TWIAD: Why You Should Index In Write-Optimized Databases

SAND2016-0692C

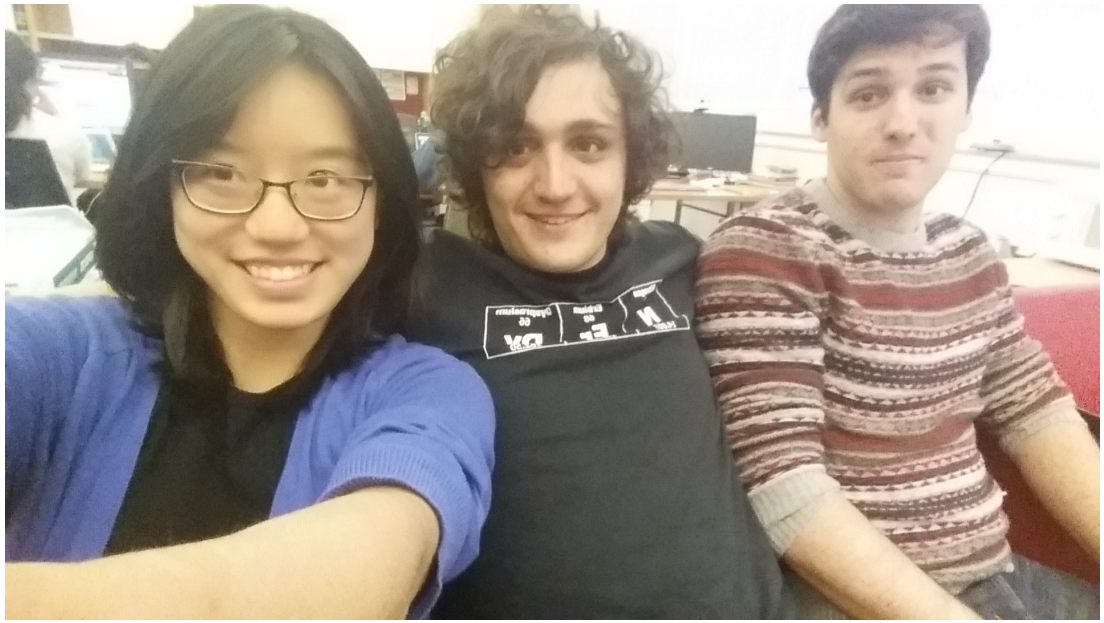
Bridger Hahn[12], Helen Xu[12], Thomas Kroeger[1], Nolan Donoghue[12], and David Zage[3]

{bridger.hahn, helen.xu, nolan.donoghue}@stonybrook.edu
tmkroeg@sandia.gov
zage@cerias.net
[1]Sandia National Laboratories, Livermore, CA, USA
[2]Department of Computer Science, Stony Brook University
[3]Intel Corporation, Santa Clara, CA, USA

