(Some) Trends in Database Research

Miguel Branco

Talk Overview

- A bit of history
  - *What goes around comes around*
- Modern Hardware
  - *Hitting the walls*
- Modern Databases
  - *One size does not fit all*

A bit of history

- Early 1970s
  - Overabundance of database offerings
  - Incompatible, exposing many implementation details
- Ted Codd proposed a new model
  - Relational model
  - Structured Query Language (SQL)
  - Implementation differences became largely irrelevant
- Uniformity brought direct competition
  - DB market dominated by 3 players

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

Today

- Recent explosion in the number of database offerings
  - Different interfaces
  - Lost uniformity

One size does not fit all!

Talk Overview

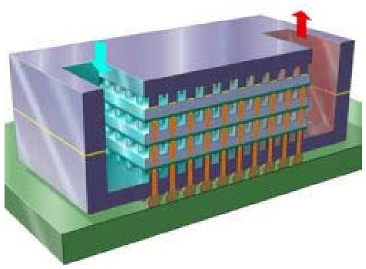
- A bit of history
  - *What goes around comes around*
- Modern Hardware
  - *Hitting the walls*
- Modern Databases
  - *One size does not fit all*

Getting the data to the CPU

- Latency and bandwidth *(source David Patterson, Oct 2004)*

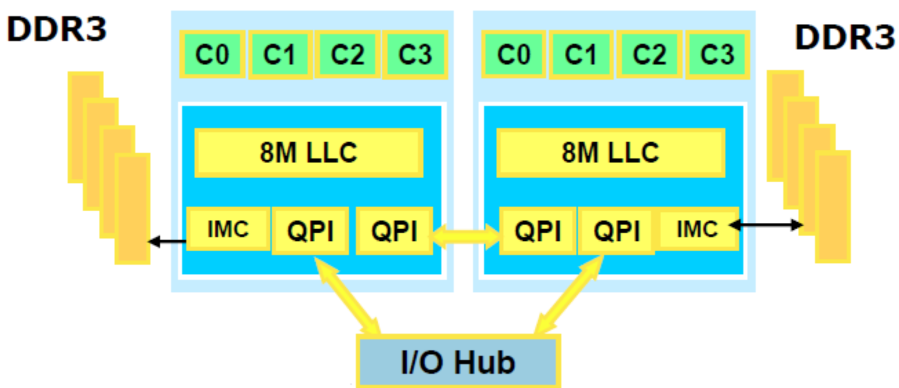
	CPU		DRAM
Annual Bandwidth Improvement	1.50	>	1.27
Annual Latency Improvement	1.17	>	1.07

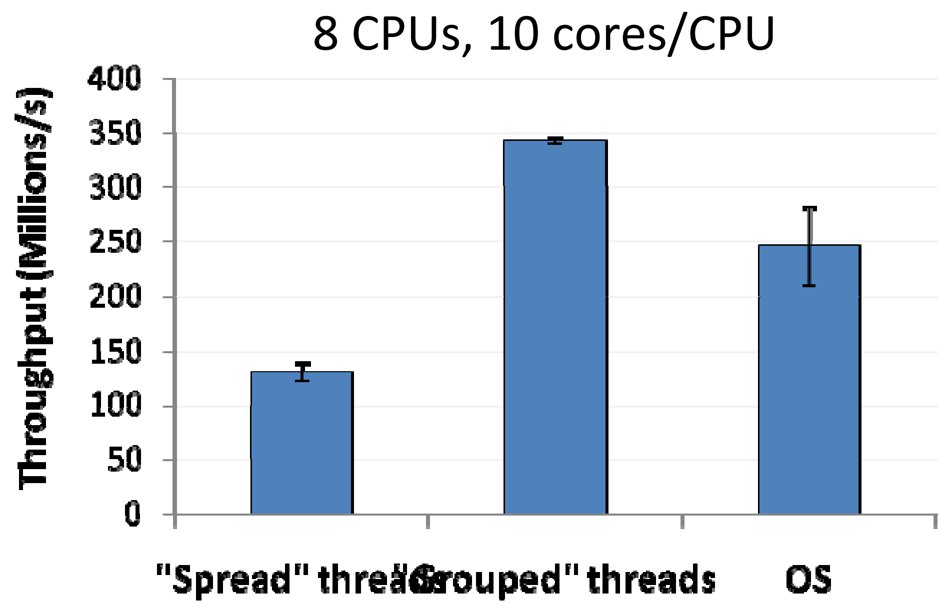
- Memory getting further away from the CPU
- Solution?
  - Cache locality
  - Stack chips vertically



Getting the ***right*** data to the ***right*** CPU

- Hardware is multicore and heterogeneous:
  - Non-Uniform Memory Architectures

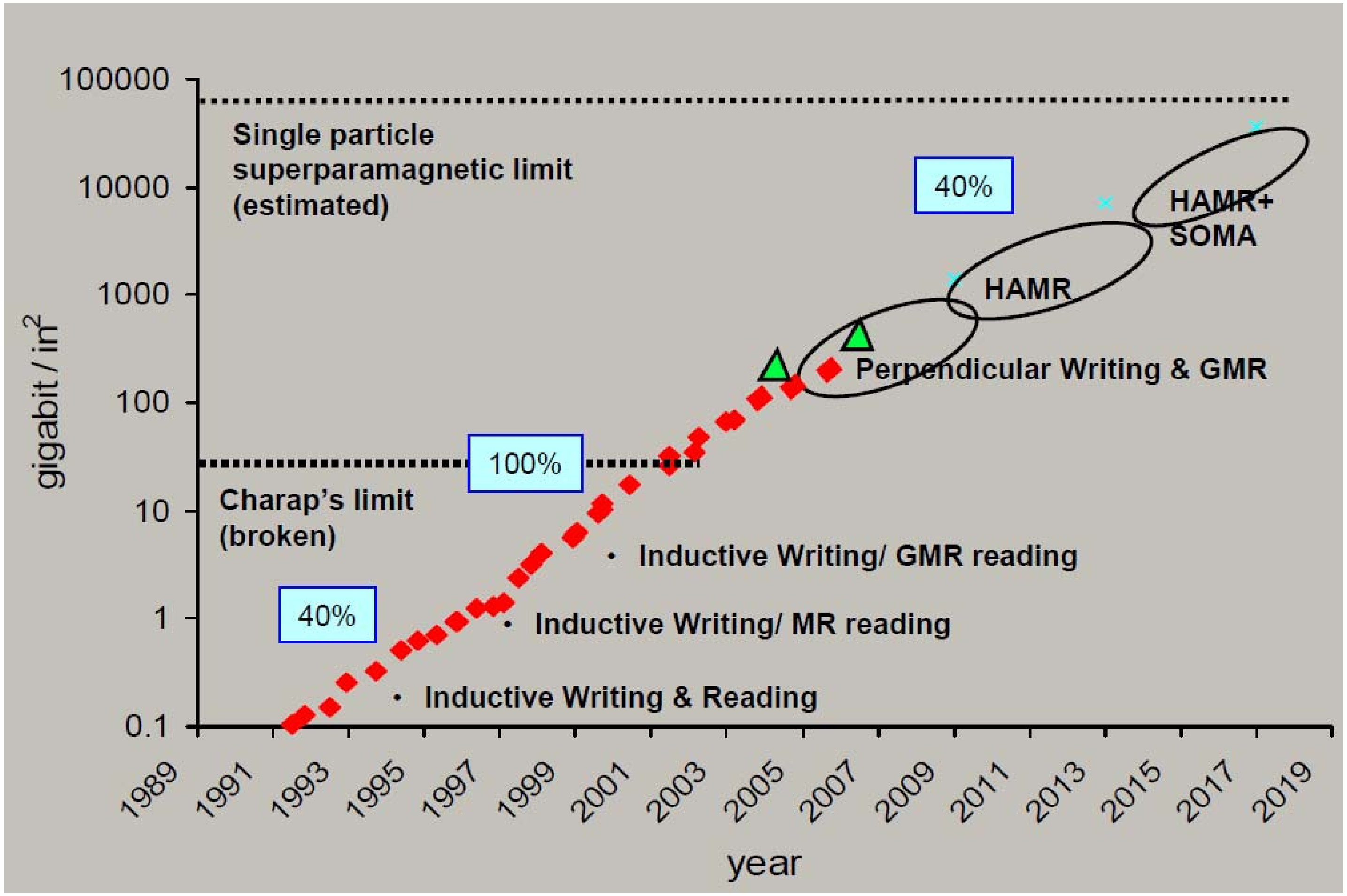




- Solution?
  - "Islands" of computation → Smarter software

Getting the data from storage

- HDD Capacity growing *(source Seagate, 2009)*



Getting the data from storage

- But HDD latency and bandwidth not increasing fast enough *(source David Patterson, Oct 2004)*

