A Quick Introduction to YCSB

YCSB -- Overview

- Yahoo! Cloud Serving Benchmark
- Extensible workload generator for benchmarking key-value stores, e.g.
  - Memory-based Key-value stores: Redis, Memcached
  - Cloud-based: Azure
  - Others: Cassandra, MongoDB
- As a client,
  - Load key-value pairs into the key-value stores (a.k.a. the Load phase)
  - Issue workloads to the key-value stores (a.k.a. the Run phase)
  - Report statistics

YCSB -- Workloads

- Default workloads
  - 6 types of workloads
    - Update-heavy: 50% read, 50% update
    - Read-mostly: 95% read, 5% update
    - Read-only
    - Read latest (delete old ones, insert new ones and read mostly the new ones)
    - Read-modify-write
    - Short range scan

YCSB -- Workloads

- Flexibility to generate different workloads
  - Setup a workload configuration file, or pass the configurations as parameters
  - Common workload parameters:
    - Key-value pairs: key size, total number of fields and their length in the value
    - Total number of key-value pairs
    - Total number of operations
    - Insert order, e.g. sequential, hashed (uniform)
    - Access distributions, e.g., sequential, uniform, zipfian, latest
    - Ratio of read, update, insert, read-modify-write, and range scan
- Support multi-thread client threads
- Support multiple clients
  - Set a manual range of key-value pairs for each client during the Load phase

YCSB -- Statistics Report

- Type of statistics
  - Percentile latency, e.g., 95-th percentile and 99-th percentile
  - Histogram buckets
  - Time series

YCSB -- Usage

- $ ycsb <command> <key-value store> [options]
  - command:
    - load: Load key-values into the key-value store
    - run: Run the workload on the key-value store
    - shell: Interactive shell for manual operations
  - key-value store:
    - e.g., basic: the implementation that dumps all operations to console

YCSB -- Usage

- $ ycsb <command> <key-value store> [options]
  - options:
    - -P <file>: Workload file
    - -cp <path>: Extra Java classpath
    - -jvm-args <args>: Extra arguments to JVM
    - -p <key=value>: Workload property (overrides the workload file)
    - -s: Print status to standard error stream
    - -target <n>: Target ops/sec
    - -threads <n>: Number of client threads

YCSB -- Example (using Redis)

1. Prepare the KV store
2. Download and configure the YCSB benchmark
    a. Download YCSB
    b. Define the KV store interface and its location
    c. Define the workload
    d. Define the runtime parameters
3. Load the data
4. Run the target workload

Reference: https://github.com/brianfrankcooper/YCSB/wiki/Running-a-Workload
(the platform used in the example is Ubuntu 16.04LTS server)

YCSB -- Example (using Redis)

1. Prepare the KV store
    a. Download Redis

        $ wget http://download.redis.io/redis-stable.tar.gz
        $ tar zxf redis-stable.tar.gz

    b. Compile Redis

        $ sudo apt-get install build-essential tcl
        $ cd redis-stable
        $ make && make test
        $ sudo make install

    c. Run Redis (in local mode)

        $ redis-server ./redis.conf

YCSB -- Example (using Redis)

2. Configure YCSB benchmark
    a. Download YCSB

        $ wget https://github.com/brianfrankcooper/YCSB/archive/0.12.0.zip
        $ sudo apt-get install unzip python maven openjdk-8-jdk
        $ unzip 0.12.0.zip

    b. Define the KV store type and its location

        "redis", "-p redis.host=127.0.0.1 -p redis.port=6379"

    c. Define the workload

        "workloada"

    d. Define runtime parameters

        "-threads 1”

YCSB -- Example (using Redis)

3. Load the data

```
$ cd YCSB-0.12.0
$ ./bin/ycsb load redis -p redis.host=127.0.0.1 -p redis.port=6379 -P workloads/workloada \
    -threads 1
```

4. Run the target workload

```
$ ./bin/ycsb run redis -p redis.host=127.0.0.1 -p redis.port=6379 -P workloads/workloada \
    -threads 1
```

YCSB -- Example (using Redis)

4. Run the target workloads
    - Statistics Report

```
[Overall] Runtime (ms)
[Overall] Throughput (ops/sec)
…
[READ], Operations
[READ], AverageLatency (us)
[READ], MinLatency (us)
[READ], MaxLatency (us)
[READ], 95thPercentileLatency (us)
[READ], 99thPercentileLatency (us)
[READ], Return=OK
...
```

YCSB -- Extensions (Workload)

- Use new parameters
  - e.g., total number of key-value pairs, alternative access distributions
  - See the workload file template
- Implement a new workload
  - e.g., different access distribution, key/value generation
  - See the guide for programming details: https://github.com/brianfrankcooper/YCSB/wiki/Implementing-New-Workloads#option-2-new-java-class

YCSB -- Extension (KV store support)

- Implement a client in Java to support
  - Standard operations: read, scan, update, insert, delete
  - Key-value store properties, e.g., location
- See the guide for more programming details:
  - https://github.com/brianfrankcooper/YCSB/wiki/Adding-a-Database

YCSB -- Others

- Reference
  - YCSB wiki
- Related tools:
  - YCSB-C: a minimal C++ version of YCSB
  - YCSB++: Extended YCSB benchmark (unfortunately the code is no longer available)

YCSB -- Supplementary

- Example on a simple hack to existing code
  - e.g. Zipfian constant in the Zipf distribution
    - Modify the file “core/src/main/java/com/yahoo/ycsb/generator/ZipfianGenerator.java”

      public static final double ZIPFIAN_CONSTANT = 0.99;

    - Recompile (all packages), $ mvn clean package