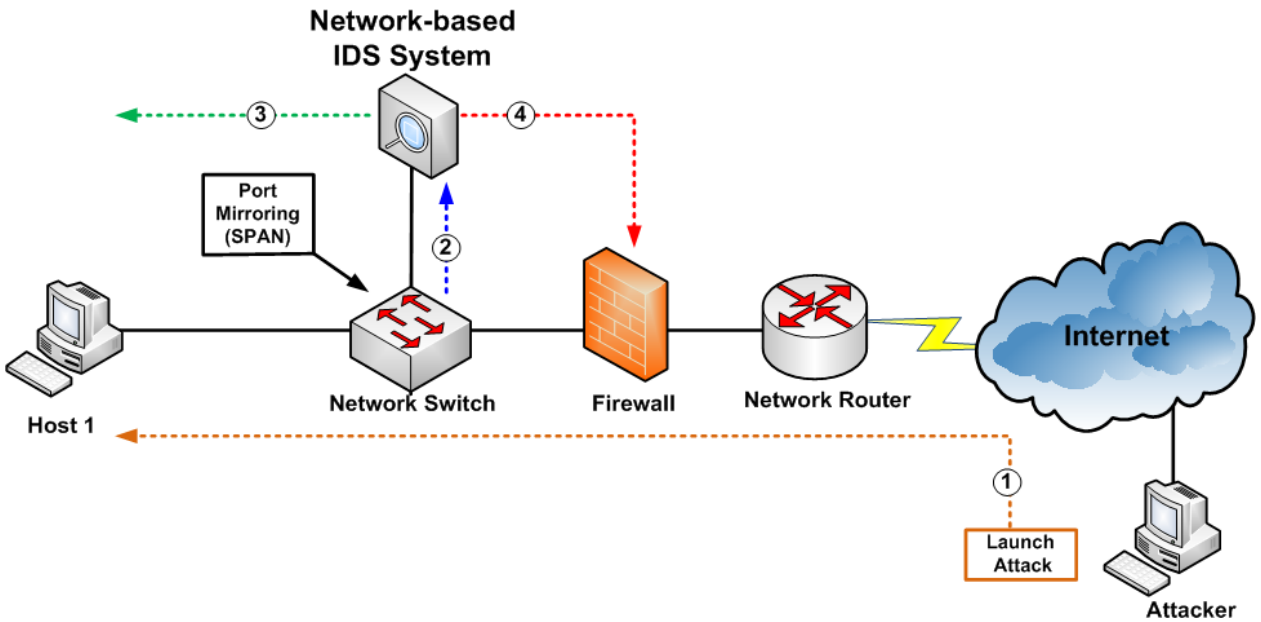
October 28, 2015

TWIAD: Why You Should Index Your IDS Data In Write-Optimized Databases

TWIAD the
Write-Optimized
IP
Address
Database



TWIAD: Why You Should Index Your **IDS** Data In Write-Optimized Databases



IDS as we know it

TWIAD: Why You Should Index Your **IDS Data** In Write-Optimized Databases

Your friendly neighborhood IDS monitor

- State of the art IDS involves logging all connections across the network
- Logging is bad but indexing is too slow
- Why should an IDS do the work of maintaining a database?

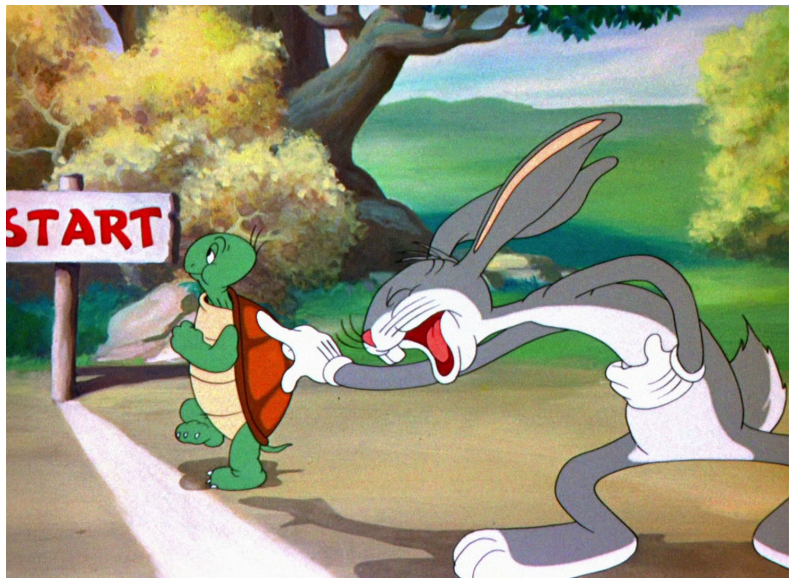
TWIAD: Why You Should **Index Your** IDS **Data In** Write-Optimized **Databases**

- Indexes facilitate faster queries
- "The right read problem is a write problem"
- Slow queries indicate bad/insufficient indexes

Indexing vs Logging

TWIAD: Why You Should Index Your **IDS Data In** Write-Optimized **Databases**

- The pipe can be very big
- Connection logs generate quickly
- Lower budget = better
- How can we keep up and store all of this data in multiple indexes?



