

KIP-564: Add new cached authorizer:change the dim of cache

- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)

Status

Current state: Accepted

Discussion thread: <https://lists.apache.org/thread.html/r93be167479a782d1a02144ad00f4d2f9bb97a805a45918b29131d714%40%3Cdev.kafka.apache.org%3E>

JIRA:

 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

infra/browse/KAFKA-5261

hboards Projects Issues Search Log in

has been LDAP enabled, if you are an ASF Committer, please use your LDAP Credentials to login. Any problems email users@infra.apache.org

Details

Type: Improvement
Priority: Major
Affects Version/s: 0.10.2.1
Component/s: security
Labels: None

Status: Critical
Resolution: Unresolved
Fix Version/s: None

People

Assignee: Unassigned
Reporter: Stephane Maerek
Votes: Vote for this issue
Watchers: Start watching this issue

Dates

Created: 17/May/17 07:14
Updated: 19/Jul/18 18:22

Description

Currently, looking at the KafkaApis class, it seems that every request going through Kafka is also going through an authorize check:

```
private def authorize(session: Session, operation: Operation, resource: Resource): Boolean =  
  authorizer.forall(_authorize(session, operation, resource))
```


The SimpleAclAuthorizer logic runs through checks which all look to be done in linear time (except on first run) proportional to the number of acls on a specific resource. This operation is re-run every time a client tries to use a Kafka Api, especially on the very often called "handleProduceRequest" and "handleFetchRequest"


I believe a cache could be built to store the result of the authorize call, possibly allowing more expensive authorize() calls to happen, and reducing greatly the CPU usage in the long run. The cache would be invalidated every time a change happens to aclCache

Thoughts before I try giving it a go with a PR?

Issue Links

links to
GitHub Pull Request #3756

Like the jira:

 Unable to render Jira issues macro, execution error.

,We met the same performance issue in our production

environment,hence, we make a revision for the mechaism of authorization, our revision have such optimizations

- 1Build a cache for authorization, which can avoid recomputation of authorization result. The authorization result will fetch on the result catch if the same result has been computed rather than compute it again
- 2Differ from the pr [GitHub Pull Request #3756](#), when we build the result cache of the authorization, we take the resource into first consideration. In this way, the authorization is recomputed only when the authorization are change of specific resource. Compared to the the frequency of recomputation can be reduced obviously.

Public Interfaces

A public interface is any change to the following:

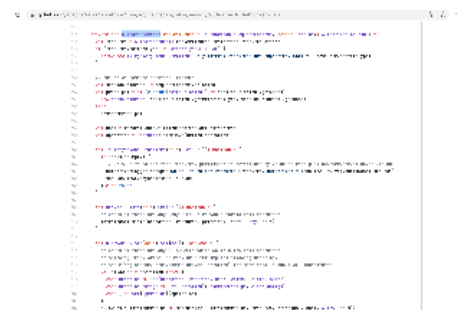
- update this old class
 - [core/src/main/scala/kafka/security/authorizer/AclAuthorizer.scala](#)
- add these new class
 - [core/src/main/scala/kafka/security/authorizer/CachedAuthorizer.scala](#)
 - [core/src/test/scala/unit/kafka/security/authorizer/CachedAuthorizerTest.scala](#)

Proposed Changes

PR Available here: <https://github.com/apache/kafka/pull/7661>

the method named authorizeAction in

The purpose of this method named authorizeAction in [AclAuthorizer.scala](#) is to calculate the AuthorizationResult under the current conditions. This method checks a lot of items and needs to traverse a lot of data: denyAcls / allowAcls / isSuperUser

A screenshot of a code editor showing the authorizeAction method in AclAuthorizer.scala. The code is in Scala and includes comments in Chinese. It defines the authorizeAction method which takes a KafkaPrincipal, a Resource, and an Acl as parameters and returns an AuthorizationResult. The method logic involves checking if the principal is a superuser, then checking if the resource is a cluster resource, and finally checking if the principal has the required permissions based on the ACLs. The code is wrapped in a try-catch block to handle exceptions.

Same calculations will only cause frequent gc and cpu rises,so we don't need to do the same calculations under the same conditions.

This optimization is to solve this problem. We cache each condition and result into the authorizerCache. The next time querying , we first get the authorizerCache. If there is, we can get the results. If not, we can use authorizeAction to calculate and add to authorizerCache

Compatibility, Deprecation, and Migration Plan

- No impact or migration plan required as this proposal is only adding new methods and not changing any current behaviour.

Rejected Alternatives

As mentioned in the Proposed changes, this is inefficient.