Modular Data Storage with Anvil

Mike Mamarella, Shant Hovsepian, Eddie Kohler

Presented by Guozhang Wang

DB Lunch, December 30th, 2009

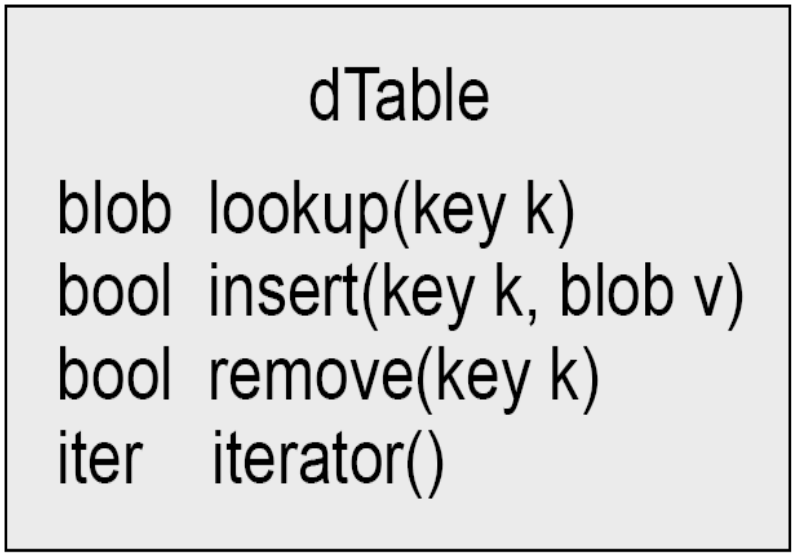
Several slides are from the authors

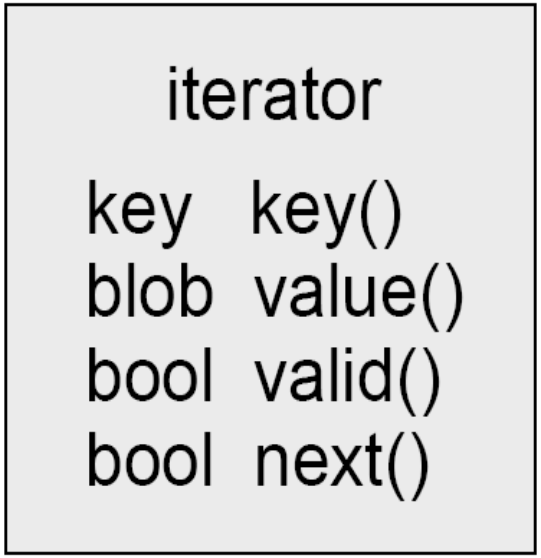
Motivation

- Custom Data Stores
  - can greatly outperform conventional systems by 100x for specific work loads
  - are often written ***monolithically***
- What if application has characteristics of both OLTP and warehousing?
- We need a modular and extensible toolkit to build new data store layouts

Anvil

- Fine-grained *dTables*: abstract key/value
  - Keys are integers, floats, or strings
  - Values are byte arrays
  - ***Iterators*** support in-order traversal
  - Most are read only



