PTM数据分析服务开发

# 数据准备

* hf\_metadata.json的模型数据
* sample.json，截取hf\_metadata.json的部分数组，作为程序调试的样本

# 消息队列发送模型信息流

### Kafka环境搭建与部署

Kafka部署在zookeeper集群上（zookeeper集群的部署会在storm部分说明）

zookeeper.connect=master:2181,slave1:2181,slave2:2181

listeners=PLAINTEXT://slave2:9092

### Kafka消息producer

创建了topic：json\_data，使用send.py脚本向Kafka消息中间件发送dump后的json字符串

### Kafka消息Consumer

用Storm流处理工具来消费消息

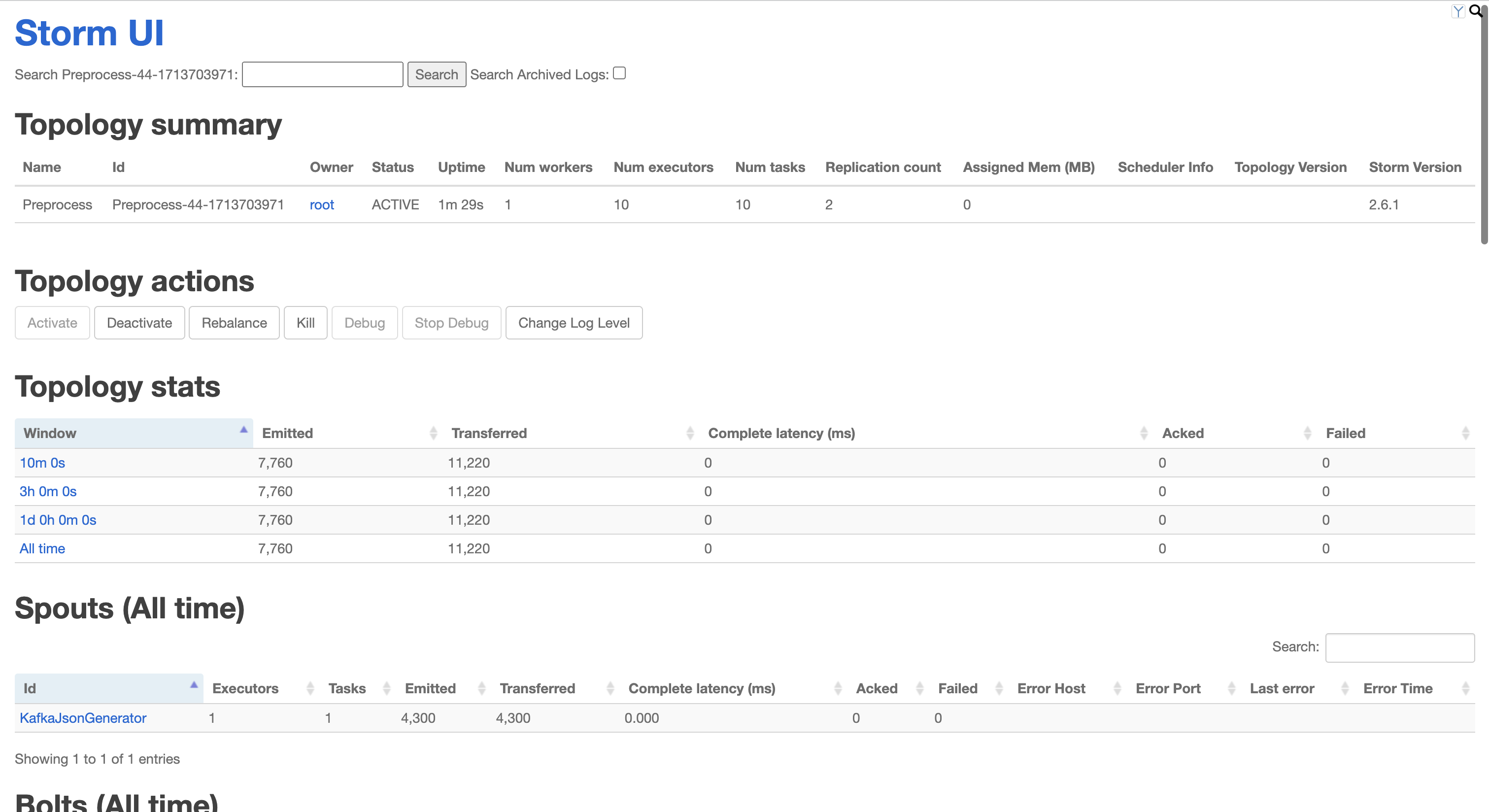
# Storm工具清洗数据

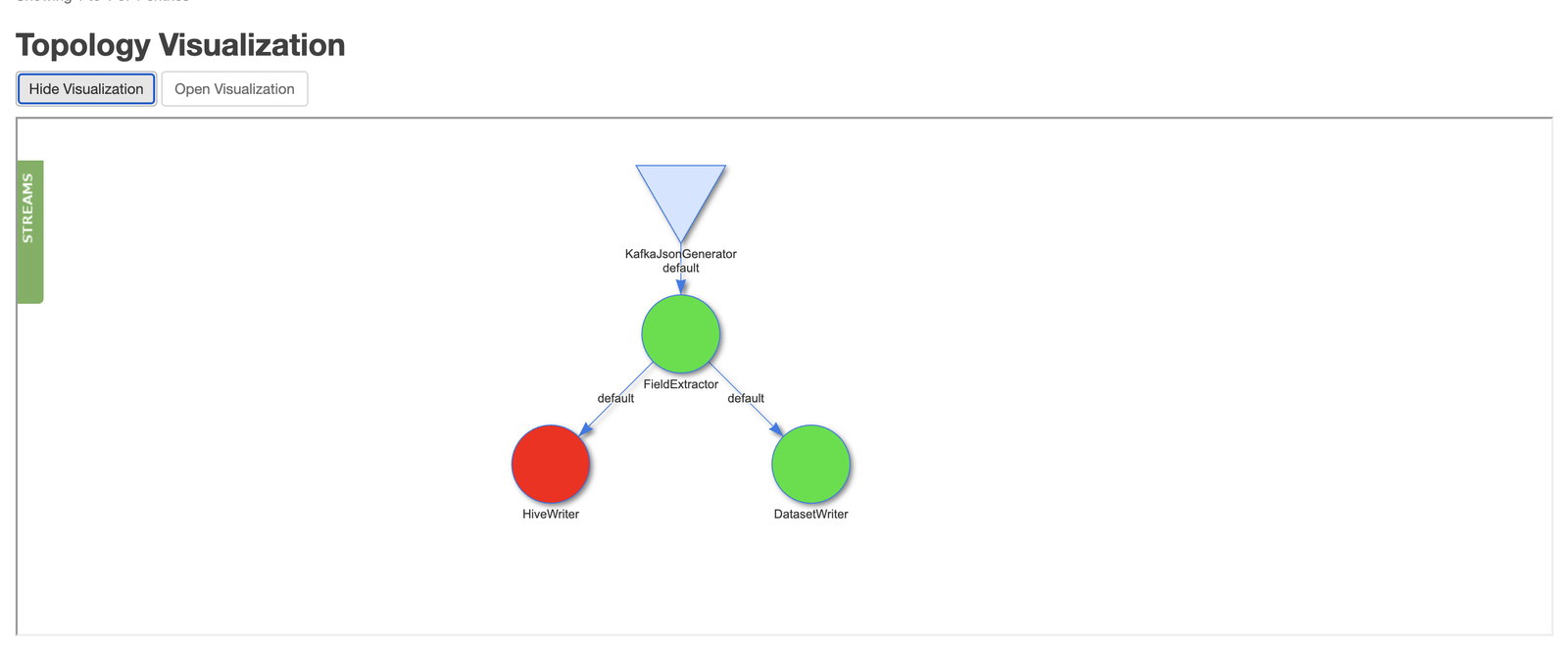
### Storm环境搭建与部署

* 搭建zookeeper集群，配置监听端口、选举端口、通讯端口
* 搭建Storm集群，配置zookeeper服务的连接，nimbus节点的ip，supervisor的port
* 启动zookeeper集群
* 在三台主机上启动nimbus，supervisor，logviewer，storm ui进程
* 使用scp命令将打包成jar文件的topology程序发送到远程主机上，并提交到集群上运行

### Storm拓扑程序

* storm运行后ui截图





根据图形化界面来说明storm的实现

* Spout节点从kafka读取json字符串，emit到ExtractFieldBolt
* ExtractFieldBolt节点用json字符串构建json对象，提取数据分析所需的字段（并且会填补缺失信息为默认值）并emit到HiveWriterBolt和HiveDatasetBolt
* HiveWriterBolt和HiveDatasetBolt将提取到的字段填写到hive数据库相应的table里，这两个Bolt分别对应的table将在“数据库设计于存储：4. 数据库设计”部分说明

后期优化：

1. ExtractFieldBolt里增加数据成员private Set<String> processedModelIds;保存处理过的modelid，实现去重功能
2. Hive数据库一次插入耗时较长，但是发现一次插入多条记录和插入单条记录耗时接近，因此使用缓存，保存HiveBolt收到的tuples，并且在tuples达到BATCH\_SIZE大小的时候将缓存批量插入到hive里

