Hedwig
A Shared-Nothing Message Broker

Hedwig Team
Yahoo! Research
What Hedwig Offers

- Guaranteed-delivery topic-based pub-sub system
  - Durability: acknowledged published get delivered
  - Subscribers guaranteed to get all publishes after subscription (even if subscribers fail and come back)
  - Messages delivered in publisher order

- 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

- Optimized for multiple data-centers
History

• Has been a relatively low-key effort

• Started gaining steam in June 2009.

• 2-2.5 engineers (+1 intern) over the summer.
Architecture

Protocol

Zookeeper subscriptions, how much they have consumed, etc.

Bookkeeper published messages

Internet

Hedwig instances in other data centers

Client

Hub

Hub

Hub

 protocol
subscriptions, how much they have consumed, etc.

published messages

Hedwig instances in other data centers
Data organized in a hierarchical namespace
- Nodes in the namespace, called znodes, can be persistent or ephemeral (znode will be deleted in service detects client failure)
- Strong durability guarantee
- Strong ordering guarantee
- API allows for clients to watch for changes
- Data stored in memory for low latency and consistent performance, but changes logged to disk for performance
- Reads processed using local server information, changes are linearized through leader.
Zookeeper

• Used for configuration storage
  – Locations of ledgers
  – Subscriber information

• Topic leader election
  – Discovering topic leader
  – Ensuring single topic leader
  – Detecting topic leader failure

• Membership
  – Discovering available bookies
• Model a write-ahead log as an append-only sequence of entries, called a ledger

• Simple interface
  – create/openLedger
  – addEntry
  – readEntry
  – deleteLedger
• Like distributed RAID 1,0 (with append-only)
  – Configurable redundancy
  – Bookkeeper servers handle append-only, hence highly optimized.

• Open-source as contrib to Zookeeper
Performance

1. append to log, sync()
2. append to file for topic, no sync

Can get close to sequential disk bandwidth due to group commit
Hedwig Hubs

- Topics horizontally partitioned across hubs
  - A topic belongs to only one hub (for per-topic ordering guarantees).
  - A hub can have multiple topics.

- Manages delivery to subscribers
  - Caches recently published data in process, to avoid trip to Bookkeeper.

- Subscribes to hubs in other data centers
Automatic Failover

Client \( \text{publish}(T) \) \rightarrow \text{Virtual IP / DNS Round Robin} \rightarrow \text{Hub A} \rightarrow \text{Hub B} \rightarrow \text{Hub C} \rightarrow \text{Zookeeper}

handle if owner
Automatic Failover

Client

Virtual IP / DNS Round Robin

Hub A

Hub B

Hub C

Zookeeper

publish(T)

lookup owner
(choose least-loaded hub if no owner exists)

redirect(B)
Automatic Failover

Client \rightarrow Hub A \rightarrow Hub B \rightarrow Hub C

publish(T) (redirected)

- handle if owner
Automatic Failover

If B fails, ephemeral node disappears, and a new owner is chosen automatically.
Publish

Client -> Hub B

publish(T) (redirected)

Hub A

Zookeeper

Hub B

Hub C

Bookkeeper

log message
Publish

Client → Hub A → Hub B → Hub C → Client

Zookeeper

Bookkeeper

ack publish

broadcast
Subscribe

Client ➔ Hub A ➔ Hub B ➔ Hub C ➔ Zookeeper

subscribe(T) (redirected)

Find last message consumed
Subscribe

Client -> Hub B

Hub A

Hub B

Hub C

Zookeeper

Retrieve messages not in cache

Bookkeeper

broadcast
Testing

• Hardware
  – Old, relatively-crappy, commodity boxes
  – 2 cores, 2.13 GHz, 4GB RAM
  – 2 disks, 7.2K rpm SATA

• Most results on 4-box farm (1 hub, 3 bookies)

• Performance Tests
• Failure Tests
• Stability Tests
Performance (Latency v/s Throughput)

(1 hub, 3 bookies, 100 topics, 1K messages, 1 subscriber per topic)
Scalability

Throughput against # of servers
(100 topics, 1K messages,
1 subscriber per topic)
Large message sizes

Percentage of NIC Bandwidth against msg size
(100 topics, 1 hub, 3 bookies, 1 subscriber/topic)
• Able to shoot down a bookie
  – Operations continue without a single failure

• Able to shoot down a hub
  – Operations going to that hub fail, but only for a second (depending on our ZK timeout)
  – Topic gets taken up automatically by another hub
Stability

• Able to run the system for days without anything going wrong.
• Recovery tools done, but just started testing.
Recent improvements

• Scaling with number of topics.
  – Currently every topic gets its own file, which doesn’t scale.
  – Patch in progress to share files among topics
  – Preliminary numbers indicate scalability of up to 10s of thousands of topics per bookie
Recent improvements

- Collection of consumed logs
- Bookie recovery
What’s Missing

• C++ Client Library  
  – Have an initial implementation
• JMX binding
• Operational/Monitoring tools (1.5 months)  
  – A promising approach is to write adapters so that existing tools just work.
Harder, Longer-term Things

- Notifications (3-4 months)

- Adaptive replication
  - Relatively easy with current design (3 weeks)

- Support for non-star topologies and changing data-center topology on the fly (4-5 months)
http://zookeeper.apache.org