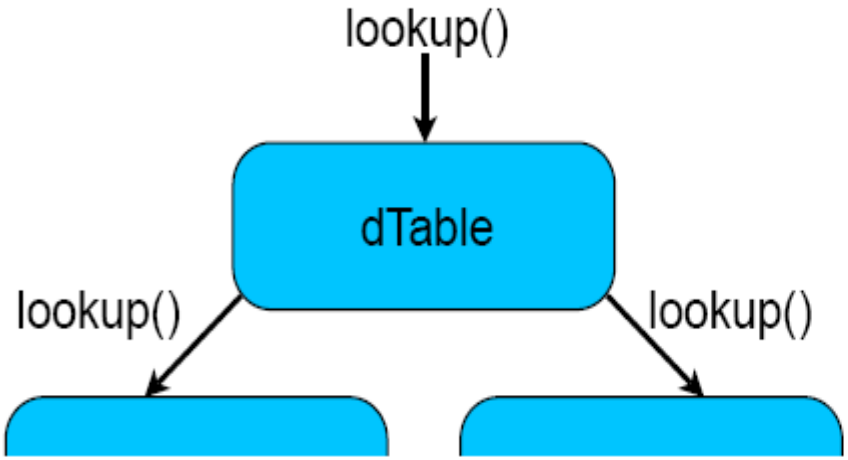


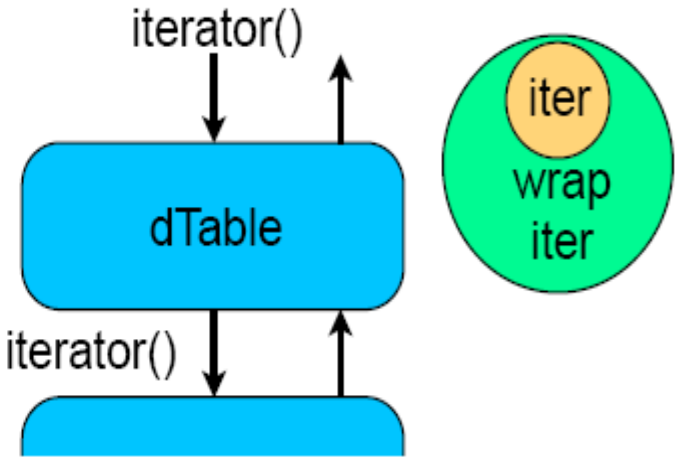
How to build DBMS from dTable

- How to build indexing, hashing, etc using dTables?
- How to handle writes efficiently?
- How to handle transactions?

#1 dTable Layering

- dTables can be built over other dTables using the same interface
  - Storage dTable
  - Performance dTable



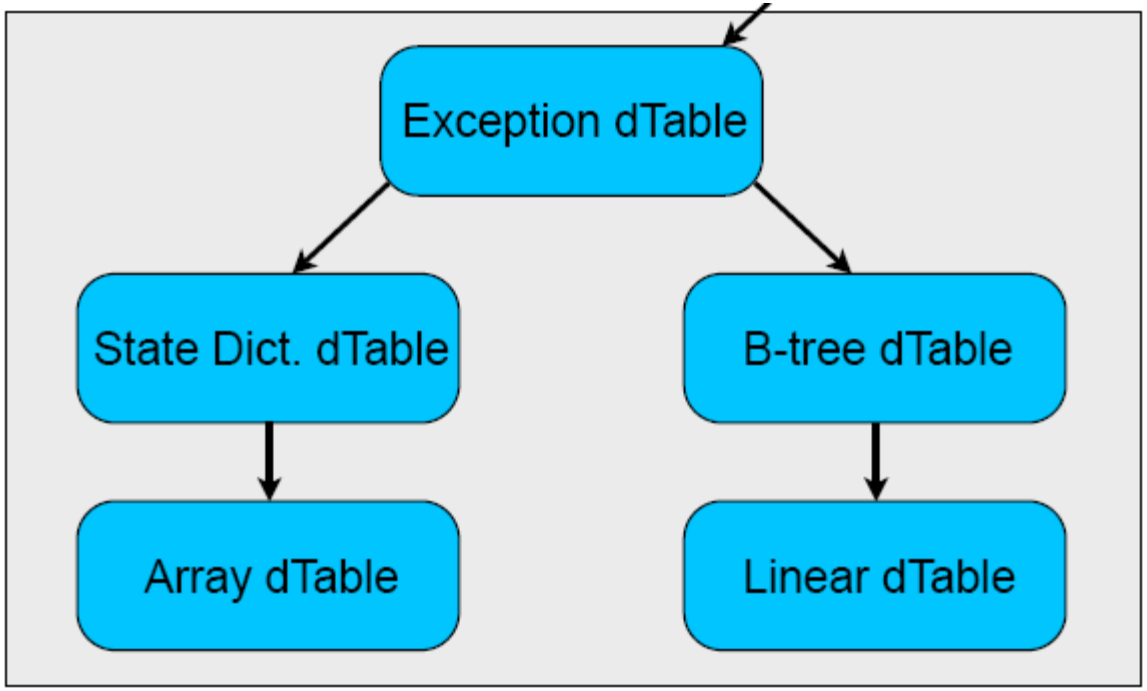


dTable Layering

- Exception dTable
  - Combines a "restricted" dTable with an "unrestricted" dTable
- E.g., want to store the state of residence of customers
  - Identified by mostly-contiguous IDs
  - Most live in the US, but a few don't

