Hive反模式的检测与修复

软件架构文档

版本 2.0

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 25/10/2020 | 1.0 | 第一次编写 | 原帅，崔楠 |
| 28/10/2020 | 2.0 | 第一次修改 | 崔楠，原帅 |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 5

3.1 概述 5

3.2 在构架方面具有重要意义的设计包 5

4. 部署视图 6

5. 技术视图 6

6. 质量属性的设计 7

软件架构文档

# 简介

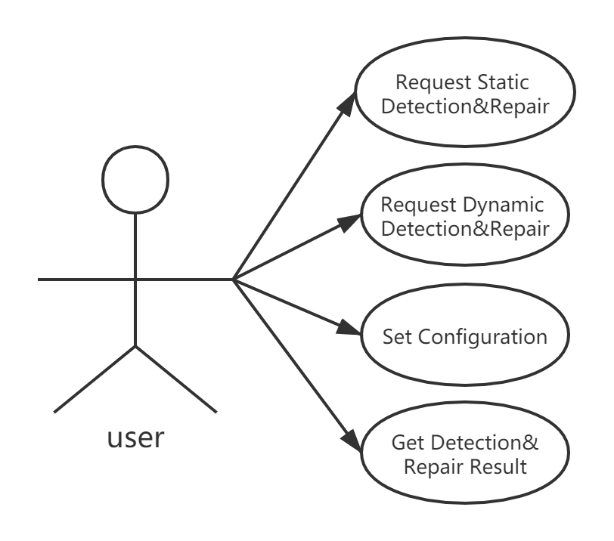
## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

