

## Doris 编译问题

我的机器配置信息是：

```
Model Name:           MacBook Air
Model Identifier:      MacBookAir10,1
Model Number:         Z127000ALZP/A
Chip:                 Apple M1
Total Number of Cores: 8 (4 performance and 4 efficiency)
Memory:               16 GB
```

我在编译后端时，一开始参考的是这个文档：<https://doris.apache.org/zh-CN/docs/dev/install/source-install/compilation-general#使用-docker-开发镜像编译推荐>。

但是会因为 OOM 出现一些错误 `fatal error: Killed signal terminated program ...`。于是我按照提示给 docker 分配了 12 GB 的内存，4GB 的 swap，和 6 个 CPU core。但是依然会发生同样的错误。另一个问题是，使用 docker image 编译的速度很慢，即使我开启了 parallel 选项，编译后端估计需要10个小时以上。

后来我参考这个文档 <https://doris.apache.org/zh-CN/docs/dev/install/source-install/compilation-mac>，尝试在我的 M1 Mac 上本地编译。这里我首先遇到的问题是，使用 brew 安装 openjdk@8 时，会出现 CPU 架构不匹配的错误。当我去掉版本号后，brew 默认安装的是 jdk 20 版本。由于 jdk 20 版本是一个 pre-release 版本，我尝试手动安装 jdk 17 LTS 版本。这里又会遇到一个问题：brew 安装 maven 时默认使用的是 jdk 20 版本，于是会发生版本冲突。因此，我只能绕过 brew，手动下载和安装 jdk 17，以及 maven 3.9.2 版本。

在安装 dependencies 后，我遇到了 `jni.h` 文件无法被找到的错误。我通过设置 `JAVA_HOME` 环境变量解决了这个错误。本地编译的速度明显比使用 docker image 编译快，在设置 `-j8` 的情况下，大概只需要十分钟左右。一个疑问是：为什么使用 docker image 编译的速度这么慢？

关于环境变量，我发现 `export JAVA_HOME` 的 shell 命令，无法在 shell 启动时自动执行。添加 maven bin 路径到 `PATH` 的命令也无法被自动执行。我怀疑是我自己的 `.bashrc` 文件设置的问题。

编译前端时没有出现错误，但是出现了一些 warnings。我暂时忽略了它们。

最后终端输出 `Successfully build doris`，我认为编译成功了。

## Doris 启动问题

我参考这个文档 <https://doris.apache.org/zh-CN/docs/dev/get-starting/>，尝试启动 Doris。

我一开始使用的启动脚本为 `doris/bin` 路径下的 `start_xx.sh` 脚本。在执行 `./start_be.sh --daemon` 时，没有抛出任何错误，但也没有任何日志输出，通过任务管理器，我发现后端并没有启动。

通过查找 `doris` 目录，我发现实际应该执行的脚本位于 `doris/output/be/bin` 路径下。这次我成功启动了后端节点，但是由于前端尚未启动，抛出了一些 warning 信息：

```

I0529 13:55:48.131352 2141892608 point_query_executor.h:260] prune lookup cache, total 0 expired items
W0529 13:55:48.680249 1844277248 local_file_system.cpp:296] [IO_ERROR]failed to list /proc/49402/task/: (2), No such file or directory
W0529 13:55:48.680534 1844277248 doris_metrics.cpp:367] failed to count thread num: [IO_ERROR]failed to list /proc/49402/task/: (2), No such file or directory
W0529 13:55:48.680608 1844277248 local_file_system.cpp:296] [IO_ERROR]failed to list /proc/49402/fd/: (2), No such file or directory
W0529 13:55:48.680649 1844277248 doris_metrics.cpp:389] failed to count fd: [IO_ERROR]failed to list /proc/49402/fd/: (2), No such file or directory
W0529 13:55:48.680724 1844277248 system_metrics.cpp:396] open /proc/stat failed, errno=2, message=0
W0529 13:55:48.680774 1844277248 system_metrics.cpp:956] open /proc/vmstat failed, errno=2, message=0
W0529 13:55:48.680824 1844277248 system_metrics.cpp:495] open /proc/diskstats failed, errno=2, message=0
W0529 13:55:48.680871 1844277248 system_metrics.cpp:584] open /proc/net/dev failed, errno=2, message=0
W0529 13:55:48.680922 1844277248 system_metrics.cpp:755] open /proc/sys/fs/file-nr failed, errno=2, message=0
W0529 13:55:48.680968 1844277248 system_metrics.cpp:680] open /proc/net/snmp failed, errno=2, message=0
W0529 13:55:48.681016 1844277248 system_metrics.cpp:796] open /proc/loadavg failed, errno=2, message=0
W0529 13:55:48.681068 1844277248 system_metrics.cpp:906] open /proc/stat failed, errno=2, message=0
I0529 13:55:48.938097 2196779008 task_worker_pool.cpp:1116] waiting to receive first heartbeat from frontend before doing task report
I0529 13:55:58.323984 2147053568 olap_server.cpp:762] cooldown producer get tablet num: 0
I0529 13:55:58.940166 2196779008 task_worker_pool.cpp:1116] waiting to receive first heartbeat from frontend before doing task report

