MichiganEngineering

Do Query Optimizers Need to be SSD-aware?

Steven Pelley, Kristen LeFevre, Thomas F. Wenisch,
University of Michigan CSE

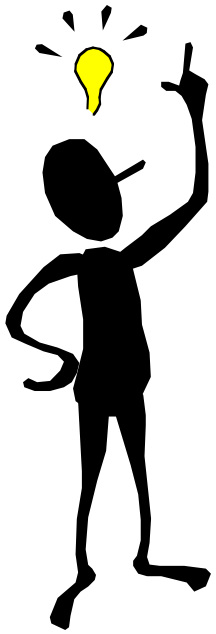
ADMS 2011

Enterprise flash: a new hope

	Disk	Flash
Model	WD 10Krpm	OCZ RevoDrive
Seq. Read	120MB/s	190MB/s
Random Read	10ms	90µs
$/GB	.55	2.5

- Flash much lower read latency than disk
- Flash eliminates performance difference between random and sequential
- Current DBMS optimized for disk

Taking advantage of SSDs

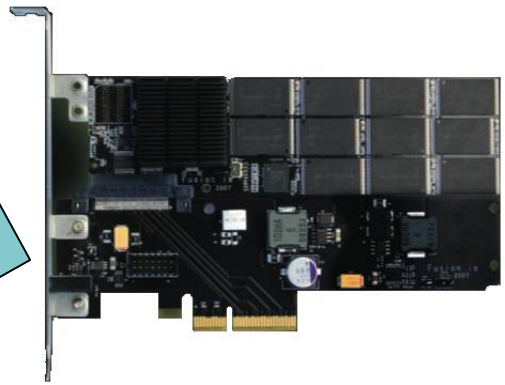


- Data may "prefer" disk or flash
- Intelligent layout:
  - Improve performance
  - Maintain low cost

Automated Layout
[Agrawal *et al.*, Ozmen *et al.*]

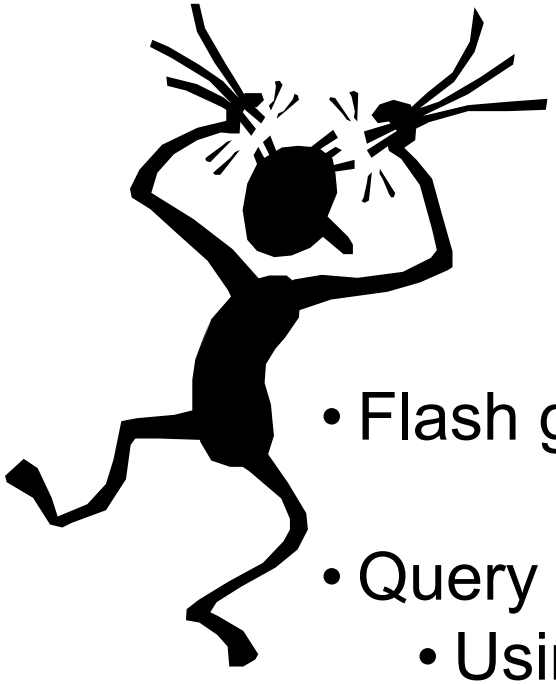
Table 1: Accessed Sequentially

Table 2: Accessed Randomly



Optimal query plan should depend on where data resides

SSD query plans unchanged



- Flash gives better performance, but…
- Query plans do not need to be changed on flash
  - Using commercial DBMS – IBM DB2
  - 0.9% of scan selectivities change access path
  - Join choice matters on disk, but not on flash
- Intelligent layout does not depend on query opt.

Outline

- Introduction
- Experimental structure and methodology
- Access path study
- Join study
- Conclusions

Experimental structure

- Focus on two key aspects of query plans
  - Access paths (scan type)
  - Join type
- Force specific query plans and measure *actual* performance
- Determine performance difference between devices

Methodology

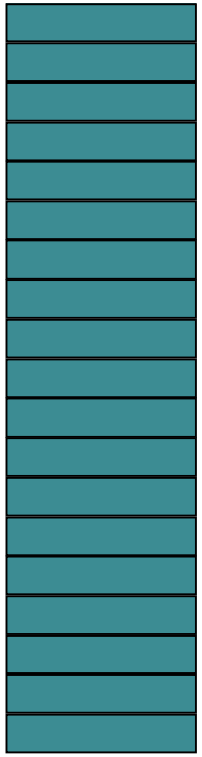
- IBM DB2 v9.7
- Intel core 2 (dual core)
- 2GB memory
- Disk: WD VelociRaptor 10Krpm drive
- Flash: OCZ RevoDrive PCIe
- Wisconsin Benchmark schema and data

Caveat: limited scope

- We focus on *existing commercial systems*
  - Use existing storage managers, algorithms, access paths, and layouts
  - Do not consider algorithms, access paths, and layouts optimized for flash
- Decision Support Services (DSS)
  - Ad hoc queries
  - Performance commonly depends on query plan