	CPU		HDD
Annual Bandwidth	1.50	>	1.28

Jim Gray in 2006:
“Tape is Dead, Disk is Tape, Flash is Disk, RAM Locality is

- Random b/w growing ~10% of sequential
  (source James Hamilton, 2011)
- HDD RPMs not improving: Impacting DBMS!
- Solutions: Flash?

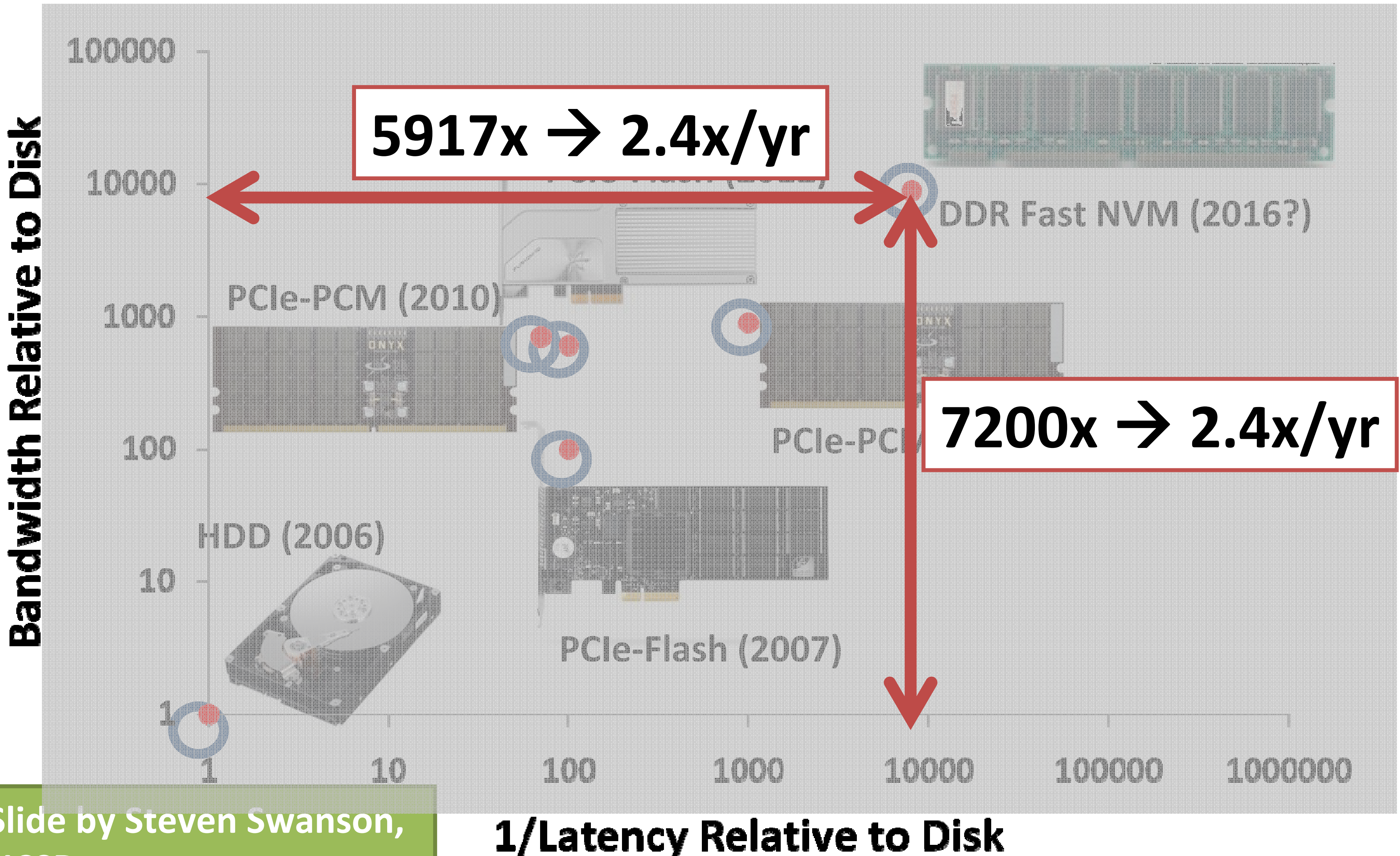
Phase-Change Memory

- PCM vs Flash

4K accesses	PCM today	PCM expected	NAND Flash circa 2011
Read BW	800 MB/s	5-6 GB/s	3 GB/s
Read latency (HW)	20 µs	4-5 µs	60 µs
Write BW	40 MB/s	600-650 MB/s	1 GB/s
Write latency (HW)	250 µs	20-200 µs	300 µs
Endurance	1M cycles	>1M cycles	100K cycles

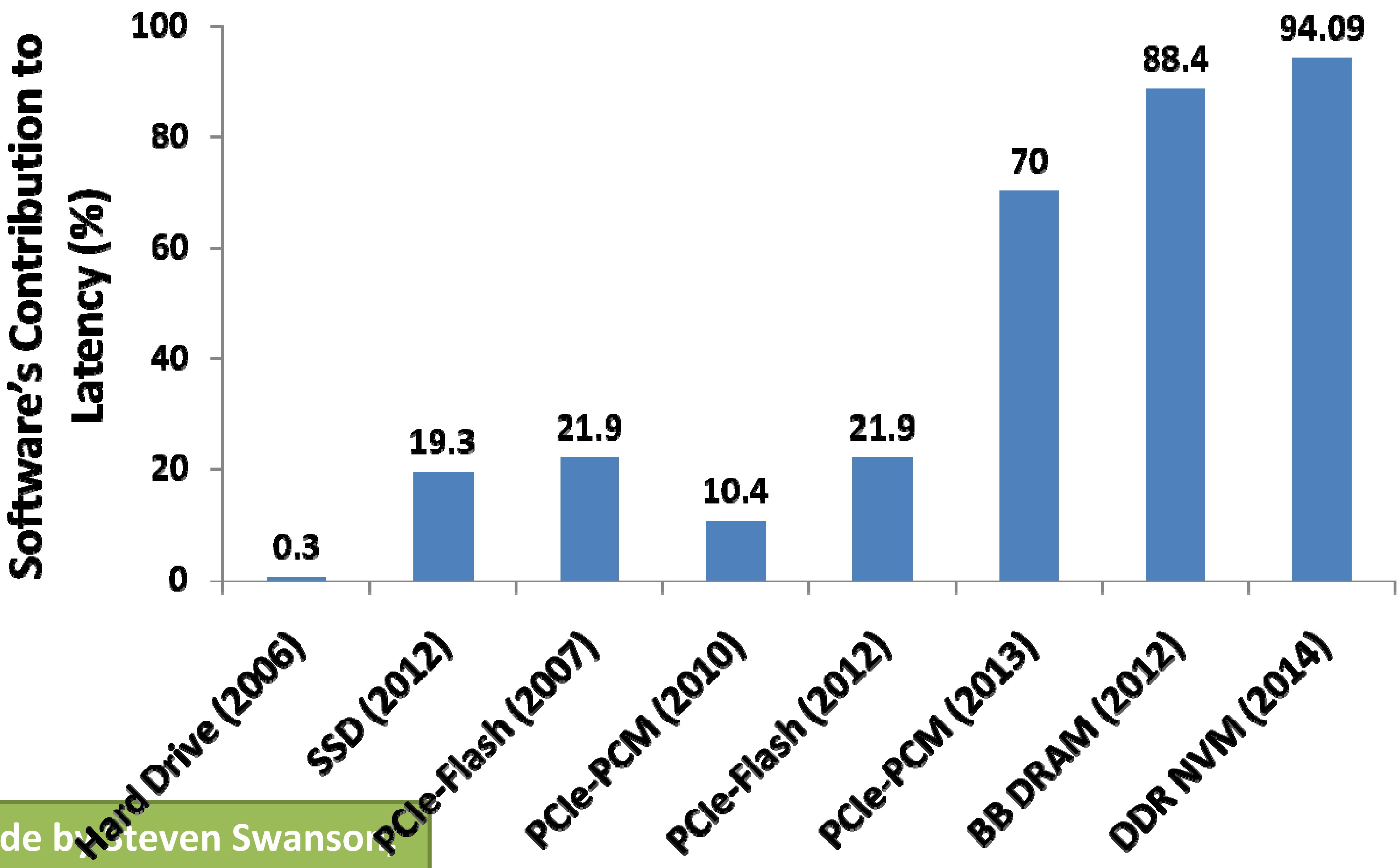
- PCM will likely offer:
  - Better endurance and latency than Flash
  - Supports writes in byte-size pieces like DRAM