```

我分析这是正常的行为。

当我尝试启动前端节点时，我首先遇到了 java jdk 版本的问题。前端启动脚本会使用一个配置文件，这个文件中定义了一些为 java jdk 9+ 版本服务的 options：`JAVA_OPTS_FOR_JDK_9="-Xmx8192m -XX:SurvivorRatio=8 -XX:MaxTenuringThreshold=7 -XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=80 -XX:SoftRefLRUPolicyMSPerMB=0 -Xlog:gc*:${DORIS_HOME}/log/fe.gc.log.$CUR_DATE:time"`

由于我使用的 java jdk 版本为 17，因此会使用如上选项启动 java。但是这些 java options 实际上已经在 java jdk 14+ 版本被遗弃了。于是我尝试将 `CMS` 开头的选项删除。删除这些选项后，前端节点可以启动，但是在启动过程中遇到了一个错误：

```

2023-05-29 11:25:50,881 ERROR (stateListener|76) [EditLog.logEdit():1068] Fatal Error : write stream Exception
java.lang.reflect.InaccessibleObjectException: Unable to make field private final java.util.concurrent.locks.ReentrantReadWriteLock$ReadLock
    at java.lang.reflect.AccessibleObject.checkCanSetAccessible(AccessibleObject.java:354) ~[?:?]
    at java.lang.reflect.AccessibleObject.checkCanSetAccessible(AccessibleObject.java:297) ~[?:?]
    at java.lang.reflect.Field.checkCanSetAccessible(Field.java:178) ~[?:?]
    at java.lang.reflect.Field.setAccessible(Field.java:172) ~[?:?]
    at com.google.gson.internal.reflect.UnsafeReflectionAccessImpl.makeAccessible(UnsafeReflectionAccessImpl.java:44) ~[gson-2.8.9.jar:?]
    at com.google.gson.internal.bind.ReflectiveTypeAdapterFactory.getBoundFields(ReflectiveTypeAdapterFactory.java:159) ~[gson-2.8.9.jar:?]
    at com.google.gson.internal.bind.ReflectiveTypeAdapterFactory.create(ReflectiveTypeAdapterFactory.java:102) ~[gson-2.8.9.jar:?]
    at com.google.gson.Gson.getDelegateAdapter(Gson.java:572) ~[gson-2.8.9.jar:?]
    at org.apache.doris.persist.gson.GsonUtils$PostProcessTypeAdapterFactory.create(GsonUtils.java:477) ~[doris-fe.jar:1.2-SNAPSHOT]
    at com.google.gson.Gson.getAdapter(Gson.java:489) ~[gson-2.8.9.jar:?]
    at com.google.gson.internal.bind.ReflectiveTypeAdapterFactory.createBoundField(ReflectiveTypeAdapterFactory.java:117) ~[gson-2.8.9.jar:?]
    at com.google.gson.internal.bind.ReflectiveTypeAdapterFactory.getBoundFields(ReflectiveTypeAdapterFactory.java:166) ~[gson-2.8.9.jar:?]
    at com.google.gson.internal.bind.ReflectiveTypeAdapterFactory.create(ReflectiveTypeAdapterFactory.java:102) ~[gson-2.8.9.jar:?]
    at com.google.gson.Gson.getDelegateAdapter(Gson.java:572) ~[gson-2.8.9.jar:?]
    at org.apache.doris.persist.gson.GsonUtils$PostProcessTypeAdapterFactory.create(GsonUtils.java:477) ~[doris-fe.jar:1.2-SNAPSHOT]
    at com.google.gson.Gson.getDelegateAdapter(Gson.java:572) ~[gson-2.8.9.jar:?]
    at org.apache.doris.persist.gson.RuntimeTypeAdapterFactory.create(RuntimeTypeAdapterFactory.java:262) ~[doris-fe.jar:1.2-SNAPSHOT]
    at com.google.gson.Gson.getAdapter(Gson.java:489) ~[gson-2.8.9.jar:?]
    at com.google.gson.Gson.toJson(Gson.java:727) ~[gson-2.8.9.jar:?]
    at com.google.gson.Gson.toJson(Gson.java:714) ~[gson-2.8.9.jar:?]
    at com.google.gson.Gson.toJson(Gson.java:669) ~[gson-2.8.9.jar:?]
    at com.google.gson.Gson.toJson(Gson.java:649) ~[gson-2.8.9.jar:?]
    at org.apache.doris.policy.Policy.write(Policy.java:139) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.journal.JournalEntity.write(JournalEntity.java:158) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.journal.bdbje.BDBJEJournal.write(BDBJEJournal.java:137) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.persist.EditLog.logEdit(EditLog.java:1064) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.persist.EditLog.logCreatePolicy(EditLog.java:1623) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.policy.PolicyMgr.createDefaultStoragePolicy(PolicyMgr.java:115) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.catalog.Env.transferToMaster(Env.java:1345) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.catalog.Env.access$1200(Env.java:278) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.catalog.Env$4.runOneCycle(Env.java:2402) ~[doris-fe.jar:1.2-SNAPSHOT]
    at org.apache.doris.common.util.Daemon.run(Daemon.java:116) ~[doris-fe.jar:1.2-SNAPSHOT]