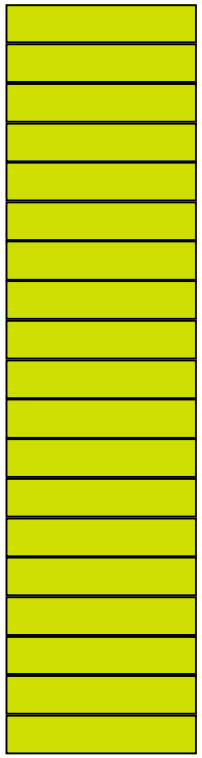
Background: DB2 access paths

Table



Background: DB2 access paths

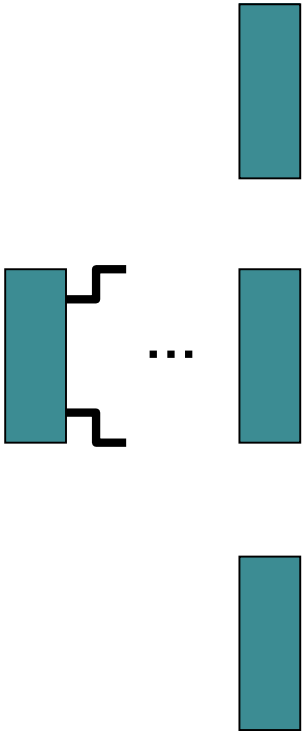
Table



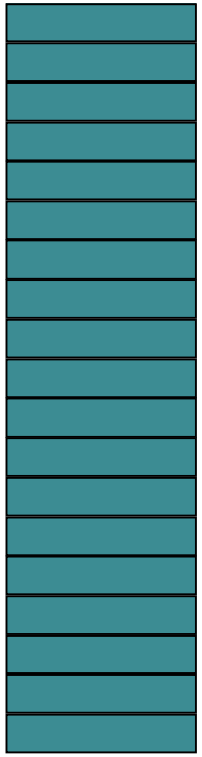
Relation scan: high selectivity

