# SOSP 2009 I/O Review

Richard P. Spillane

Stony Brook University

http://www.fsl.cs.sunysb.edu/

STONY BROOK
STATE UNIVERSITY OF NEW YORK

# Summary

- Three papers in SOSP I/O
  - Modular Data Storage with Anvil
  - Operating System Transactions
  - Better I/O Through Byte-Addressable, Persistent Memory (PCM)
- I will only discuss the first two

# Anvil

- Implement some common data stores
- Maximize code re-use by:
  - Employing a modular architecture
  - Using a single interface (dTable)
  - Providing a bestiary of various dTables
  - Separation of concerns
- Once we're done, let others use these modules also to make their own stores

# What is a dTable?

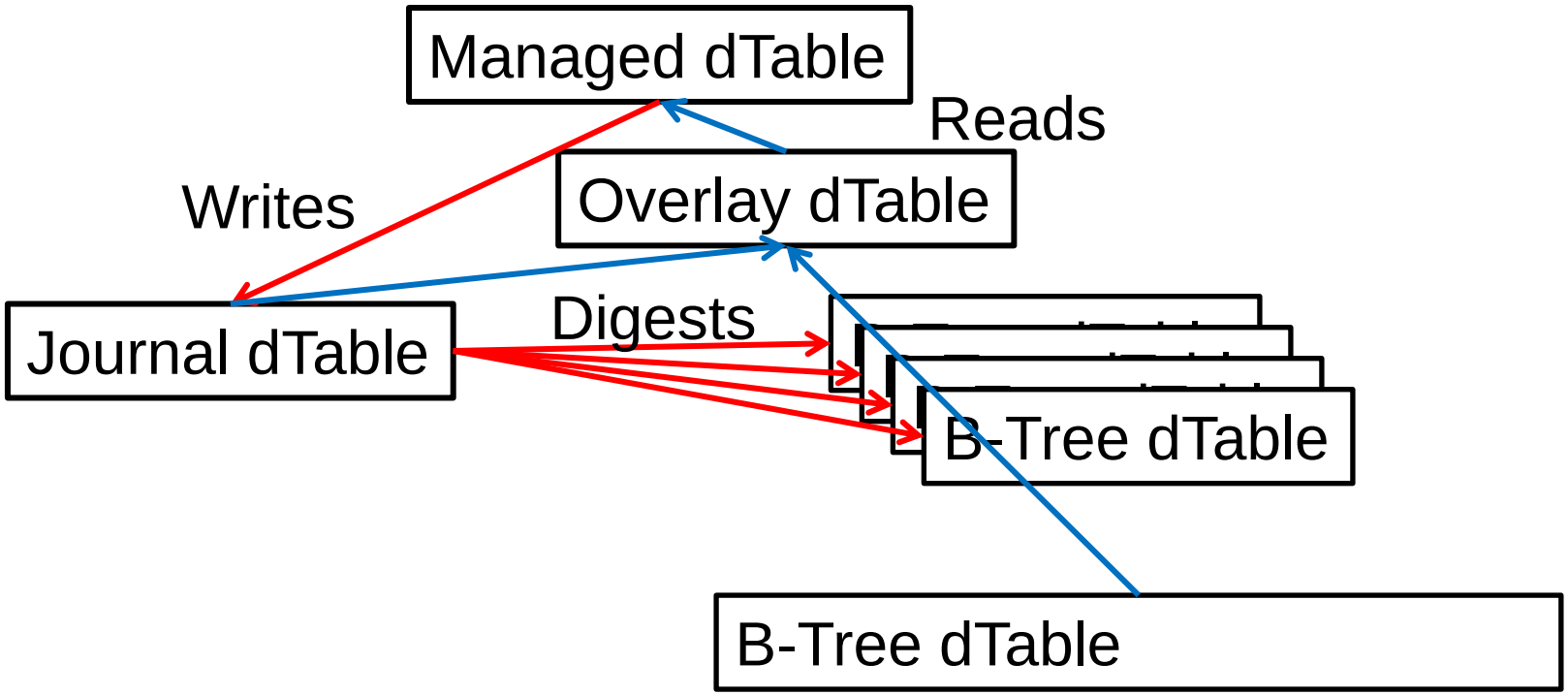
- A dTable is a standard key-value store interface:
  - bool **contains**(key)
  - value **find**(key)
  - Iter **iterator**()
    - bool iterator::**seek**(key)
    - …
  - int **insert**(key, value)
  - int **remove**(key)
  - …