```

由于我没有 java 编程的经验，我目前并没有解决这个错误。

我怀疑是 java 版本的问题。

## Doris 使用问题

根据文档 <https://doris.apache.org/zh-CN/docs/dev/get-starting/>，我尝试使用 MySQL 连接 Doris 前端。

这里遇到的问题是，文档中所提供的 MySQL 链接对应的是 linux 平台下 x86 架构的 MySQL binary 文件。但我使用的 arm 架构的 M1 Mac，于是并不能使用这个 binary 文件。

由于我没有 MySQL 的使用经验，在浏览了 MySQL 官网后，我准备使用 MySQL Shell。

由于我尚未成功启动 Doris 前端节点，我暂时搁置了这个问题。

## 任务要求

为 Doris SQL 解析和执行添加对 `array_contains_all` 函数的支持。

在 clickhouse 中，有一个对等的函数 `hasAll`，它的定义和用例如下：

### hasAll

Checks whether one array is a subset of another.

```
hasAll(set, subset)
```

#### Arguments

- **set** – Array of any type with a set of elements.
- **subset** – Array of any type with elements that should be tested to be a subset of **set**.

#### Return values

- **1**, if **set** contains all of the elements from **subset**.
- **0**, otherwise.

#### Peculiar properties

- An empty array is a subset of any array.
- **Null** processed as a value.
- Order of values in both of arrays does not matter.

## Examples

`SELECT hasAll([], [])` returns 1.

`SELECT hasAll([1, Null], [Null])` returns 1.

`SELECT hasAll([1.0, 2, 3, 4], [1, 3])` returns 1.