Background: DB2 access paths

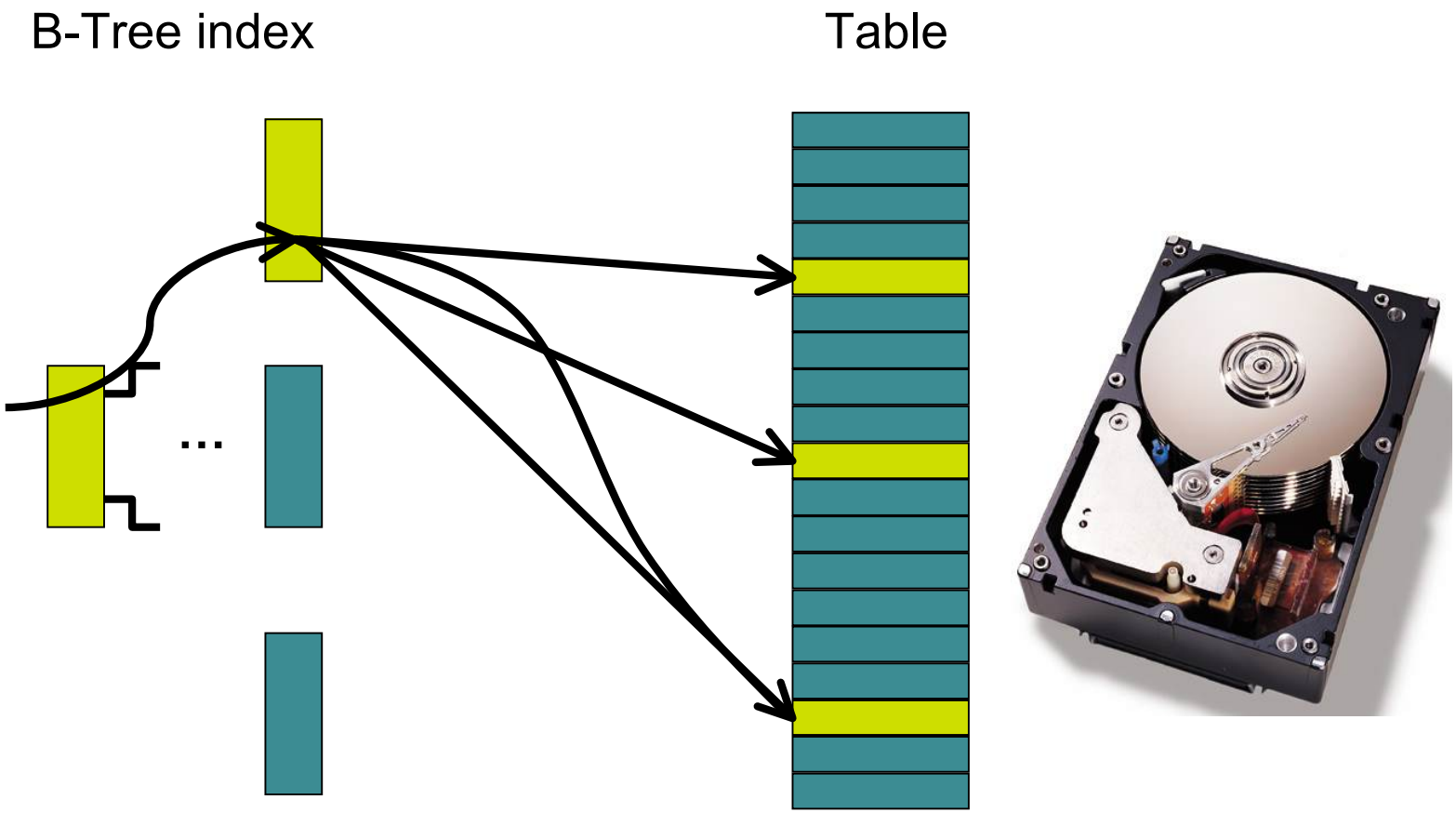
B-Tree index



Table



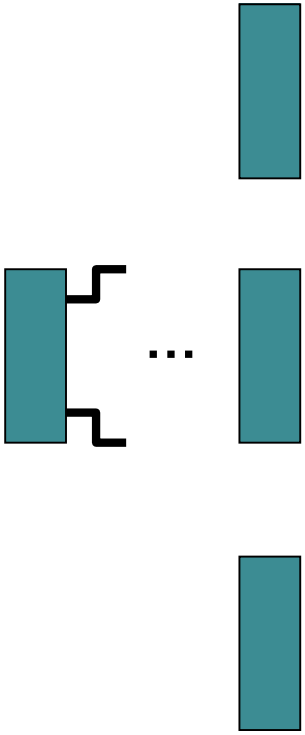
Background: DB2 access paths



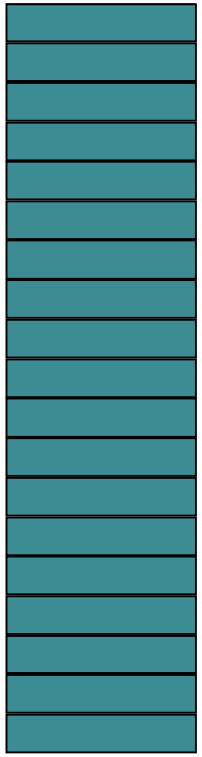
Index scan: low selectivity

Background: DB2 access paths

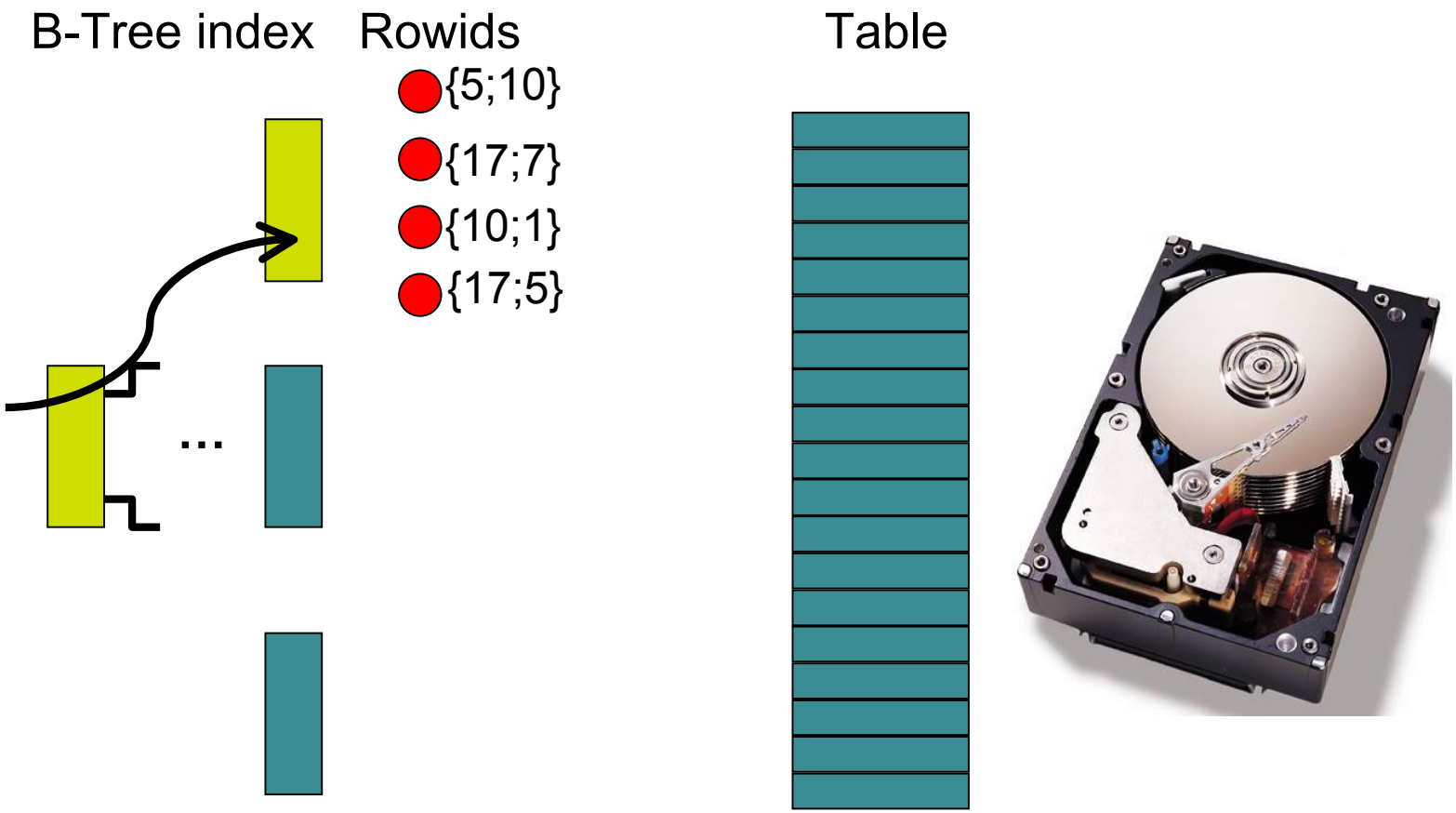
B-Tree index



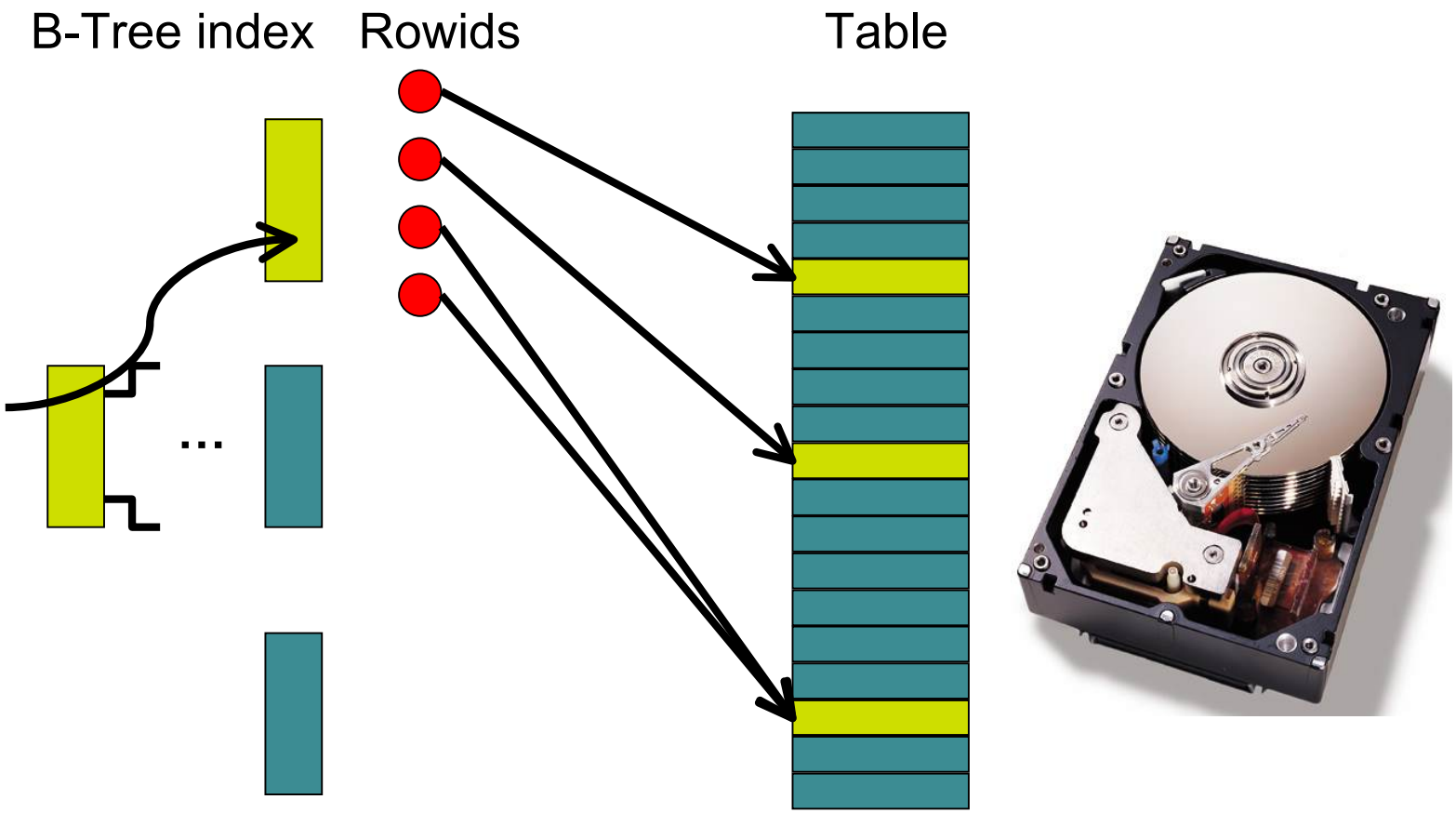
Table



Background: DB2 access paths



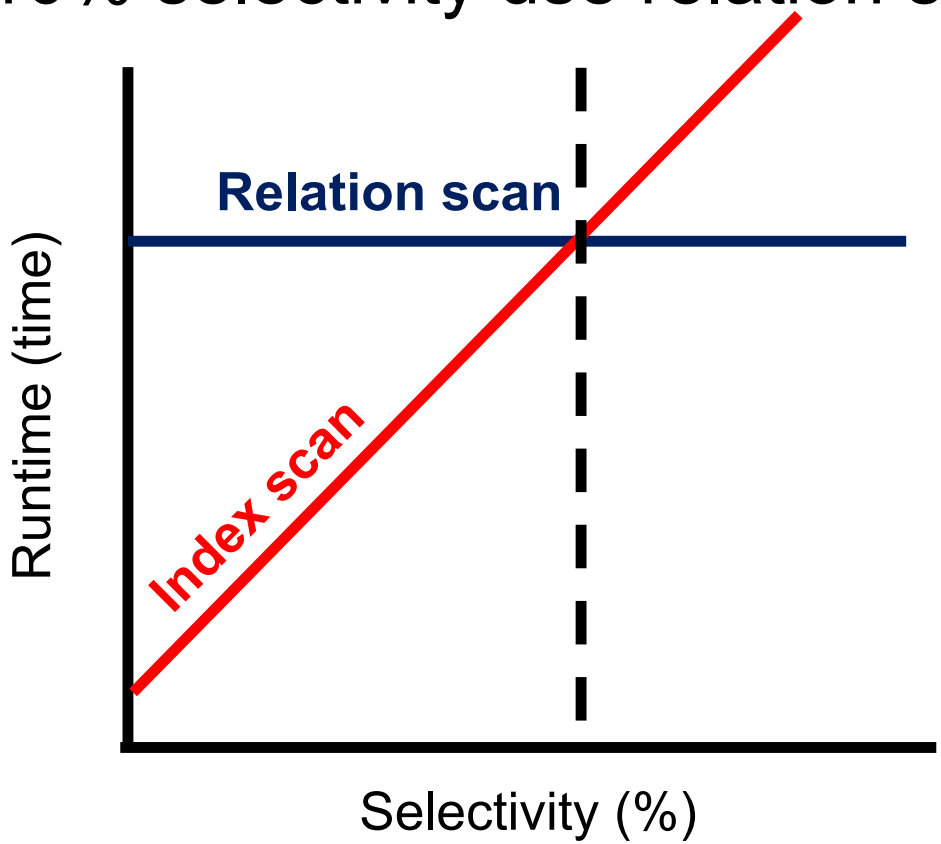
Background: DB2 access paths



