KIP-858: Handle JBOD broker disk failure in KRaft

Status

Current state: Accepted

Discussion thread: https://lists.apache.org/thread/8dqvfhzcyy87zyy12837pxx9lgsdhvft

Vote thread: https://lists.apache.org/thread/4pqjp8r7n94lnymv3xc689mfw33lz3mj

JIRA: ⚠ Unable to render Jira issues macro, execution error.

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Support for multiple log directories per broker, aka JBOD (*Just a Bunch Of Disks*) came in KIP-112 and since then JBOD has been an important feature in Kafka, allowing it to run on large deployments with multiple storage devices per broker.

To ensure availability, when a partition leader fails, the controller should elect a new leader from one of the other in-sync replicas. But the controller does not check whether each leader is correctly performing its duties, instead the controller simply assumes that each broker is working correctly if it is still an active member of the cluster. In KRaft, cluster membership is based on timely heartbeat requests sent by each broker to the active controller. In ZooKeeper, cluster membership is based on an ephemeral zNode under `/brokers/ids`.

In KRaft mode, when a single log directory fails, the broker will be unable to be either a leader or a follower for any partitions in that log directory, but the controller will have no signal that it needs to update leadership and ISR for the replicas in that log directory, as the broker will continue to send heartbeat requests. [1]

In ZooKeeper mode when a log directory fails, the broker sends a notification to the controller which then sends a full `LeaderAndIsr` request to the broker, listing all the partitions for all log directories for that broker. The controller relies on per-partition error results from the broker to update leadership and ISR for the replicas in the failed log directory. Without this notification, the partitions with leadership on that log directory will not get a new leader assigned and would remain unavailable.

Support for KRaft in JBOD, was proposed and accepted back in KIP-589 — with a new RPC from the broker to the controller indicating the affected topic partitions in a failed log directory — but the implementation was never merged and concerns were raised with possible large requests from the broker to the controller.

KIP-833 was accepted, with plans to mark KRaft as production ready and deprecate ZooKeeper mode, but JBOD is still a missing feature in KRaft. This KIP aims to provide support for JBOD in KRaft, while avoiding any RPC having to list all the partitions in a log directory.

Public interfaces

Command line tools

The `format` sub-command in the the `kafka-storage.sh` tool already supports formatting more than one log directory — by expecting a list of configured `log.dirs` — and "formatting" only the ones that need so. A new property will be included in `meta.properties` — `directory.id` — which will identify each log directory with a UUID. The UUID is randomly generated for each log directory.

meta.properties

A new property — `directory.id` — will be expected in the `meta.properties` file in each log directory configured under `log.dirs`. The property indicates the UUID for the log directory where the file is located. If any of the `meta.properties` files does not contain `directory.id` one will be randomly generated and the file will be updated upon Broker startup. The `kafka-storage.sh` tool will be extended to generate this property as described in the previous section.

Reserved UUIDs

The following UUIDs are excluded from the random pool when generating a log directory UUID:

- `UUID.UNASSIGNED_DIR` – `new Uuid(0L, 0L)` – used to identify new or unknown assignments.
- `UUID.LOST_DIR` - `new Uuid(0L, 1L)` – used to represent unspecified offline directories.
- `UUID.MIGRATING_DIR` - `new Uuid(0L, 2L)` – used when transitioning from a previous state where directory assignment was not available, to designate that some directory was previously selected to host a partition, but we're not sure which one yet.

The first 100 UUIDs, minus the three listed above are also reserved for future use.

Metadata records

`RegisterBrokerRecord` and `BrokerRegistrationChangeRecord` will have a new field:

```
{ "name": "LogDirs", "type":  "[]uuid", "versions":  "3+", "taggedVersions": "3+", "tag": "0",
  "about": "Log directories configured in this broker which are available." }
```

`PartitionRecord` and `PartitionChangeRecord` will both have a new `Directories` field

```
{ "name": "Directories", "type": "[]uuid", "versions": "1+",
  "about": "The log directory hosting each replica, sorted in the same exact order as the Replicas field."}
```

Although not explicitly specified in the schema, the default value for `Directory` is `Uuid.UNASSIGNED_DIR (Uuid.ZERO)`, as that's the default *default value* for UUID types. [2] A directory assignment to `Uuid.UNASSIGNED_DIR` conveys that the log directory is not yet known, the hosting Broker will eventually determine the hosting log directory and use `AssignReplicasToDirs` to update this the assignment.

RPC requests

`BrokerRegistrationRequest` will include the following new field:

```
{ "name": "LogDirs", "type":  "[]uuid", "versions":  "2+",
  "about": "Log directories configured in this broker which are available." }
```

`BrokerHeartbeatRequest` will include the following new field:

```
{ "name": "OfflineLogDirs", "type":  "[]uuid", "versions": "1+", "taggedVersions": "1+", "tag": "0",
  "about": "Log directories that failed and went offline." }
```

A new RPC named `AssignReplicasToDirs` will be introduced with the following request and response:

```
{
  "apiKey": <TBD>,
  "type": "request",
  "listeners": ["controller],
  "name": "AssignReplicasToDirsRequest",
  "validVersions": "0",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "BrokerId", "type": "int32", "versions": "0+", "entityType": "brokerId",
      "about": "The ID of the requesting broker" },
    { "name": "BrokerEpoch", "type": "int64", "versions": "0+", "default": "-1",
      "about": "The epoch of the requesting broker" },
    { "name": "Directories", "type":  "[]DirectoryData", "versions": "0+", "fields":  [
      { "name":  "Id", "type": "uuid", "versions": "0+", "about": "The ID of the directory" },
      { "name": "Topics", "type": "[]TopicData", "versions": "0+", "fields": [
        { "name":  "TopicName", "type": "uuid", "versions": "0+",
          "about": "The name of the assigned topic" },
        { "name": "Partitions", "type": "[]PartitionData", "versions": "0+", "fields": [
          { "name": "PartitionIndex", "type": "int32", "versions": "0+",
            "about": "The partition index" }
        ]}
      ]}
    ]}
  ]
}
```

```
{
  "apiKey": <TBD>,
  "type": "response",
  "name": "AssignReplicasToDirsResponse",
  "validVersions": "0",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The top level response error code" },
    { "name": "Directories", "type":  "[]DirectoryData", "versions": "0+", "fields":  [
      { "name":  "Id", "type": "uuid", "versions": "0+", "about": "The ID of the directory" },
      { "name": "Topics", "type": "[]TopicData", "versions": "0+", "fields": [
        { "name":  "TopicId", "type": "uuid", "versions": "0+",
          "about": "The name of the assigned topic" },
        { "name": "Partitions", "type": "[]PartitionData", "versions": "0+", "fields": [
          { "name": "PartitionIndex", "type": "int32", "versions": "0+",
            "about": "The partition index" },
          { "name": "ErrorCode", "type": "int16", "versions": "0+",
            "about": "The partition level error code" }
        ]}
      ]}
    ]}
  ]
}
```

A `AssignReplicasToDirs` request including an assignment to `Uuid.LOST_DIR` conveys that the Broker is wanting to correct a replica assignment into a offline log directory, which cannot be identified.

This request is authorized with CLUSTER_ACTION on CLUSTER.

Proposed changes

Metrics

MBean name	Description
kafka.server:type=KafkaServer, name=QueuedReplicaToDirAssignments	The number of replicas hosted by the broker that are either missing a log directory assignment in the cluster metadata or are currently found in a different log directory and are queued to be sent to the controller in a `AssignReplicasToDirs` request.

Configuration

The following configuration option is introduced

Name	Description	Default	Valid Values	Priority
`log.dir.failure.timeout.ms`	If the broker is unable to successfully communicate to the controller that some log directory has failed for longer than this time, and there's at least one partition with leadership on that directory, the broker will fail and shut down.	30000 (30 seconds)	[1, ...]	low

Storage format command

The `format` subcommand will be updated to ensure each log directory has an assigned UUID and it will persist a new property `directory.id` in the `meta.properties` file. The value is base64 encoded, like the cluster UUID.

The `meta.properties` version field will stay set to 1, to allow for a downgrade after an upgrade on a non JBOD KRaft cluster. [3] The UUIDs for each log directory are automatically generated by the tool if there isn't one assigned already in an existing `meta.properties` file.

Having a persisted UUID at the root of each log directory allows the broker to identify the log directory regardless of the mount path.