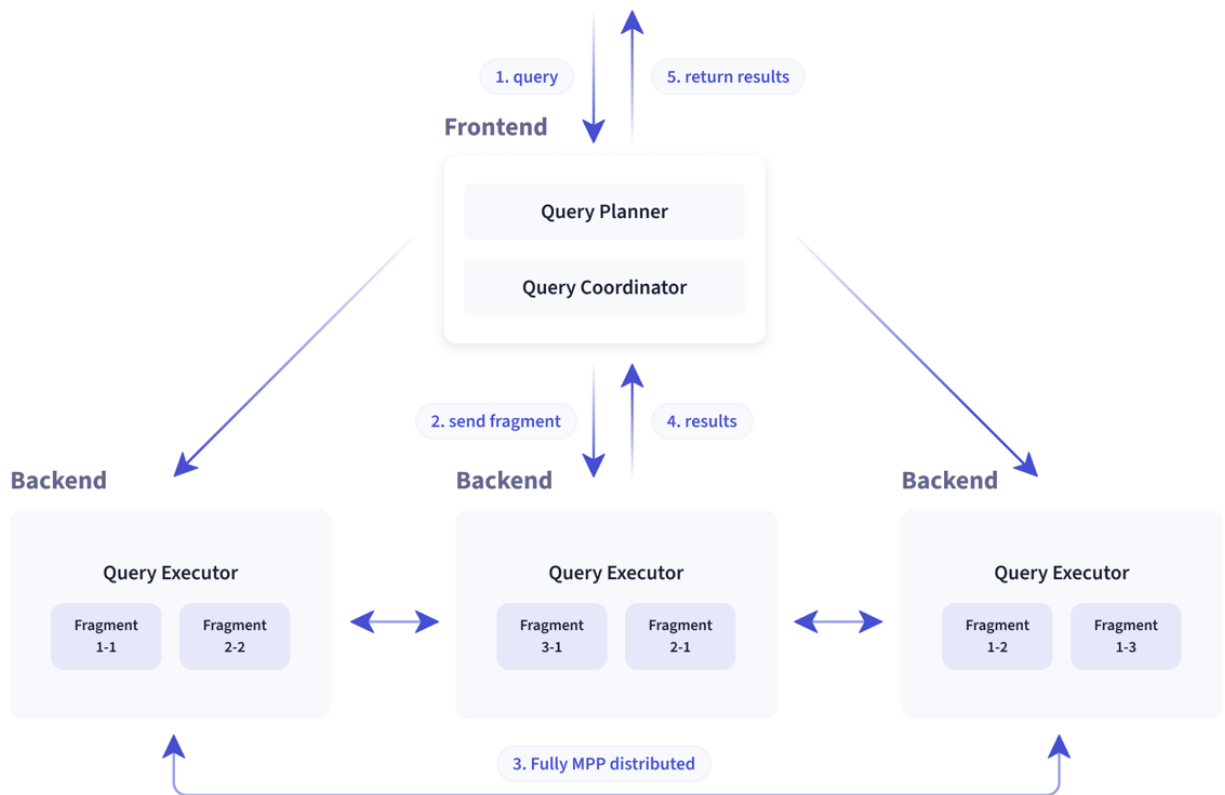
`SELECT hasAll(['a', 'b'], ['a'])` returns 1.

`SELECT hasAll([1], ['a'])` returns 0.

`SELECT hasAll([[1, 2], [3, 4]], [[1, 2], [3, 5]])` returns 0.

通过 `hasAll` 函数的用例，我基本了解了所需实现的 `array_contains_all` 函数的语义。

## Doris 架构



client 发一条 SQL 给 Doris 前端。前端对 SQL 进行解析，再生成执行计划。前端将这个执行计划拆分成多个 plan fragment，以 fragment 为单位发送给 Doris 后端。Doris 后端并发地执行 plan fragment，再将执行结果返回给前端。前端经过一定处理后，交付给 client。

在查找 Doris 架构时，我搜集到了如下我认为比较有用的文档或博客：

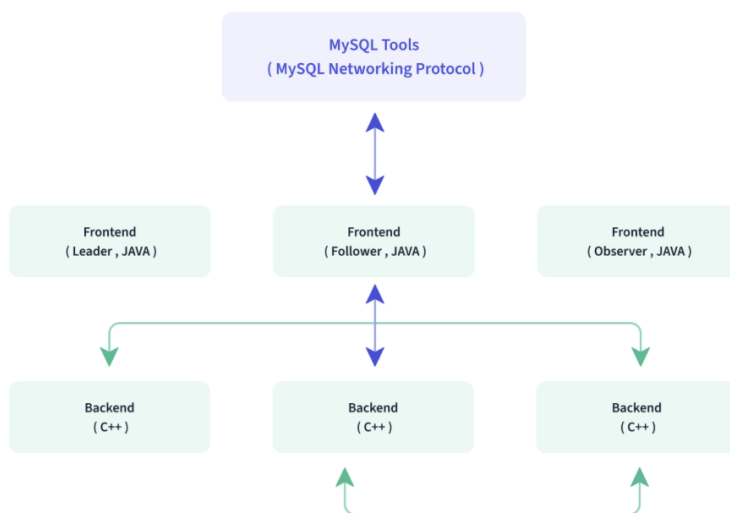
关于 Doris 架构

## Technical Overview

As shown in the figure below, the Apache Doris architecture is simple and neat, with only two types of processes.

- Frontend (FE): user request access, query parsing and planning, metadata management, node management, etc.
- Backend (BE): data storage and query plan execution

Both types of processes are horizontally scalable, and a single cluster can support up to hundreds of machines and tens of petabytes of storage capacity. And these two types of processes guarantee high availability of services and high reliability of data through consistency protocols. This highly integrated architecture design greatly reduces the operation and maintenance cost of a distributed system.