Rowid-sort scan: mid selectivities

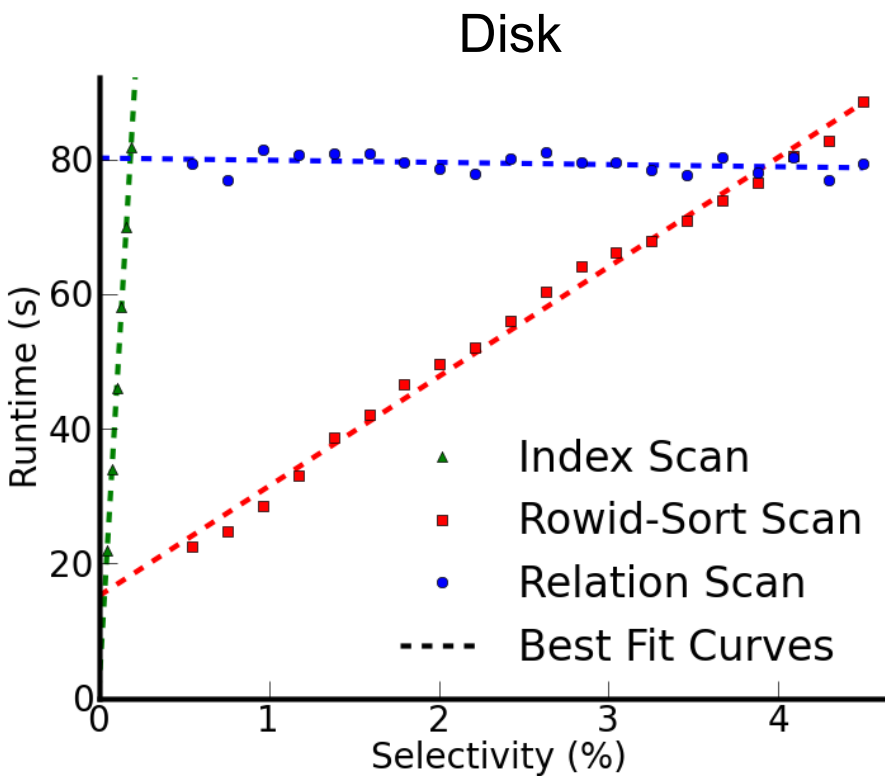
Background: Selecting access path

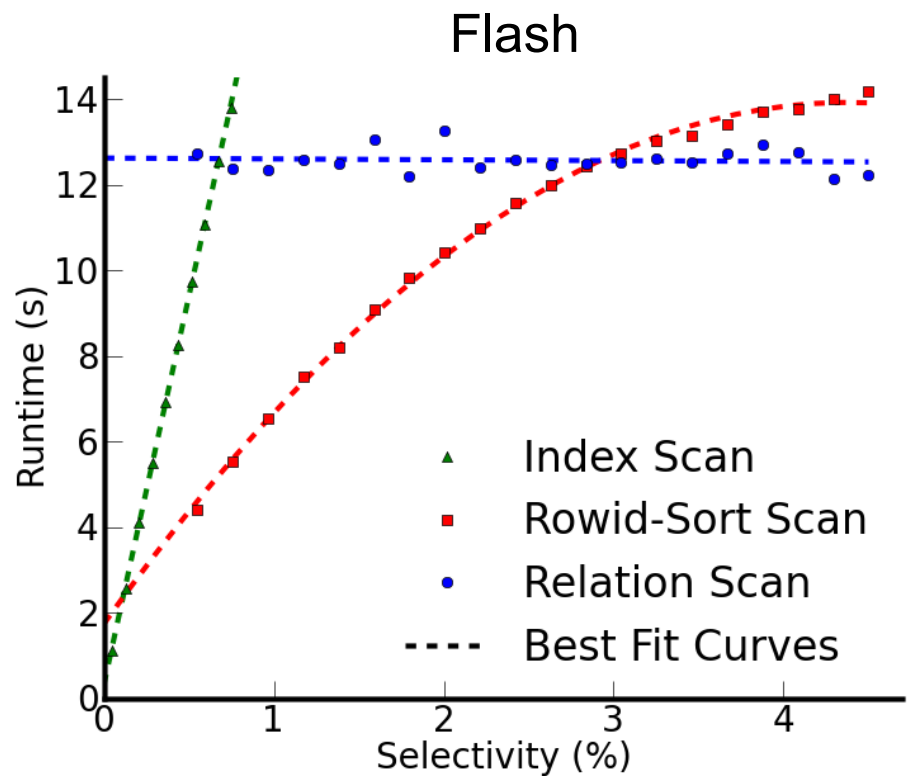
- Below 10% selectivity use index scan
- Above 10% selectivity use relation scan



