BookKeeper

Flavio Junqueira
Yahoo! Research, Barcelona

Hadoop in China 2011
What’s BookKeeper?

- Shared storage for writing fast sequences of byte arrays
- Data is replicated
- Writes are striped
- Many processes can access it
Motivation

• Recoverable systems
  ✓ Journal/write-ahead log
  ✓ Integrity and durability
  ✓ Efficient: sequential synchronous writes

• Why is writing sequentially important?
  ✓ To avoid random seeks
More motivation

• Examples
  ✓ Many databases (e.g., Postgres)
  ✓ Hbase region server
  ✓ ZooKeeper
  ✓ HDFS namenode
  ✓ Hedwig hubs
HDFS at a glance

- Main components: namenode and datanode
  - Single name node
  - A number of data nodes
- Namenode
  - Manages FS namespace
  - Regulates access to the FS
  - Mapping of blocks to data nodes
- Datanode
  - Stores blocks
  - Serves reads and writes

http://hadoop.apache.org/common/docs/current/hdfs_design.html
Namenode

• File system state
  ✓ Metadata, block map
  ✓ In memory

• Checkpoint
  ✓ On disk
  ✓ Snapshot of the service state

• Edit log
  ✓ Persists changes to the file system metadata
  ✓ Written to disk
Namenode

- Edit log is a journal
  ✓ Local disk
  ✓ NFS server

- Production use
  ✓ Enterprise-class NFS
  ✓ Expensive devices
  ✓ E.g., Netapp Filer
  ✓ Robust, but still a single point of failure

BookKeeper kicks in!
Making the namenode highly available

- Backup node
  - ✓ One step ahead
  - ✓ Receives a stream of updates
  - ✓ Warm standby

- Shortcomings
  - ✓ Cannot guarantee consistency
  - ✓ Difficult to have multiple backups
Making the namenode highly available

1) Communication among processes to coordinate

2) Communication with shared data store
Making the namenode highly available

- Replicate the functionality of the name node
  - Performance penalty
  - Not scalable
- Write log to external device
  - NFS
    - Avatarnode
      - Replication is not transparent
  - External high-performance logging/journaling service
    - BookKeeper
BookKeeper

- Shared storage for logs
- Design goals
  - Efficient sequential writes
  - Fault tolerance
  - Scalability
BookKeeper architecture

- **Bookie**: Storage node
- **Ledger**: log file
- **Ensemble**: group of bookies storing a ledger
- Writes to quorums of Bookies
- Parallel writes to quorums
- Reads from the same quorum
The anatomy of a bookie

- **Transaction log**
  - ✓ Pre-allocates, batches
  - ✓ Return upon write/sync to disk

- **Index**
  - ✓ Position of entry

- **Entries**
  - ✓ Written sequentially to entry log
Scalability of writes

- Write quorums do not necessarily intersect
- Assuming that:
  1. Each bookie performs $e$ entries/s
  2. Number of bookies: $r$
  3. Write quorum: $q$ bookies
- Ideal maximum throughput: $\frac{r \times e}{q}$
- In practice, network bandwidth or cpu limits the total capacity in bytes written per second
API at a glance

• createLedger
• openLedger
• addEntry
  ✓ Async and sync
• readEntries
  ✓ Async and sync
• closeLedger
  ✓ Writes the last entry id to ZooKeeper
Why keep last entry id?

- Acknowledgement
  - Ledger closed properly
- Agreement
  - Two readers don’t read different sets of entries
- What if no last entry id has been written?
Recovery procedure

- Reader client executes a **ledger recovery** procedure
- Hints on ledger entries
- **Procedure**
  - ✓ Request last entry hint from bookies
  - ✓ Try to read as many entries greater than the hint
  - ✓ Make sure entries are written to a quorum
How to use it

• Application writer
  ✓ Creates a ledger
  ✓ Add entries to the ledger
  ✓ Return upon confirmation from quorum
  ✓ Closes the ledger

• Application readers
  ✓ Open ledger
  ✓ Read from the ledger

• Application does not reopen to append
BookKeeper service

- Service
  - ✔ Bookies in the cloud
  - ✔ Through ZooKeeper

- ZooKeeper
  - ✔ Bookies online
  - ✔ Ledger metadata
Performance
Setup

- Cluster of identical machines
- 2 Quad Core Intel Xeon 2.5GHz
- 16GB of RAM
- Four SATA disks, 7,200 RPMs
- 1Gbit/s network interface
BookKeeper performance

- Single writer
BookKeeper performance

- Multi-writer
  - ✓ Aggregate throughput

- Concurrent ledgers
  - ✓ Up to 40k ledgers

<table>
<thead>
<tr>
<th>bytes</th>
<th>2Q</th>
<th>3Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>87k</td>
<td>116K</td>
</tr>
<tr>
<td>1024</td>
<td>31k</td>
<td>54k</td>
</tr>
<tr>
<td>4096</td>
<td>8k</td>
<td>16k</td>
</tr>
</tbody>
</table>

add operations/s
BookKeeper and the Namenode

The diagram compares latency in ms to throughput (1000 x txns/s) for different storage options:

- BookKeeper
- Local File
- NFS
- NoPersist

The y-axis represents latency in ms, while the x-axis shows throughput in terms of transactions per second (1000 x txns/s). The graph illustrates the performance differences among these storage options under varying levels of throughput.
Hedwig
Hedwig

• Multi-region pub/sub system
• Guaranteed-delivery topic-based pub-sub system
• Extremely High Performance
• Elastically scalable
  ✓ Deployed over commodity machines
  ✓ Capacity can be added on-the-fly by adding machines
• Low Operational Complexity
  ✓ Tolerate failures without manual intervention
  ✓ Automatic load balancing
• Designed for multiple data-centers
Wrap up
Advanced features

• Opening without recovery
  ✓ Warm standbys
  ✓ Must know what you’re doing

• Fencing
  ✓ Consistency despite concurrent accesses
  ✓ Prevents new successful writes once recovered
Status

- Release on the way
  ✓ Candidate should be out this week
- BookKeeper and the namenode
  ✓ Watch HDFS-1580 and HDFS-234
The team

- Dhruba Borthakur (Facebook)
- Flavio Junqueira (Yahoo!)
- Ivan Kelly (Yahoo!)
- Benjamin Reed (Yahoo!)
- Utkarsh Srivastava (Twitter)
Contributing

• Sign up for the lists
• Discuss with the community
• Propose improvements
  ✓ Bug fixes
  ✓ New features

http://zookeeper.apache.org/bookkeeper
Questions?

http://zookeeper.apache.org/bookkeeper