Item 219
git_comments:

git_commits:

1. **summary:** for #675 desktop comment
   message: for #675 desktop comment

github_issues:

1. **title:** Using stream buffer to process COM_QUERY & STMT_EXECUTE response
   body: For MySQL protocol, Sharding-Proxy finish COM_QUERY Response's fetch all data, but GET_MORE_CLIENT_DATA still not implemented yet.
2. **title:** Using stream buffer to process COM_QUERY & STMT_EXECUTE response
   body: For MySQL protocol, Sharding-Proxy finish COM_QUERY Response's fetch all data, but GET_MORE_CLIENT_DATA still not implemented yet.
3. **title:** Using stream buffer to process COM_QUERY & STMT_EXECUTE response
   body: For MySQL protocol, Sharding-Proxy finish COM_QUERY Response's fetch all data, but GET_MORE_CLIENT_DATA still not implemented yet.
   label: code-design
4. **title:** Using stream buffer to process COM_QUERY & STMT_EXECUTE response
   body: For MySQL protocol, Sharding-Proxy finish COM_QUERY Response's fetch all data, but GET_MORE_CLIENT_DATA still not implemented yet.
5. **title:** Using stream buffer to process COM_QUERY & STMT_EXECUTE response
   body: For MySQL protocol, Sharding-Proxy finish COM_QUERY Response's fetch all data, but GET_MORE_CLIENT_DATA still not implemented yet.
   label: requirement
6. **title:** Using stream buffer to process COM_QUERY & STMT_EXECUTE response
   body: For MySQL protocol, Sharding-Proxy finish COM_QUERY Response's fetch all data, but GET_MORE_CLIENT_DATA still not implemented yet.

github_issues_comments:

1. After a short investigation, I found that GET_MORE_CLIENT_DATA is talking about to transfer a file to MySQL server. Did you intend to do this? As to get more data from MySQL, I tried 'setFetchDirection' and 'setFetchSize' method in JDBC, and found nothing differenct from packet captured by tcpdump. I doubt the MySQL server don't support this feature. I'll start a deeper study for it later.
2. I finally find out what JDBC do to avoid the client occure an OUT_OF_MEMORY exception in ResultSet section of https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-reference-implementation-notes.html. JDBC read one row at a time in client TCP buffer, and MySQL server will block in IO until client TCP buffer is not full. This mechanism don't depend on the MySQL protocol. We can avoid proxy from being OUT_OF_MEMORY just change the parameter of stme: stmt = conn.createStatement(java.sql.ResultSet.TYPE_FORWARD_ONLY, java.sql.ResultSet.CONCUR_READ_ONLY); stmt.setFetchSize(Integer.MIN_VALUE); Please note that this approach will lead to a lower performance, since each time the clint call next() the data will be copied from native to heap. I'll test the memory usage between these two different statments, and upload the test result.
3. **body:** This test base on my runnable code just completed, and shows the difference of memory usage between stream ResultSet and non-stream ResultSet. sharding-proxy are running on my PC with VM options: -server -Xmx1g -Xms1g -Xmn128m -Xss256k -XX:+DisableExplicitGC -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:LargePageSizeInBytes=128m -XX:+UseFastAccessorMethods -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=99 5 clients connect to a sharding-proxy, each of the clients select 150000 rows concurrently. non-stream ResultSet: ![resultset](https://user-images.githubusercontent.com/24643893/39034408-de27d700-44a8-11e8-9998-7b875fdef6b6.png) stream ResultSet: ![resultsetstreaming](https://user-images.githubusercontent.com/24643893/39034426-f2fa6d5a-44a8-11e8-9185-5b64e792ead8.png) Stream ResultSet proved to be more efficiency in space and time.
   label: code-design

4. fixed at 2.1.0

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

jira_issues_comments: