li

DigestJoin: Exploiting Fast Random Reads for Flash-based Joins [C]// Lee W C, King C T, Pitoura E. Proceedings of the 10th International Conference on Mobile Data Management. TaiPei: IEEE Computer Society Press, 2009: 152-161.



Our method : Sub-Join

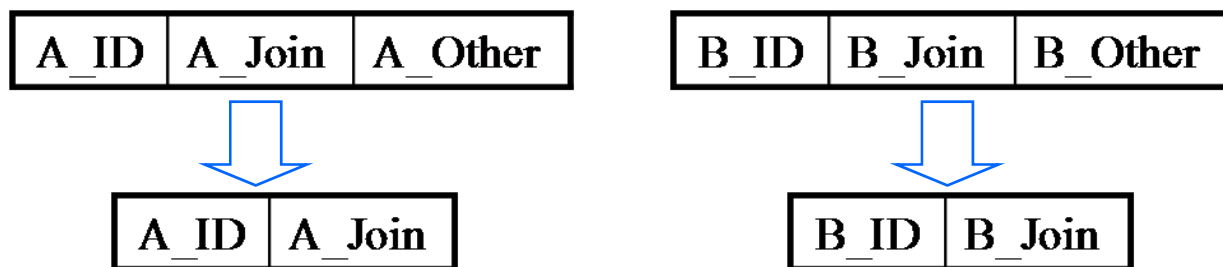
➤ The Framework of Sub-Join Algorithm





Our method : Sub-Join (cont.)

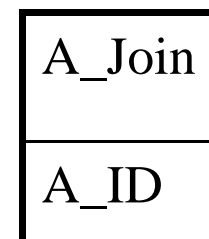
➤ Sub-Table Construct:



- less data volume to read → lower read cost
- more tuples in main memory → less I/O operations

➤ Column-Stores in Sub-Table

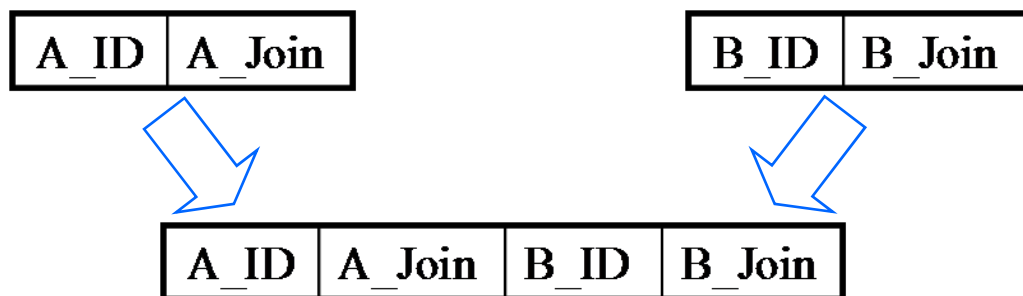
- avoid random write → lower write cost
- sequential read → faster data access





Our method : Sub-Join (cont.)

➤ Sub-Table Join:

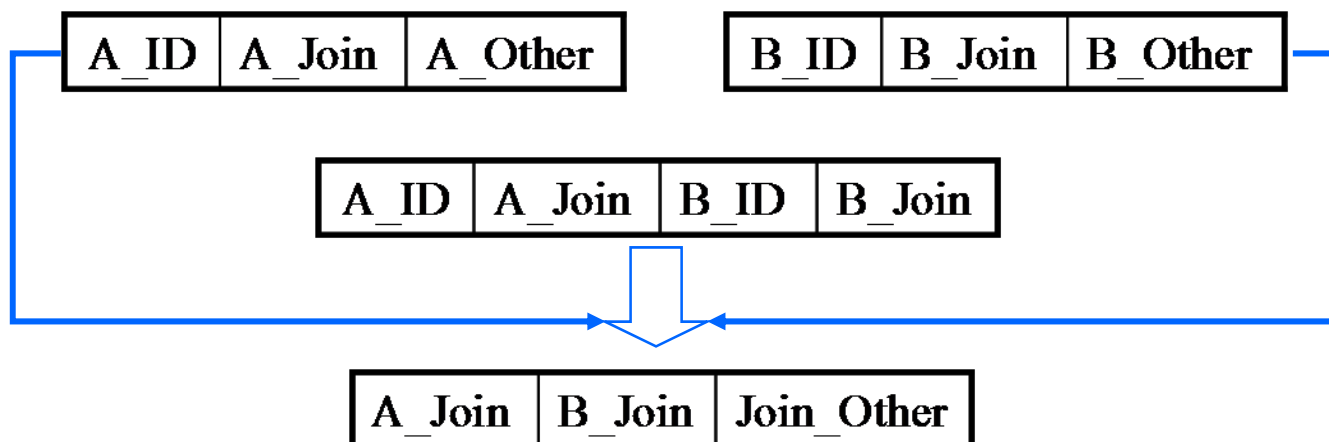


- only join column is necessary → reduce read column further → more tuples in main memory → less I/O operations



Our method : Sub-Join (cont.)