Exceptional dTable

- Restricted handled by array dTables (contiguous integer keys, fixed size values)
- Unrestricted handled by linear dTables

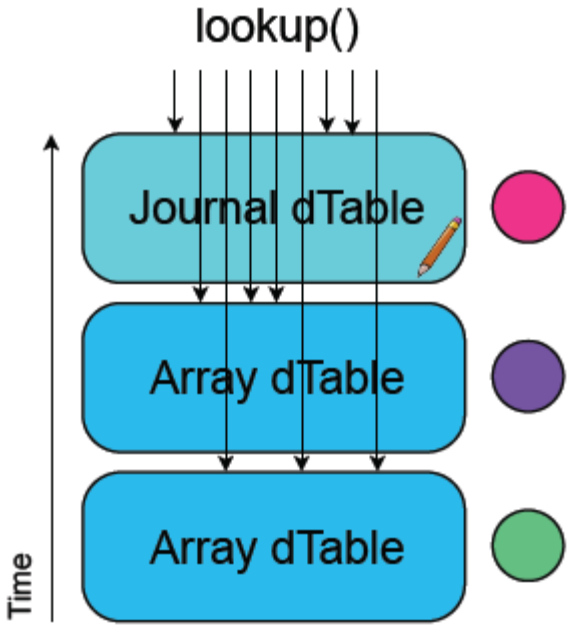


#2 Writable dTables

- Isolates all writing to dedicated writable dTables
- Journal dTable
  - Append-only store for new/updated data
  - Periodic "digestion" to read-only dTables when it gets large
- Combine write-optimized and read-only dTables into single logical dTable: ***Overlay***

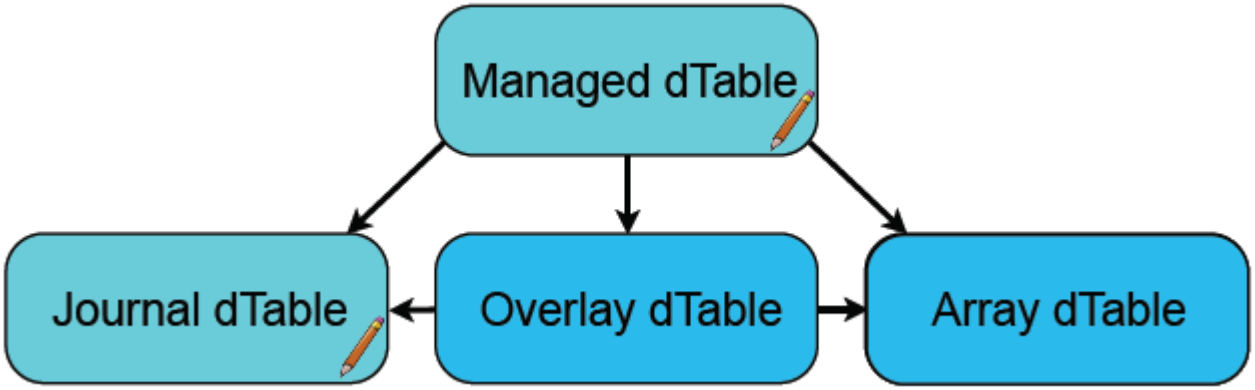
Overlay dTable

- Built over two or more dTables, usually one writable and multi read-only.
- Iterator merges all underneath dTables' iterators for reads
- Older "lower" data can be overridden by newer "higher" data



