Doris load data设计

需求采集

```
LOAD DATA
    [LOW_PRIORITY | CONCURRENT] [LOCAL]
    INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var
        [, col_name_or_user_var] ...)]
    [SET col_name={expr | DEFAULT}
        [, col_name={expr | DEFAULT}] ...]
```

目前Doris只支持Broker方式的load data, 也支持通过StreamLoad的方式导入本地文件, 但是不支持直接通过MySQL客户端导入本地文件, 而有些用户使用mysql就已经有用到本地导入的功能, 因此希望Doris也能够支持通过客户端方式执行本地导入.

语法说明

MySQL :: MySQL 8.0 Reference Manual :: 13.2.7 LOAD DATA Statement

【转载】Mysql load data infile用法(万级数据导入，在几秒之内) - NewLife365 - 博客园

阿里云ADB(MySQL) LOAD DATA

TIDB 的 LOAD DATA

Oceanbase的Load DATA

网络报文

MySQL协议分析

MySQL: LOCAL INFILE Data协议信息

浅谈MySQL load data local infile细节 -- 从源码层面 - lispking - 博客园

实现设计

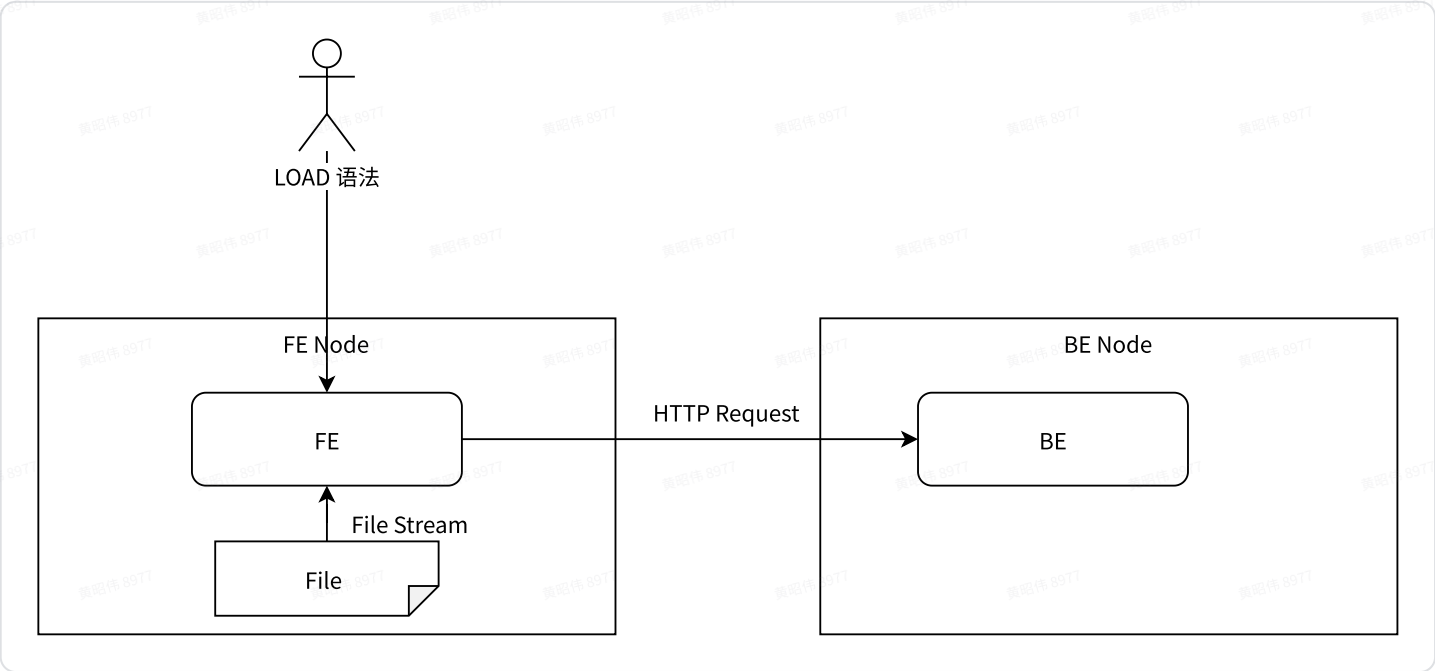
整体思路是复用先现有的StreamLoad导入逻辑, 将本地导入语法转化为后端的StreamLoad任务.

MySQL的LOAD DATA语法支持上传服务端数据和客户端数据, 这两个需要分开来讨论.

服务端文件

服务端的场景, 是数据文件已经上传到FE节点, 用户明确知道文件是在FE的本地节点上.