Bandwidth / Latency Improvement



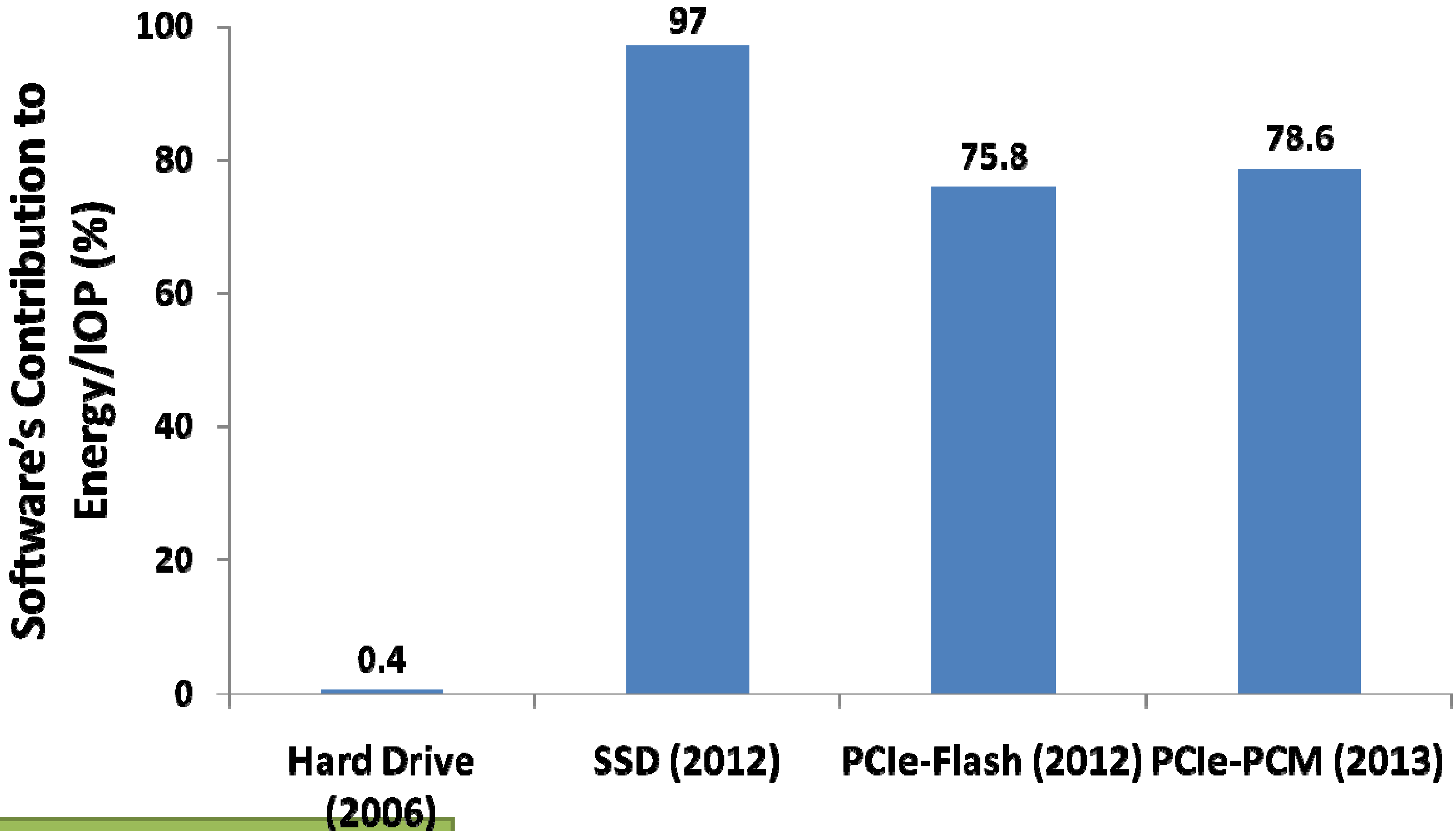
Slide by Steven Swanson, UCSD

Software Latency Will Dominate



Slide by Steven Swanson, UCSD

Software Energy Will Dominate



Slide by Steven Swanson, UCSD

Overall

- Memory further away from CPU
- Some CPUs/cores very close, others further away

  … but our s/w is not prepared for that!

- HDDs very far from the CPUs
- Flash/PCM better
- Future NVMs even better

  … but our s/w is not prepared for that!