# 数据库设计与存储

### 数据库的选择

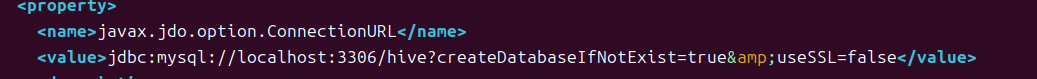
本次大作业面对的数据集数据条目较多，且不需要挖掘数据间关系、对于实时性的要求较低，遂采用Hive数据库进行数据的存储，其接口采用类SQL语法，可有效提高开发速度，降低学习成本，其对于大数据的处理相对于小数据具有更大的优势。

### 数据库的环境搭建

1. 在三台机器上安装jdk8、MySQL8和Hadoop，在主节点安装Hive
2. 对三台机器的Hadoop进行配置文件的编写，构成以其中一台（Master）为主节点的Hadoop集群



1. 在主节点的MySQL中创建Hive数据库，对主节点的Hive配置文件进行编写，使其将MySQL中的Hive作为存储元数据的数据库，至此Hive环境建立完成。

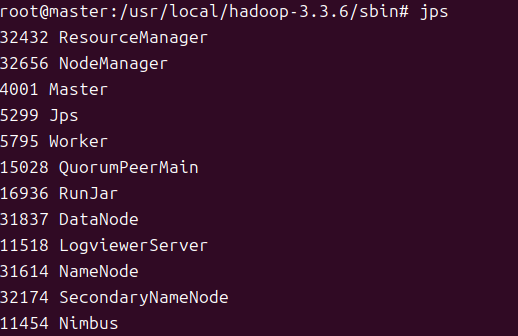


### 数据库的连接使用

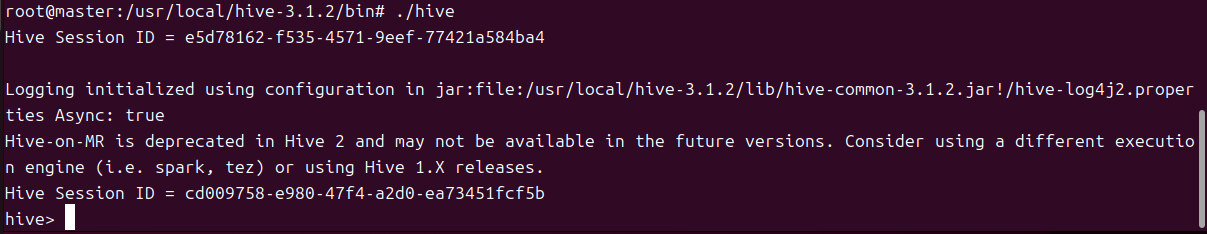
Hive数据库连接启动使用步骤如下：

1. service mysql start启动MySQL数据库；
2. 在主节点运行Hadoop下start-all.sh启动Hadoop集群；
3. nohup ./hive --service hiveserver2&后台启动hiveserver2以实现hive的远程连接（hive --service hiveserver2前台启动会占用端口，遂后台启动）；
4. 启动Hive，可直接在hive命令行中使用命令和脚本进行写入，也可以使用jdbc远程连接进行数据的写入和读取。