#3 Managed dTable

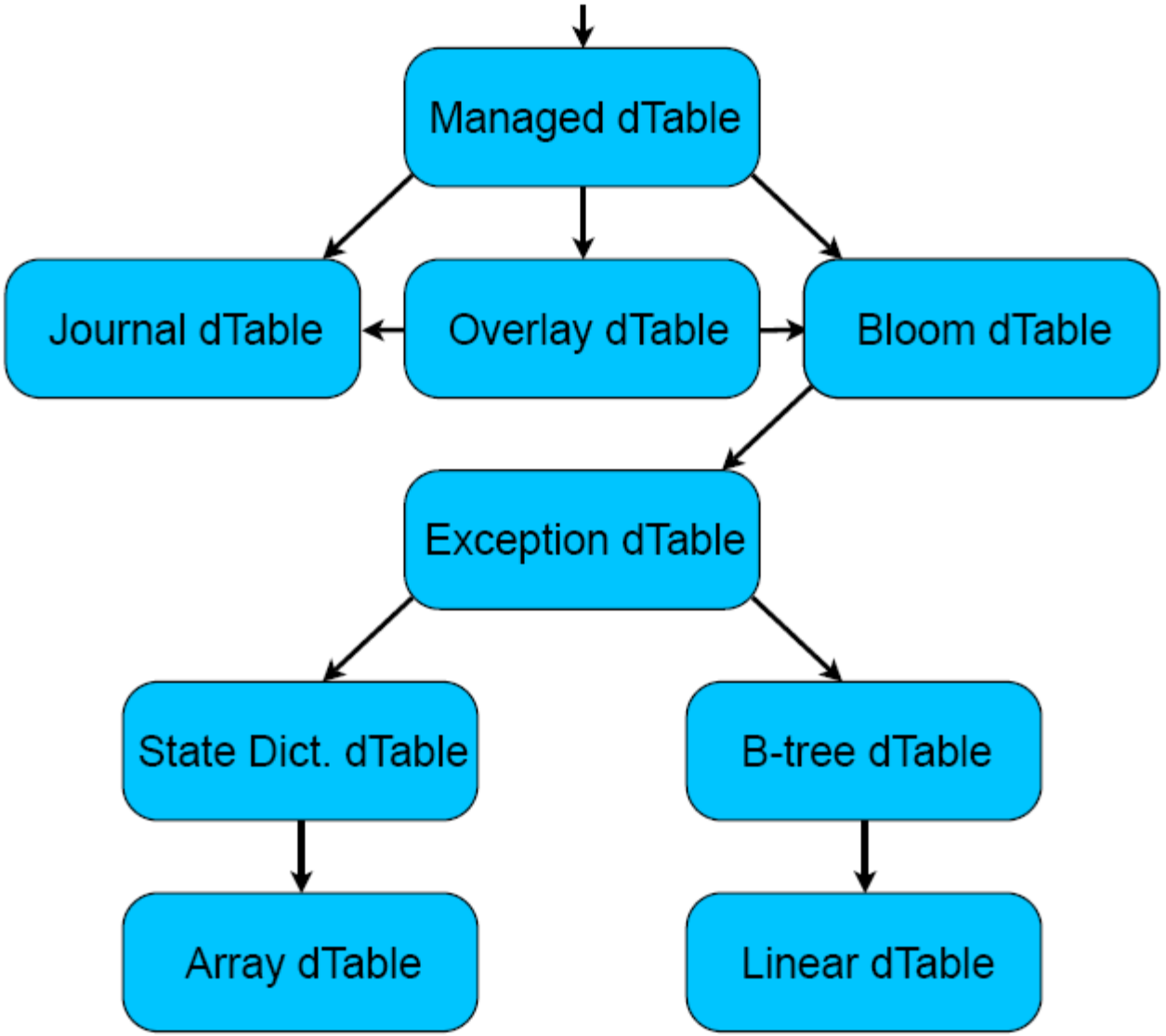
- Interfaces with transaction library, which keeps transaction logs
  - Always consistent
  - User decide durability
- Also decides policy for digesting journal dTables and combining read-only dTables