Example

Given the following `server.properties`:

```
(... other non interesting properties omitted ...)
process.roles=broker
node.id=8
metadata.log.dir=/var/lib/kafka/metadata
log.dirs=/mnt/d1,/mnt/d2
```

The command `./bin/kafka-storage.sh format -c /tmp/server.properties --cluster-id 41QSStLtR3qOekbX4ZlbHA` would generate three `meta.properties` files that could look like the following:

`/var/lib/kafka/metadata/meta.properties` :

```
#
#Thu Aug 18 15:23:07 BST 2022
node.id=8
version=1
cluster.id=41QSStLtR3qOekbX4ZlbHA
directory.id=e6umYSUsQyq7jUUzL9iXMQ
```

```
/mnt/d1/meta.properties :
#
#Thu Aug 18 15:23:07 BST 2022
node.id=8
version=1
cluster.id=41QSStLtR3qOekbX4ZlbHA
directory.id=b4d9ExdORgaQq38CyHwWTA
```

```
/mnt/d2/meta.properties :
#
#Thu Aug 18 15:23:07 BST 2022
node.id=8
version=1
cluster.id=41QSStLtR3qOekbX4ZlbHA
directory.id=P2aL9r4sSqqyt7bC0uierg
```

Each directory, including the directory that holds the cluster metadata topic — `metadata.log.dir` — has a different and respective value as the directory ID.

In the example above, we can identify the following directory mapping:

- `/var/lib/kafka/metadata` has log directory UUID `e6umYSUsQyq7jUUzL9iXMQ`
- `/mnt/d1` has log directory UUID `b4d9ExdORgaQq38CyHwWTA`
- `/mnt/d2` has log directory UUID `P2aL9r4sSqqyt7bC0uierg`

Brokers

Broker lifecycle management

When the broker starts up and initializes `LogManager`, it will load the UUID for each log directory (`directory.id`) by reading the `meta.properties` file at the root of each of them.

- If there are any two log directories with the same UUID, the Broker will fail at startup
- If there are any `meta.properties` files missing `directory.id`, a new UUID is generated, and assigned to that directory by updating the `meta.properties` file.

The set of all loaded log directory UUIDs is sent along in the broker registration request to the controller as the `LogDirs` field.

Metadata caching

Currently, Replicas are considered offline if the hosting broker is offline. Additionally, replicas will also be considered offline if the replica references a log directory UUID (in the new field `partitionRecord.Directories`) that is not present in the hosting Broker's latest registration under `LogDirs` and either:

- the log directory UUID is `UUID.LOST_DIR`
- the hosting broker's registration indicates multiple online log directories. i.e. `brokerRegistration.LogDirs.length > 1`

If neither of the above conditions are true, we assume that there is only one log directory configured, the broker is not configured with multiple log directories, replicas all live in the same directory and neither log directory assignments nor log directory failures shall be communicated to the Controller.

Handling log directory failures

When multiple log directories are configured, and some (but not all) of them become offline, the broker will communicate this change using the new field `OfflineLogDirs` in the `BrokerHeartbeat` request — indicating the UUIDs of the new offline log directories. The UUIDs for the accumulated failed log directories are included in every `BrokerHeartbeat` request until the broker restarts. If the Broker is configured with a single log directory, this field isn't used, as the current behavior of the broker is to shutdown when no log directories are online.

Log directory failure notifications are queued and batched together in all future broker heartbeat requests.

If the Broker repeatedly fails to communicate a log directory failure, or a replica assignment into a failed directory, after a configurable amount of time — `log.dir.failure.timeout.ms` — and it is the leader for any replicas in the failed log directory the broker will shutdown, as that is the only other way to guarantee that the controller will elect a new leader for those partitions.

Replica management

When configured with multiple `log.dirs`, as the broker catches up with metadata, and sees the partitions which it should be hosting, it will check the associated log directory UUID for each partition (`partitionRecord.Directories`).