descript



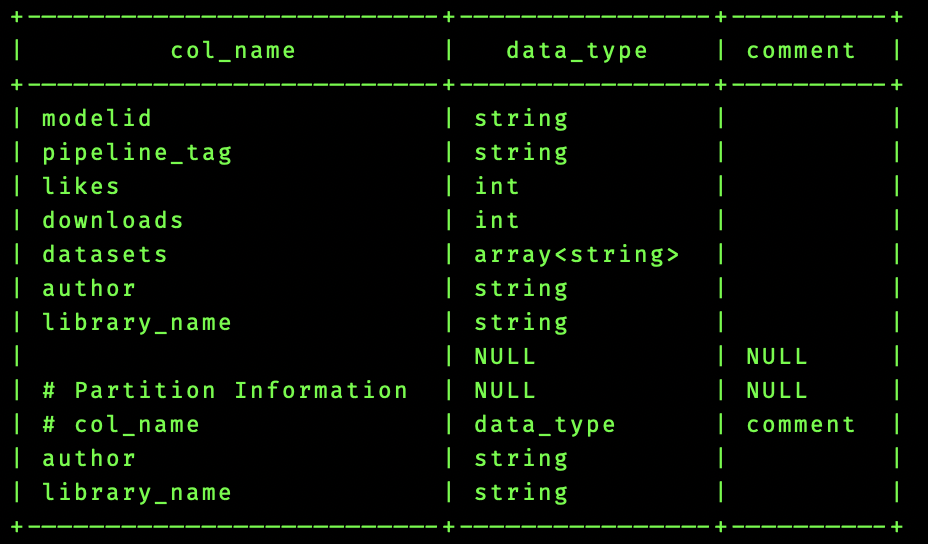
descript



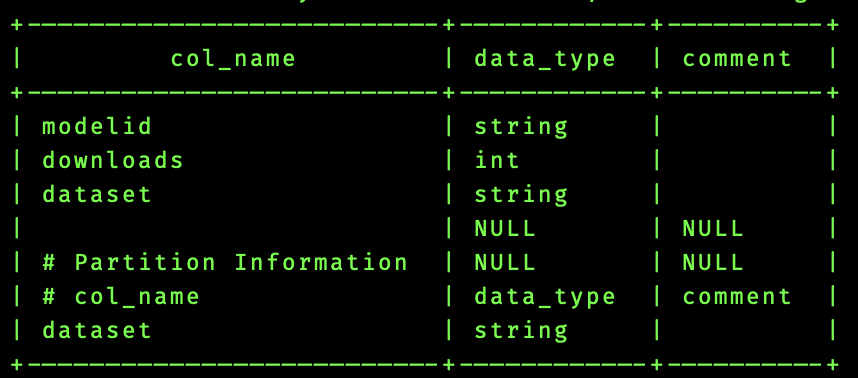
### 数据库设计

在ptm数据库下设计了两张table：model和dataset

* model表包含了所有数据分析服务所需要的信息，如下图所示，其中datasets是数组，保存了训练数据集的集合（如果没有将会填入一个"null\_dataset"）。另外为了提高后面按发布作者下载量和模型家族下载量的分析的效率，在author和library\_name这两个字段上建立了分区，提高查找性能。



* 由于在分析环节没有办法用FineBI工具来分割array<string>单独处理，因此单独建了一张表专门存储modelid和dataset的一一对应关系。为了便于按数据集划分的分析，将dataset字段设为partition分区



# 分析展示

1. **分析看板软件的选择**

由于要对数据进行分析与可视化操作，我们考虑使用Tableau、Apache Superset、FineBI等数据分析软件对Hive中的数据进行读取分析，考虑到成本问题，以及在数据挖掘作业中已经事先对FineBI进行基本了解与操作，我们最终选择FineBI作为本次作业分析与可视化工具。

1. **对数据进行的分析操作**
   1. **数据筛选**

在前面的操作中，我们得到了存储在hive中的表格数据，但是在观察得到的数据中，我们发现一些数据的以下特征：

数据中包含噪声：噪声来源于一些不符合预期的数据值，以及一些调试过程中产生的干扰数据。

数据取值两级分化：就下载量一项来说，整体数据分布两级分化，取值很大和取值很小的模型都是在数据中较多的。数据方差很大。

针对上面的数据特点，我们在创建看板之前还需要对数据进行二次筛选。具体操作如下：

1. 删去不符合预期的取值。在获得的数据中再次进行过滤，例如相同名称的模型出现两条数据集不一致的数据行，采取随机保留一条的策略。
2. 另外的一部分噪点则通过我们手动删除的方式予以删去。
   1. **选取数据可视化指标**

关于数据可视化的指标，我们选取了以下指标作为我们需要观测的内容：按家族划分的模型总量。按作者划分的模型下载量排行，模型总下载量排行，按照数据集划分的模型下载量排行。