Era of "software oblivious" speed ups is over

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

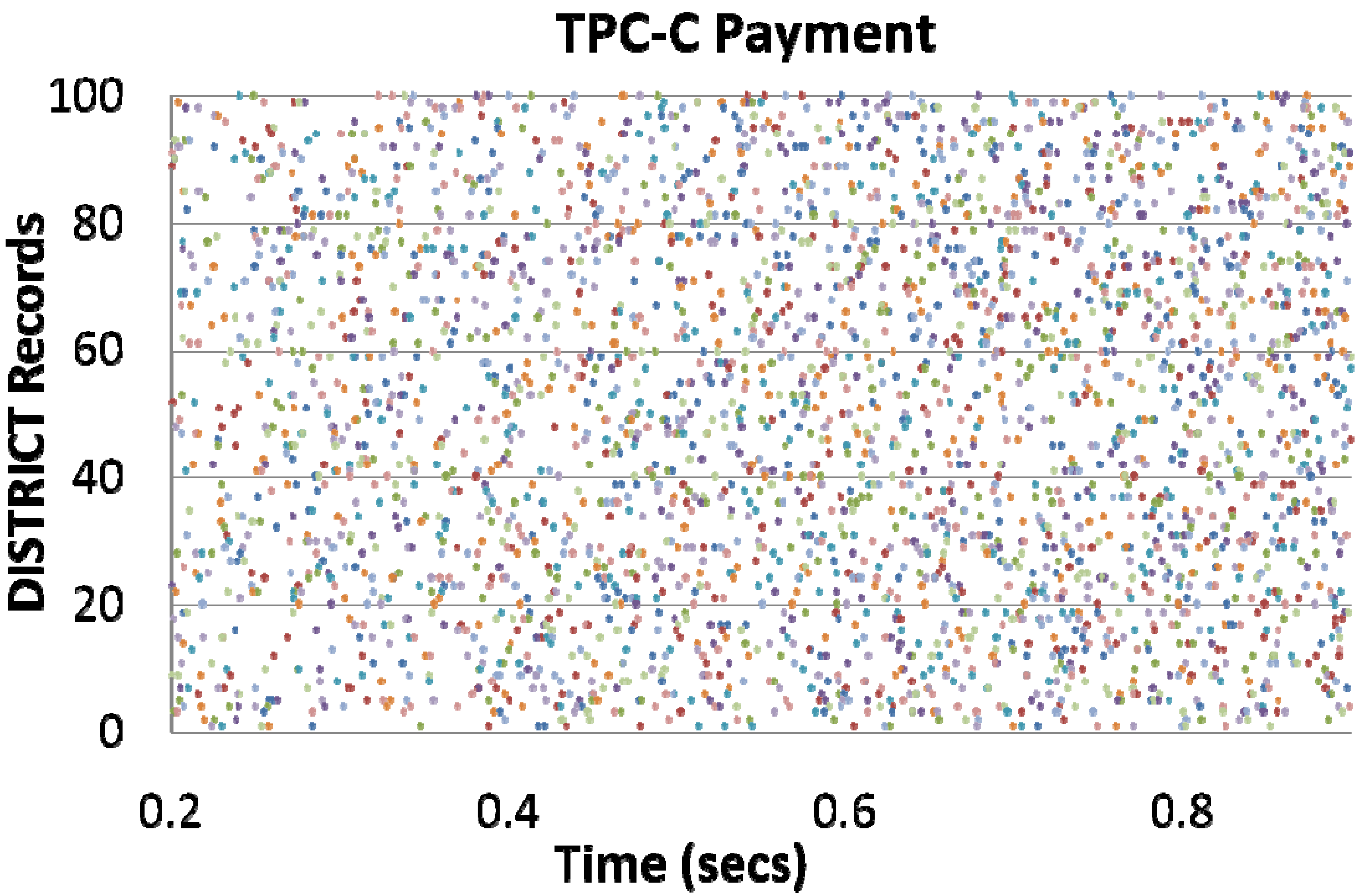
Talk Overview

- A bit of history
  - *What goes around comes around*
- Modern Hardware
  - *Hitting the walls*
- Modern Databases
  - *One size does not fit all*

Scaling Up
Scaling Out
Scientific DBs
Usability & Maintenance

Scaling Up the DB Engine

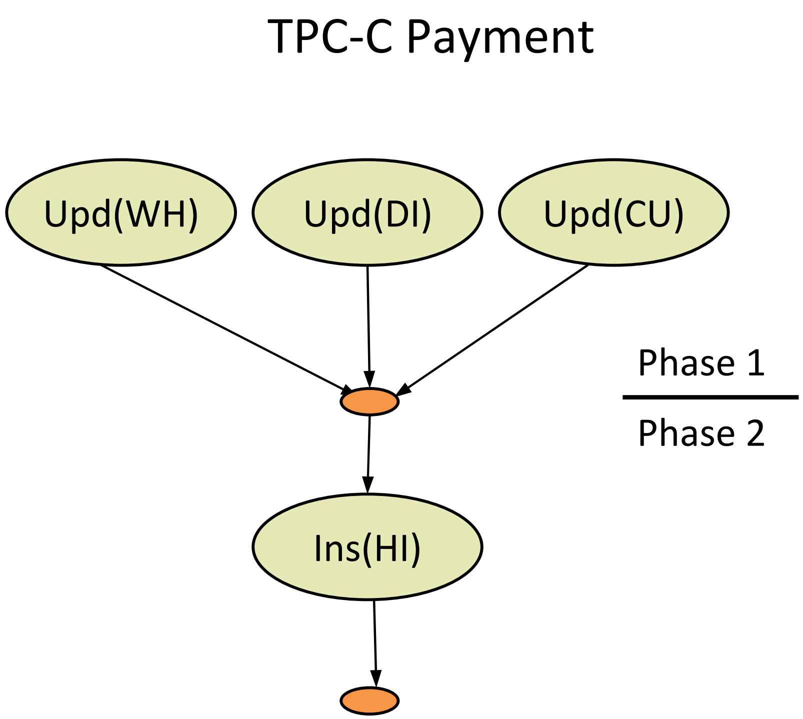
Traditional Transaction Execution



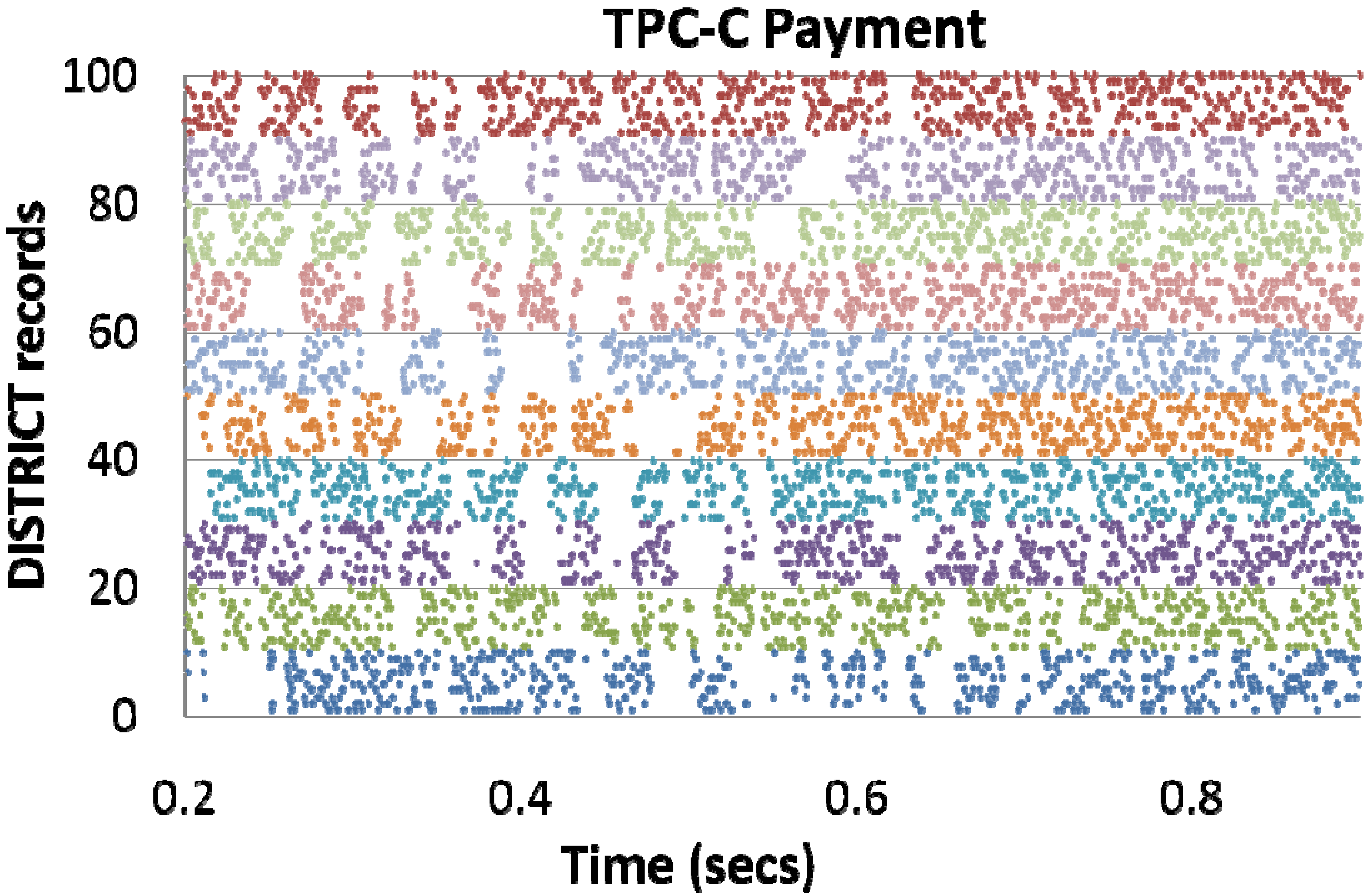
- Unpredictable access pattern
- Source of contention