- If the partition is not assigned to a log directory (refers to `Uuid.UNASSIGNED_DIR`)
  - If the partition already exists, the broker uses the new RPC — `AssignReplicasToDirs` — to notify the controller to change the metadata assignment to the actual log directory.
  - If the partition does not exist, the broker selects a log directory and uses the new RPC — `AssignReplicasToDirs` — to notify the controller to create the metadata assignment to the actual log directory.
- If the partition is assigned to an online log directory
  - If the partition does not exist it is created in the indicated log directory.
  - If the partition already exists in the indicated log directory and no future replica exists, then no action is taken.
  - If the partition already exists in the indicated log directory, and there is a future replica in another log directory, then the broker starts the process to replicate the current replica to the future replica.
  - If the partition already exists in another online log directory and is a future replica in the log directory indicated by the metadata, the broker will replace the current replica with the future replica after making sure that the future replica is fully caught up with the current replica.
  - If the partition already exists in another online log directory, the broker uses the new RPC — `AssignReplicasToDirs` — to the controller to change the metadata assignment to the actual log directory. The partition might have been moved to a different log directory whilst the broker was offline.
- If the partition is assigned to an unknown log directory or refers to `Uuid.LOST_DIR`
  - If there are offline log directories, no action is taken — the assignment refers to a a log directory which may be offline, we don't want to fill the remaining online log directories with replicas that existed in the offline ones.
  - If there are no offline directories, the broker selects a log directory and uses the new RPC — `AssignReplicasToDirs` — to notify the controller to create the metadata assignment to the actual log directory.

If instead, a single entry is configured under `log.dirs` or `log.dir`, then the `AssignReplicasToDirs` RPC is only sent to correct assignments to `UUID.LOST_DIR`, as described above.

If the broker is configured with multiple log directories it remains FENCED until it can verify that all partitions are assigned to the correct log directories in the cluster metadata. This excludes the log directory that hosts the cluster metadata topic, if it is configured separately to a different path — using `metadata.log.dir`.

Assignments to be sent via `AssignReplicasToDirs` are queued and batched together, handled by a log directory event manager that also handles log directory failure notifications.

Intra-broker replica movement

Support for replica movement between directories was introduced in KIP-113. This functionality is maintained, but altered slightly so that the controller remains correctly informed of the log directory for any moving replica.

The existing `AlterReplicaLogDirs` RPC is sent directly to the broker in question, which starts moving the replicas using `ReplicaAlterLogDirsThread` – this remains unchanged. But when the future replica first catches up with the main replica, instead of immediately promoting the future replica, the broker will:

1. Asynchronously communicate the log directory change to the controller using the new RPC – `AssignReplicasToDirs`.

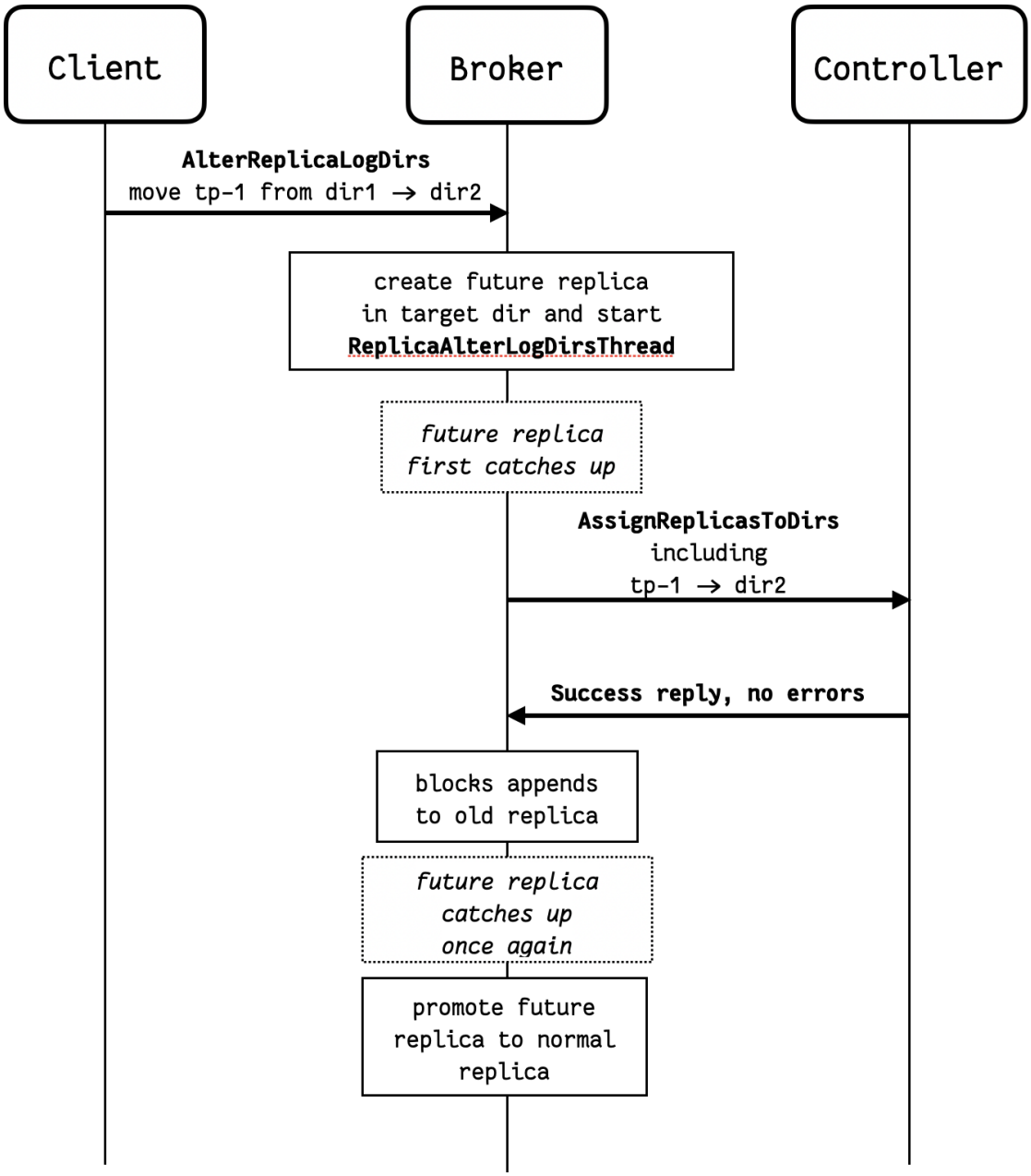
2. Keep the `ReplicaAlterLogDirsThread` going. The future replica is still the future replica, and it continues to copy from the main replica – which still in the original log directory – as new records are appended.