Flash's low latency should shift break-even point to right

Actual performance

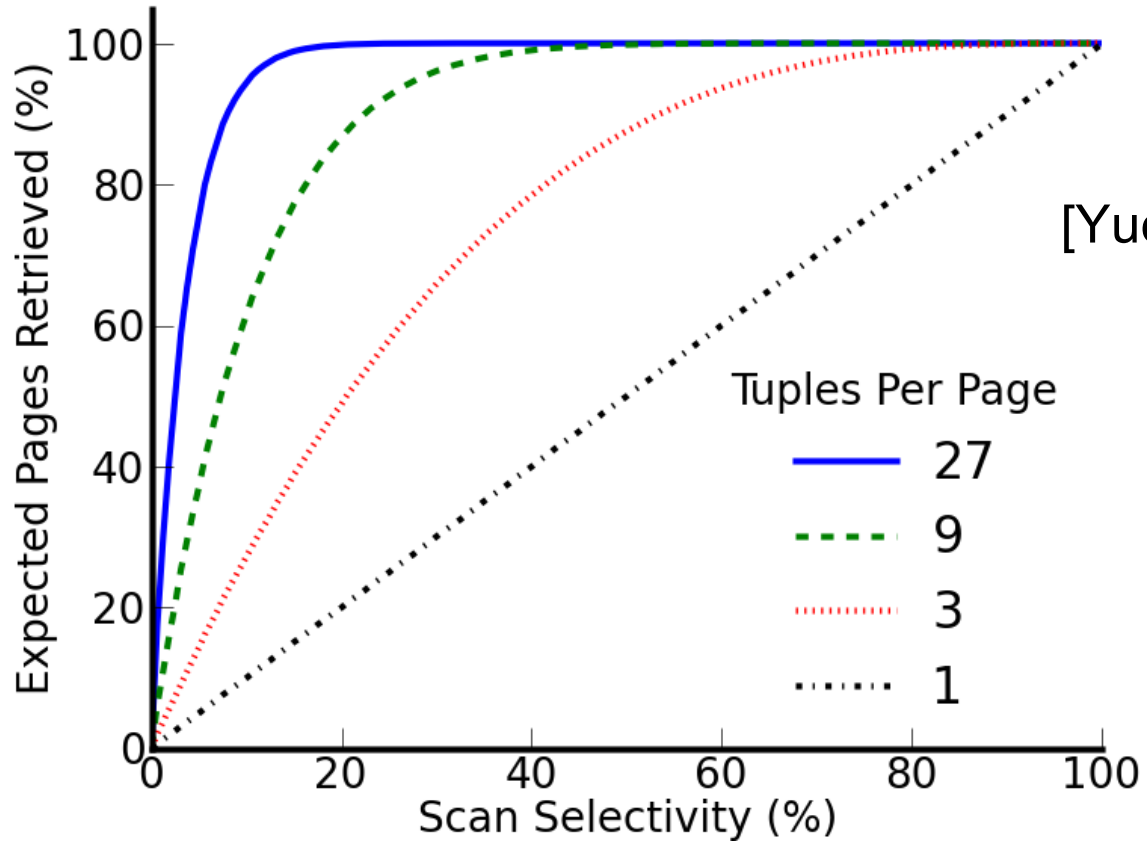




- Break-even points at lower selectivity than expected
- Break-even points shift for only 0.9% selectivities

Why is the 10% rule wrong?

- 10% rule really applies to *page selectivity*
- Index scan, uniform random layout:

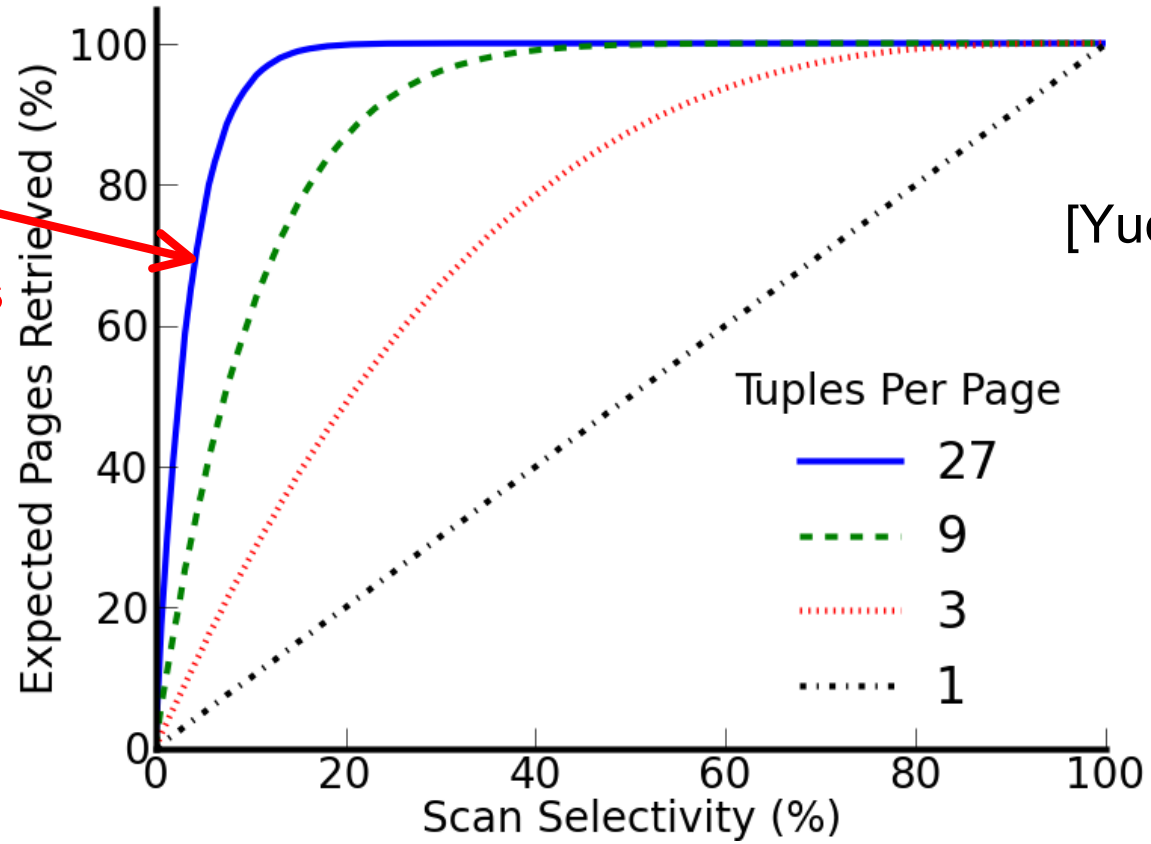


[Yue *et al.*]

Why is the 10% rule wrong?

