KIP-612: Ability to Limit Connection Creation Rate on Brokers

- Status
- Motivation
- Public Interfaces
    - Broker configurations for broker-wide and per-listener connection rate limits
    - Dynamic configurations for a new entity type IPs
    - Tools
    - Connection rate throttling behavior
    - Metrics
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives
    - Broker configurations to set limits on per IP connection attempt rates
    - Use incrementalAlterConfigs/describeConfigs for IP quotas

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA:

⚠ Unable to render Jira issues macro, execution error.

⚠ *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

Creating a new connection adds CPU overhead to the broker. KIP-306 mitigated the issue of connection storm due to un-authorized connections (e.g., misconfigured clients). However, connections storms may also come from (mostly) well-behaved clients. One example is when deploying a new application may cause temporary connection storm due to a large number of clients starting up and creating connections to the cluster at the same time. Another example is clients that create a new connection for each produce/consume request, causing high connection rate to the brokers. A very high connection creation rate may stop broker from doing other useful work, causing high request latencies or even URPs.

To address this issue, the KIP proposes to add the ability to set a limit on the rate with which the broker accepts new connections. To ensure that some clients do not take over most of the connection creation rate quota, the KIP also adds the ability to set a limit on connection creation rate per IP.

Public Interfaces

Broker configurations for broker-wide and per-listener connection rate limits

A new broker configuration option will be added to limit the total rate at which non-inter-broker connections will be accepted on the broker. Connections on the inter-broker listener will be permitted even if the configured broker-wide limit is reached. This will be a dynamic config that can be updated without restarting the broker.

Config option: Name: `max.connection.creation.rate` Type: `Int` Default value: `Int.MaxValue`

The config may be prefixed with a listener prefix to specify a different listener-specific limit: `listener.name.{listenerName}.max.connection.creation.rate`. Listener-specific limits will be applied in addition to the broker-wide limit. If a listener-specific limit is not specified, each listener can create connections with the rate up to the broker-wide limit as long as the total rate is also within the broker-wide limit. If a broker has multiple listeners, connections on the inter-broker listener will always succeed as long as connection creation rate is within that listener's rate limit. The behavior of the proposed broker-wide and per-listener configs is consistent with max.connections broker configuration (KIP-402).

Dynamic configurations for a new entity type IPs

Per-IP connection creation rate quota will be configured as a dynamic quota for entity type `ips`.

- Default quota for <IPs> will be stored in Zookeeper at /config/ips/<default>
- Quota for a specific IP address for which quota override is defined will be stored in Zookeeper at /config/ips/<ip-address>

If per-IP or default IP quota is not configured, we will use Int.MaxValue as a default to ensure backward compatibility.

We will use the `AlterClientQuota` and `DescribeClientQuota` APIs to manage IP connection rate quotas. Adding support for IP quotas will not require any changes to the protocol, since entity type and name are represented flexibly as Strings. For ease of use, we will define a new recognized IP entity type in **org.apache.kafka.common.quota.ClientQuotaEntity:**

```
/**
 * Describes a client quota entity, which is a mapping of entity types to their names.
 */
public class ClientQuotaEntity {

    /**
     * The type of an entity entry.
     */
    public static final String USER = "user";
    public static final String CLIENT_ID = "client-id";
        public static final String IP = "ip";
}
```

Since IP quotas are orthogonal with user/client ID quotas the following is to disambiguate how the new `IP` entity will interact with the existing `client-id` and `user` entities.

A `DescribeClientQuotasRequest` constructed with a filter for both ip entities and either of user or client-id represents a logical `AND` of IP quotas and user/client quotas. However, since there are no quotas that satisfy both IP and either of user/client-id, such a request is likely due to user error. In such a case we will return an `INVALID_REQUEST` error.

Likewise, an `AlterClientQuotasRequest` with both an IP entity and either of user or client-id entity will return an `INVALID_REQUEST` error, as there are no shared quota properties between the two entity types.

Tools

`kafka-configs.sh` will be extended to support per-IP connection rate quotas. A new entity type `IPs` will be added with the following key (and a value of type Int):

- `connection_creation_rate` : The total connection rate limit for the connections from a specific IP address.

For example:

```
bin/kafka-configs  --bootstrap-server localhost:9091 --alter --add-config 'connection_creation_rate=100' --entity-name 93.284.53.13 --entity-type ips
```

```
bin/kafka-configs  --bootstrap-server localhost:9091 --alter --add-config 'connection_creation_rate=100' --ip 93.284.53.13
```

Default connection rate quotas for an IP address can be configured via the following:

```
bin/kafka-configs  --bootstrap-server localhost:9091 --alter --add-config 'connection_creation_rate=100' --entity-type ips --entity-default
```

```
bin/kafka-configs  --bootstrap-server localhost:9091 --alter --add-config 'connection_creation_rate=100' --ip-defaults
```

Connection rate throttling behavior

The connection creation rate limits will be applied to the same quota window configuration (`quota.window.size.seconds` with 1 second default) as existing produce/fetch quotas and request rate quota (KIP-124). Since limit on connection creation rate on the broker is also a type of quota, this approach will keep it consistent with the existing quota implementations on the broker. If connection creation rate on the broker exceeds the broker-wide limit, the broker will delay accepting a new connection by an amount of time that brings the rate within the limit. If a listener-specific limit is specified, and the connection rate on that listener exceeds the limit, the broker will delay accepting new connection on that listener by an amount of time that brokers the rate of the listener within the listener limit. The maximum delay applied will be the quota window size (which also means that the minimum connection rate limit is effectively 1 connection creation / second).