Once the broker receives confirmation of the metadata change – indicated by a successful response to `AssignReplicasToDirs` – then it will:

1. Block appends to the main (old) replica and waits for the future replica to fully catch up once again.
2. Makes the switch, promoting the future replica to main replica and cleaning up the old one.

By delaying the metadata change until the future replica has caught up we minimize the chance of a log directory failure happening with an incorrect replica to log directory assignment in the metadata.

The diagram below illustrates the sequence of steps involved in moving a replica between log directories.



In the diagram above, notice that if `dir1` fails after the `AssignReplicasToDirs` RPC is sent, but before the future replica is promoted, then the controller will not know to update leadership and ISR for the partition. If the destination directory has failed, it won't be possible to promote the future replica, and the Broker needs to revert the assignment (cancelled locally if still queued). If the source directory has failed, then the future replica might not catch up, and the Controller might not update leadership and ISR for the partition. In this exceptional case, the broker issues a `AssignReplicasToDirs` RPC to the Controller to assignment the replica to `UUID.LOST_DIR` - this lets the Controller know that it needs to update leadership and ISR for this partition too.

Controller

Replica placement

For any new partitions, the active controller will use `Uuid.UNASSIGNED_DIR` as the initial value for log directory UUID for each replica – this is the default (empty) value for the tagged field. Each broker with multiple `log.dirs` hosting replicas then assigns a log directory UUID and communicates it back to the active controller using the new RPC `AssignReplicasToDirs` so that cluster metadata can be updated with the log directory assignment. Brokers that are configured with a single log directory to not send this RPC.

Handling log directory failures

When a controller receives a `BrokerHeartbeat` request from a broker that indicates any UUIDs under the new `OfflineLogDirs` field, it will:

- Persist a `BrokerRegistrationChange` record, with the new list of online log directories.
- Update the Leader and ISR for all the replicas assigned to the failed log directories, persisting `PartitionChangeRecords`, in a similar way to how leadership and ISR is updated when a broker becomes fenced, unregistered or shuts down.

If the any of the listed log directory UUIDs is not a registered log directory then the call fails with error 57 — `LOG_DIR_NOT_FOUND`.

Handling replica assignments

The controller accepts the `AssignReplicasToDirs` RPC and persists the assignment into metadata records.