关于 Doris 向量化

### 活动回顾 | Apache Doris 向量化技术实现与后续规划

本文主要与大家分享社区期待已久的 Doris 向量化技术，包括 Doris 向量化的设计与实现、当前开发情况以及后续规划。


<https://mp.weixin.qq.com/s/0TVjp-TiPsk1EQWQ3d4C4A>

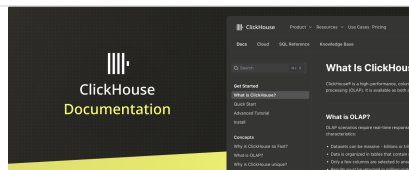


关于 clickhouse 架构

ClickHouse 架构概述 | ClickHouse Docs

clickhouse-jia-gou-gai-shu}

 <https://clickhouse.com/docs/zh/development/architecture#block>



## Doris 前端如何解析查询语句中的内置函数

这是我首次接触 Apache Doris，因此首要的任务显然是弄清楚 Doris 是如何解析和执行 SQL 的。特别的，需要重点弄清楚 Doris 是如何解析和执行一个 built-in 函数的。我搜集到了如下我认为很有用的文档或博客：

关于 Doris 文件组织

新手指南 | 如何从零开始参与 Apache 顶级开源项目？（二）

作者：苏奕嘉 | SelectDB 生态研发工程师写在开头上一篇文章中 <https://zhuanlan.zhihu.com/p/559460373>我们介绍了 Apache Doris 社区的工作机制、如何参与社区贡献以及如何完成第一个 PR，更多是从大而全的角度来...

 <https://zhuanlan.zhihu.com/p/564386656>



关于 Doris SQL 解析

Doris全面解析：Doris SQL 原理解析 - Apache Doris

<!--

 <https://doris.apache.org/zh-CN/blog/principle-of-Doris-SQL-parsing/>

浏览了这些博客后，我进一步询问 ChatGPT，要求其从源码角度给我分析 Doris 是如何解析 SQL 的。从它的回答中得知，前端的 `fe/fe-core/src/main` 文件夹下的 `antlr4`，`cup`，`jflex` 文件夹与 SQL 解析相关。在简单 Google 后，我得知 `antlr4` 是一个集词法分析、语法分析等功能的生成器，`jflex` 是一个词法解析器，`cup` 是一个语法解析器。

由于我没有编译原理、编译器相关的基础，我并不知道它们是怎么协同工作的。但是我之前参加过 Oceanbase minioB 比赛，在其中接触过基于 flex 和 yacc 的词法、语法解析。于是，我可以读懂 `antlr4` 文件夹下的 `DorisLexer.g4` 和 `DorisParser.g4` 文件。

下面以一条 select 语句为例，简单介绍一个内置函数是如何被解析的：

```
selectClause
: SELECT selectHint? DISTINCT? selectColumnClause
;
```

```
selectColumnClause
: namedExpressionSeq
| ASTERISK EXCEPT LEFT_PAREN namedExpressionSeq RIGHT_PAREN
;
```

一条 select 语句的组成为：SELECT 关键字、selectHint、DISTINCT 关键字、selectColumnClause。其中 selectHint 为指导查询优化的 hint，DISTINCT 表示是否对查询结果去重，selectColumnClause 表示待查询的列或表达式。

```

primaryExpression
: name=(TIMESTAMPDIFF | DATEDIFF)
    LEFT_PAREN
        unit=datetimeUnit COMMA
        startTimestamp=valueExpression COMMA
        endTimestamp=valueExpression
    RIGHT_PAREN
#timestampdiff
| name=(TIMESTAMPADD | DATEADD)
    LEFT_PAREN
        unit=datetimeUnit COMMA
        startTimestamp=valueExpression COMMA
        endTimestamp=valueExpression
    RIGHT_PAREN
#timestampadd
| name =(ADDDATE | DAYS_ADD | DATE_ADD)
    LEFT_PAREN
        timestamp=valueExpression COMMA
        (INTERVAL unitsAmount=valueExpression unit=datetimeUnit
        | unitsAmount=valueExpression)
    RIGHT_PAREN
#date_add
| name=(SUBDATE | DAYS_SUB | DATE_SUB)
    LEFT_PAREN
        timestamp=valueExpression COMMA
        (INTERVAL unitsAmount=valueExpression unit=datetimeUnit
        | unitsAmount=valueExpression)
    RIGHT_PAREN
#date_sub
| CASE whenClause+ (ELSE elseExpression=expression)? END
| CASE value=expression whenClause+ (ELSE elseExpression=expression)? END
#searchedCase
#simpleCase
| name=CAST LEFT_PAREN expression AS dataType RIGHT_PAREN
#cast
| constant
#constantDefault
| ASTERISK
#star
| qualifiedName DOT ASTERISK
#star
| functionIdentifier LEFT_PAREN ((DISTINCT|ALL)? arguments+=expression
(COMMA arguments+=expression)* (ORDER BY sortItem (COMMA sortItem)*)?)? RIGHT_PAREN
(OVER windowSpec)?
#functionCall
| LEFT_PAREN query RIGHT_PAREN
#subqueryExpression
| ATSIGN identifier
#userVariable
| DOUBLEATSIGN (kind=(GLOBAL | SESSION) DOT)? identifier
#systemVariable
| identifier
#columnReference
| base=primaryExpression DOT fieldName=identifier
#dereference
| LEFT_PAREN expression RIGHT_PAREN
#parenthesizedExpression
| EXTRACT LEFT_PAREN field=identifier FROM (DATE | TIMESTAMP)?
source=valueExpression RIGHT_PAREN
#extract
;

```

primaryExpression 表示表达式的主体部分，即什么样的 token 组合可以被解析为一条有效的表达式树。其中一种有效的组合为函数调用：

```

functionIdentifier LEFT_PAREN ((DISTINCT|ALL)? arguments+=expression
(COMMA arguments+=expression)* (ORDER BY sortItem (COMMA sortItem)*)?)? RIGHT_PAREN
(OVER windowSpec)?

```

上面这个语法表示一个有效的函数调用可以为：`functionIdentifier(arg1, arg2, ...)`。

至于 functionIdentifier，只要不是保留字，都是合法的。如果查询到这个调用与某个内置函数的签名（函数名、参数个数、参数类型）匹配，则知道这是 function 是一个内置函数。

由于我缺乏 Java 编程和 SQL 优化和处理的经验，因此前端的相关代码就暂时搁置。

## Doris 后端如何执行 Plan Fragment 中的函数

后端的 `service/backend_service.h` 文件中定义了后端提供的一些服务接口，其中一个为 `exec_plan_fragment`。以这个函数为 entry point，我们可以比较简单地 trace 函数调用栈。大致如下：

- backend service 的 `exec_plan_fragment` 接口被前端所调用。
- service 利用 exec env 创建一个 fragment manager，由 manager 管理 plan fragment 的执行。
- manager 创建一个执行任务，交付给线程池异步执行。

- 一个 plan fragment 由一个 `PlanFragmentExecutor` 执行。执行时会构建一个算子树，树中的节点为 `ExecNode`，每个 node 有不定数量的 children。我没有细究，不过我猜 Doris 和 CMU 15445 一样使用的是火山执行模型，因此每个节点被抽象为一个迭代器。在阅读这一块的代码时，我发现这里的文档和函数似乎有点混乱，行存、列存相关的代码、文档并没有很好地区分开。
- 由于 Doris 实现了 batching 和 vectorization，因此每个 exec node 会以 block 为单位进行数据的处理。exec node 中设置了数据处理时所依据的表达式。
- 表达式由 `VExpr` 表示，每个表达式被拆解成一颗树，因此每个 `VExpr` 都会有不定数量的子表达式。`VExpr` 是一个抽象类，通过继承它以定义表达式的具体行为。`VectorizedFnCall` 是 `VExpr` 的一个实现，用于处理函数表达式。
- 每个函数表达式有一个 `_function` 成员，其类型为指向 `IFunctionBase` 类的指针。在实现函数时，我们会定义一个函数类，这个类会继承 `IFunctionBase` 类，因此通过 `_function` 成员就可以调用具体的函数实现。

## 代码实现

根据 mentor 的建议，我首先在 Github issue 和 PR 中搜索其它 array 函数。基于我所搜索到的内容，我有如下一些实现想法：

### 使用集合操作

Doris 目前已经支持了集合的并、交、差操作。于是一个很自然的想法为：对于输入集合 A、B，为判断 A 是否完全包含 B，可以先做集合的交，得到  $A \cap B = C$ 。再做集合的差，得到  $B - C = D$ 。如果 D 为空集，则 A 完全包含 B 命题为真，否则为假。

但是通过阅读相关源码后，我发现目前的 array set 相关操作要求输入的两个集合包含相同类型的元素（NULL 也可以），这显然与 clickhouse `hasAll` 函数的语义不符。虽然如此，这样的实现可以完成一部分功能。

关于 `array_set`，`array_union`，`array_except`，`array_intersect`

<https://github.com/apache/doris/commit/3744321f01436c68099ec1db28da9777c8f913da>

### 参考 clickhouse `hasAll` 函数的源码

`hasAll` 函数的源码：

ClickHouse/hasAllAny.h at master · ClickHouse/ClickHouse

ClickHouse® is a free analytics DBMS for big data. Contribute to ClickHouse/ClickHouse development by creating an account on GitHub.