# What's the Magic Sauce?

- Not every dTable is a B-Tree
  - We have 'Exception' dTable
  - We have 'Journal' dTable
  - We have 'Bloom' dTable
  - …
- Many of these dTables aren't stores at all and are implemented in terms of lower dTables (Overlay)

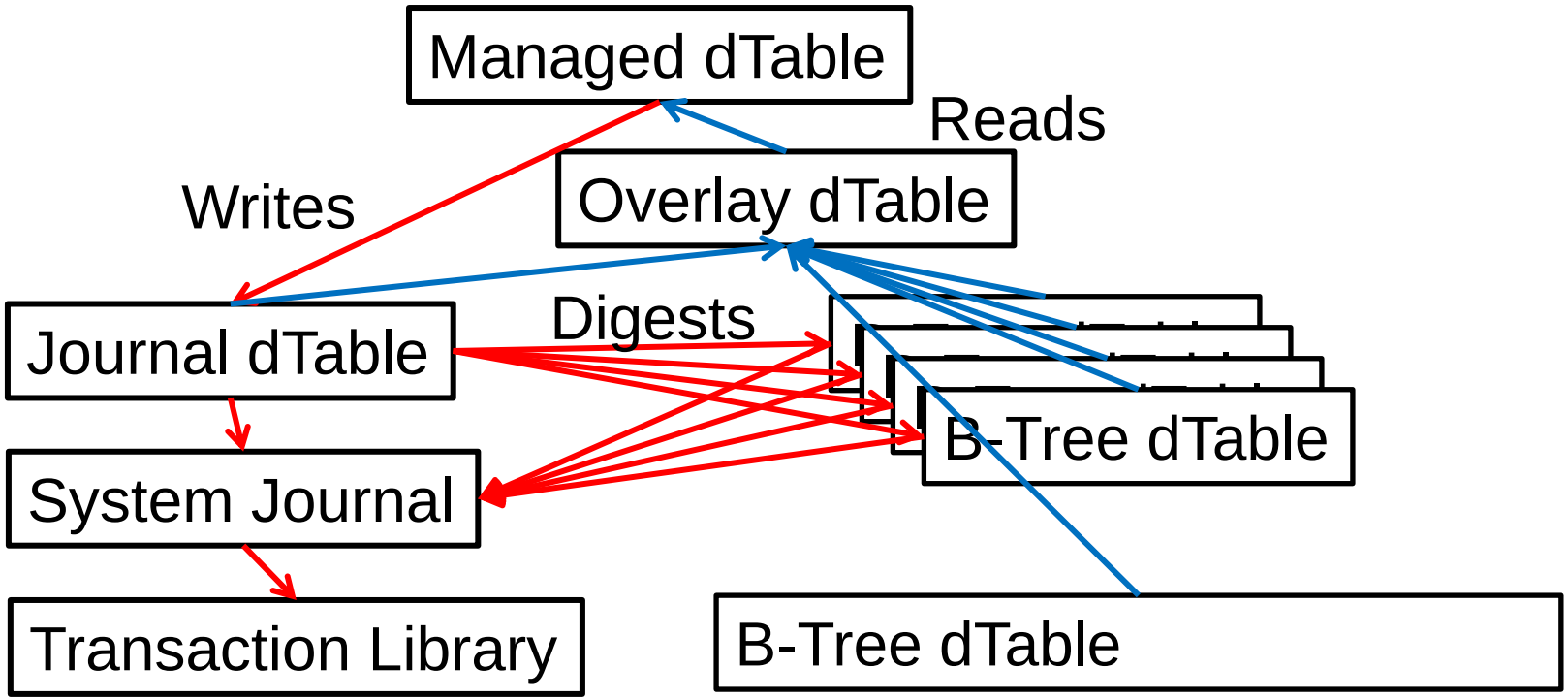
# Anvil's Key-Value Store

- Anvil uses a Managed dTable for key-value store



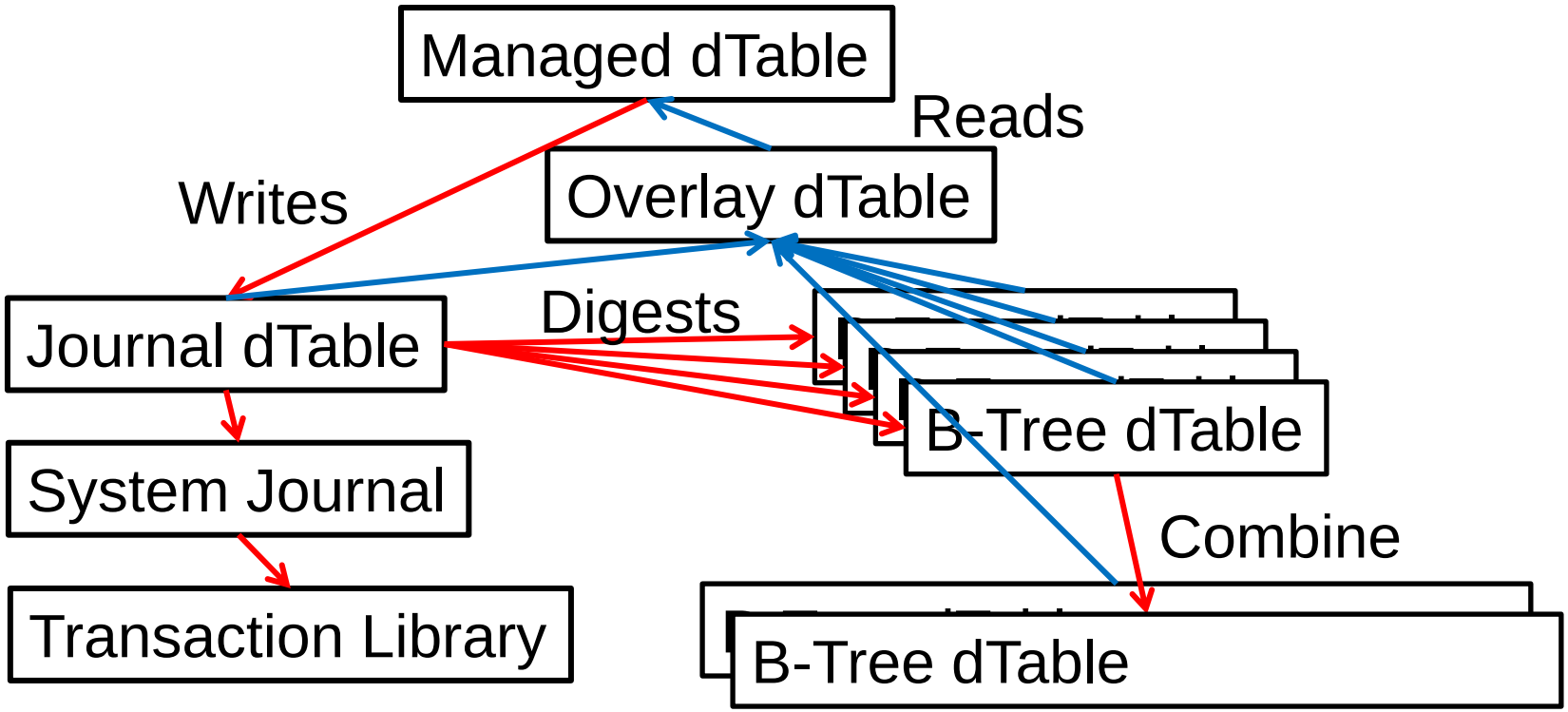
# Anvil's Key-Value Store

- Anvil uses a Managed dTable for key-value store



# Anvil's Key-Value Store

- Anvil uses a Managed dTable for key-value store



# Other dTables:

- Exception
- B-Tree (Index)
- Linear (Packed Key-Value pairs)
- Array dTable
- Fixed-size dTable
- Small Integer dTable
- Delta Integer dTable
- Bloom Filter dTable…

# Message Passing

- Use RAM to communicate messages
  - Upper dTable calls lower dTable routines
  - Lower dTable routines return to upper

# Anvil Performance

- Hypothesis: Anvil provides access to most performance tradeoffs
  - Different Anvil configs tested
  - Look for expected results (e.g., Bloom filter increases lookup of some keys)
- Hypothesis: Anvil beats typical SQLite back-end when tweaked
  - On DBT2 TPC-C implementation
    - Anvil: 5066 TPM, Reqsz 24.68KiB, 1077.5 W/s
    - Orig.: 905 TPM, Reqsz 8.52KiB, 437.5 W/s

# Thanks!

# Questions (I have)

- dTable can be implemented in BDB or some other DB library?
- dTable performance over SQLite is huge, and details on that benchmark and its marked difference are scant
  - SQLite the right DB to compare against?
    - Benchmark implementation constraints?
  - What is Anvil's point-query performance?
    - Shouldn't be good, Overlay has to do log_2(N) queries across disparate B-Trees