Kafka replication system tests

A. Overview

According to Kafka Replication Design document, "The purpose of adding replication in Kafka is for stronger durability and higher availability. We want to guarantee that any successfully published message will not be lost and can be
consumed, even when there are server failures. Such failures can be caused by machine error, program error, or more commonly, software upgrades."

1. **Design documentation**
   a. https://issues.apache.org/jira/secure/attachment/12487175/kafka_replication_highlevel_design.pdf
   b. https://cwiki.apache.org/confluence/display/KAFKA/kafka+Detailed+Replication+Design+V3

B. Kafka Replication Testing Plan

B.1 Test Contract:

1. Produce and consume messages to x topics and y partitions.
2. This test sends m messages to n replicas.
3. At the end verifies the log size and contents as well as using a consumer to verify that there is no message loss.

B.2 Test dimensions: Varying each parameters to provide different test scenario

Parameter	**Value Set**
No. of partitions	1, 5, 10
No. of replica factors	1 ~ 6
Log segment sizes	1K, 2K, 10K
No. of topics	1, 5, 10, 100
Producer compression	On / Off
Producer acks	-1, 1
Producer mode	Sync, Async
Failure Type (Applicable in Failure Testcases)	• Controlled Failure (kill -15) • Hard Failure (kill -9) • Soft Failure (long pause during GC)

C. Test Cases

Functional Test	**Description**
C.1 Replication Basic	1. **Setup**: Configure 1 Zookeeper, 1 ~ 6 brokers, 1 producer, 1 consumer 2. **Test Description**: a. Follow the steps in B.1 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas

C.2 Replication Leader Election	1. **Setup**: Configure 1 Zookeeper, 1 ~ 6 brokers, 1 producer, 1 consumer 2. **Test Description**: a. Follow the steps in B.1 b. ***During the test session, find leader from brokers' log4j message and introduce failure to Leader*** c. Leader re-election will be triggered 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that new leader is re-elected by parsing the brokers' log4j messages log files
C.3 Replication with Leade r Failure	1. **Setup**: Configure 1 Zookeeper, 1 ~ 6 brokers, 1 producer, 1 consumer 2. **Test Description**: a. Follow the steps B.1 b. ***During the test session, find leader from brokers' log4j message and introduce failure to Leader*** c. The no. of failures can be specified in the corresponding testcase_<n>_properties.json d. The type of failures are defined in Test Dimentions in B.2 (Controlled, Hard, Soft) 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas
C.4 Replication with Follow er Failure	1. **Setup**: Configure 1 Zookeeper, 1 ~ 6 brokers, 1 producer, 1 consumer 2. **Test Description**: a. Follow the steps in B.1 b. ***During the test session, find leader from brokers' log4j message and exclude that broker and introduce failure to one of the other brokers which are Followers***. c. The no. of failures can be specified in the corresponding testcase_<n>_properties.json d. The type of failures are defined in Test Dimentions in B.2 (Controlled, Hard, Soft) 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas
C.5 Replication with Contro ller Failure	1. **Setup**: Configure 1 Zookeeper, 1 ~ 6 brokers, 1 producer, 1 consumer 2. **Test Description**: a. Follow the steps in B.1 b. ***During the test session, find Controller from either brokers' log4j messages or querying the Bean and introduce failure to Controller***. c. The no. of failures can be specified in the corresponding testcase_<n>_properties.json d. The type of failures are defined in Test Dimentions in B.2 (Controlled, Hard, Soft) 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas
C.6 Replication with Mirror Maker Failure	1. **Setup**: Configure 2 Clusters with 1 Mirror Maker: a. Source: 1 Zookeeper, 1 ~ 6 brokers, 1 producer b. Mirror Maker to replicate data from Source to Target c. Target: 1 Zookeeper, 1 ~ 6 brokers, 1 consumer 2. **Test Description**: a. Follow the steps in B.1 b. ***During the test session, introduce failure to Mirror Maker***. The no. of failures can be specified in the corresponding testcase_<n>_properties.json c. The type of failures are defined in Test Dimentions in B.2 (Controlled, Hard, Soft) 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas

C.7 Replication with Backward Compatibility (0.7 & 0.8 Kafka jars) / Migration Tool	1. **Setup**: Configure 2 Clusters with 1 Mirror Maker: a. Source: 1 Zookeeper, 1 ~ 6 brokers, 1 producer (***running in 0.7 Kafka jar***) b. Mirror Maker to replicate data from Source to Target c. Target: 1 Zookeeper, 1 ~ 6 brokers, 1 consumer (***running in 0.8 Kafka jar***) 2. **Test Description**: a. Follow the steps in B.1 3. **Test Dimensions**: Varying the parameters within the Value Set to observe the behavior of replication against different combinations 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas
C.8 Replication with Production Setup	1. **Setup**: Configuration as follows: a. Zookeeper: 5 nodes cluster b. Brokers: 8 nodes cluster c. log segment size: 1GB d. Producer compression: On e. Async Producer: Yes f. Producer Acks: -1 g. Replica Factor: 3 h. No. Topics: 1000 i. No. Partitions: 10 2. **Test Description**: a. Repeat the steps in B.1 b. During the test session, randomly introduce failure to **Leader, Follower, Mirror Maker or Controller** constantly. This would be a reliability / stress test which should last a few hours. c. The type of failures are defined in Test Dimentions in B.2 (Controlled, Hard, Soft) 3. **Test Dimensions**: Fixed at the set up stage. 4. **Validation**: a. Verify that all message id are matching in both producer log and consumer log b. Verify that all corresponding topic-partition log segment files checksums are matching across all replicas