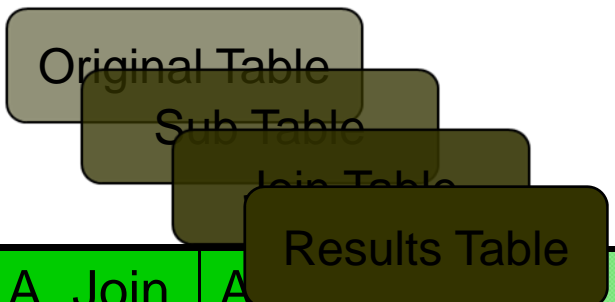
➤ Data Call Back



- retrieve the data according their key → fully make use of the fast random read

Our method : Sub-Join (cont.)

➤ Example



A_ID	A_Join	A_...
1	3	3
2	3	6
3	5	5
4	6	1
5	6	2
6	10	4

Join	Other_a	Other_b
3	aaaaa	bbbbbb
3	aaaaa	bbbbbb
3	aaaaa	bbbbbb
3	aaaaa	bbbbbb
3	aaaaa	bbbbbb
5	aaaaa	bbbbbb
6	aaaaa	bbbbbb
6	aaaaa	bbbbbb



Performance Evaluation

➤ Setup for Experiment

- CPU: Intel Core 2 Pentium 4 Duo CPU 2.83GHz, 2GB RAM
- SSD: Mtron SATA-3035-016, 16GB
- Database: Oracle Berkeley DB

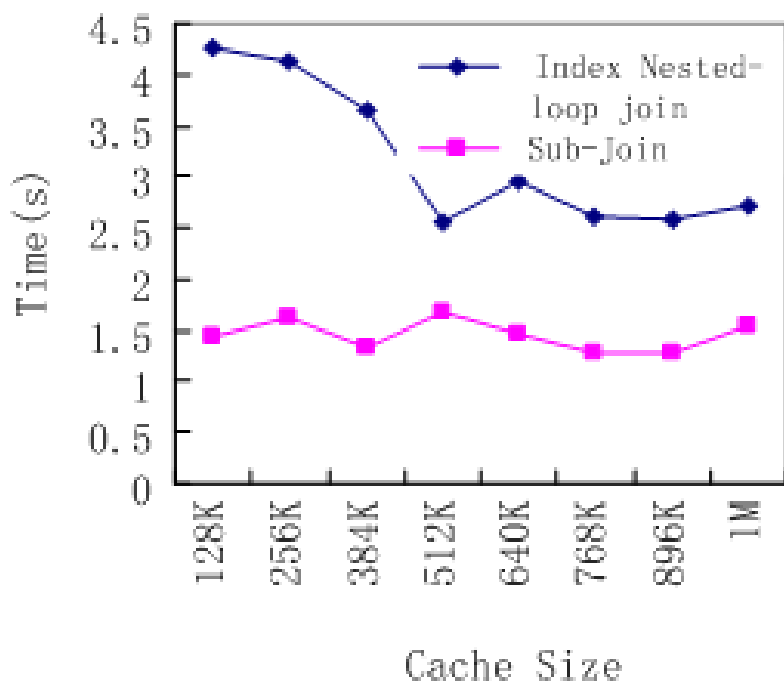
➤ Implementation

- Index Nested-loop Join
- Sub-Join
- Select A.join, other_a, other_b, ...
from A, B
where A.join=B.join