If the indicated log directory UUID is not one of the Broker's online log directories, then the replica is considered offline and the leader and ISR is updated accordingly, same as when the `BrokerHeartbeat` indicates a new offline log directory.

Broker registration

Upon a broker registration request the controller will persist the broker registration as cluster metadata including the online log directory list and offline log directories flag for that broker. The controller may receive a new list of online directories and offline log directories flag — different from what was previously persisted in the cluster metadata for the requesting broker.

- If there are no indicated online log directory UUIDs the request is invalid and the controller replies with an error 42 – `INVALID_REQUEST`.
- If multiple log directories are registered the broker will remain fenced until the controller learns of all the partition to log directory placements in that broker - i.e. no remaining replicas assigned to `Uuid.UNASSIGNED_DIR` . The broker will indicate these using the AssignReplicasToDirs RPC.
    - The broker remains fenced by not wanting to unfence itself in heartbeat requests until the number of mismatching replica to log directory assignments is zero. This number is represented by the new metric `QueuedReplicaToDirAssignments`.
- If multiple log directories are registered and some of them are new (not present in previous registration) then these log directories are assumed to be empty. If they are not, the broker will use the `AssignReplicasToDirs` RPC to correct assignment and choose not to become UNFENCED before the metadata is correct.

Brokers whose registration indicates that multiple log directories are configured remain FENCED until all log directory assignments for that broker are learnt by the active controller and persisted into metadata.

Compatibility, Deprecation, and Migration Plan

The metadata.version will be bumped to gate changes to the RPCs and metadata records.

Migrating a cluster in KRaft without JBOD

The cluster needs to be upgraded before configuring multiple entries in `log.dirs`. After the upgrade, the `metadata.version` feature flag needs to be upgraded using `kafka-features.sh`. Then the brokers can be reconfigured with multiple entries in `log.dirs`.

Upon being reconfigured with multiple log directories, brokers will update and generate `directory.id` in `meta.properties` as necessary to reflect the new log directories. Brokers will then register the log directories with the controller via `BrokerRegistration` and use `AssignReplicasToDirs` to create the partition-logdirectory assignments in the cluster metadata before becoming UNFENCED.

Migrating a cluster in ZK mode running with JBOD

Migration into KRaft mode is addressed in KIP-866. That migration is extended in the following way:

- As per KIP-866, a separate Controller quorum is setup first, and only then the existing brokers are reconfigured and upgraded.
- When configured for the migration and while still in ZK mode, brokers will:
    - update meta.properties to generate and include `directory.id;`

    - send `BrokerRegistrationRequest` including the log directory UUIDs;
    - shutdown if any directory fails;
    - sends assignments via the `AssignReplicasToDirs` RPC.
- During the migration, the controller:
    - persists log directories indicated in broker registration requests in the cluster metadata;
    - persists directory assignments received via the `AssignReplicasToDirs` RPC.
- The brokers restarting into KRaft mode will want to stay fenced until their log directory assignments for all hosted partitions are persisted in the cluster metadata.
- The active controller will also ensure that any given broker stays fenced until it learns of all partition to log directory assignments in that specific broker via the new `AssignReplicasToDirs` RPC.
- During the migration, existing replicas are assumed and assigned to log directory `Uuid.MIGRATING_DIR` until the actual log directory is learnt by the active controller from a broker running in KRaft mode.

Replica management

Existing replicas without a log directory are either:

- Assumed to live in a broker that isn't yet configured with multiple log directories, and so live in a single log directory. It is not possible to trigger a log directory failure from a broker that has a single log directory, as the broker would simply shut down if there are no remaining online log directories. Or
- Assigned to a log directory as of yet unknown, in a broker that remains FENCED. As the broker remains FENCED it cannot assume leadership for any partition, and so a log directory failure would be handled by the current partition leader.

The two assumptions above eliminate the risk of having a broker which is not shutting down, but is unable to continue its leadership responsibilities due to the partition being persisted in a log directory that is broken or otherwise unavailable and the active controller not being aware of such an issue.

Storage formatting

The changes to storage formatting simply ensure the existence of two new fields of in an existing metadata file – `meta.properties` – at the log directory roots. The new fields are ignored by earlier versions of Kafka.

Test plan

The system test for log directory failures will be extended to KRaft mode.

This feature has been modeled in TLA+.