dTables in summary

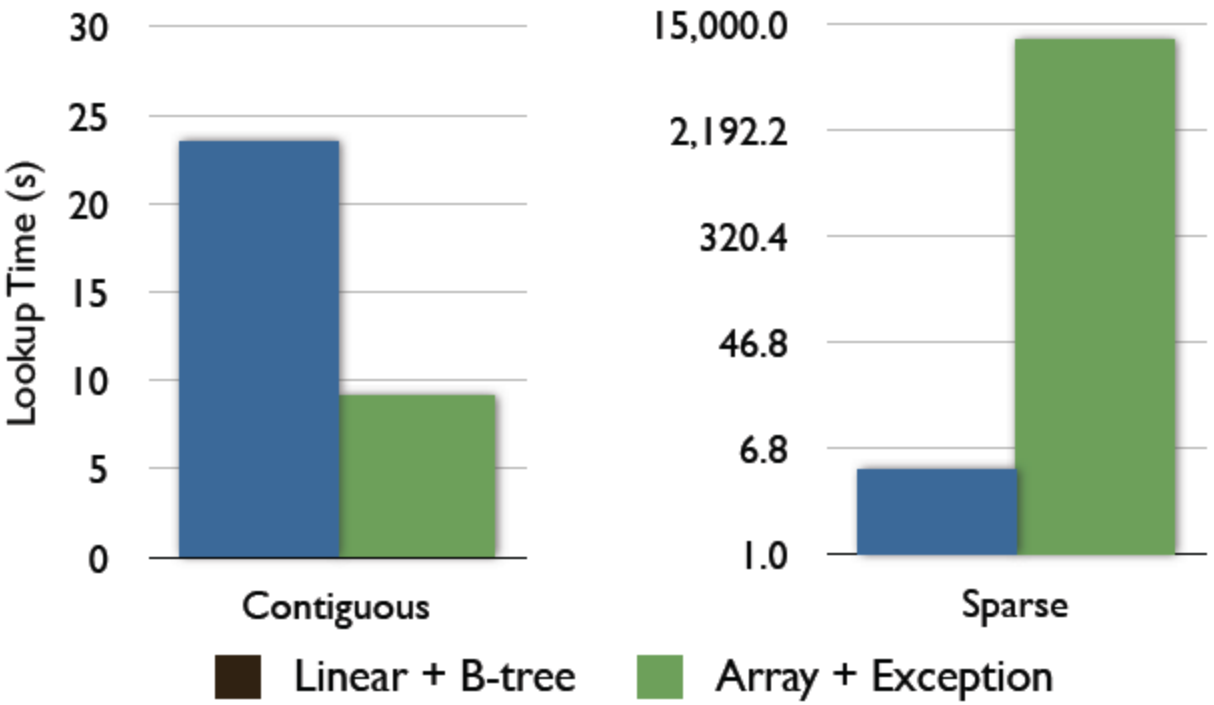
- Storage dTables: linear, fix-sized, array, memory, journal, etc
- Performance dTables: b-tree, bloom filter, cache, etc
- Unifying dTables: exception, overlay, managed