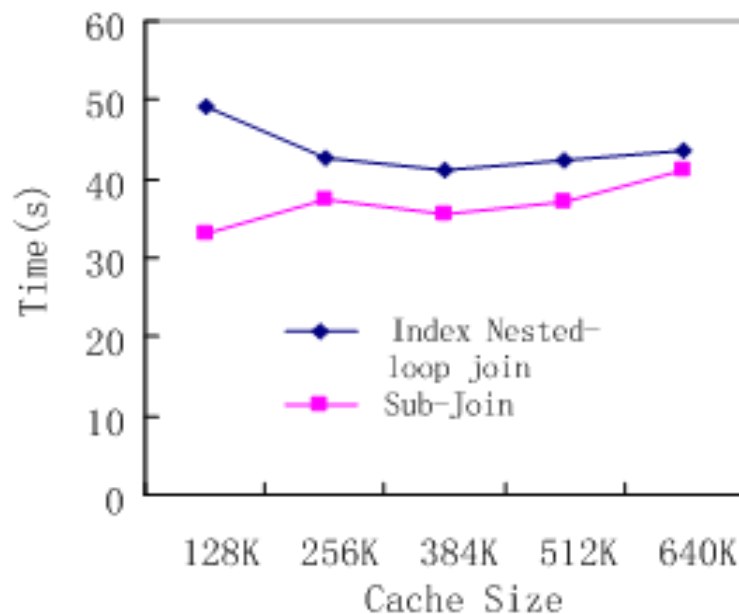


Performance Evaluation (cont.)

➤ Selectivity



Selectivity=0.01, Record=2KB

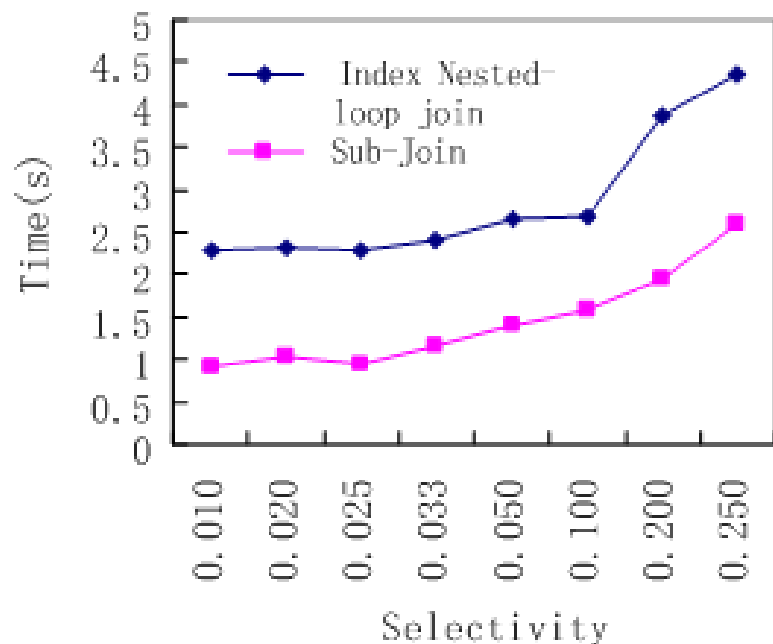


Selectivity=0.2, Record=2KB

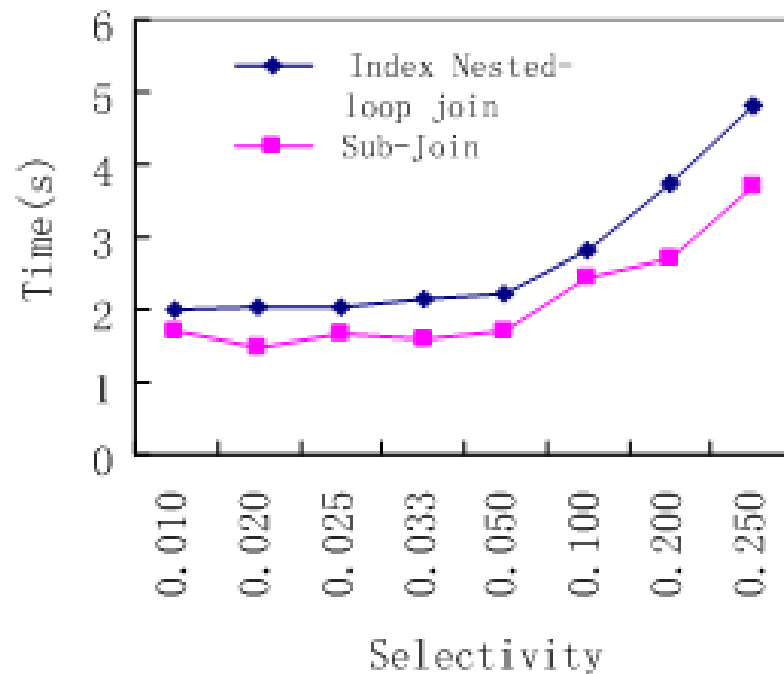


Performance Evaluation (cont.)

➤ Cache size



Cache=50MB,Record=2KB

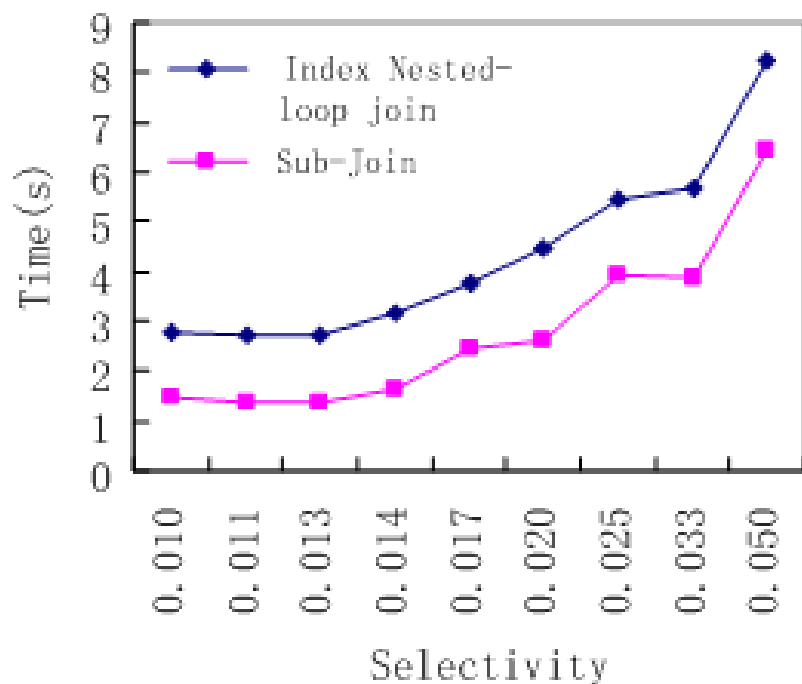


Cache=128KB,Record=2KB

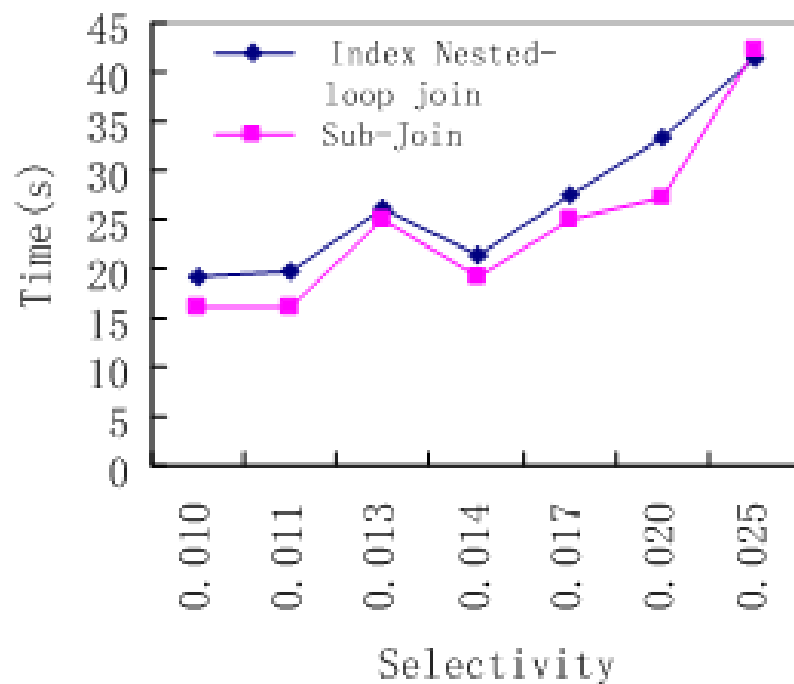


Performance Evaluation (cont.)

➤ Record size



Cache=1MB,Record=2KB



Cache=1MB,Record=32KB



Summary

- We have discussed the defect of the join algorithm adopted in the traditional RDBMS from the perspective of flash memory
- We have proposed a novel query optimization algorithm for flash-based database : Sub-Join
- Experiments results show that Sub-Join outperforms traditional indexed nested-loop join



Q&A

Thank you!