> 这种场景相对较少



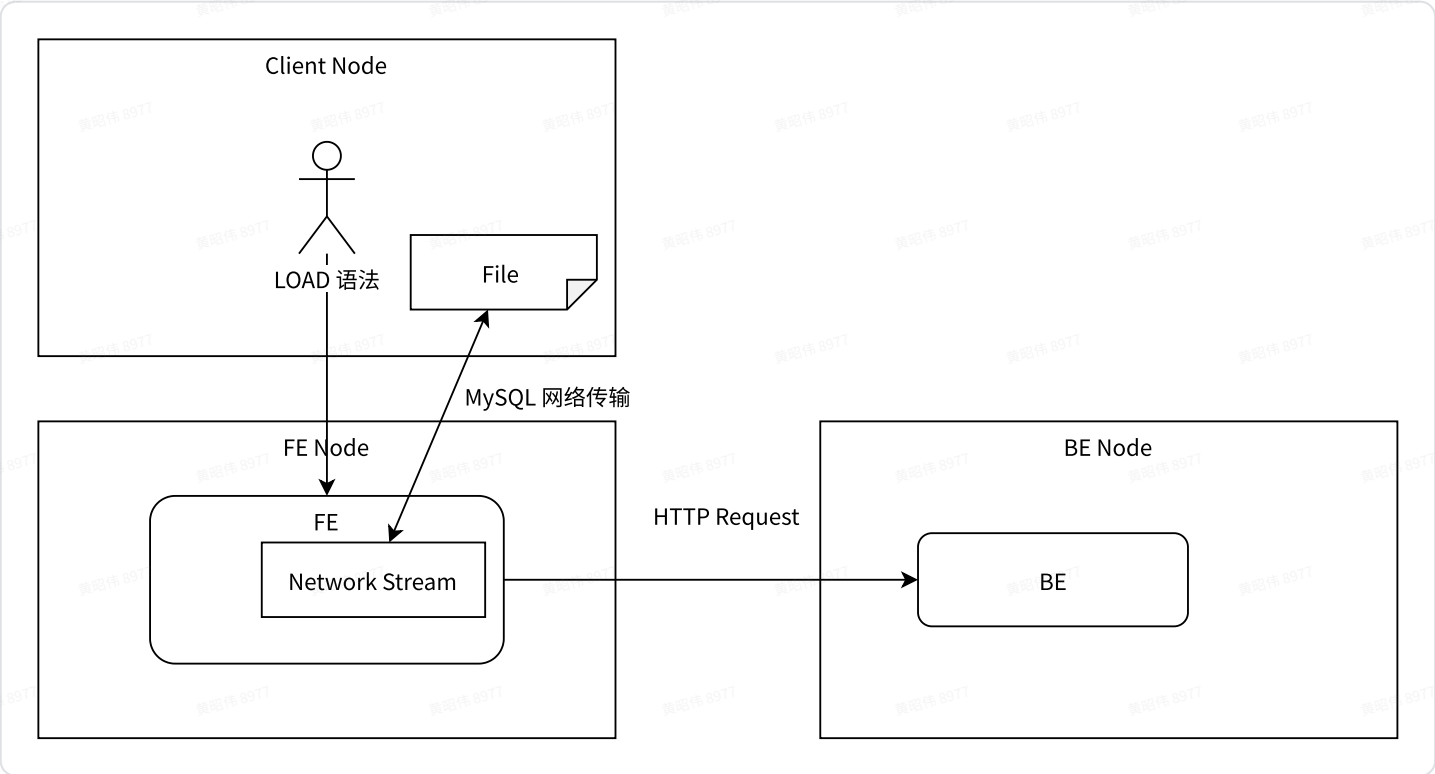
此时, 用户使用LOAD语法后, FE会构造StreamLoad的请求, 发送给BE, 然后会把StreamLoad的任务返回任务结果.

```
1 public static RequestBody createInputStreamRequestBody(final InputStream stream)
2     return new okhttp3.RequestBody() {
3         @Override
4         public MediaType contentType() {
5             // 设置body mime类型，这里以二进制流为例
6             return MediaType.get("application/octet-stream");
```