Data-oriented Transaction Execution

- Each transaction input is a graph of Actions & RVPs
- Actions
  - Table/Index it is accessing
  - Subset of routing fields
- Rendezvous Points
  - Decision points (commit/abort)
  - Separate different phases
  - Counter of the # of actions to report
  - Last to report initiates next phase
  - Enqueue the actions of the next phase

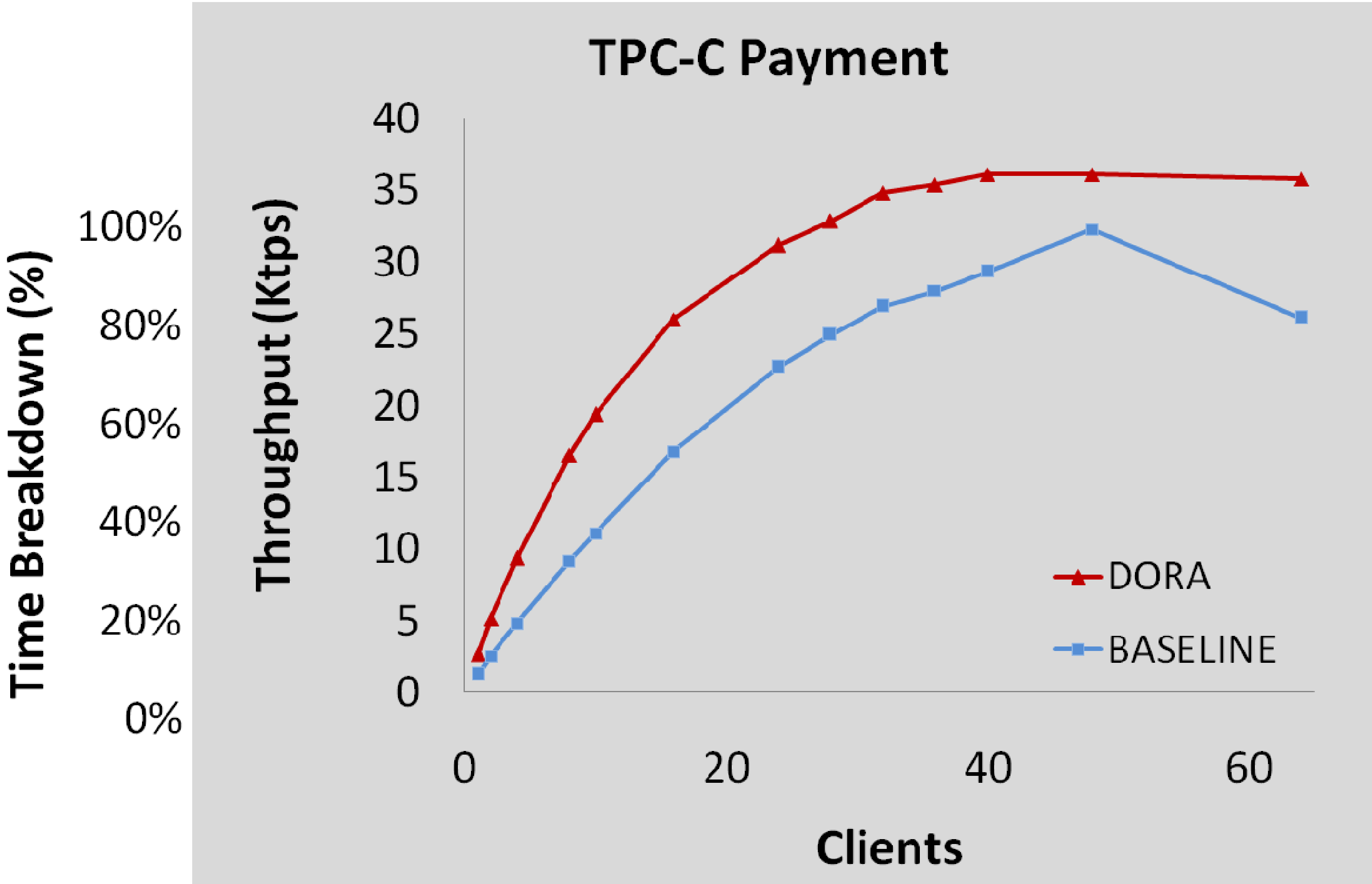


Data-oriented Transaction Execution



- Predictable access patterns
- Optimizations possible (e.g. no centralized locks)

DORA vs. Conventional – At 100% CPU



un Niagara T2
HW contexts

ra

ck Manager
nt
ck Manager

her Cont

ork

- Eliminate contention on the centr. lock manager
- Significantly reduced work (lightweight locks)

Rows, Columns and Hybrids

- Given the table:

C1	C2	C3	C4

- And the linear representation in memory:

- Let’s SUM(C1)

Memory wall: CPU spends ~95% of time waiting for data!

Rows, Columns and Hybrids

- Option 2: Column-store

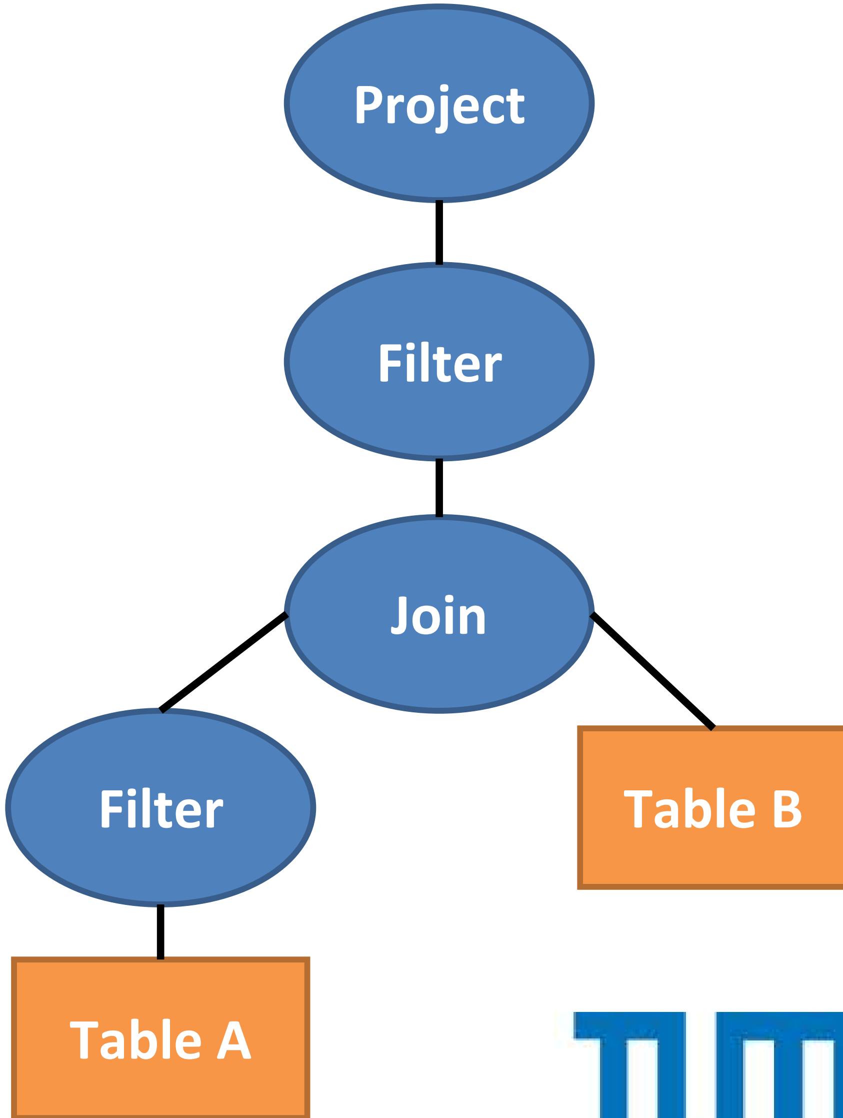
- Cache locality is King
- Major RDBMS remain row-stores but adding col-store features
  - Row-stores great for transactional workloads
  - Column-stores great for analytics (e.g. aggregations)

monetdb INGRES vectorwise VERTICA An HP Company .....