At first, logging is faster

TWIAD: Why You Should Index Your IDS Data In Write-Optimized **Databases**

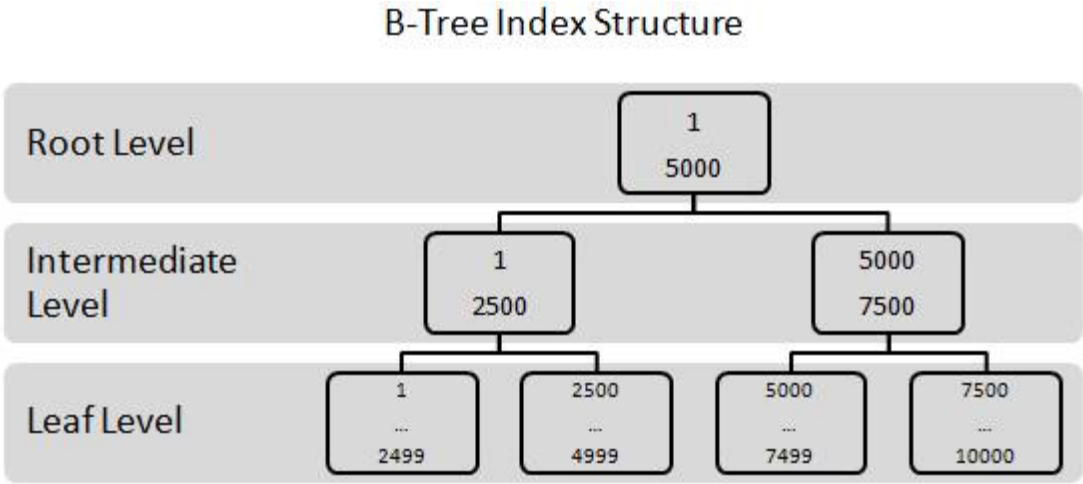
Goal: Ingest and query network data quickly

- More indexes expedite query but slow insertion
- Age-old tradeoff
- What if we could ingest more quickly to offset this tradeoff?

Logging falls flat when you get to queries

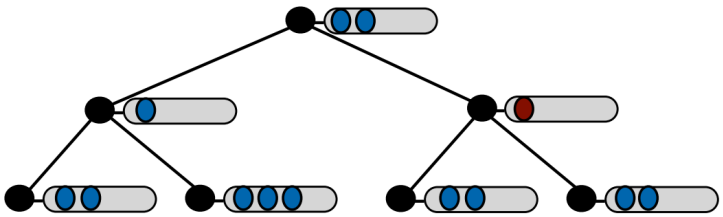
TWIAD: Why You Should Index Your IDS Data In **Write-Optimized Databases**

- Start with a B-Tree
- Data stored at the leaves
- Insertion requires tree traversal
- Same cost as query

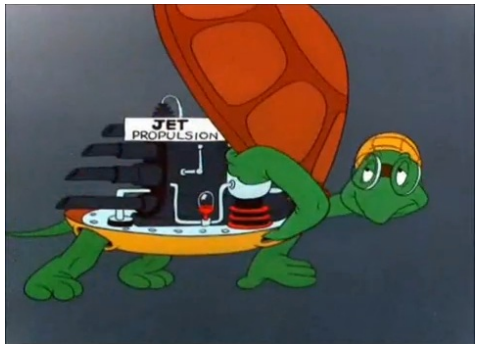


B Tree

TWIAD: Why You Should Index Your IDS Data In **Write-Optimized Databases**



B^ϵ Tree



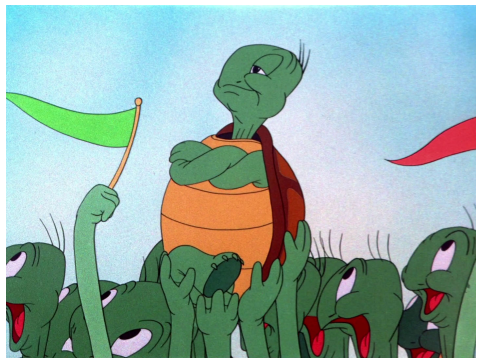
- Add buffers at every node
- Insert into root buffer
- When buffers fill, flush down one level
- Amortized insertion cost

TWIAD: Why You Should Index Your IDS Data In **Write-Optimized Databases**

	insert	point query
Optimal tradeoff (function of ε=0...1)	$O\left(\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}}\right)$	$O\left(\log_{1+B^\varepsilon} N\right)$
B-tree (ε=1)	$O\left(\log_B N\right)$	$O\left(\log_B N\right)$
ε=1/2	$O\left(\frac{\log_B N}{\sqrt{B}}\right)$	$O\left(\log_B N\right)$
ε=0	$O\left(\frac{\log N}{B}\right)$	$O\left(\log N\right)$

10x-100x faster inserts

Performance Bounds



Clear Winner: Indexing With Write-Optimization

TWIAD: Why You Should **Index Your IDS Data In Write-Optimized Databases**

Common IDS Queries:

- "Have I seen x IP before?"
- "Have I seen x IP between times y and z?"
- "Have I seen x subnet before?"