Customer State Residence Example



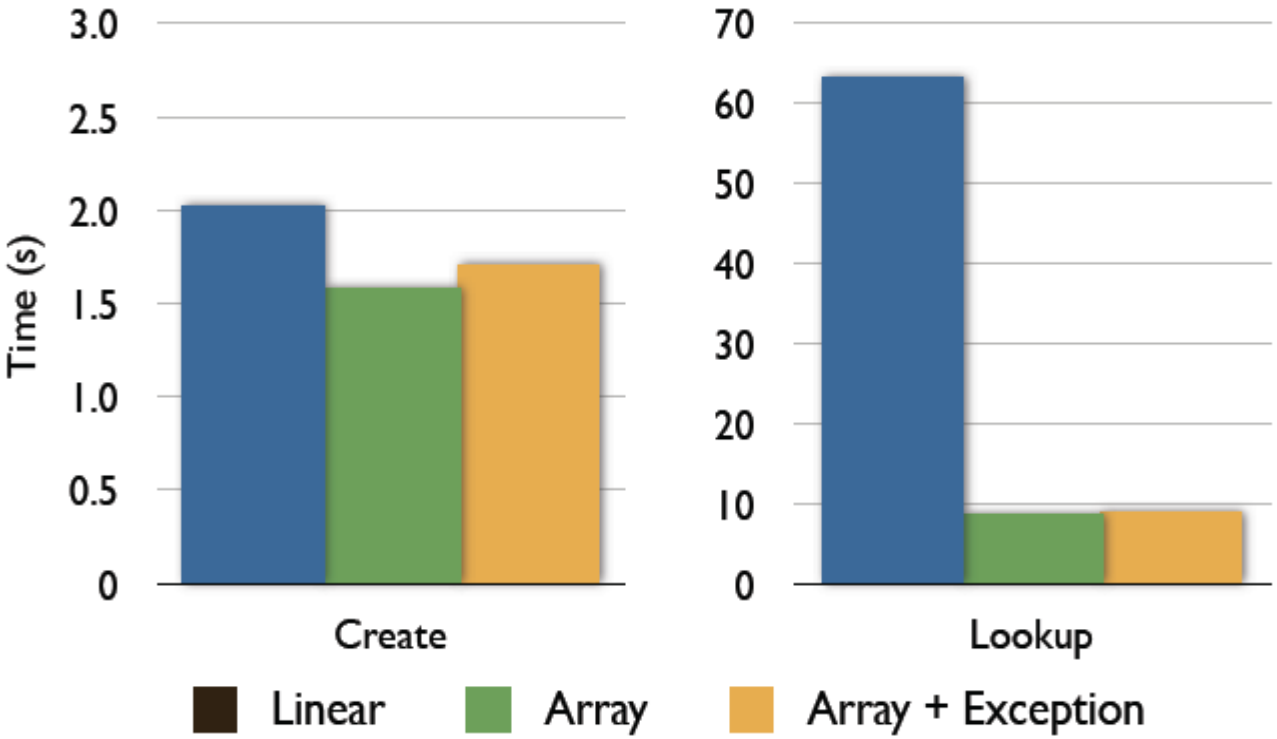
Modularity

- Linear + B-tree vs. Array + Exception
  - Keys: contiguous or spaced 1000 apart



Exception dTable Low Overhead

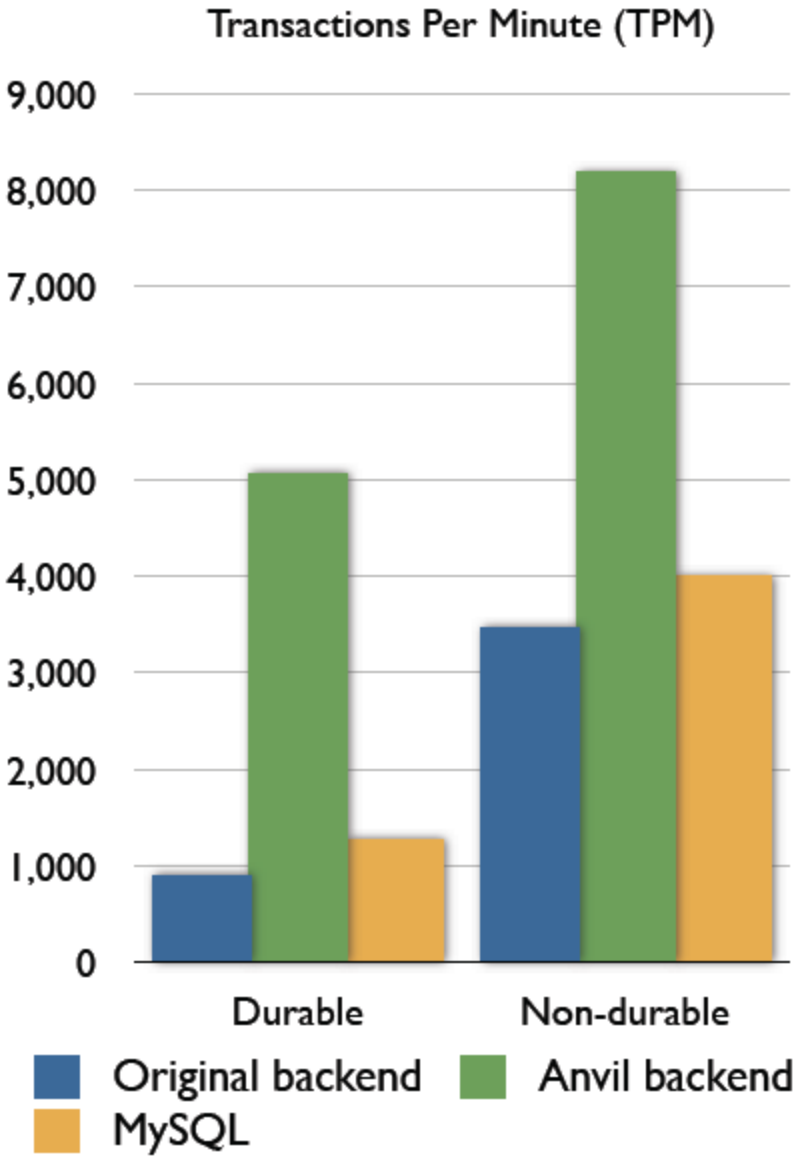
- Linear vs. Array vs. Array + Exception



- Exception dTable is low overhead vs. array but restores full functionality

Read/Write Separation

- Anvil's durable and non-durable config outperformes origina durable and non-durable config



Questions ?