Report for assignment 4

Project

Name: Trino

URL: https://github.com/DD2480-Group26/trino (forked version)
https://github.com/trinodb/trino (original version)

Official repository of Trino, the distributed SQL query engine for big data, formerly known as PrestoSQL (https://trino.io)

Onboarding experience

Did you choose a new project or continue on the previous one?
For assignment 3 we worked on another project called json-iterator.

If you changed the project, how did your experience differ from before?
When working with the previous assignment a huge time was spent on setting up the project. Json-iterator was our first maven project and meanwhile there was no documentation on set up for developers. This in combination with everyone in our team having different computer setup (operative system and hardware) made the set up tedious and difficult. When working with our current project trino we experienced a much faster set up time. Much to thank the set up guide given in the repos README.

In general we found that the trino project provided more information and guidance for developers wanting to contribute. The trino project even included a page with development guidelines, a page on how to contribute and a Security Policy that included information about reporting security vulnerabilities. Even though our scope of contribution was relatively small we found the documentation informing and made us more secure in some code decisions. Overall we think that having this documentation for open-source helps keep the code base consistent.

Effort spent

For each team member, how much time was spent in

WHAT	Carl	Julia	Victor	Elisabeth	Hemen

1. plenary discussions/ meetings;	2h + 1.5h + 1h + 0.5h	4h 9min (3 meetings)	2h + 1.5h + 1h	4h 30min	2h + 1.5h + 1h (Meetings)
2. discussions within parts of the group;		2h 20min (discussing integration, peer reviews, writing on discord)	2h (pair programmin g)	3h (including pair programmin g)	1h (reviewing pull request)
3. reading documentati on;		3h 36min (Mostly spent reading about SpringBoot, Maven and project documentati on)	3h	2h 35min	2.5 h (reading the project documentati on and MAVEN)
4. configuration and setup;	3h building, each build fails in different stages. Added jUnit dependency to hive mvn project to solve analyze-only dep. error.	1:04 (I had some problems running the tests. Got help from the team and thus less time than the rest)	2h30h (I had to update mava version, upgrade ubuntu subsytem, dowlonoad docker and VS code, try to solve building issues…)	2h 30 min (including time to learn different tools used for setup and how to solve errors)	4h (building the project and got many errors to deal with locally, need to use IntelliJ IDEA and set it up as MAVEN project)
5. analyzing code/output;	1h (pull request code review)	4h 23min (Figuring out the code flow, server setup, solving broken tests)	40 min	50 min	2.5 h analyzing the test function
6. writing documentati on;	2h (UML)	40min	30 min	1h	30 min
7. writing code;		2h 41 min	10h (including different ways to do	12h 40min (including searching on internet)	9 h (writing test cases and solving broken tests)

			the timer that are not in the final code)	(excluding pair programmin g)	
8. running code?		1h 25 min	1h	30 min (running tests or setting up a server)	1 h (trying to run the server)

For setting up tools and libraries (step 4), enumerate all dependencies you took care of and where you spent your time, if that time exceeds 30 minutes.
A common problem for most group members was getting Exceptions (mostly MojoFailureException that indicate a problem within a plugin) thrown during build. Time to solve these exceptions by googling and experimenting with the code took time. For some group members, it also took time for installing or updating all the different tools that are used and get the tools to work in an OS that does not fulfill the requirement of this project (the requirement was using Linux or Mac, so Windows user also need to use a VM tool).

Overview of issue(s) and work done.

Title: Timeout mechanism for file listing
URL:https://github.com/trinodb/trino/issues/1723

Add a configurable timeout mechanism to `CachingDirectoryLister.list` to prevent large list operation from blocking other operations

Functionality:

1. Have a timeout mechanism, i.e. stop the code after a certain time has passed and continue the original operations.
2. Throw an Exception if the limited timeout time has passed.
    - Point 1-2 are created in the method `list` in class `CachingDirectoryLister`
3. The time used as timeout should be configurable
    - Created a filed variable in class `CachingDirectoryLister` as timeout variable and are configurable in class `HiveConfig` in method `setFileListingTimeout`.