TWIAD: Why You Should **Index Your IDS Data In Write-Optimized Databases**

TWIAD Key Construction:

- Origin IP - 4 bytes
- Timestamp - 8 bytes
- Origin Port - 3 bytes
- Destination IP - 4 bytes
- Destination Port - 3 bytes

22 byte key

- All inserts are stored twice
- Origin and Destination are reversed in the second insert
- Origin and Destination are now interchangeable

TWIAD: Why You Should **Index Your IDS Data In Write-Optimized Databases**

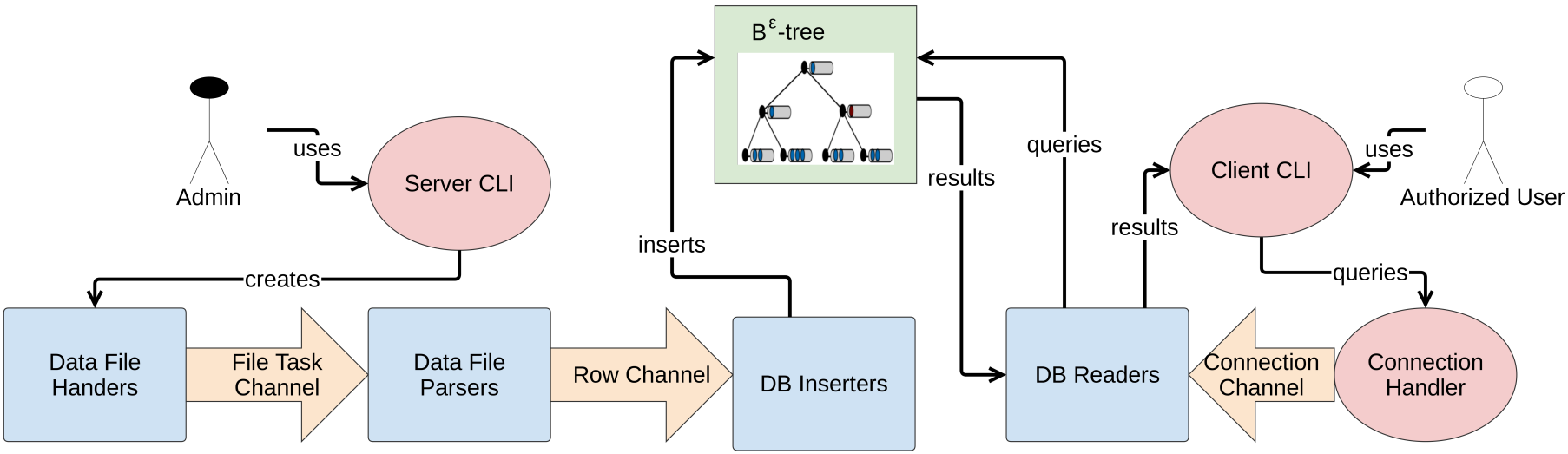
TWIAD Value Fields:

- Protocol - 4 bytes
- Duration - 8 bytes
- Origin Bytes - 8 bytes
- Response Bytes - 8 bytes
- Connection State - 5 bytes
- Origin Packets Bytes (sans header) - 4 bytes
- Response Packets Bytes (sans header) - 4 bytes
- Version - 6 bytes
- IsReversed Bit - 1 byte

48 byte value

- We include most BRO Connection Log fields
- Bytes are payload bytes vs Packets Bytes are all IP level bytes
-

TWIAD: Why You Should **Index Your IDS Data In Write-Optimized Databases**



TWIAD Architecture Diagram

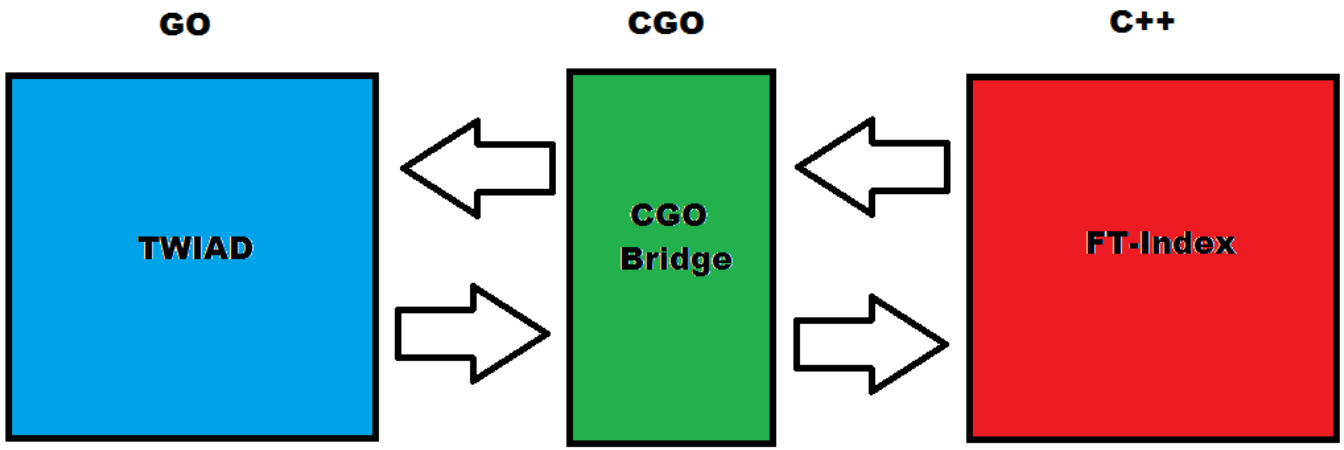
TWIAD: Why You Should Index Your IDS Data In Write-Optimized Databases

