Performance Debugging
Scalable Table Stores
using YCSB++

Swapnil Patil
M. Polte, W. Tantisiriroj, K. Ren, L. Xiao,
J. Lopez, G. Gibson, A. Fuchs *, B. Rinaldi *
Carnegie Mellon University
* National Security Agency

ISTC-CC Retreat 2011
Importance of scalable table stores

- For data processing and analysis
- For systems services (e.g., metadata in Colossus)
Growing complexity of table stores

Growing set of HBase features

- **2008**
  - HBase release

- **2009**
  - RangeRowFilters
  - Batch updates

- **2010**
  - Bulk load tools
  - RegEx filtering
  - Scan optimizations

- **2011+**
  - Co-processors
  - Access Control

Simple, lightweight → complex, feature-rich stores

- ![Up arrow] Supports a broader range of applications
- ![Down triangle] Hard to debug performance issue and complex component interactions
State of table store benchmarking

YCSB: Yahoo Cloud Serving Benchmark

- Modular design to test different table stores
- Great for CRUD (create-read-update-delete) benchmarking, but not for sophisticated features

Need richer tools for understanding advanced features in table stores…
This talk: YCSB++ tool

Mechanisms to ease performance debugging

- Abstractions to write parallel tests with multiple clients and heterogeneous phases
- Integrated monitoring that correlates performance with system behavior

Used to test advanced features

- Bulk loading, table pre-spitting, batch writing
- Weak consistency, server-side filtering, security

Tool released at http://www.pdl.cmu.edu/ycsb++
Talk Outline

• Motivation
• YCSB++ architecture
• Illustrative examples of using YCSB++
• Summary and ongoing work
Original YCSB framework

Configurable workload generation to test stores

- API adaptor converts $\text{read}(K)$ to $\text{hbase\_get}(K)$
YCSB++ supports new table store

New DB adaptor for Apache Accumulo table store
  • New parameters and workload executor extensions
Coordinated & multi-phase tests

ZooKeeper-based coordination & synchronization
- Enables heavy workloads and asymmetric testing
Coordinated & multi-phase YCSB++

- Distributed, multi-client tests
  - Allows clients to co-ordinate their test actions
  - Rely on shared data structures in ZooKeeper
  - Used to study weak consistency models

- Multi-phase tests
  - Can construct tests comprising of different phases
  - Built on ZooKeeper-based barrier-synchronization
  - Used for analyze high-speed ingest features
Collective monitoring in YCSB++

Fine-grained resource monitoring using Otus\cite{Ren2011}

- Collects from YCSB, table stores, HDFS and /proc
Example of OTUS monitoring

Monitoring Resource Usage and TableStore Metrics

- Accumulo Avg. StoreFiles per Tablet
- HDFS DataNode CPU Usage
- Accumulo TabletServer CPU Usage

Accounts resources used by different services
  - Useful when each machine has multiple services
Talk Outline

• Motivation
• YCSB++ architecture
• Illustrative examples of using YCSB++
  • Case study: HBase and Accumulo
  • Both are Bigtable-like table stores
• Summary and ongoing work
Primer on Bigtable-like stores

(1) Incoming mutation logged in memory (unsorted order)

(2) MINOR COMPACTION
Memtables written to sorted, indexed store files in HDFS

(3) MAJOR COMPACTION
LSM-tree based file merging (in background)
Accumulo table store

- Started at NSA; now an Apache project
  - Built for high-speed ingest and scan workloads

- New features in Accumulo
  - Iterator framework for user-specified programs placed in different stages of DB pipeline
    - E.g., Supports joins and stream processing
  - Also provides fine-grained cell-level access control
  - Fault tolerant cross-server operations
Features Tested by YCSB++

- Table bulk loading
- Batch writing
- Weak consistency
- Table pre-splitting
- Server-side filtering
- Access control

Illustrative Example

Table bulk loading

- High-speed ingestion through minimal data migration
- Need careful tuning and configuration [Sasha2002]
Table bulk loading in action

(2) IMPORT ... 
... store files into table stores to make data available for users

(1) FORMAT ... 
... existing data files to store-file specific format using Hadoop
8-phase bulk load test in YCSB++

Measurement phase
- Light mix of Read/Update operations
- Interleaved to study performance over time

Pre-load data
- Insert 6M rows in empty table

Load data
- Load 48M rows in existing table

Sleep
- Let servers finish balancing work
Multi-phase tests show variation

10x latency variation; lasts for a long time!

Uniformly low latency after store is steady (no inserts)
Monitoring rebalancing at servers

Let's take a closer look at correlating performance with server-side state
Correlate latency with server-side work

StoreFiles and Tablets increase with splitting
Background compactions reduce number of store files
FEATURES TESTED BY YCSB++

- Batch writing
- Weak consistency
- Table bulk loading
- Table pre-splitting
- Server-side filtering
- Access control

Details in ACM SOCC 2011 paper
FAQ: Are workloads realistic?

YCSB++ uses synthetic workloads
  • Uses YCSB’s synthetic workload generator based on record size, distribution and operation types

Goal: How can we create real workloads?
  • Trace real applications running on table stores
    – E.g., Twitter, photo stores, RSS, data mining
  • YCSB++ can replay these (anonymized) traces
Need real application traces

Example workload: Monitoring application

- All OpenCloud cluster monitoring information is stored in HBase
- OTUS tool queries HBase to create graphs
- Trace these calls to create a “monitoring” workload

ISTC-CC: How and what can we trace?

- Looking for ways to collect traces from table store deployment with real applications
Summary: YCSB++ tool

- For benchmarking & debugging performance of advanced features using extensions to YCSB

<table>
<thead>
<tr>
<th>Weak consistency semantics</th>
<th>Distributed clients using ZooKeeper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast insertion (pre-splits, bulk loads)</td>
<td>Multi-phase testing (with Hadoop)</td>
</tr>
<tr>
<td>Server-side filtering</td>
<td>New workload generators and database client API extensions</td>
</tr>
<tr>
<td>Fine-grained access control</td>
<td></td>
</tr>
</tbody>
</table>

- Two case-studies: HBase & Accumulo
- Download at http://www.pdl.cmu.edu/ycsb++