<https://github.com/ClickHouse/ClickHouse/blob/master/src/Functions/array/hasAllAny.h>



在浏览相关源码后，我发现 clickhouse 的代码框架与 Doris 有不小的区别。于是我暂时搁置了这种实现思路。

### 参考 Doris 其它 array 函数的实现

关于 `array_contains`


<https://github.com/apache/doris/pull/14564>




关于 `array_contains_any`

[Feature](array-function) add support for array\_contains\_any by david990917 · Pull Request #18931 · apache/doris

Proposed changes Issue Number: #17361 Problem summary Describe your changes. Checklist(Required) Does it affect the original behavior Has unit tests been added Has document been added or modifi...


 <https://github.com/apache/doris/pull/18931/files#diff-1b13fe18507780e1e19df79df1322a0308794b9794790eb9af996ab95a9bccac>



关于 `array_concat`


[Feature](array-function) Support array\_concat function (#17436) · yagagagaga/doris@2a14910


Apache Doris is an easy-to-use, high performance and unified analytics database. - [Feature](array-function) Support array\_concat function (#17436) · yagagagaga/doris@2a14910


 <https://github.com/yagagagaga/doris/commit/2a14910c8ab2709489b5ba91f67ed5393142da51?diff=split>

yagagagaga/doris

**[Feature](array-function) Support array\_concat function...**

381 lines changed +278 -3 


 gitcoi committed March 8, 2023 -> 2a14910

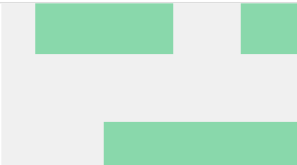


关于 `array_pushback`

[Feature](array-function) Support array\_pushback function #17417 by liangjiawei1110 · Pull Request #19988 · apache/doris

Proposed changes Issue Number: close #17417 Problem summary Implement array\_pushback. mysql&gt; select array\_pushback([1, 2], 3); +-----+ | array\_pushback(ARRAY(1, 2))...

 <https://github.com/apache/doris/pull/19988/files#diff-ff2bb2442674b98cc2559741dcc340d88f1262073cbaa19f277b8e0d3de98c71>



通过分析以上这些函数所修改的文件，容易得知我所需要修改的文件，以及大概是如何修改的。

## 我的实现思路

我一开始计划将 `array_contains_all` 函数实现为如下行为：

- 首先判断输入的集合 A, B 是否为 array 类型。如果任意一个不是 array，抛出错误。
- 判断集合 B 是否为空集，如果是，立即返回 true。
- 判断集合 A 的元素数量是否大于等于 B 的元素数量，如果否，立即返回 false。
  - 这个策略实际并不可用，因为根据 clickhouse 的测试用例 `SELECT hasAll([NULL], [NULL, NULL])`，多个 NULL 会被归约到一个 NULL，因此在计算元素数量时，需要将 NULL 单独处理。
- 判断集合 A 的元素类型所组成的集合是否是集合 B 的元素类型所组成的集合的超集。如果否，立即返回 false。
  - 这个策略实际并不可行，因为根据 clickhouse 的测试用例 `SELECT hasAll([1.0, 2, 3, 4], [1, 3]) returns 1`。来看，这里应该允许类型转换。通过查阅 Doris 关于 data type 相关的源码，我发现 Doris 通过 super type 来间接判断是否允许类型转换。
  - 进一步，我发现 Doris 在 `vec/data_type` 文件夹下提供了很多关于 data type 的类和函数，其中很多都对实现 `array_contains_all` 函数非常有用。
- 对于嵌套数组，即多维数组的处理，我使用 clickhouse 测试了几个用例：
  - `SELECT hasAll([[NULL]], [NULL])`：这个测试用例会报错，因为输入集合类型不匹配的问题。
  - `SELECT hasAll([[]], [[]])`：返回 true。
  - `SELECT hasAll([[]], [[], []])`：返回 true。不理解。
  - `SELECT hasAll([[NULL]], [[NULL]])`：返回 true。

- `SELECT hasAll([[NULL]], [[NULL, NULL]])`：返回 false。不理解。
- `SELECT hasAll([[1.0]], [[1]])`：返回 true。
- 遍历集合 B 中的每个元素，尝试在集合 A 中找到同样的元素（类型、值都相同）。如果没找到，立即返回 false。如果找到了，则在一个 mask 中将集合 A 中这个元素标记为已匹配。如果集合 B 中的所有元素都能被找到，那么返回 true。