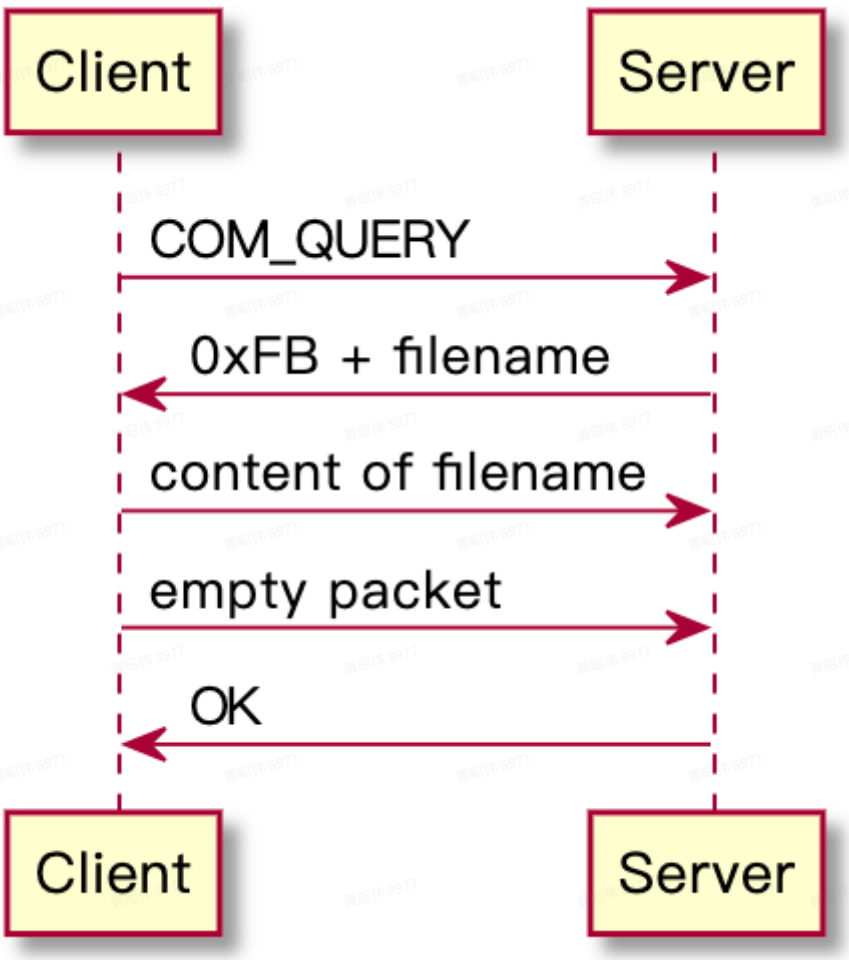
```
        }

        @Override
        public long contentLength() throws IOException {
            // 返回-1表示body长度未知，将开启http chunk支持
            // RequestBody中默认返回也时-1
            return -1;
        }

        @Override
        public void writeTo(BufferedSink sink) throws IOException {
            try (Source source = Okio.source(stream)) {
                sink.writeAll(source);
            }
        }
    };
}

public static void main(String[] args) throws IOException {
    OkHttpClient client = new OkHttpClient().newBuilder().build();
    byte[] bytes = "3\t2".getBytes();
    ByteArrayInputStream inputStream = new ByteArrayInputStream(bytes);
    RequestBody body = createInputStreamRequestBody(Files.newInputStream(Paths.g
    Request request = new Request.Builder()
            .url("http://doris:8040/api/hzw/t2/_stream_load")
            .method("PUT", body)
            .addHeader("Expect", "100-continue")
            .addHeader("Authorization", "Basic cm9vdDo=")
            .addHeader("Content-Type", "text/plain")
            .build();
    Response response = client.newCall(request).execute();
    System.out.println(response.body().string());
}
```

客户端文件

如果是客户端文件的话, 相对于服务端会更加复杂, 我们需要考虑MySQL协议的网络交互.



由于文件在客户端, 那么需要通过MySQL网络协议将数据从客户端传输到FE节点, 具体实现可以参考TiDB的实现.



FE->BE的实现, 需要实现一个InputStream, 数据由MySQL网络流写入, 使用与FileInputStream一致.

然后通过调用BE的StreamLoad接口实现, 数据的最终写入.

最后返回任务结果.

> 看MySQL的协议, 客户端导入应该是个同步请求, 就需要看看返回值是不是一样.

```
public static RequestBody createInputStreamRequestBody(final InputStream stream)
    return new okhttp3.RequestBody() {
        @Override
        public MediaType contentType() {
            // 设置body mime类型，这里以二进制流为例
            return MediaType.get("application/octet-stream");
        }

        @Override
        public long contentLength() throws IOException {
            // 返回-1表示body长度未知，将开启http chunk支持
            // RequestBody中默认返回也时-1
            return -1;
        }

        @Override
        public void writeTo(BufferedSink sink) throws IOException {
            try (Source source = Okio.source(stream)) {
                sink.writeAll(source);
            }
        }
    };
}

public static void main(String[] args) throws IOException {
    OkHttpClient client = new OkHttpClient().newBuilder().build();
    byte[] bytes = "3\t2".getBytes();
    ByteArrayInputStream inputStream = new ByteArrayInputStream(bytes);
    RequestBody body = createInputStreamRequestBody(inputStream);
    Request request = new Request.Builder()
            .url("http://doris:8040/api/hzw/t2/_stream_load")
            .method("PUT", body)
            .addHeader("Expect", "100-continue")
            .addHeader("Authorization", "Basic cm9vdDo=")
            .addHeader("Content-Type", "text/plain")
            .addHeader("label", "12344")
            .build();
    Response response = client.newCall(request).execute();
    System.out.println(response.body().string());
}
```

任务拆解

- 服务端文件的语法支持
- 客户端文件传输协议支持
- 支持IGNORE *number*{LINES | ROWS}语法
- 支持[REPLACE | IGNORE]语法