- Active research into compression, hybrids, switching between representations, qry optimiz.

Query Compilation

- Background:
  - Database engines convert declarative user queries to a tree of operators
  - Data flows bottom-up
- Idea: generate query-specific machine code at optimization time
  - Reducing Instruction Cache Misses
  - Shorter code paths
  - Query-specific optimizations

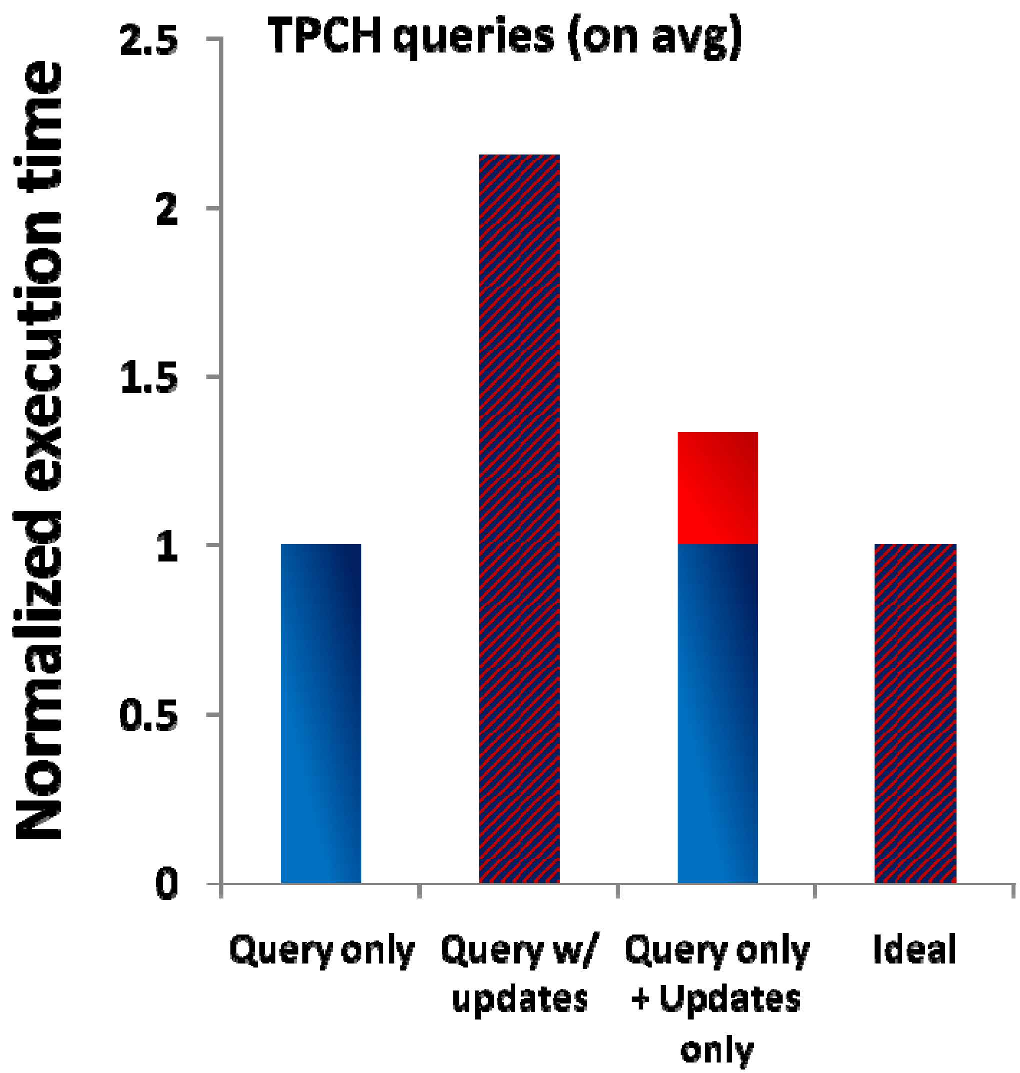


(Hardware Query Compilation)

- Idea: translate entire queries or commonly used operations to hardware circuits
- e.g., field-programmable gate arrays for data processing:
  - Low latency
  - High throughput
  - Low power
- But historically difficult:
  - Workload changes are costly!
  - But recent progress with pre-defined building blocks

SSDs and Databases

- How leverage SSDs to improve DBMS perf.?
- Example: Supporting on-line updates in a data warehouse
  - Negligible query overhead
  - Higher update rate



