“One Size Fits All”, The Death of

Michael Stonebreaker, Ugur Cetintemel

This Paper

- Current state: One RDBMS does everything
  - OLAP, OLTP, ...

This Paper

- Current state: One RDBMS does everything
  - OLAP, OLTP, ...
- Existence proof that streaming implemented on top of RDBMS engine is slow.
  - Made by a company founded by both authors.

This Paper

- Current state: One RDBMS does everything
  - OLAP, OLTP, ...
- Existence proof that streaming implemented on top of RDBMS engine is slow.
  - Made by a company founded by both authors.
- Fundamentally different abstractions are needed for fast stream processing.
  - Time to specialize database engines.

Talk about three papers

- ICDE05: One Size Fits All
  - Bad for stream processing.

Talk about three papers

- ICDE05: One Size Fits All
  - Bad for stream processing.
- CIDR07: One Size Fits All - Part 2
  - Bad for text search (Google), XML, OLAP, Scientific Data

Talk about three papers

- ICDE05: One Size Fits All
  - Bad for stream processing.
- CIDR07: One Size Fits All - Part 2
  - Bad for text search (Google), XML, OLAP, Scientific Data
- VLDB07: The End of an Architectural Era
  - Bad for OLTP

Talk about three papers

- ICDE05: One Size Fits All
  - Bad for stream processing.
- CIDR07: One Size Fits All - Part 2
  - Bad for text search (Google), XML, OLAP, Scientific Data
- VLDB07: The End of an Architectural Era
  - Bad for OLTP

The academic NoSQL movement

What Changed (b/w 1970-2005)

- Workload: Streaming, text search, even OLTP queries
  - Different consistency requirements, different performance requirements

What Changed (b/w 1970-2005)

- Workload: Streaming, text search, even OLTP queries
  - Different consistency requirements, different performance requirements
- Memory capacity
  - Enables push based architecture, reduces time per OLTP query, etc.

What Changed (b/w 1970-2005)

- Workload: Streaming, text search, even OLTP queries
  - Different consistency requirements, different performance requirements
- Memory capacity
  - Enables push based architecture, reduces time per OLTP query, etc.
- Cluster computing
  - Transaction mechanisms, failure recovery (recover from active replica)

What gets specialized

- The query interface: windowing, OLAP differences, etc.

What gets specialized

- The query interface: windowing, OLAP differences, etc.
- The types of indices that are used.

What gets specialized

- The query interface: windowing, OLAP differences, etc.
- The types of indices that are used.
- Note: Physical plans are different but I don’t know if this is a specialization
  - Depends on query and input data source.

What gets specialized

- The query interface: windowing, OLAP differences, etc.
- The types of indices that are used.
- Note: Physical plans are different but I don't know if this is a specialization
  - Depends on query and input data source.
- Arguably these are already specialized and different in RDBMSes today.

What changes universally?

- Support (arbitrary) stored procedures running in RDBMS process space
  - Context switch adds to latency, this is faster.

What changes universally?

- Support (arbitrary) stored procedures running in RDBMS process space
  - Context switch adds to latency, this is faster.
- Consider entire workload when optimizing query plan and data placement.

What changes universally?

- Support (arbitrary) stored procedures running in RDBMS process space
  - Context switch adds to latency, this is faster.
- Consider entire workload when optimizing query plan and data placement.
- Eliminate redo logs: recover by replication (cannot eliminate undo)

What changes universally?

- Support (arbitrary) stored procedures running in RDBMS process space
  - Context switch adds to latency, this is faster.
- Consider entire workload when optimizing query plan and data placement.
- Eliminate redo logs: recover by replication (cannot eliminate undo)
- Support different data structures and algorithms
  - E.g., partition data to avoid multi-threaded access, transaction differently.

One Size Fits All...

- We need to change the architecture of databases
  - But do we need to separate them out into different unrelated programs?

One Size Fits All...

- We need to change the architecture of databases
  - But do we need to separate them out into different unrelated programs?
- Why did we share in the first place:

One Size Fits All...

- We need to change the architecture of databases
  - But do we need to separate them out into different unrelated programs?
- Why did we share in the first place:
  - Code reuse

One Size Fits All...

- We need to change the architecture of databases
  - But do we need to separate them out into different unrelated programs?
- Why did we share in the first place:
  - Code reuse
  - Data reuse: similar argument to Tachyon, etc.

One Size Fits All…

- We need to change the architecture of databases
  - But do we need to separate them out into different unrelated programs?
- Why did we share in the first place:
  - Code reuse
  - Data reuse: similar argument to Tachyon, etc.
  - Management and maintainability.

One Size Fits All...

- We need to change the architecture of databases
  - But do we need to separate them out into different unrelated programs?
- Why did we share in the first place:
  - Code reuse
  - Data reuse: similar argument to Tachyon, etc.
  - Management and maintainability.
  - Economics?

What does specialization buy us anyhow

- Performance gains (how much?)

What does specialization buy us anyhow

- Performance gains (how much?)
- Simpler code (see also Unix philosophy)

What does specialization buy us anyhow

- Performance gains (how much?)
- Simpler code (see also Unix philosophy)
  - But more code.

What does specialization buy us anyhow

- Performance gains (how much?)
- Simpler code (see also Unix philosophy)
  - But more code.
- Pick and choose from different vendors?

What does specialization buy us anyhow

- Performance gains (how much?)
- Simpler code (see also Unix philosophy)
  - But more code.
- Pick and choose from different vendors?
  - I don’t know if this can realistically happen.

Where are we Today

- Where are we today: the architecture changed
  - OLAP :- Move to column stores.
  - OLTP :- SAP Hana, Hekaton, etc. (memory based)
  - All of the stuff we talked about this semester.

Final Questions

- Stored procedures: why are we giving up on isolation?
- Should we be separating out execution engines for cluster computing
  - Remember Spark Streaming, GraphX, Naiad, ...