SIP-20: Separation of Compute and Storage in SolrCloud

Status

Current state: Under Discussion

Discussion thread: [DISCUSS] SIP-20: Separation of Compute and Storage in SolrCloud

JIRA: SOLR-17125 - SIP-20 ZERO replicas, separation of compute and storage

Released: -

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast). Confluence supports inline comments that can also be used.

Motivation

Provide a separation of compute (query/indexing) and storage for SolrCloud, with a single stored copy of each shard (on S3/GCS/etc) and support for stateless nodes. This makes running SolrCloud in a public cloud environment a lot more efficient (in addition to an Operator and other public cloud specific aspects).

Nodes do have a local copy (cache) on their local disk of the data as it is used.

This allows scaling compute (more queries, more indexing) independently of storage (more collections). Most useful for use cases in which indexes are used intermittently. Elasticity is also made simpler: shutting down a node is possible at any time without data loss (all acknowledged data has been stored remotely and remains accessible). Adding a node to a saturated cluster does not increase inter-node communication in the cluster or the load on already saturated nodes: indexes needed by new nodes are fetched from the shared storage.

Public Interfaces

A fourth replica type called ZERO is introduced (in addition to existing replica types NRT, TLOG and PULL). At Collection creation time, it is possible to specify that the collection exclusively uses replicas of type ZERO rather than being a "normal" collection that uses NRT/TLOG/PULL.

A shared repository called the “Zero store” storing the ZERO replicas data is configured in a similar way to backup repositories.

Proposed Changes

Initial code drop (in branch jira/solr-17125-zero-replicas, Jira SOLR-17125, documentation in jira/solr-17125-zero-replicas/solr/ZERO-REPLICAS.adoc) is the actual implementation of the ZERO replica type, of storage and access of a single shard copy in the Zero store, with adequate protections against overwrites and data loss (the multiple concurrent writers problem).

Such code has been running for almost 3 years at high scale in production (100+ clusters). The branch as visible is a cleaned-up (made more readable, other customizations removed, ported to main) version of that original code.

Changes planned but not yet implemented: transform the transaction log which is currently per replica and not used for ZERO replicas into a per shard transaction log compatible with ZERO replicas and stateless nodes, and have it also stored on remote storage, independent of any specific node. This allows achieving the goal of stateless nodes without forcing update batches to commit and write the new segments to the remote store before acknowledging to the client as is currently the case with the proposed implementation (lower efficiency and slower indexing).

Compatibility, Deprecation, and Migration Plan

Compatibility of the new code with existing clusters and data (on disk and in ZooKeeper) is total. There is no deprecation resulting from this change and given the compatibility no need for a migration plan.

A cluster running the new code with a configured Zero store can have new Collections created to use ZERO replicas. Existing collections not using ZERO replicas continue functioning normally without change and new “normal” collections can be created as well. A backup followed by a restore can allow transforming an existing collection into a ZERO collection.

Some SolrCloud features are not yet supported by ZERO replicas. This is not directly a compatibility issue but a limitation for migrating an existing collection to ZERO replicas if unsupported features are used. There is no fundamental reason preventing supporting all SolrCloud features on ZERO replicas collections, but more work is required.

Security considerations

The Zero store is implemented via reuse of existing BackupRepository implementations, so no added vulnerabilities related to the added Zero store. New bugs are of course likely given the amount of new code.
Test Plan

As mentioned, code implementing ZERO replicas is running in production at large scale. It is possible that regressions were introduced with the cleanup preceding this code contribution. Functional and longevity tests will have to be run (which I assume is needed for any major Solr version upgrade anyway).

Rejected Alternatives

HdfsDirectory allows some separation of compute and storage. Its fundamental difference with the current proposal is that it associates a “disk” per SolrCloud node, and therefore stores each replica multiple times on remote storage, as opposed to only once (storing the “shard”).

From an implementation perspective, it would have been awesome to not have to introduce a new replica type and instead plug the implementation at the Directory level. Although very tempting, the idea proved non realistic (main problem is when a former leader that “thinks” it is still leader and the current leader for a shard both try to update the shared storage concurrently, the implementation must guarantee no data can be lost or overwritten), or at least no concrete viable proposal was made.