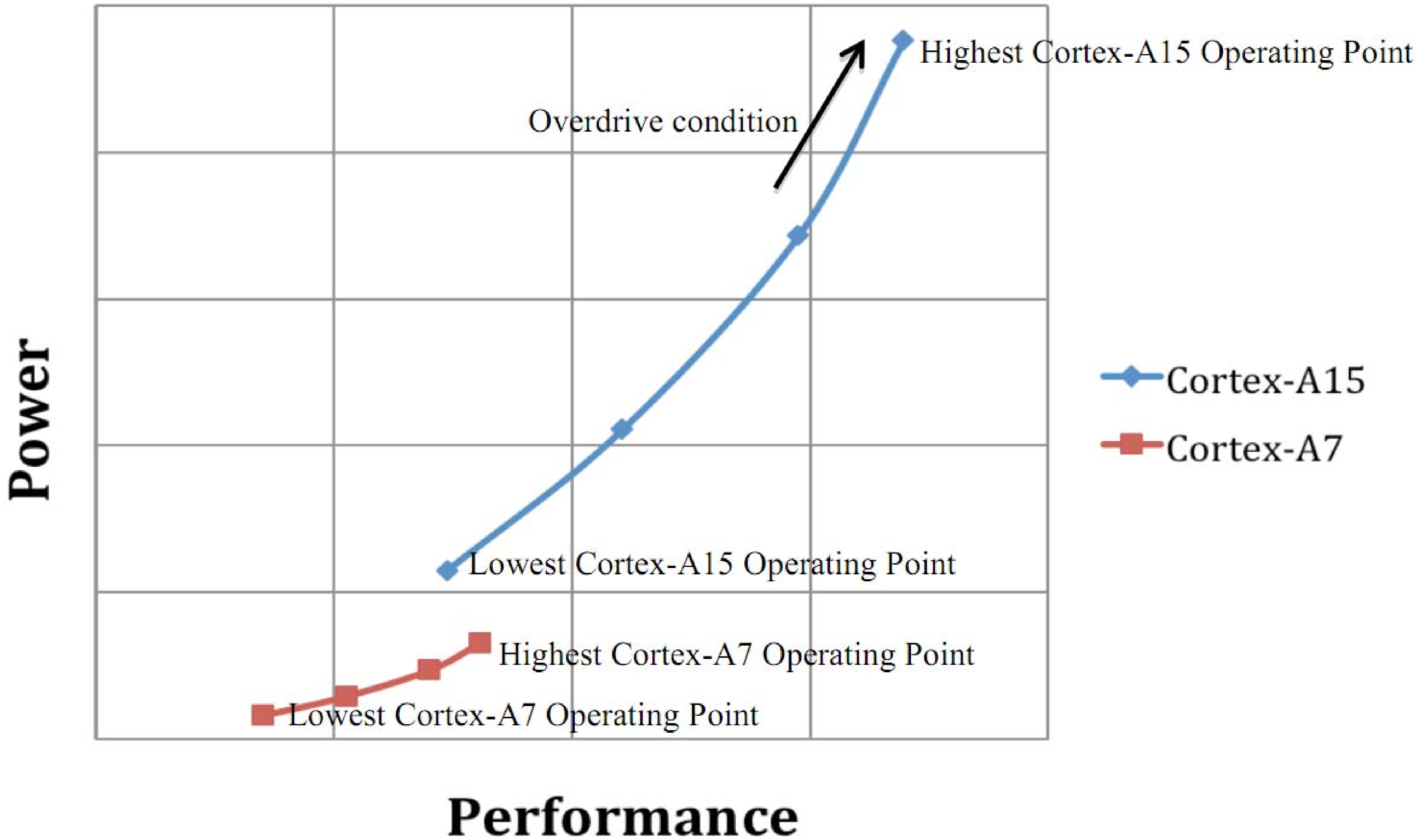


Figure 4 Cortex-A15-Cortex-A7 DVFS Curves

Programmability?

Scaling Out the DB Engine

Data-Intensive Applications and Systems

(CAP Theorem)

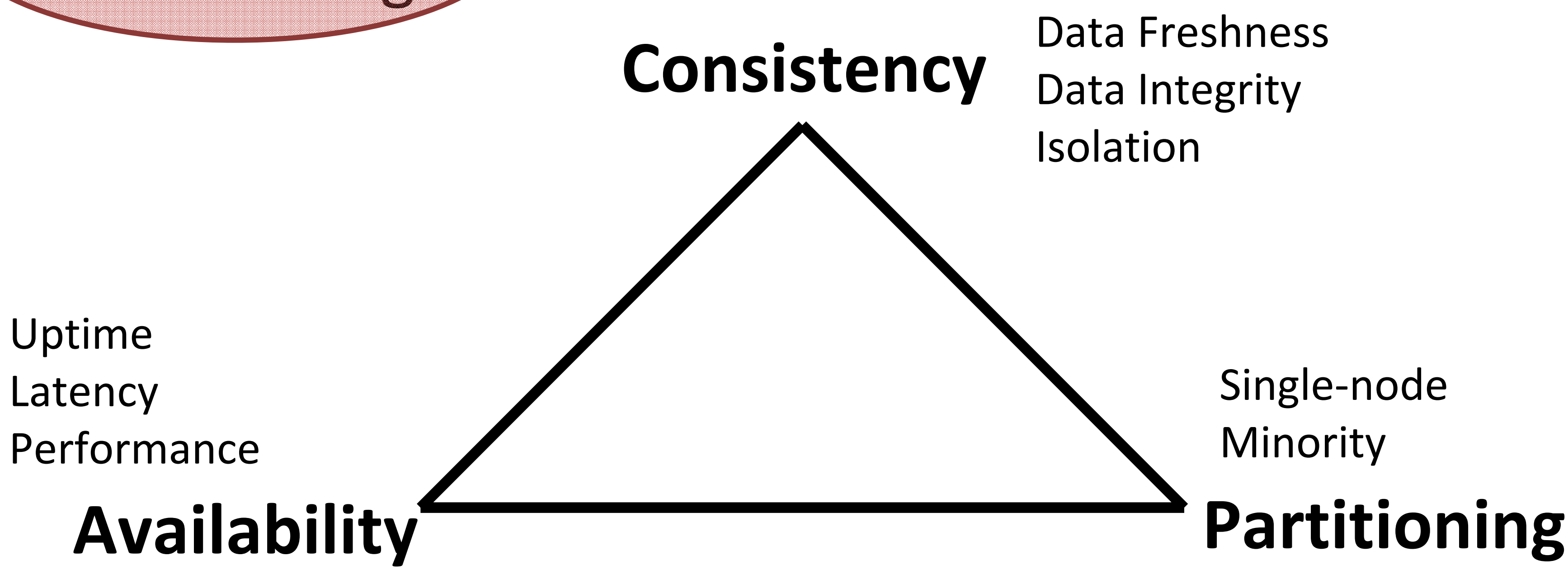
- CAP Theorem says “choose any two of

  Consistency

  Availability

  Partitioning”

~NoSQL



Use CAP as a *guide* not as a *rule*

(Eventual Consistency)

1. Be wary of ***guaranteed eventual consistency***
2. Eventual consistency is often not enough
   – Tunable knobs to strengthen consistency
3. Not easy to strengthen consistency later on, at the database or application-side

"Eventual consistency is eventually not enough"
Mehul Shah (Nou Data)

Start from consistent "core" and relax it, not the other way around

NoSQL

- NoSQL simplifies programming model and relaxes constraints
- Different design points from classical RDBMS:
  - Relational model encourages joins: may be expensive and unpredictable at scale
  - Drop or relax atomicity, consistency, isolation, durability
  - Limited support for schemas
- Individual design choices lead to very different products
- Ideal for *some workloads* and *some hardware configurations*
- NoSQL's low-entry barrier appealing
  - Despite many heterogeneous, incompatible offerings

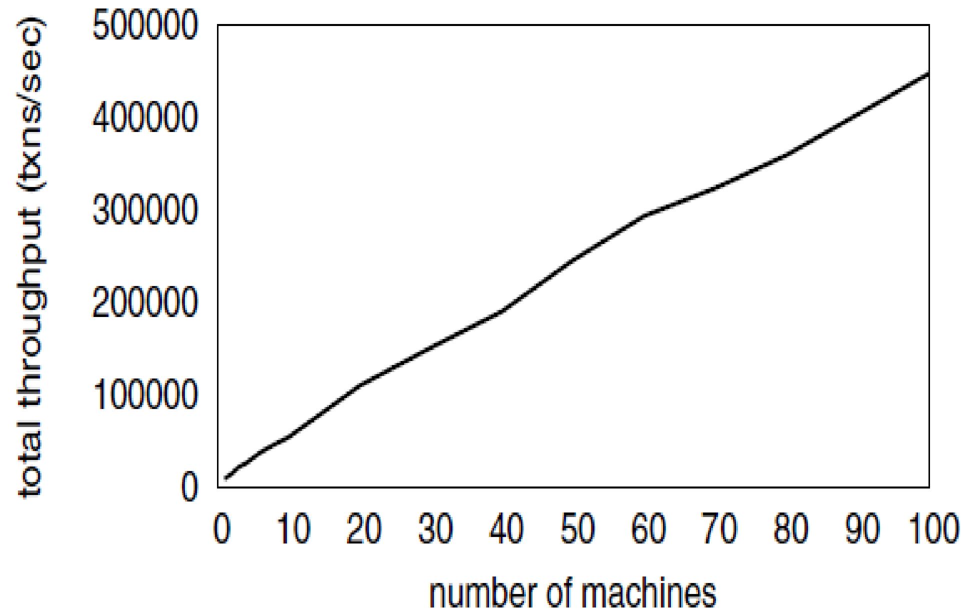
NewSQL

- ACID-compliant databases
- Highly scalable
- How: data partitioning → simplified code paths (e.g. no locks)
- But data partitioning difficult, suffer under skews
- Research: partitioning techniques, minimizing 2PC overhead, …

Other scaling-out ideas

- Giving up determinism for scalability: *Calvin*
- Give up freedom to:
  - Non-deterministically abort
  - Reorder requests on the fly
- Get
  - Scalability
  - ACID-compliance
- How?
  - Order requests in advance
  - Execute deterministically

Calvin



Yale University

Data-Intensive Applications and Systems

MapReduce

- Widely used for:
  - Extract, Transform, Load
  - *In situ* big data processing
  - Ad hoc queries
- MapReduce community:
  - Implementing indexes, views, query optimization
  - Friendlier DB-like declarative interfaces: Pig Latin, Hive
  - Combining MapReduce with RDBMS:
    - MapReduce for task distribution
    - RDBMS for queries within a worker node