- 10% rule really applies to *page selectivity*
- Index scan, uniform random layout:

27 tuples/page at 4% row sel. expect 67% pages



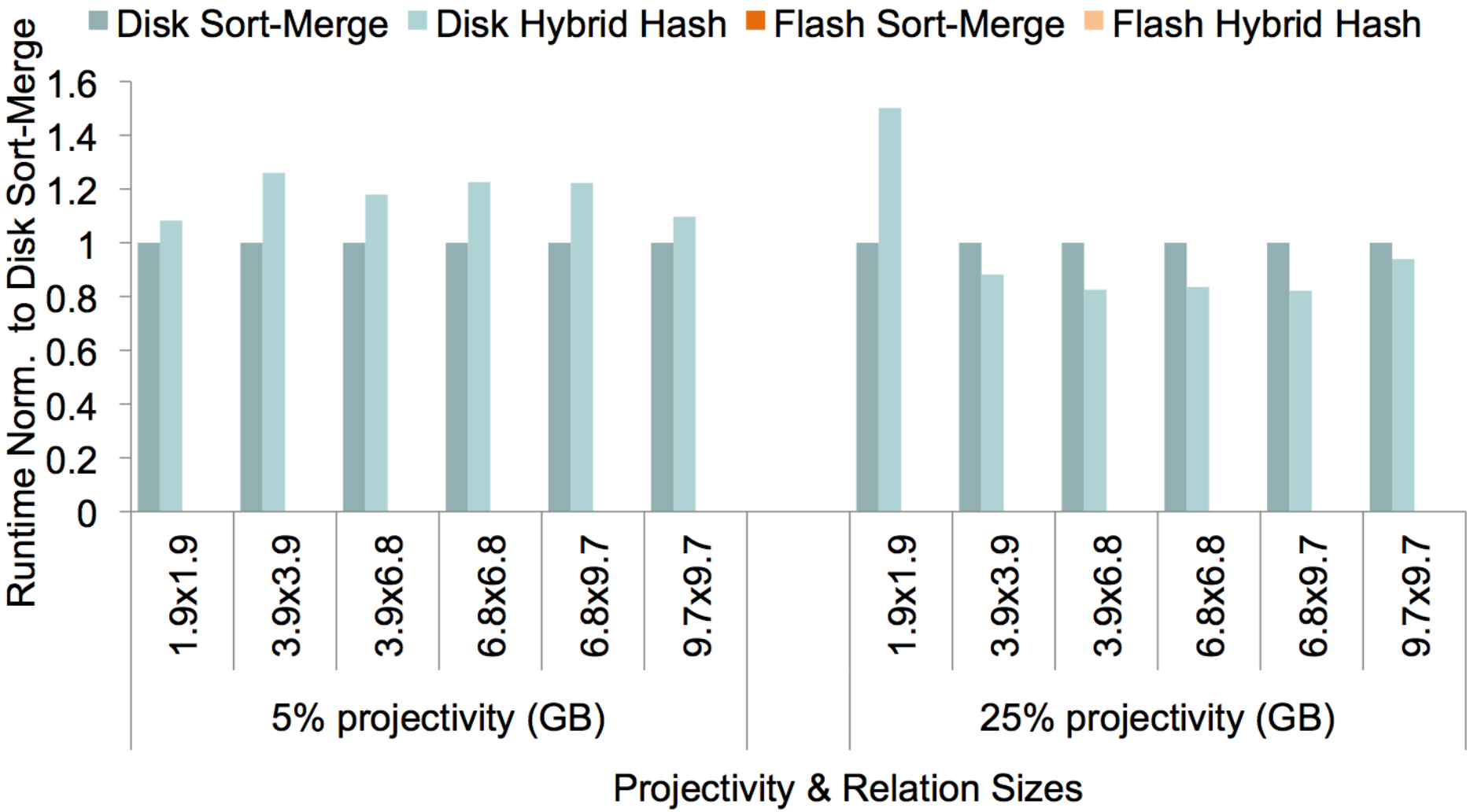
[Yue *et al.*]