Write Optimized databases should provide a faster infrastructure for storing IDS data

This lends itself to faster intrusion response with less hardware and lower costs

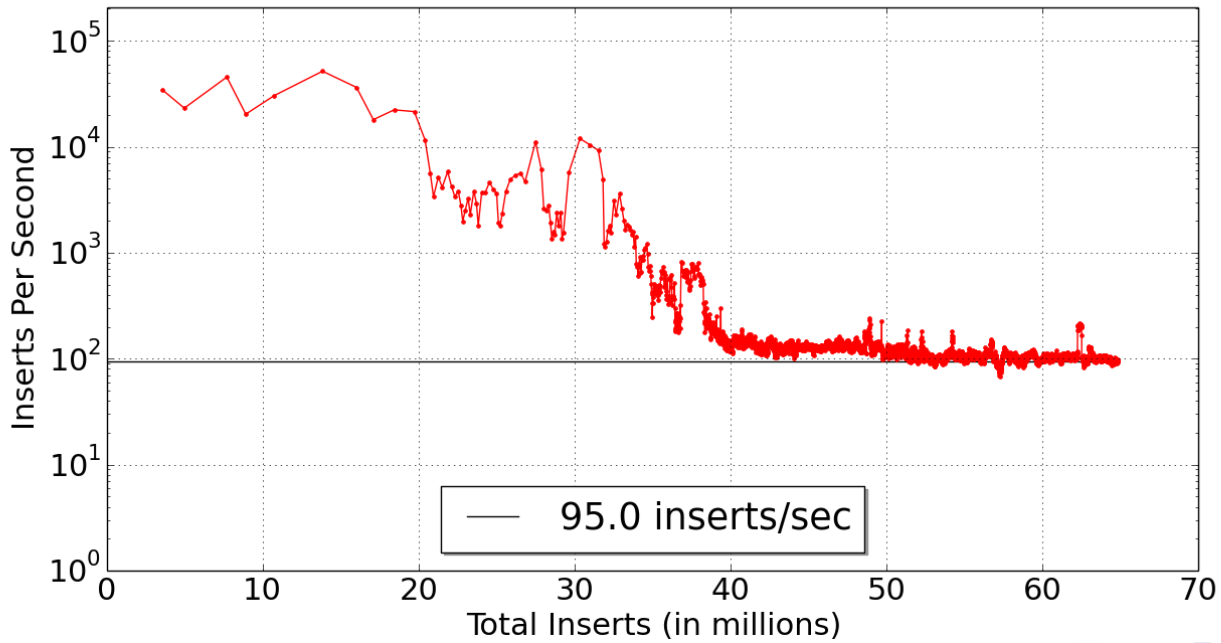
Preliminary TWIAD Results

- When designing TWIAD, we decided to build on top of an open source Write-Optimized index: FT-Index
- We wrote TWIAD in GO, though FT-Index is written in C++
- CGO acts as an intermediary to link and execute C/C++ code with GO code
- GO has many convenient features for parallel computation: GoRoutines, Channels, Scheduling, etc.
- However, the CGO bridge to C++ may have hurt our performance



Preliminary TWIAD Insertion Results

- TWIAD started with an insertion rate on the order of 10,000 inserts/second
- TWIAD's insertion rate then declined until it steadied out on the order of 100 inserts/second
- These results are from a desktop with 32GB of RAM and a 3.4GHz processor



