CEP-41 (DRAFT) Apache Cassandra Unified Rate Limiter

Status

Current state: *Draft*

Discussion thread:

JIRA:

Released: <Cassandra Version>

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

Motivation

When we run Cassandra as a service at scale having thousands of unique workloads, from time to time, these workloads put pressure on Cassandra due to a variety of reasons:

- A sudden spike in the number of requests
- Hot shards
- Anti-patterns

Once a Cassandra server is bombarded, it melts down, and due to cascading failures, the server goes down and takes down the entire Cassandra ring. So, one or two slower nodes create a ripple effect on other healthy peers and take those nodes down; eventually, all Cassandra nodes become inoperable.

Recovery often takes quite a while, which requires a heavy operational burden. The idea is to prevent the server-side meltdown and keep the impact contained to the hot nodes only so it does not spread across the entire ring.

Audience

This enhancement proposal unlocks newly adopting Apache Cassandra users, who often must make significant investments before using it.

Goals

Apache Cassandra already has rate limiters supported at various layers, e.g., an incoming request-based one as part of CASSANDRA-15013, a Native transport one as part of CASSANDRA-16663, etc. However, the existing rate limiter only solves the targeted scope, e.g., focusing on request rate, native transport, etc.

Often, we see that a couple of busy Cassandra server nodes take down the entire Cassandra ring due to the ripple effect. The goal is to develop a unified and comprehensive rate limiter that works considering system signals, such as CPU, and internal signals, such as various thread pools. The rate limiter will have the capability to exclude critical traffic, such as system tables, and replication, as well as the capability to throttle on a percentage basis for a safe rollout, filtering query types, circuit breaker, etc.

1. Align on various signals to consider for a unified rate limiter framework design.
2. Consider the depth (at the native transport layer, thread pool layer, etc.) at which the signals would be measured because the effectiveness of the rate limiter decreases if we measure too deep/late in the stack.
3. Rate limits the read and/or mutation replication traffic from peer nodes.
4. Table-level circuit-breakers.
5. Various filtering policies apply to the rate limiter.
6. The safest way to incrementally roll out the rate limiter at scale in a large-managed fleet.
7. The solution should be extremely easy for an operator to manage, especially the configuration portion.
8. Extremely conservative to begin with. No false negatives, false positives are ok to begin with.

The new framework should be complementary to the existing solutions, in other words, if anyone is using the rate-limiter solutions mentioned above, then this framework adoption should not break the existing ones.

Non-Goals

1. Capable of covering 100% of the scenarios in which one feels the need for the rate limiter.

Proposed Changes

Solution#1 Multiple Design Proposals From Dev Mailing List

https://lists.apache.org/thread/swo9n7s5y9ldm4dsvy34q1lcjns4wxkl

Solution#2 Unified Rate Limiter in Apache Cassandra

Open-source Cassandra (CASSANDRA-15013) has an elementary built-in memory rate limiter based on the incoming payload from user requests. There is another rate limiter targeted towards native transport(CASSANDRA-16663). These rate limiters activate if any incoming user request's payload exceeds certain thresholds or native transport exceeds certain limits. These existing rate limiters solve targeted issues.

Cassandra's server-side meltdown due to overload is a known problem. Often, a couple of busy nodes take down the entire Cassandra ring due to the ripple effect. This document proposes a unified-purpose comprehensive rate limiter that works considering system signals, such as CPU, and internal signals, such as thread pools. The rate limiter will have knobs to filter out internal traffic, system traffic, replication traffic, and furthermore based on the types of queries.

Design doc Apache Cassandra Unified Rate Limiter

New or Changed Public Interfaces

TODO

Compatibility, Deprecation, and Migration Plan

TODO

Test Plan

TODO

Rejected Alternatives

TODO