KIP-188 - Add new metrics to support health checks

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *KAFKA-5746*

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

Motivation

For Kafka Ops, it will be useful to have some more metrics to support health checks so that any issues can be identified early.

Public Interfaces

Broker-side metrics

Error rates

There are currently no metrics to report error rates in the broker. It will be useful to monitor errors returned to clients for each request type, so that alerts can be generated if requests fail consistently. Each error code for each request type will be measured separately to provide flexibility in terms of how errors are processed by downstream tools.

This metric will be a meter in the same group as existing request metrics `RequestsPerSec` etc.

MBean: `kafka.network:type=RequestMetrics,name=ErrorsPerSec,request=api_key_name,error=error_code_name`

Since only a small number of error codes are used per request, entries will be created dynamically. Unused entries will be expired similar to quota metric expiry.

When responses contain multiple errors (e.g. per-partition error in Produce response or per-topic error in CreateTopics), all the errors in the response are counted. So there is a 1:n mapping between responses and errors for some responses. The scope of errors in each response is shown in this table.

ApiKey	Scope of error	Request:Errors Mapping
UpdateMetadata	request	1:1
ControlledShutdown	request	1:1
FindCoordinator	request	1:1
JoinGroup	request	1:1
Heartbeat	request	1:1
LeaveGroup	request	1:1

SyncGroup	request	1:1
ListGroups	request	1:1
SaslHandshake	request	1:1
ApiVersions	request	1:1
InitProducerId	request	1:1
AddOffsetsToTxn	request	1:1
EndTxn	request	1:1
DescribeAcls	request	1:1
Produce	partition	1:n
Fetch	partition	1:n
Offsets	partition	1:n
OffsetCommit	partition	1:n
OffsetFetch	partition	1:n
DeleteRecords	partition	1:n
OffsetForLeaderEpoch	partition	1:n
AddPartitionsToTxn	partition	1:n
WriteTxnMarkers	partition	1:n
TxnOffsetCommit	partition	1:n
LeaderAndIsr	partition + request	1:n
StopReplica	partition + request	1:n
Metadata	topic	1:n
CreateTopics	topic	1:n
DeleteTopics	topic	1:n
DescribeGroups	group	1:n
CreateAcls	acl	1:n
DeleteAcls	acl	1:n
DescribeConfigs	resource	1:n
AlterConfigs	resource	1:n

Message conversion rate and time

Down conversions are expensive since the whole response has to be read into memory for conversion. It will be useful to monitor the rate of down conversion and the time spent on conversions.

Fetch and produce message conversion rates will be meters in the same group as existing topic metrics `TotalFetchRequestsPerSec` etc.

MBean: `kafka.server:type=BrokerTopicMetrics,name=FetchMessageConversionsPerSec,topic=([-.\w]+)`

MBean: `kafka.server:type=BrokerTopicMetrics,name=ProduceMessageConversionsPerSec,topic=([-.\w]+)`

It will also be useful to know the time taken for down conversions. Fetch down conversion time metric will be a histogram alongside other request time metrics. This time will also be included in request logs so that clients requiring expensive down conversions can be identified. Conversion time will also be added for produce requests.

MBean: `kafka.network:type=RequestMetrics,name=MessageConversionsTimeMs,request={Produce|Fetch}`

Request size and temporary memory size

Large messages can cause GC issues in the broker, especially if down conversions are required. Maximum message batch size can be configured per topic to control this, but that is the size after compression. Since the batches are decompressed to validate produce requests and for fetch down conversion, it will be useful to have metrics for produce message batch size.

Request metrics will be added for request size as well as the temporary memory size for processing the request. These two metrics will be histograms. For produce messages the two metrics will indicate message batch size before and after decompression as well as the compression ratio. The values will also be included in request logs so that clients requiring a lot of temporary memory space can be identified.

MBean: `kafka.network:type=RequestMetrics,name=RequestBytes,request=<apiKey>`

MBean: `kafka.network:type=RequestMetrics,name=TemporaryMemoryBytes,request=<apiKey>`

Authentication success and failure rates

Rate of failed authentications are useful to identify misconfigured or malicious connection attempts. Successful connection rates may also be helpful for each listener.

These metrics will be Kafka metrics added to the same group as network metrics like `connection-creation-rate`.

MBean: kafka.server:`type=socket-server-metrics,listener=<listenerName>,networkProcessor=<processorIndex>`

New attributes:

1. successful-authentication-rate
2. failed-authentication-rate

ZooKeeper status and latency

It will be good to monitor latency of ZooKeeper requests so that any issues with ZooKeeper communication can be detected early.

This will be a histogram in a new group ZooKeeperClient.

MBean: `kafka.server:type=ZooKeeperClientMetrics,name=ZooKeeperRequestLatencyMs`

It will also be useful to see the current status of broker's connection to ZooKeeper.

This will be a String Gauge in the existing group SessionExpireListener which currently shows the rate of each state (eg. DisconnectsPerSec)

MBean: `kafka.server:type=SessionExpireListener,name=SessionState`

State will be one of `Disconnected|SyncConnected|AuthFailed|ConnectedReadOnly|SaslAuthenticated|Expired`

Client-side metrics

Client versions

We currently have a MBean for client version that gives commit id and version, but this is not exposed as a metric. In order to optimize upgrades and debug issues, it will be useful to have a gauge for client versions to monitor the versions used by clients.

Two metrics will be added as `Attributes` `commit-id` and `version` with String value indicating the commit-id/version.

MBean: `[kafka.admin.client|kafka.consumer|kafka.producer]:type=app-info,client-id=ClientId`

Proposed Changes

New yammer metrics will be added on the broker-side and Kafka metrics on the client-side.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*

These are new metrics and there will be no impact on existing users.

Rejected Alternatives