Requirements for the new feature or requirements affected by functionality being refactored

The class `CachingDirectoryLister` should:

- have a timeout mechanism for listing files in the method `list`.

- throw a timeout exception if listing files takes more time than the set limit in the method `list`.
- have a timeout limit field with a default value.
    - The timeout limit field should have a setter.
    - The limit value should be updatable when the server is started.
- Have corresponding test cases to test this timeout mechanism.

Optional (point 3): trace tests to requirements.

Code changes

Patch

Complete patch link (it includes the git diff):
https://github.com/DD2480-Group26/trino/compare/oldMaster...DD2480-Group26:master

Optional (point 4): the patch is clean.

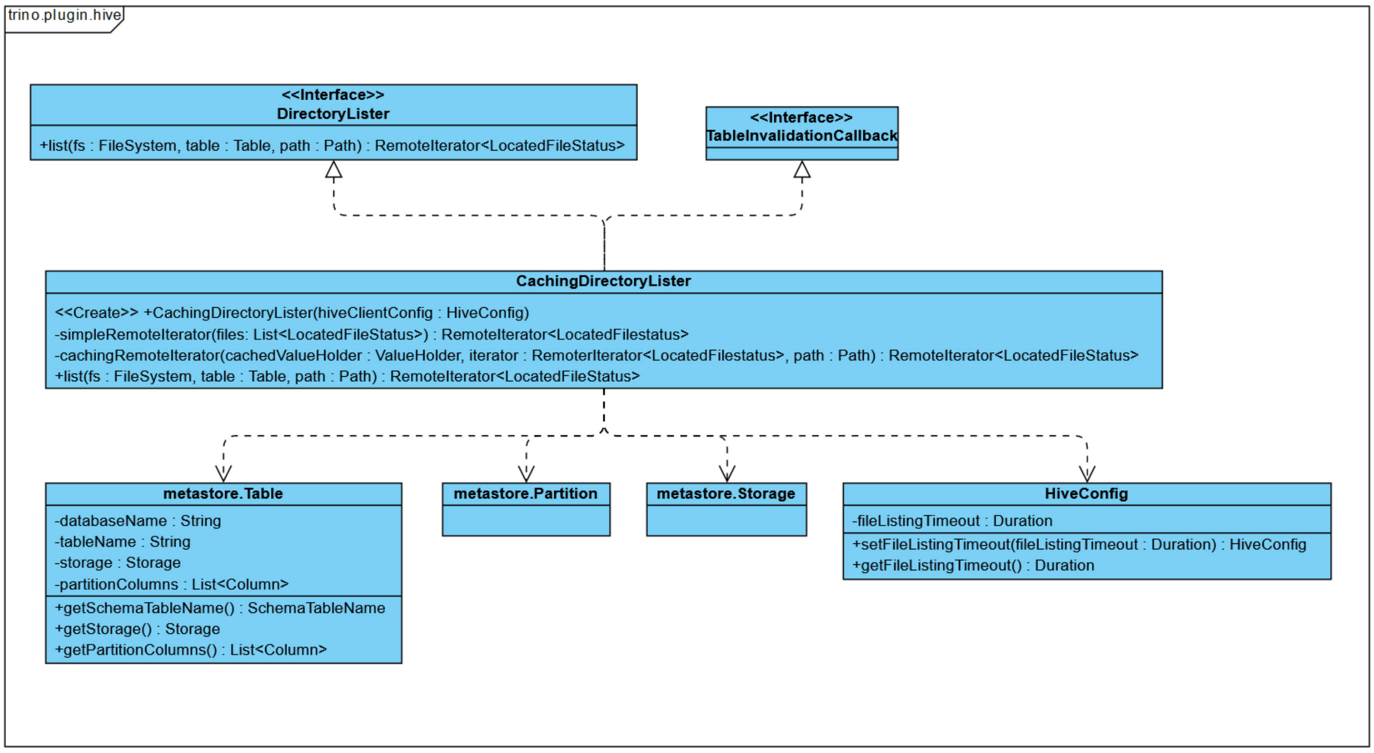
Optional (point 5): considered for acceptance (passes all automated checks).