Implication: relation scan preferred even on flash

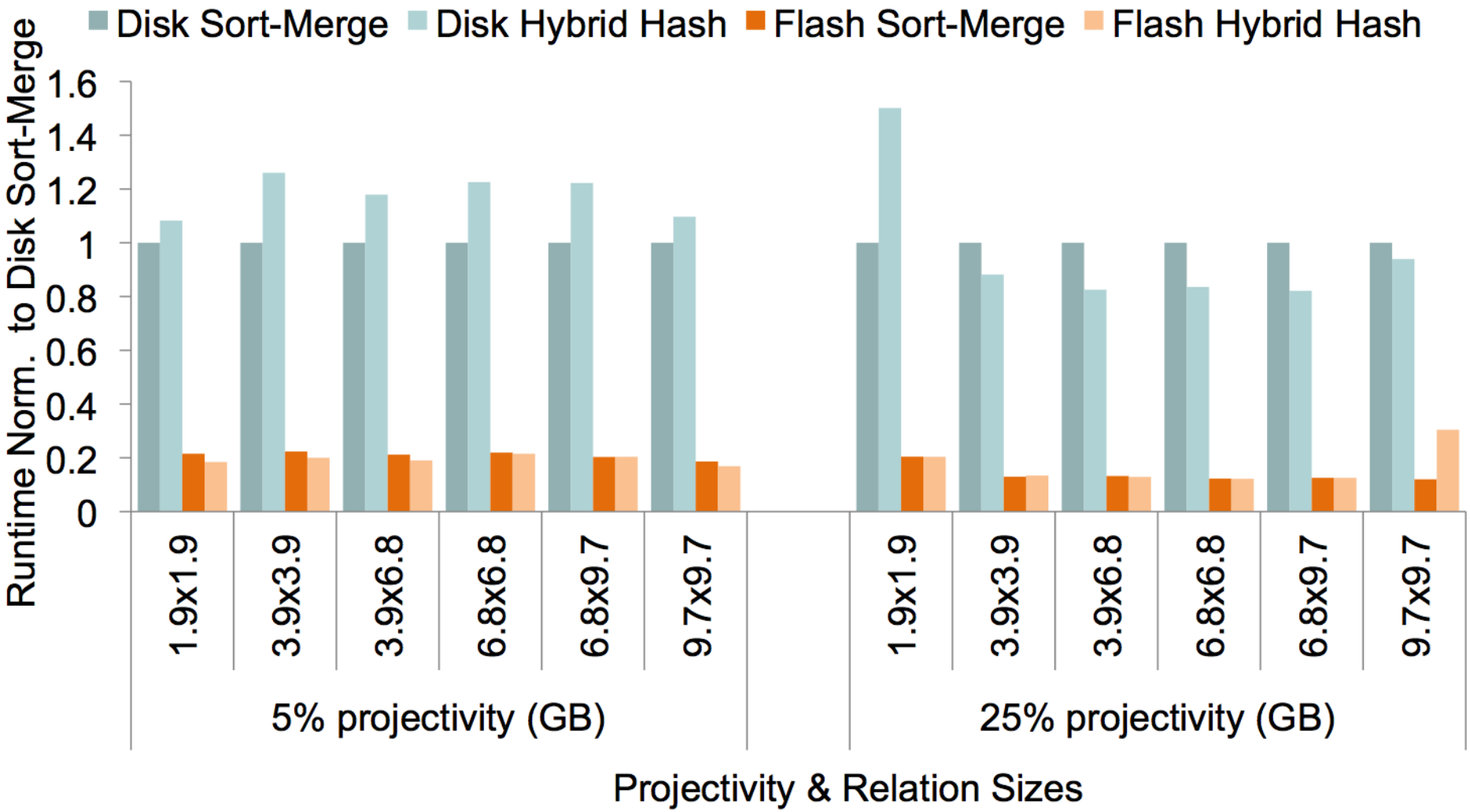
Join

- Join performance on disk depends on:
  - Algorithm
  - Relation sizes
  - Projectivity
- These determine
  - Device access patterns
  - Data read and written to device

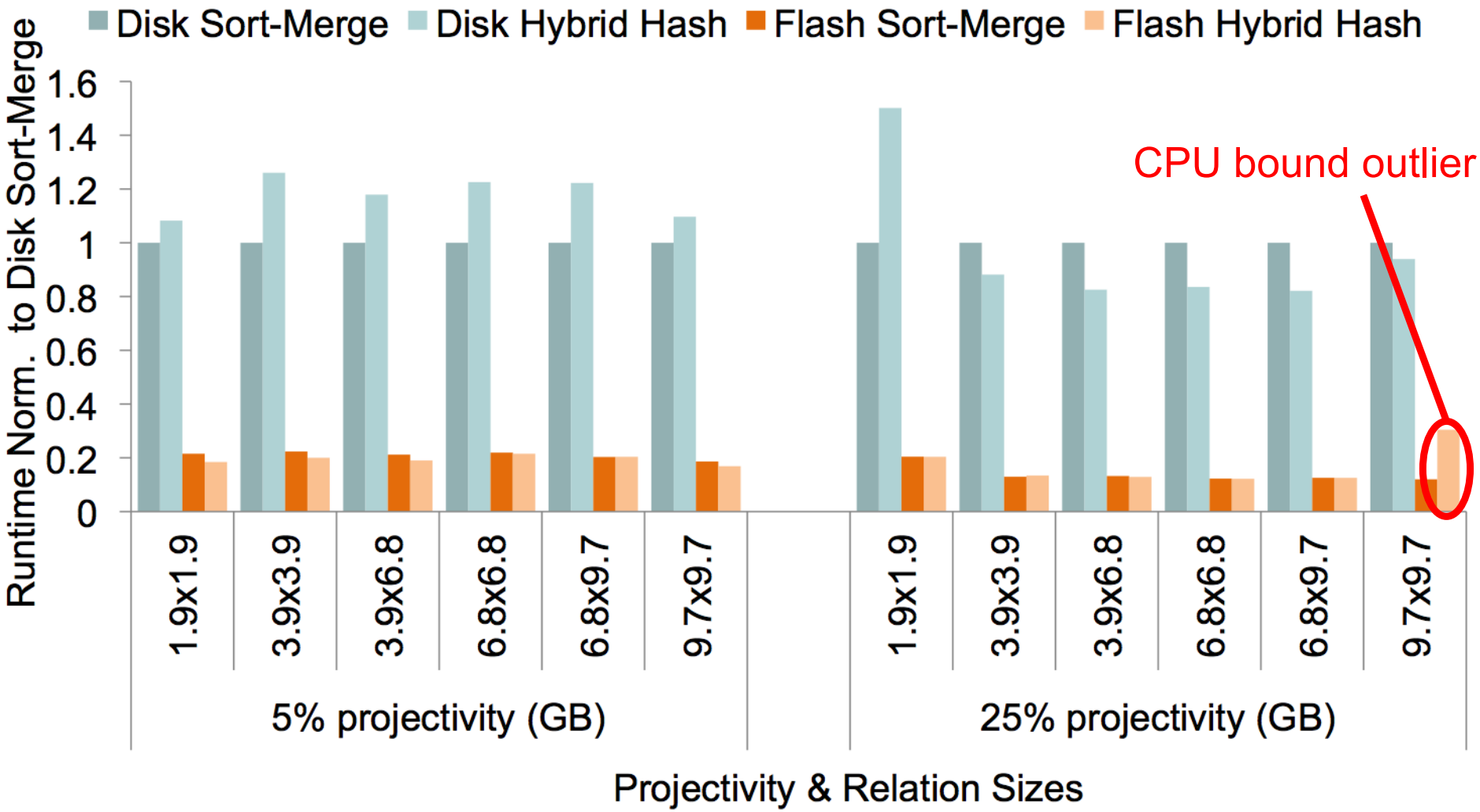
Join performance



Join performance



Join performance



Flash insensitive to join type – 5.4% avg. difference w/o outlier

What does this really say

- Little opportunity for taking advantage of device-dependent query optimization
- Large access granularity interferes with flash's low random read latency
- DBs have been designed for disk's high latency (buffer pool, algorithms, optimizer, etc.)
  - Need redesign before considering rewriting query optimizer

Thank You!