KIP-786: Emit Metric Client Quota Values

Status

Current state: Under Discussion

Discussion thread: Discussion Thread

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

Motivation

We have had many situations where clients have been throttled due to exceeding their produce/fetch or request quota. Unfortunately, with possibly varying quotas assigned across clients, the only means of preforming capacity planning around throttling requires dynamically fetching the Zk quota config. While this can be done using the `AdminClient` under KIP-546 it leads to disjoint data sources - requiring additional work to implement alerting and capacity planning for clients nearing their quota.

Goals

- Emit the quota (upper-bound) value tagged at the `client-id/user` granularity as an additional attribute to the `kafka.server` MBean
- Provide broker level configuration to enable the recording of the quota value.
  - In the case the configuration is not set, no additional recording overhead should be added to the user request (i.e. the caching and creation of an additional sensor).

Public Interfaces

Configuration

A new boolean configuration property `"client.quota.value.metric.enable"` has been added as a static kafka.server.`KafkaConfig`. As expected, this config must be set to `"true"` in order to enabled emitting the quota metric.

Monitoring

An additional attribute is added to each of of the `MBeans`:

MBean Name	Current Attributes	Proposed New Attribute
`kafka.server:type={Produce|Fetch},user=([-.\w]+),client-id=([-.\w]+)`	`byte-rate`, `throttle-time`	`quota-value`
`kafka.server:type=Request,user=([-.\w]+),client-id=([-.\w]+)`	`request-time, throttle-time`	`quota-value`

Proposed Changes

Configuration

As mentioned above, the following boolean config will be added to `kafka.server.KafkaConfig` with the property defined as "`client.quota.metric.value.enable`".

Quota Managers

To facilitate emitting the new metric, an additional `Sensor` is needed that records the quota value. To fit into the mechanism used for caching client sensors, as well as obtaining the write lock for recording. This sensor is only created in the case where the quota is enabled.

ClientQuotaManager.scala

```
 def getOrCreateQuotaSensors(session: Session, clientId: String): ClientSensors = {
  [...]
    Option(config.quotaValueMetricEnable).collect{ case true =>
      sensorAccessor.getOrCreate(
      getQuotaSensorName(metricTags, "Value"),
      ClientQuotaManagerConfig.InactiveSensorExpirationTimeSeconds,
      clientQuotaValueMetricName(metricTags),
      Some(getQuotaBaseMetricConfig),
      new Value
    )}
  )
  [...]
}
```

Following this, on each `Fetch` or `Produce` API request the quota value is then recorded.

ClientQuotaManager.scala

```
def recordAndGetThrottleTimeMs(session: Session, clientId: String, value: Double, timeMs: Long): Int = {
  val clientSensors = getOrCreateQuotaSensors(session, clientId)
  try {

    Option(quotaCallback.quotaLimit(clientQuotaType, clientSensors.metricTags.asJava)).foreach(
        q => clientSensors.quotaValueSensor.foreach(s => s.record(q.toDouble, timeMs))
    )

    clientSensors.quotaSensor.record(value, timeMs)
    0
  } catch {
    case e: QuotaViolationException =>
      val throttleTimeMs = throttleTime(e.value, e.bound, windowSize(e.metric, timeMs)).toInt
      debug(s"Quota violated for sensor (${clientSensors.quotaSensor.name}). Delay time: ($throttleTimeMs)")
      throttleTimeMs
  }
}
```

Compatibility, Deprecation, and Migration Plan

No impact will be seen on existing users and no migration plan will be necessary.

Rejected Alternatives

From a design perspective, recording quota values as a metric isn't ideal in the sense that they are largely static (as metrics go). In the case it would make more sense to recording the "available capacity" that a client has available at a given time as a rate. However, in order for the rate to have the correct value some additional work would be needed and it should be noted that as long as the sensor configs (window size and number of windows) this corresponding rate can be calculated after the fact easily.