Test results

Test result BEFORE patch

Test result AFTER patch (currently 2 new tests fails, but all original tests pass)

UML class diagram


trino.plugin.hive
<<Interface>>
DirectoryLister
+list(fs : FileSystem, table : Table, path : Path) : RemoteIterator<LocatedFileStatus>
<<Interface>>
TableInvalidationCallback
CachingDirectoryLister
<<Create>> +CachingDirectoryLister(hiveClientConfig : HiveConfig)
-simpleRemoteIterator(files: List<LocatedFileStatus>) : RemoteIterator<LocatedFilestatus>
-cachingRemoteIterator(cachedValueHolder : ValueHolder, iterator : RemoterIterator<LocatedFilestatus>, path : Path) : RemoteIterator<LocatedFileStatus>
+list(fs : FileSystem, table : Table, path : Path) : RemoteIterator<LocatedFileStatus>
metastore.Table
-databaseName : String
-tableName : String
-storage : Storage
-partitionColumns : List<Column>
+getSchemaTableName() : SchemaTableName
+getStorage() : Storage
+getPartitionColumns() : List<Column>
metastore.Partition
metastore.Storage
HiveConfig
-fileListingTimeout : Duration
+setFileListingTimeout(fileListingTimeout : Duration) : HiveConfig
+getFileListingTimeout() : Duration

UML Description
As indicated by the enveloping line, all classes are found in the trino.plugin.hive folder. CachingDirectoryLister contains the List function which the issue is mainly about, and thus is in focus. The CachingDirectoryLister implements two interfaces, DirectoryLister and TableInvalidationCheck. The List function is a requirement of the former interface, while the latter interface has no relevancy to List. The List function uses the two private methods simpleRemoteIterator and cachingRemoteIterator in CachingDirectoryLister, which are thus shown in the diagram.

List also makes use of function from the metastore.Table class (the class is called Table, but is found in the trino.plugin.hive.metastore folder), by calling some of its ‘get’ functions. The similar imports metastore.Partition and metastore.Storage are not used in List, but are shown in the diagram since they are some of the few project-related imports in the CachingDirectoryLister class.

Finally, a HiveConfig object is sent as an argument to one of the two constructors in CachingDirectoryLister, and this constructor is one part used to implement the issue feature, by making use of the new fileListingTimeout attribute in HiveConfig (the shown operations in HiveConfig are also new in our patch).

Key changes/classes affected

We had to create a timer for the function CachingDirectoryLister.list(). The best way to do it was to store the code of this function in another function CachingDirectoryLister.listHelper(). Then in the function CachingDirectoryLister.list() we create an ExecutorService and launch the helper in another thread. By doing this, all the call of the list() function remains up to date with our new version of the code. The new thread is launched with a timer that is defined by a specific field. This field has its own getter and setter and can be configurable. The configuration is achieved in the HiveConfig class. This Class contains server configuration that is set during server set up. We thus updated the HiveConfig to include a configuration called “hive.file-listing-timeout”. This means that the HiveConfig class got a field and setter for this value.

Overall experience

This project was very learning and helpful for us to improve our skills in diving into a huge repository and try to understand how the overall project has been built-in, reading the documentation part of such a project was the hard part and took a lot of time to set up the project according to the requirements. We learned how to refactor the code with many dependencies and how to overcome the difficulties and make the code better. Collaboration in this assignment was very helpful as well as needful because we all were working on the same issue, and we can say that it went very smoothly and the pair programming into the smaller groups was really effective. Given this we are in the Formed state.