Scientific Database(s)

Array DBs

- RDBMS lack support for arrays
- But array are common:
  - Scientific applications
  - Financial services
- Array DBs support:
  - Integrated storage and computation
  - Declarative languages: SciDB, SciQL
  - Query optimization

Relational Database		
I	**J**	**value**
0	0	32.5
1	0	90.9
2	0	42.1
3	0	96.7
0	1	46.3
1	1	35.4
2	1	35.7
3	1	41.3
0	2	81.7
1	2	35.9
2	2	35.3
3	2	89.9
0	3	53.6
1	3	86.3
2	3	45.9
3	3	27.6

48 cells

Array Database			
32.5	46	81.7	54
90.9	35	35.9	86
42.1	36	35.3	46
96.7	41	89.9	28

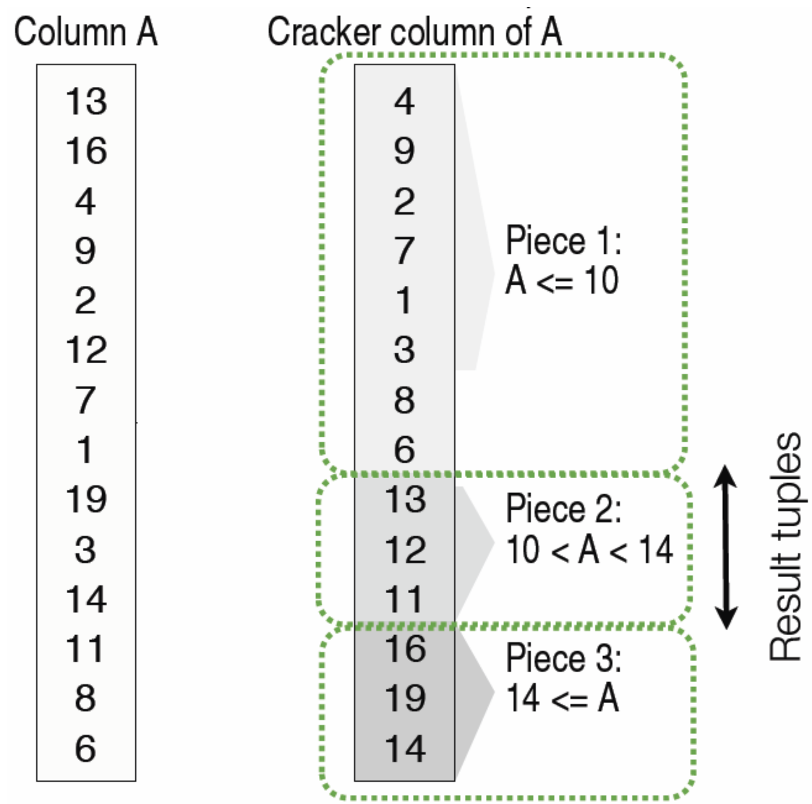
16 cells

Usability & Maintenance

Auto-tuning

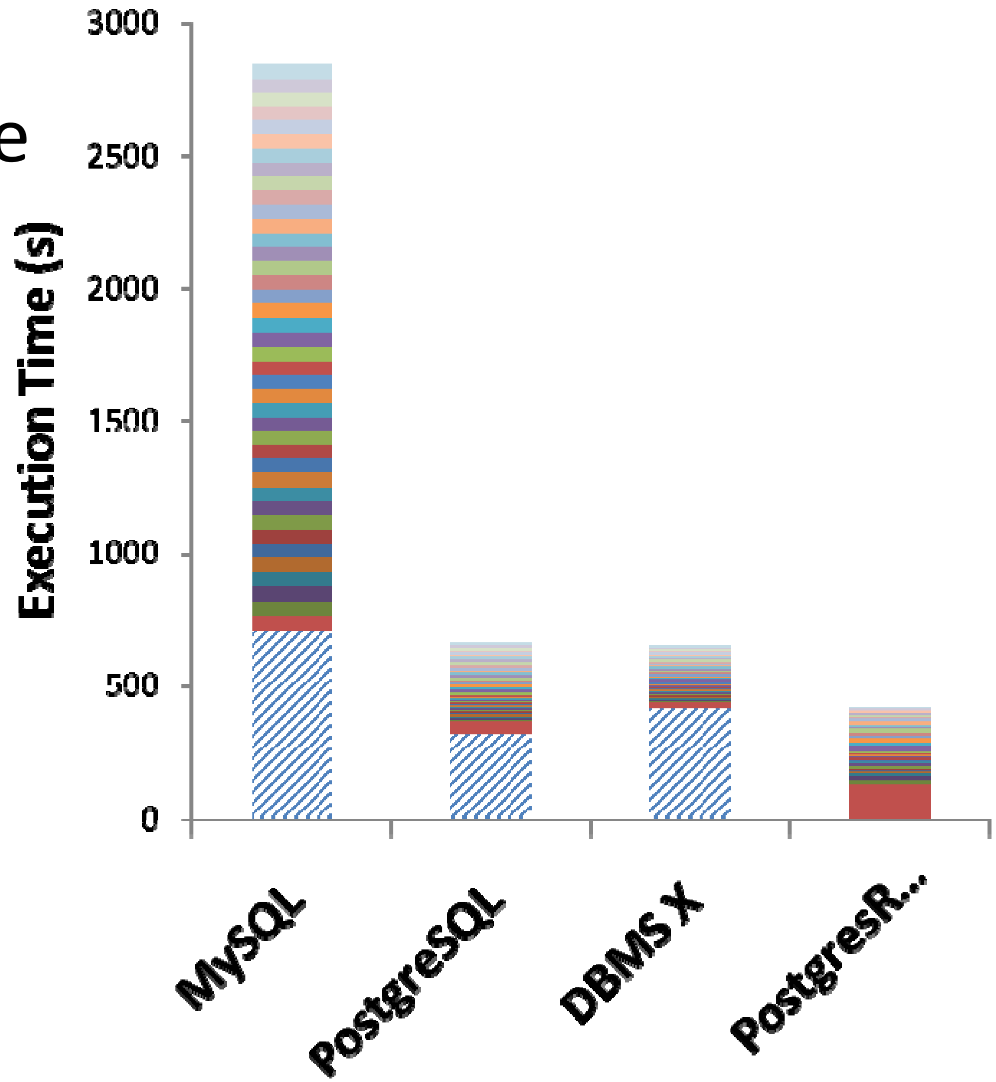
- Database tuning decisions:
  - What indexes to create?
  - Which parts of the data?
  - When to create them?
- Traditionally:
  - Offline indexing
- More recently:
  - Online indexing
  - Cracking

```
select *
from R where
R.A > 10 and
R.A < 14
```



In situ query processing

- Idea: drop proprietary DB file format, use qry. engine with seamless support for existing data files/formats/tools
- No data loading → no replication
- No vendor lock-in
- No explicit indexing → continuous adaptation, evolutionary data layout



Cloud

- The Cloud is where all previous technologies come together in easy to manage manner
  - “Smart” RDBMS
  - Storage technologies
  - NoSQL
  - MapReduce
- But:
  - Predictability?
  - Privacy?

One size does not fit all

- *Cambrian Explosion in Database Research*
  - Column stores, Arrays, MapReduce, NoSQL, Cloud, …
  - Plus many areas not mentioned: GPUs, Networking
- Disruption in hardware, software and even economics
- Expect to see a zoo of database services
  - Convergence still far out
  - Data integration a growing problem
- Expect more database services to move to the cloud
  - Even database components to be deployed as services
- Expect more h/w-driven developments
  - Dennard scaling
- Expect the DBMS paradigm – declarative data processing – to be widely adopted to ease parallel programming