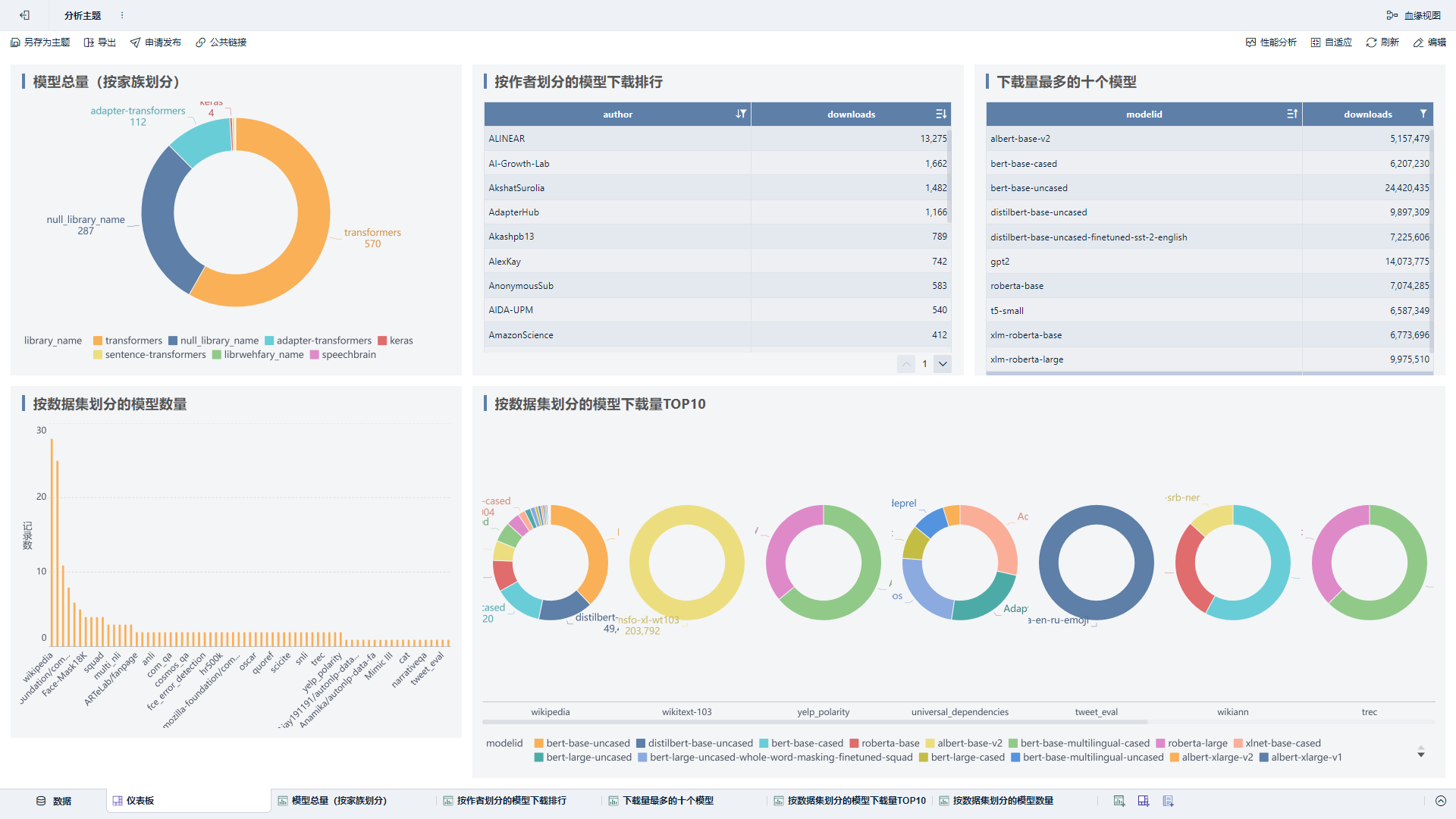
为能够观测出上面五个内容，需要统计的指标包括但不限于：

1. 统计模型名称出现次数
2. 统计模型家族中的模型名称
3. 统计每个作者的模型下载量
4. 每个模型的下载量总和
5. 每个数据集的下载量总和
6. 每个数据集在模型中的出现次数

对于每项指标的计算则是通过从数据库中进行条件查询可得。

1. **分析看板结果展示**

分析看板则通过FineBI中的组件和仪表盘制作完成。最终效果如下：



# 分布式与单机对比

1. **Storm的分布式与单机对比**

**规模与性能**

分布式Storm：可以在多台机器上运行，通过分布式部署实现水平扩展，处理大规模数据和高吞吐量。它利用分布式计算资源来处理大量的数据和并发任务，因此能够提供更高的性能和吞吐量。

单机Storm：在单个计算节点上运行，适用于小规模数据处理和简单应用场景。通常用于测试、开发和小型项目，因为它不能提供分布式部署所带来的性能和规模优势。

**可靠与容错**

分布式Storm：具有更高的可靠性和容错性，因为它可以通过在多个节点上复制任务来保证任务的执行。当某个节点发生故障时，任务可以在其他节点上重新启动，从而保证数据处理的连续性和可靠性。