## 参考资料

《Hive反模式的检测与修复立项建议书》

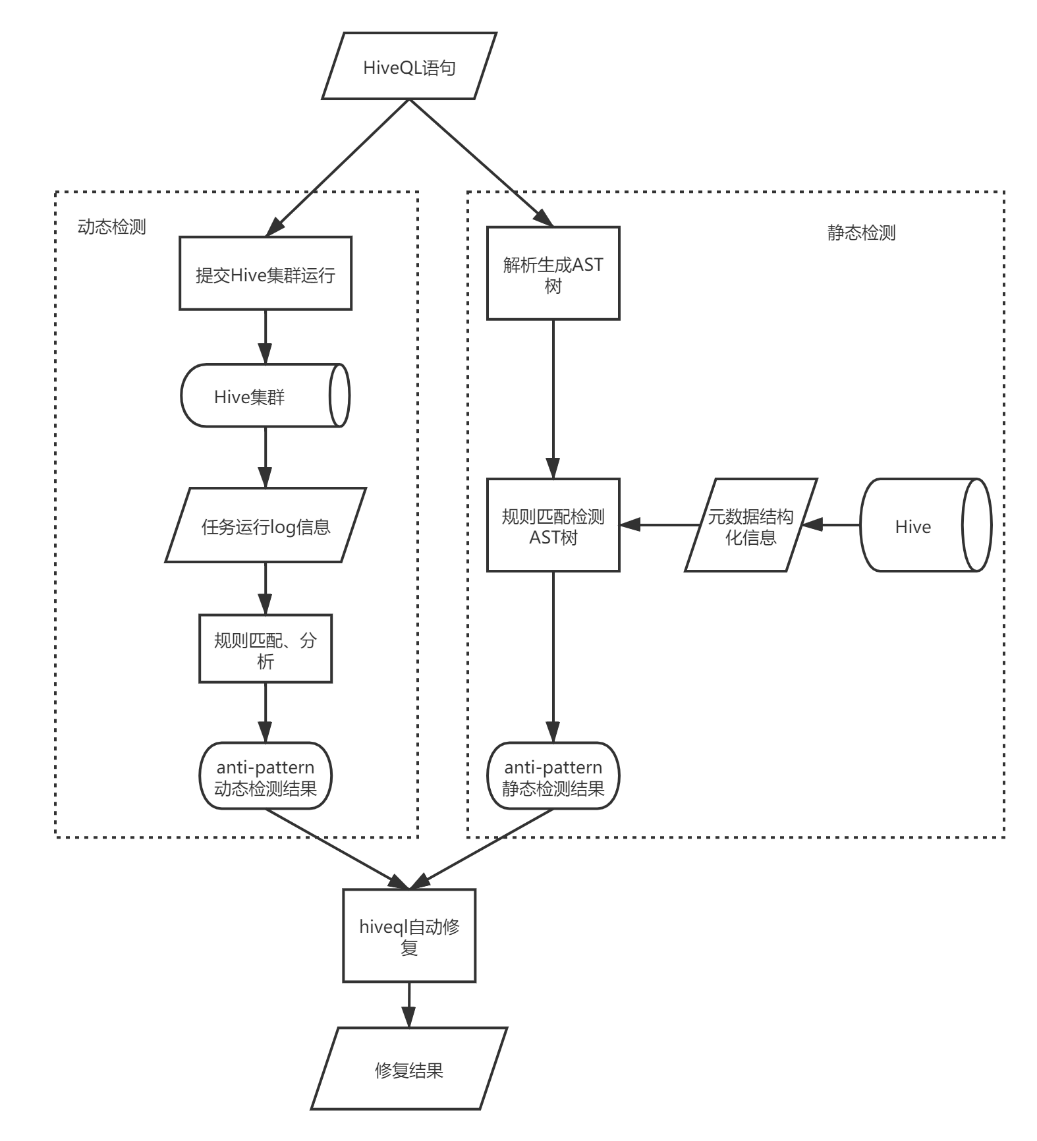
# 用例视图



本用例视图中包含用户（user）和Hive反模式检测修复系统两个Actor和请求静态或动态检测与修复，配置设置和获取检测与修复结果这四个Case。用户可向系统请求对待测HiveQL进行检测和修复，并获取系统返回的相应结果。用户还可设置相关的配置项。

# 逻辑视图

## 概述



## 在构架方面具有重要意义的设计包

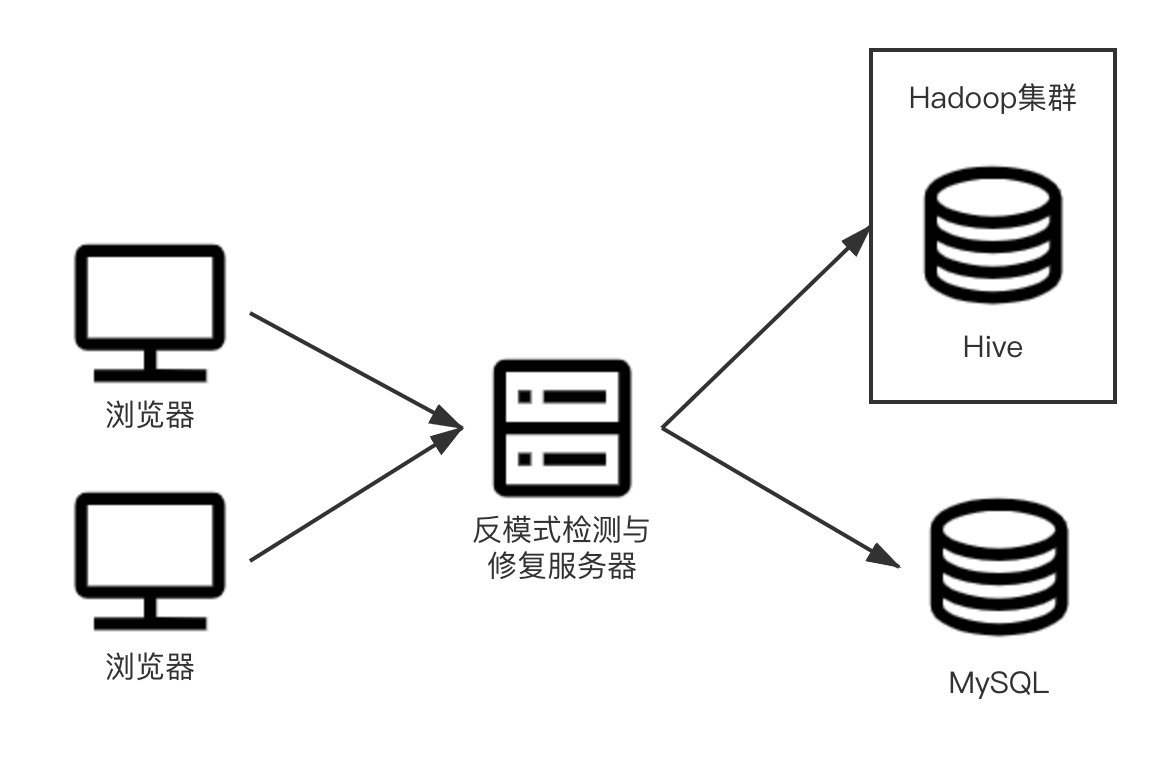
左侧动态检测和右侧静态检测两个分支是最为重要的两个设计包。

静态检测先将查询语句转换为抽象语法树（即Abstract Syntax Tree，下文简称AST），使用了Antlr4作为AST树转化与解析的工具，Antlr是一个可以根据输入自动生成语法树并可视化显示出来的开源语法分析器，它为多种语言提供了一个通过语法描述来自动构造自定义语言的识别器、编译器和解释器的框架。遍历AST是基于Antlr提供的监听器实现的，用户可以通过自定义的Listener类继承BaseListener类，重写其中的部分接口对语法树进行深度优先遍历，在监听器方法中编写规则来进行反模式的检测。