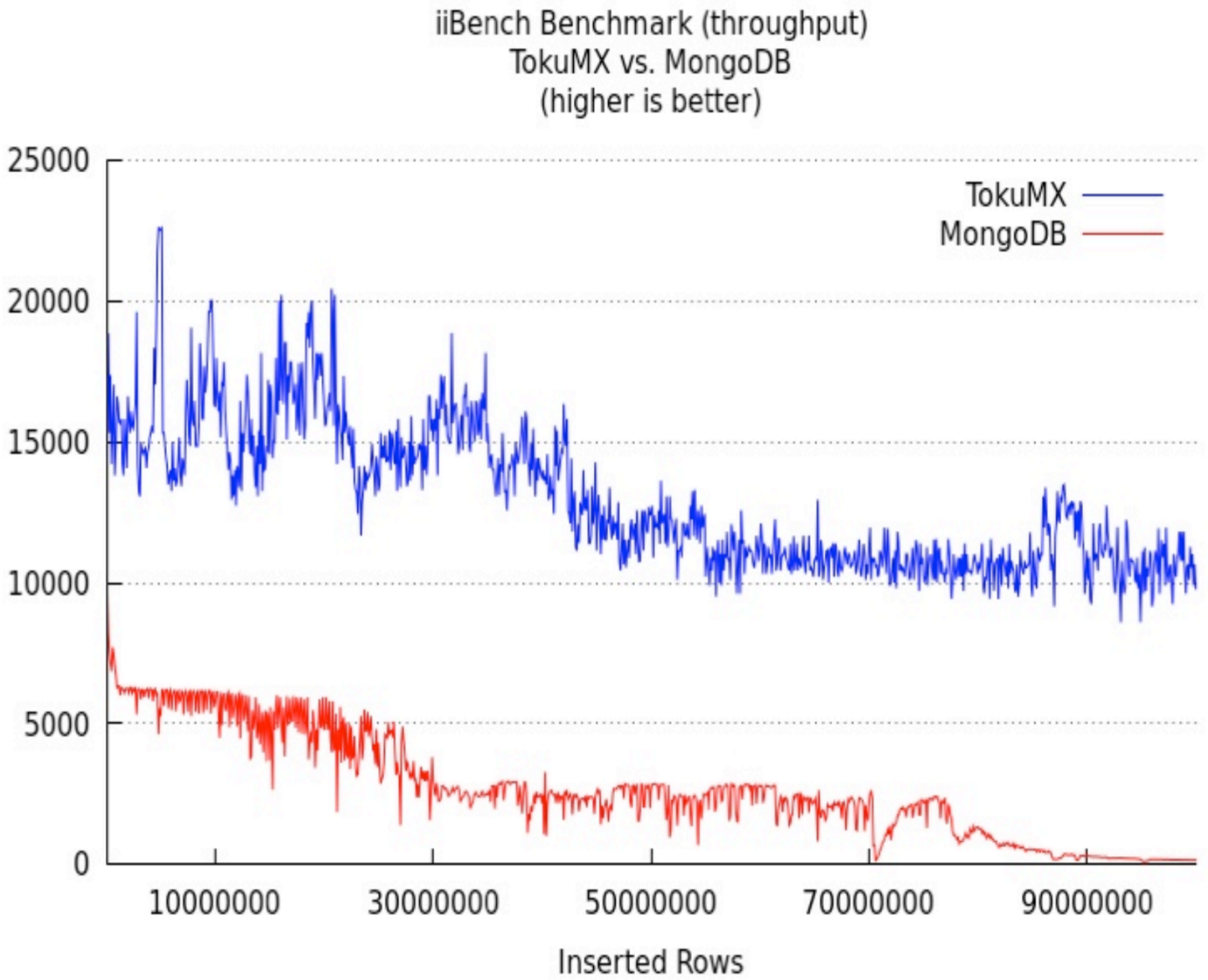
Preliminary TWIAD Query Results

Table: TWIAD Querying 43M Entry DB For Fixed IP Over Time Range

Number of Entries Returned	Time (ms)
0	1318
1	14
35	192
1665	192
2132	222

- TWIAD boasts sub-second query responses to non-empty queries on a nontrivial DB
- TWIAD has order of one second query responses to empty queries on a nontrivial DB