Since the broker has to accept the connection in order to know the IP address, limiting connection creation rate per IP means that the enforcement of the limit has to happen after connection is accepted. Dropping the connection right away as broker does for per-IP connection count limits (KIP-402) does not work as well for connection rate limiting, because the offending client is likely to immediately reconnect. At the same time, connection rate limiting by only delaying processing of connections would not work for cases where the connection creation rate is continuously higher than the set limits, which would create a large backlog. This case could be more common for clients that come through a proxy, and where there could be potentially a large number of incoming connections with the same IP. To address these two issues, our approach is as follows. If connection creation rate is reached for a specific IP address, the broker will delay processing the connection by an amount of time that brings the rate within the limit or 1 second, whichever is earliest. After the delay, if the per-IP quota is still violated, the connection will be cleaned up; otherwise, the connection will be accepted.

Metrics

New metrics that track broker-wide or per-listener rate of accepting connections, and are used by SocketServer to detect connection acceptance rate quota violation:

- `kafka.network:type=socket-server-metrics,name=connection-accept-rate,listener={listenerName}`
  - Type: Rate
  - Description: Rate of connections accepted per second on a given listener
- `kafka.network:type=socket-server-metrics,name=broker-connection-accept-rate`
  - Type: Rate
  - Description: Broker-wide rate of connections accepted per second
- `kafka.network:type=socket-server-metrics,name=connection-accept-rate,ip={ipAddress}`
  - Type: Rate
  - Description: Rate of connections accepted per second for a given IP

New metrics that track average throttle time of accepting a new connection due to reaching connection acceptance rate limit:

- `kafka.network:type=socket-server-metrics,name=connection-accept-throttle-time,listener={listenerName}`
  - Type: SampledStat.Avg
  - Description: Average throttle time due to violating per-listener or broker-wide connection acceptance rate quota on a given listener.
- `kafka.network:type=socket-server-metrics,name=ip-connection-accept-throttle-time,listener={listenerName}`
  - Type: SampledStat.Avg
  - Description: Average throttle time due to violating ip connection rate quota on a given listener. IP throttling is reported at a listener granularity rather than at an IP granularity to keep a smaller metrics overhead.

The existing metric (`kafka.network:type=Acceptor,name=AcceptorBlockedPercent,listener={listenerName}`) that tracks the amount of time `Acceptor` is blocked from accepting connections will now additionally include the amount of time `Acceptor` is blocked due to hitting connection create limit (in addition to the time blocked due to hitting the maximum limit on currently active connections).

Proposed Changes

The broker will track connection acceptance rates, broker-wide, per every listener and per IP, via sensors that wrap the `Rate` metric with the `MetricConfig`. `MetricConfig#quota` will be set to the corresponding configured connection creation rate limit. When `Acceptor` accepts a new connection, the broker-wide and the corresponding listener's metric will be incremented. When the actual connection creation rate exceeds either broker-wide or listener-specific quota, quota violation exception will be thrown. On quota violation, the broker will calculate the delay needed to bring the metric within quota by using the same formula implemented in `ClientQuotaManager.throttleTime`. The `Acceptor` thread will wait for the delay duration before accepting new connections. The maximum delay applied will be the quota window size (1 second by default).

Most of this logic will be added to `ConnectionQuotas` class, which currently throttles `Acceptor` thread to limit the number of active connections. `ConnectionQuotas` class be extended to enforce both the number of active connections and connection creation rate. This proposal adds another condition when the `Acceptor` thread waits, which will be implemented as delaying accepting a new connection based on whichever limit is reached first:

- If the number of active connections is below the limit, but broker hits the connection rate limit, the `Acceptor` will wait for the calculated delay that brings the connection creation rate metric within quota.
- If there are no available active connection slots, the broker waits for the new slot independent of whether connection rate exceeds quota or not.

When quota violation happens due to reaching the limit for a IP address, we will calculate the delay = min(delay required to bring the rate within the quota, 1 second). We will re-use the same mechanism implemented with KIP-306 where the broker delays the response for failed client authentication. When the delay passes, we will check for quota violation again. The connection will be cleaned up (dropped) if the quota is still violated, or the connection will be added to one of the Processor queues for processing (accepted).

Compatibility, Deprecation, and Migration Plan

The feature is backward compatible, because the default configuration will have a legacy behavior: No connection creation rate limit.

There will be no impact on existing users.

Rejected Alternatives

Broker configurations to set limits on per IP connection attempt rates

Add broker configurations for setting the default per-IP connection attempt limit and per-IP limit overrides:

Config option: Name: `max.connection.creation.rate.per.ip` Type: `Int` Default value: `Int.MaxValue`

Config option: Name: `max.connection.creation.rate.per.ip.overrides` Type: `String` Default value: ""

We chose the option of making per-IP quota similar to other quotas, configured as a dynamic quota for entity type "ip" as well as defaults.

Use incrementalAlterConfigs/describeConfigs for IP quotas

`DescribeConfigsRequest` requires the user to specify a list of `ConfigResource` which consists of a `Type` and `Name` for which to retrieve the desired configs. An empty name indicates that we want to describe the configs for the entity type default. This works well for describing broker and topic configs, because the full list of brokers and topics is retrievable via other requests.

However, for IPs there is no mechanism for IP discovery, and `DescribeConfigsRequest` has no way to indicate that a request should list all configs for an entity type without also giving a list of entity names. This would make discovery of which IPs have a connection rate quota unfeasible, so this was rejected.