HiveQL提交到Hive后并解析为AST树后，经过Hive框架优化后生成一系列的Map，Reduce任务，之后将这些任务提交到Hadoop集群中，由Hadoop进行资源分配和任务的执行，并记录每个任务运行过程中的详细的日志。通过一定的规则，对运行时任务的运行时间，负载等日志信息进行分析来找出可能的反模式。

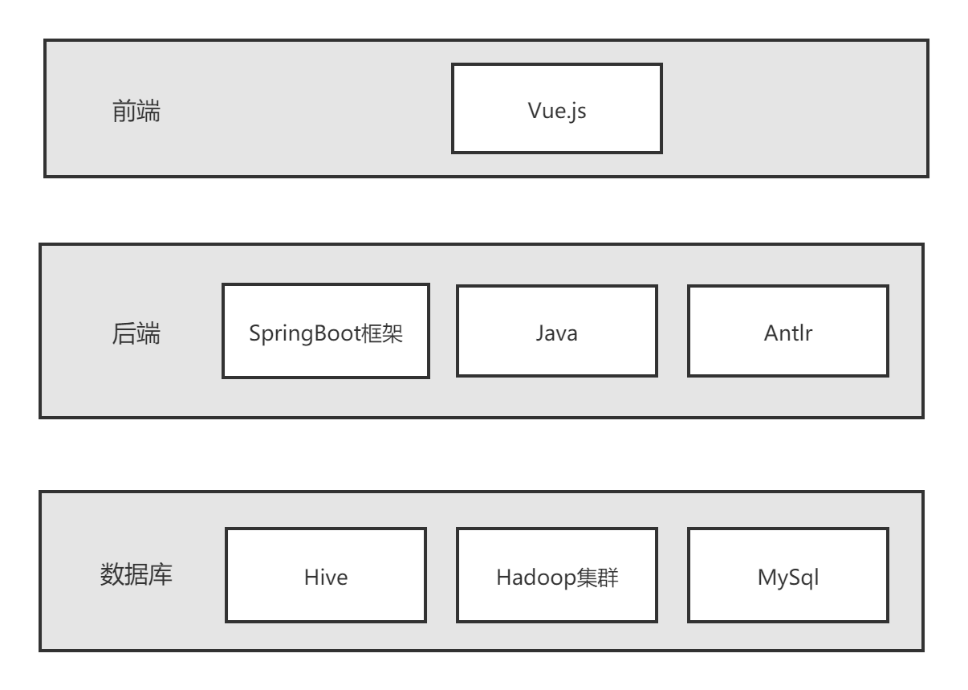
在得到检测结果后，本工具将利用修复模型对HiveQL做出相应的修复，并将检测与修复结果返回给用户。

# 部署视图



本软件采用B/S部署架构风格，在客户端使用浏览器Web应用程序进行软件使用，通过在服务器端的反模式检测修复服务，结合底层的Hadoop集群和MySql元数据库，返回待测HiveQL的检测和修复结果。

# 技术视图



本软件在数据库层面使用了Hadoop集群，并对目标检测对象Hive进行了相应的配置，使用MySql存储元数据。在后端的实现中，我们使用Java作为算法编程语言，并结合了Antlr语法解析器等工具进行系统实现，并使用SpringBoot作为后端框架，在前端使用Vue作为前端框架。

# 质量属性的设计

1）准确性

我们使用了Java这一成熟的多平台语言和广泛使用的语法解析器Antlr，为检测与修复算法的实现打下了良好的基础；在逻辑视图中使用了管道与过滤器风格，对检测与修复算法做出了较为清晰的定义。

2）服务器处理能力与可靠性

我们在软件架构中使用了许多成熟的框架，如SpringBoot，Vue，这些框架有助于提升软件的可靠性。而服务器处理性能，则与Hadoop集群性能和具体任务有密切联系。

3）可扩展性

我们在逻辑视图中对软件逻辑做了清晰地划分，这有助于在后期对某些功能环节作出扩展或增强。

4）易用性

我们设计了目前流行的B/S架构，并对用户可使用的功能做出了明确的定义，这有助于增强软件的易用性。