单机Storm：由于只有一个节点在执行任务，因此在节点发生故障时可能会导致数据丢失或任务中断，可靠性和容错性较差。

**部署与管理**

分布式Storm：部署和管理相对复杂，需要配置和管理多个节点，以及处理节点之间的通信和协调。但是一旦配置完成，它可以提供更好的性能和扩展性。

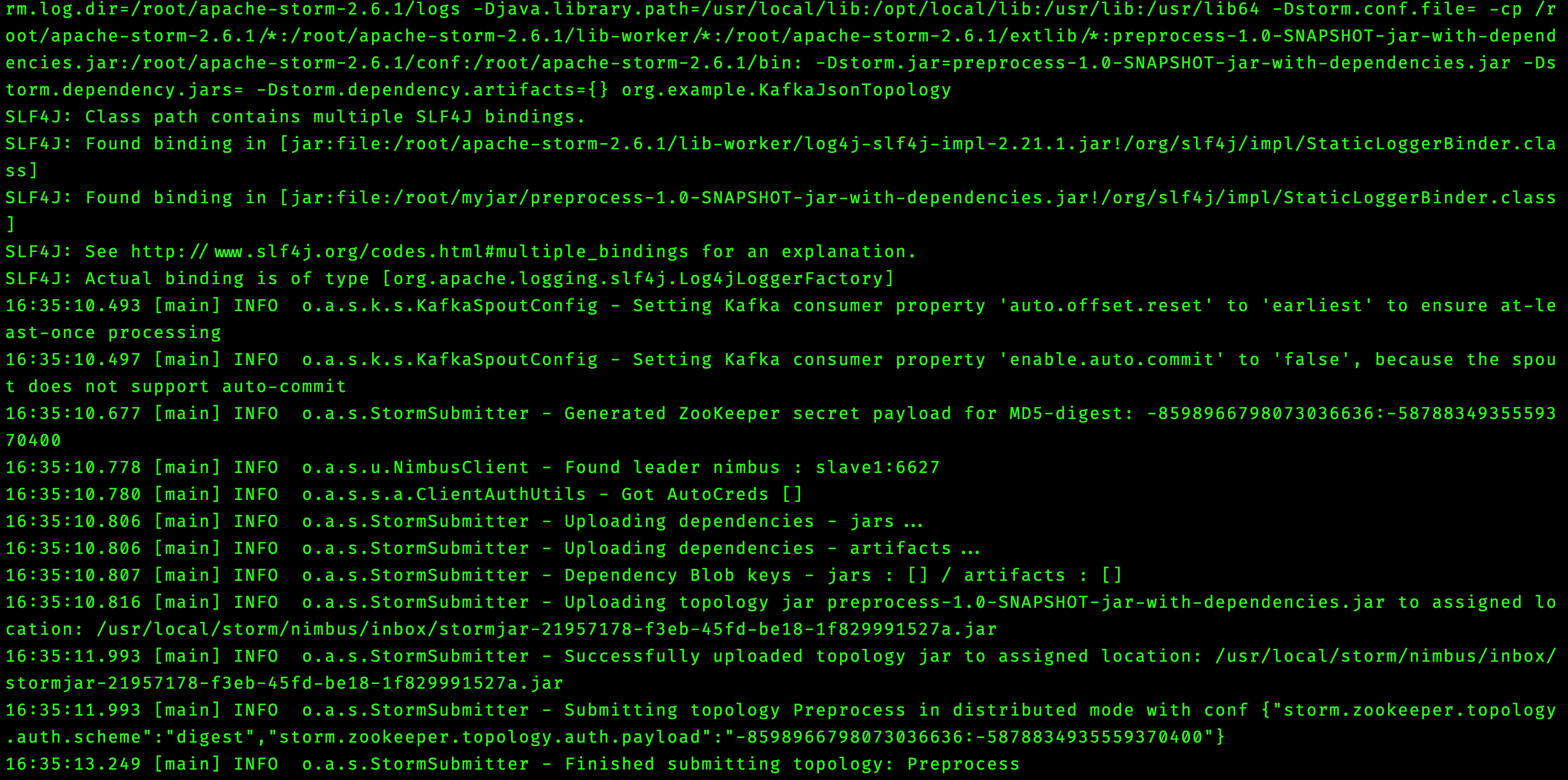
单机Storm：部署和管理相对简单，只需要在单个节点上安装和配置即可。这使得单机Storm更适合于快速原型开发和小规模部署，但在处理大规模数据时则显得力不从心。