但是通过在 clickhouse 上进行测试，我发现我对 `array_contains_all` 函数语义的理解与 clickhouse 有很大出入。因此我最终决定参考 clickhouse `sliceHasImplAnyAllGenericImpl` 函数的实现：

```
template <
    ArraySearchType search_type,
    typename FirstSliceType,
    typename SecondSliceType,
    bool (*isEqual)(const FirstSliceType &, const SecondSliceType &, size_t, size_t)>
bool sliceHasImplAnyAllGenericImpl(const FirstSliceType & first, const SecondSliceType & second, const UInt8 * first_null_map, const UInt8
{
    const bool has_first_null_map = first_null_map != nullptr;
    const bool has_second_null_map = second_null_map != nullptr;

    const bool has_second_null = hasNull(second_null_map, second.size);
    if (has_second_null)
    {
        const bool has_first_null = hasNull(first_null_map, first.size);

        if (has_first_null && search_type == ArraySearchType::Any)
            return true;

        if (!has_first_null && search_type == ArraySearchType::All)
            return false;
    }

    for (size_t i = 0; i < second.size; ++i)
    {
        if (has_second_null_map && second_null_map[i])
            continue;

        bool has = false;

        for (size_t j = 0; j < first.size && !has; ++j)
        {
            if (has_first_null_map && first_null_map[j])
                continue;

            if (isEqual(first, second, j, i))
            {
                has = true;
                break;
            }
        }

        if (has && search_type == ArraySearchType::Any)
            return true;

        if (!has && search_type == ArraySearchType::All)
            return false;
    }

    return search_type == ArraySearchType::All;
}
```

附上 ChatGPT 对该函数的解释：

这段代码看起来是一个通用的函数实现，用于在两个切片（slice）中搜索元素。函数根据传入的参数和搜索类型进行搜索，并返回相应的结果。下面是对该代码的解释和说明：

- `search_type` 是一个枚举类型，表示搜索的类型，可能是 `ArraySearchType::Any`（找到任意一个匹配）或 `ArraySearchType::All`（找到所有匹配）。

- `FirstSliceType` 和 `SecondSliceType` 是模板参数，分别表示第一个和第二个切片的类型。
- `isEqual` 是一个函数指针，指向用于比较两个切片元素是否相等的函数。该函数接受两个切片和两个索引参数，返回一个布尔值表示比较结果。

该函数的实现逻辑如下：

1. 首先，函数检查是否存在第一个和第二个切片的空值映射（null map）。如果存在第二个切片的空值映射，进入下一步处理。
2. 如果存在第二个切片的空值映射，函数检查第一个切片的空值映射是否存在。
  - 如果第一个切片也存在空值映射，并且搜索类型为 `ArraySearchType::Any`，则直接返回 `true`。
  - 如果第一个切片不存在空值映射，并且搜索类型为 `ArraySearchType::All`，则直接返回 `false`。
3. 接下来，函数使用两个嵌套的循环遍历第二个切片中的元素。
  - 如果存在第二个切片的空值映射，并且当前元素为 `null`，则跳过该元素。
  - 对于每个第二个切片的元素，函数在第一个切片中查找匹配的元素。
    - 如果存在第一个切片的空值映射，并且当前元素为 `null`，则跳过该元素。
    - 使用 `isEqual` 函数比较第一个切片中的元素与第二个切片中的元素。
      - 如果两个元素相等，则将标志位 `has` 设置为 `true`，并跳出内部循环。
  - 根据搜索类型判断是否找到匹配。
    - 如果 `has` 为 `true`，并且搜索类型为 `ArraySearchType::Any`，则直接返回 `true`。
    - 如果 `has` 为 `false`，并且搜索类型为 `ArraySearchType::All`，则直接返回 `false`。
4. 如果搜索类型为 `ArraySearchType::All`，表示需要找到所有匹配，此时返回 `true`。
  - 注意这里的返回是因为遍历完第二个切片，没有找到任何不匹配的元素。

## 关于测试

因为有 clickhouse `hasAll` 函数供参考，因此我根据 <https://clickhouse.com/docs/en/install#quick-install> 文档，下载了一个本地版本的 clickhouse。将我能想到的一些测试用例，输入到 clickhouse 中，记录 clickhouse 的行为。以此来指导我在 Doris 中的实现。

## TODO

- 成功编译 Doris 前端
- 参考其他 array function 和 Doris 关于 block, column, field, data type 等相关的源码，实现与 clickhouse `hasAll` 相同语义的 `array_contains_all` 函数。
- 写测试。
- 写文档。