Future work

- Partition reassignment across directories and across brokers involves different API calls — `AlterPartitionReassignments` and `AlterReplicaLogDirs`. Whilst reassigning partitions across brokers into a specific log directory is already possible, it involves an intricate sequence of prior calls to `AlterReplicaLogDirs` and expecting errors as a successful result. Once this work is done we can consolidate these two API calls by extending `AlterPartitionReassignments` to allow target log directories to be specified and deprecate `AlterReplicaLogDirs`. This can be done as part of a future KIP.
- The only way to know which log directory UUID corresponds to which log directory path is by reading the `meta.properties` files in each broker. A future KIP should expand the `DescribeLogDirs` RPC response to include log directory UUIDs along with the system path for each log directory.
- Partition initialization can be optimized, by having the controller preselect a log directory for new partitions. This would avoid having to wait for the broker to send a `AssignReplicasToDirs` request to indicate the chosen log directory before it is safe for the broker to assume leadership of the partition. Maybe the Controller could also take available storage in each log directory into account if the Broker indicates the available storage space for each log directory as part of broker registration. This may be be proposed in a future KIP, but we'd need to figure out a way to distinguish between a Controller initiated move, and a user manual move of a partition between log directories when the Broker is offline. [4]

Rejected alternatives

- Keeping the scope of the log directory to the broker — while this would mean a much simpler change, as was proposed in KIP-589, if only the broker itself knows which partitions were assigned to a log directory, when a log directory fails the broker will need to send a potentially very large request enumerating all the partitions in the failed disk, so that the controller can update leadership and ISRs accordingly.
- Having the controller determine the log directory for new replicas — this would avoid a further RPC from the broker upon selecting a new log directory for new replicas, and reduce the time until it is safe for the broker to take leadership of the replica. However the broker is in a better position to make a choice of log directory than the controller, as it has easier access to e.g. disk usage in each log directory. The controller could also have this information if the broker were to include it the broker registration. But to keep this change manageable and timely, this optimization is best left for future work. It would also be trickier to manage the ZKKRaft migration if we had some brokers forwarding the RPC to the controllers and others handling it directly.
- Changing how log directory failure is handled in ZooKeeper mode — ZooKeeper mode is going away, KIP-833 proposed its deprecation in a near future release.
- Using the system path to identify each log directory, or storing the identifier somewhere else — When running Kafka with multiple log directories, each log directory is typically assigned to a different system disk or volume. The same storage device can be made accessible under a different mount, and Kafka should be able to identify the contents as the same disk. Because the log directory configuration can change on the broker, the only reliable way to identify the log directory in each broker is to add metadata to the file system under the log directory itself.

- Identifying offline log directories. Because we cannot identify them by mount paths, we cannot distinguish between an inaccessible log directory is simply unavailable or if it has been removed from configured – think of a scenario where one log dir is offline and one was removed from `log.dirs`. In ZK mode we don't care to do this, and we shouldn't do it in KRaft either. What we need to know is if there are any offline log directories, to prevent re-streaming the offline replicas into the remaining online log dirs. In ZK mode, the 'isNew' flag is used to prevent the Broker from creating partitions when any logdir is offline unless they're new. A simple boolean flag to indicate some log dir is offline is enough to maintain the functionality.

Footnotes

1. The exception is the cluster metadata log directory, which can be configured separately via `metadata.log.dir` . If the metadata log directory fails, then the broker cannot continue to run. If the broker isn't running it won't send any heartbeats and the controller will know to reassign leadership and update ISRs. The main benefit of support for multiple log directories is to allow brokers to continue operating if any single one of them fails. Typically, each log directory is mapped to an independent storage device while this critical metadata log directory would instead be mapped to the one of the main system partitions.
2. Yes, double default, not a typo. The default setting, for the default value of the field.
3. If an existing, non JBOD KRaft cluster is upgraded to the first version that includes the changes described in this KIP, which write these new fields, and is later downgraded, the `meta.properties` file needs to still be readable. There's currently a hard check on the version number which would fail for a new version number.
4. Despite not being an advertised feature, currently replicas can be moved between log directories while the broker is offline. Once the broker comes back up it accepts the new location of the replica. To continue supporting this feature, the broker will need to compare the information in the cluster metadata with the actual replica location during startup and take action on any mismatch.