**运行对比展示**

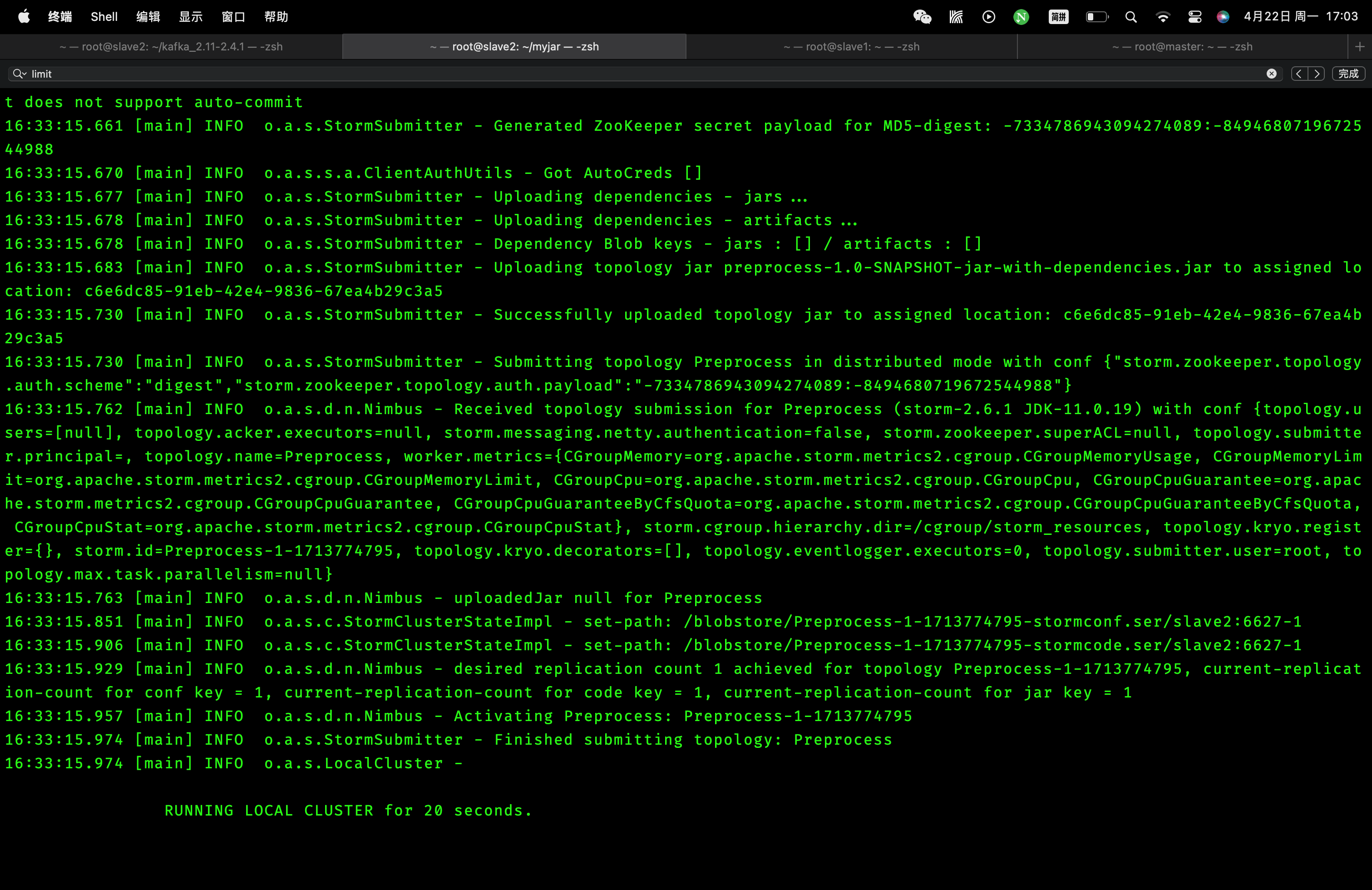
实际运行中的体会：

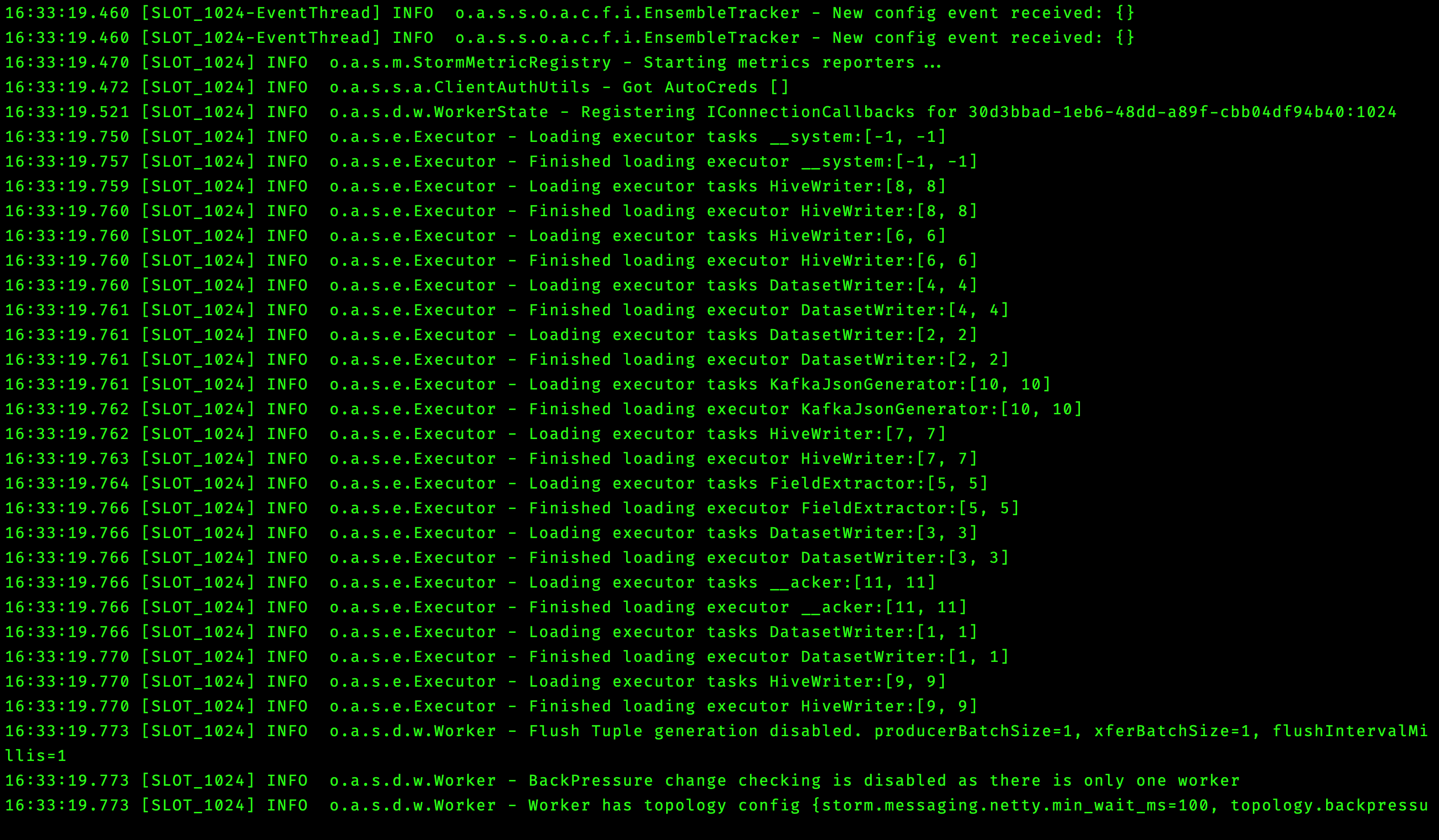
* LocalCluster模式可以在命令行直接输出System.out.println()的内容，比较便于调试。但是集群模式可以通过throw new RuntimeException的方法在ui显示信息，也可以调试
* localCluster模式运行可以看到更多log，例如下面的截图里显示了每个节点的任务数量和线程数量，可以帮助我们更好地了解storm拓扑执行的细节。

集群运行



单机运行





1. **Hive的分布式与单机对比**

**规模与性能**

分布式Hive：可以在多台机器上运行，通过分布式存储和计算资源来处理大规模数据。它能够利用集群中的多个节点并行处理查询，从而提供更好的性能和扩展性，适用于处理大规模数据集。

单机Hive：在单个计算节点上运行，适用于小规模数据集和简单查询。它通常用于测试、开发和小型数据分析任务，因为它不能提供分布式部署所带来的性能和规模优势。

**可靠与容错**

分布式Hive：具有更高的可靠性和容错性，因为它可以通过在多个节点上复制数据和任务来保证数据处理的连续性和可靠性。当某个节点或任务发生故障时，数据可以在其他节点上进行备份和恢复，从而避免数据丢失或任务中断。

单机Hive：由于只有一个节点在执行任务，因此在节点发生故障时可能会导致数据丢失或任务中断，可靠性和容错性较差。

**部署与管理**

分布式Hive：部署和管理相对复杂，需要配置和管理多个节点，以及处理节点之间的通信和协调。但是一旦配置完成，它可以提供更好的性能和扩展性，适用于处理大规模数据的企业级应用。

单机Hive：部署和管理相对简单，只需要在单个节点上安装和配置即可。这使得单机Hive更适合于快速原型开发、个人学习以及小规模数据分析任务。

# 分工与贡献

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 分工 | 贡献 |
| 王乐峰 | 520030910316 | 云主机环境准备；Zookeeper，Storm，Kafka环境搭建与部署；定时发送信息流实现；Storm拓扑程序实现；Hive数据库设计与存储实现。 | 50% |
| 杨俊杰 | 521021910628 | 数据库环境准备，MySQL、Hadoop集群、分布式Hive环境搭建与部署，参与进行数据分析。 | 25% |
| 邓可 | 519030910305 | 数据分析，数据清洗，看板设计和制作。 | 25% |