TokuMX Results

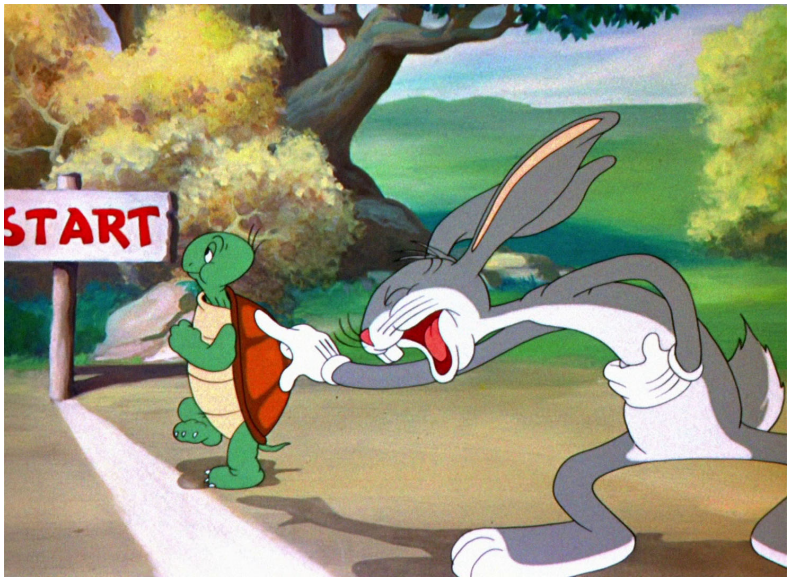


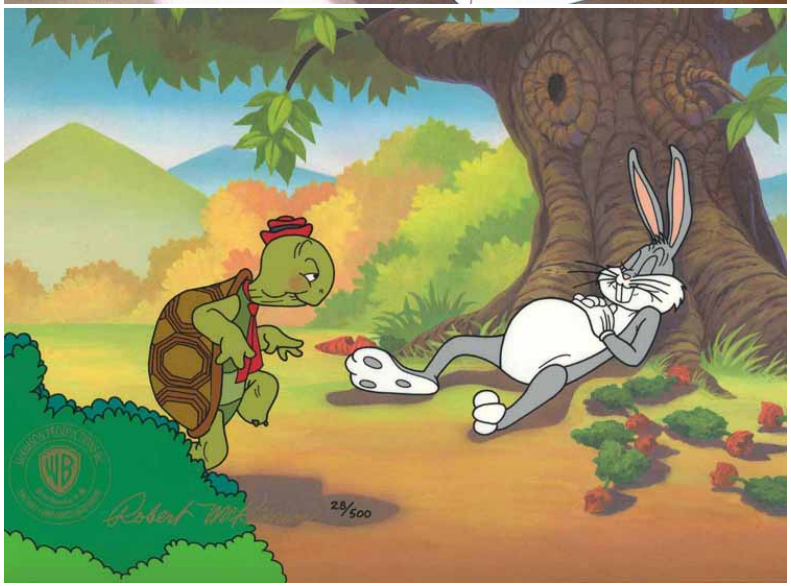
Lessons Learned & Future Work

- Executing across several languages impacts performance
- Write Optimized indexes demand familiarity and tuning
- We plan to experiment with TWIAD running on top of alternative write-optimized indexes
- Also plan to experiment with inserting into multiple indexes and investigating more interesting IDS queries

