CQS Product Requirements

CQS Product Requirements

Overview

This document specifies the high-level requirements for CQS that all stakeholders can agree upon. One goal of this page is provide criterion for evaluating various design options.

Functional Requirements

These assumes normal operating circumstances.

- Implement all actions in the Amazon SQS spec
- The Queues do not have to support FIFO behavior under any condition. No effort is done to make CQS behave like a traditional Queue data-type
- HTTP based protocol (as specified by Amazon SQS)
- Guaranteed delivery of all messages at least once
- Redelivery of messages that are not deleted in a configurable timeout.
- Redelivery of deleted messages during or after some failure conditions like Cassandra node failure.
- For the ReceiveMessage() API call, don't return the same message multiple times in more than 5% of the cases.
- A ReceiveMessage request for n messages might return less than n messages even though system has n messages in less than 10% of cases
- Message size no more than 64k

Response Time Requirements

These assumes normal operating circumstances.

- For the Send/ReceiveMessage of 1 message, the response time should be ~10ms in 95% of cases within a data-center.
- For the Send/ReceiveMessage of 1 message, the response time should be ~100ms in 99% of the cases within a data-center
- For the Send/ReceiveMessage of 1 message, the response time should be ~8ms in the median cases
- The time between a succesfull response to SendMessage() and the time a RecieveMessage() can return that message must not be more than 500ms in the 95%

Availability Requirements

All the below requirements assume that all the underlying infrastructure behaves like we expect in a failover scenario (Cassandra replication, load-balancers switching, network, etc)

- System needs to be highly available (99.999%) and deployed in multiple data centers
- No message loss in the event of node or data-center failure.
- In data-center failure case, The messages should be made available in a different data-center instantly with possible loss of hidden state.
- When a single-node failure occurs within a data-center ALL the messages should be made available in less than 60 seconds

Usage Pattern

- In Normal mode of operation, message producers and message consumers of a queue are co-located in the same data-center.
- In data-center failure case, the messages need to be available in some other data-center where the consumers can be directed to consume the messages. The consumers would be redirected through a load-balancer.

Scalability Requirements

- Unlimited number of queues
- Horizontally scalable. Adding more API and Cassandra nodes should increase throughput and capacity as linearly as feasible
- Must handle a peak load of 500k sendMessage() requests/sec
- Must have a peak load of 100k receiveMessage() requests/sec.

Security Requirements

For the first version of SQS we are not planning on implementing the AddPermision command or the CreateQueue's 'Policy' attribute. In subsequent releases, we can do these.

Client Usage Recommendation

Since the SQS will return messages out of order and could occasionally return messages previously returned by the Queue, the clients should be prepared to re-order messages and ignore duplicates.