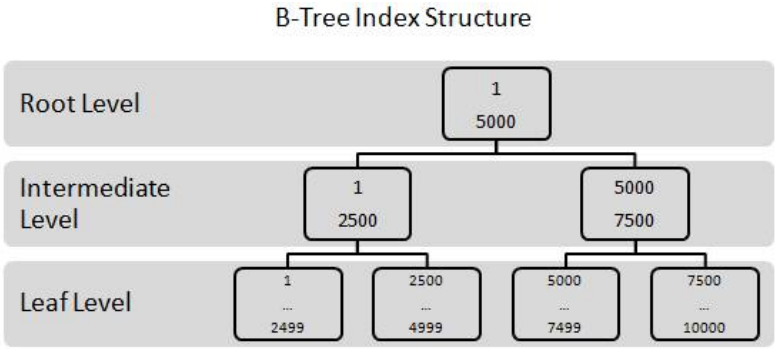
Credits

Warner Brothers Entertainment

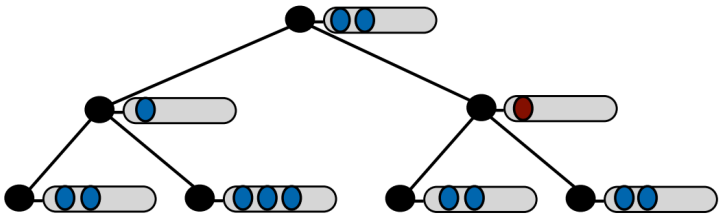




Credits



simple-talk.com

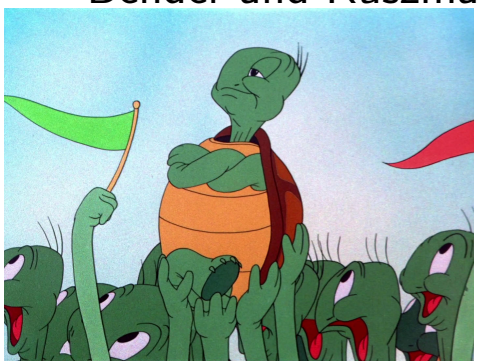


Bender and Kuszmaul

	insert	point query
Optimal tradeoff (function of ε=0...1)	$O\left(\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}}\right)$	$O\left(\log_{1+B^\varepsilon} N\right)$
B-tree (ε=1)	$O\left(\log_B N\right)$	$O\left(\log_B N\right)$
ε=1/2	$O\left(\frac{\log_B N}{\sqrt{B}}\right)$	$O\left(\log_B N\right)$
ε=0	$O\left(\frac{\log N}{B}\right)$	$O\left(\log N\right)$

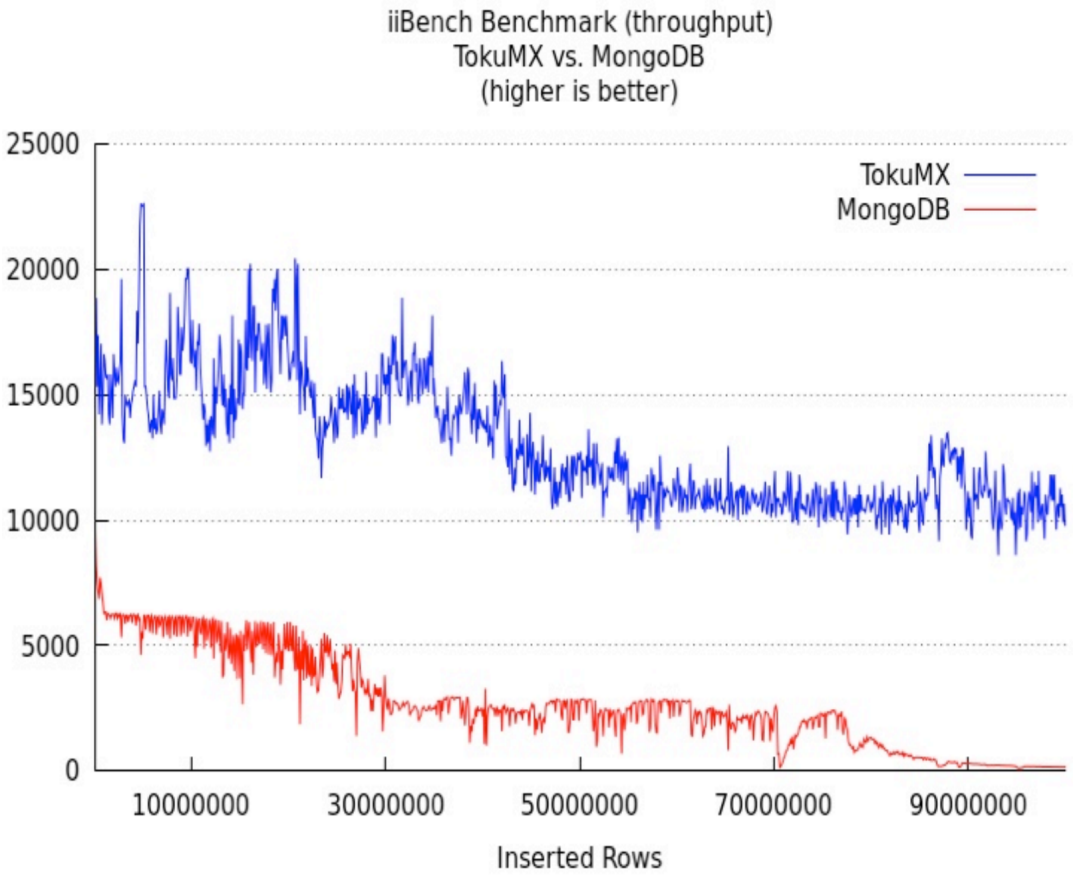
10x-100x faster inserts

Bender and Kuszmaul



Warner Brothers Entertainment

Credits



Bender and Kuszmaul