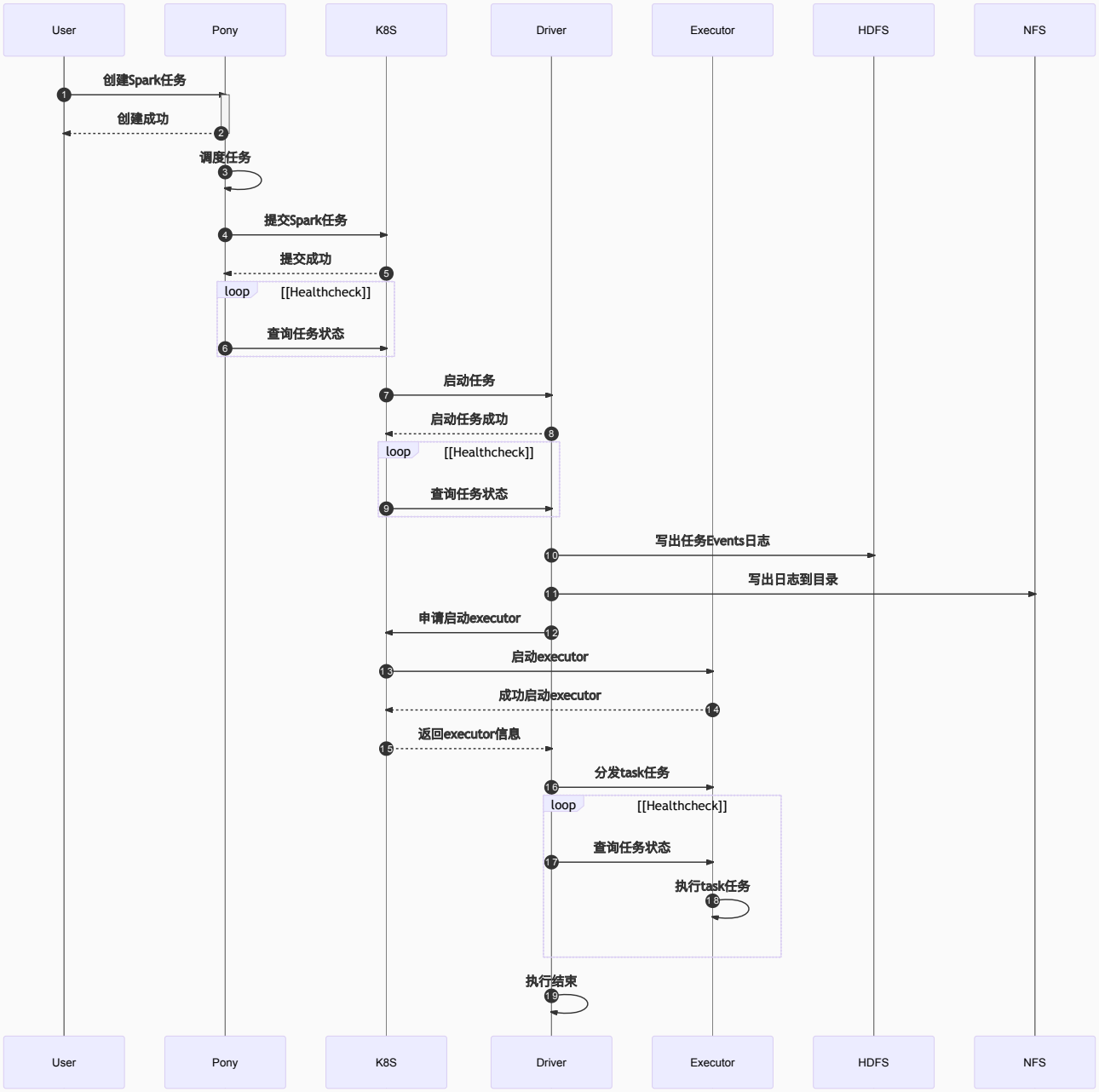
工作交接-吴婷

1. SPARK计算任务
   1.1 Spark任务流程
      1.1.1 交接说明
   1.2 Spark任务资源管理器-K8S
      1.2.1 交接说明
      1.2.2 简介
      1.2.3 资源配置设计
   1.3 Spark History Server
      1.3.1 交接说明
      1.3.2 地址
      1.3.3 配置
      1.3.4 部署
   1.4 Spark Log Server
      1.4.1 交接说明
      1.4.2 地址
      1.4.3 Spark任务日志
      1.4.4 NFS上Spark日志
   1.5 构建 PySpark 基础镜像
      1.5.1 交接说明
      1.5.2 spark-official 构建流程
      1.5.3 pyspark-official 构建流程
      1.5.4 pyspark-eigen 构建流程
      1.5.5 pyspark-eigen镜像测试
2.函数编程范式数据清洗工具-ETLSDK
      2.1 交接说明
      2.2 相关链接及地址
3. 标准数据上传下载工具-data_connector
      3.1 交接说明

3.2 相关链接及地址
4. HADOOP组件使用和维护
4.1 交接说明
4.2 HDFS
4.2 KAFKA
4.3 HBASE
4.4. EMR集群机器
4.5 监控管理
5. Ray
5.1 交接说明
5.2 相关文档及链接

1. SPARK计算任务

1.1 Spark任务流程



1.1.1 交接说明

Spark任务流程经历几次改动。从提交到Yarn资源管理器修改到K8S，目的是为了容器化ETL任务，解决依赖不同python环境的项目之间环境隔离问题。实时流运行模式从默认的cluster模式修改为local模式，目的是为了减少单个实时流类型的任务长驻资源，解决实时流任务数量增多集群资源紧缺问题。

目前越来越多的实时流任务，Spark只作实时读取数据源，UDF函数中调用Collie服务，实际数据处理在Collie服务上，Collie服务提供了丰富的debug数据的功能。

当前存在的问题：

a. HDFS的Datanode存储大量小于64MB的小文件，NameNode管理着整个HDFS文件系统的元数据内存报警。因为大部分任务涉及到使用shuffle类型算子，或者repartition Dataframe的partition数量提高并发执行效率，产生大量的小文件写出到HDFS。

1.2 Spark任务资源管理器-K8S

1.2.1 交接说明

Spark使用K8S作为资源管理器是因为Spark为了项目实现容器化隔离，并且相比在Yarn上做容器化K8S公司技术支持更多有优势降低运维成本。目前是维护状态，这块基本是@金鸿彬 在关注。比较容易出现的问题是K8S资源在零点任务数量比较多时资源不够，目前解决方案是实现Pony Spark 任务队列。

交接接收人：金鸿彬

1.2.2 简介

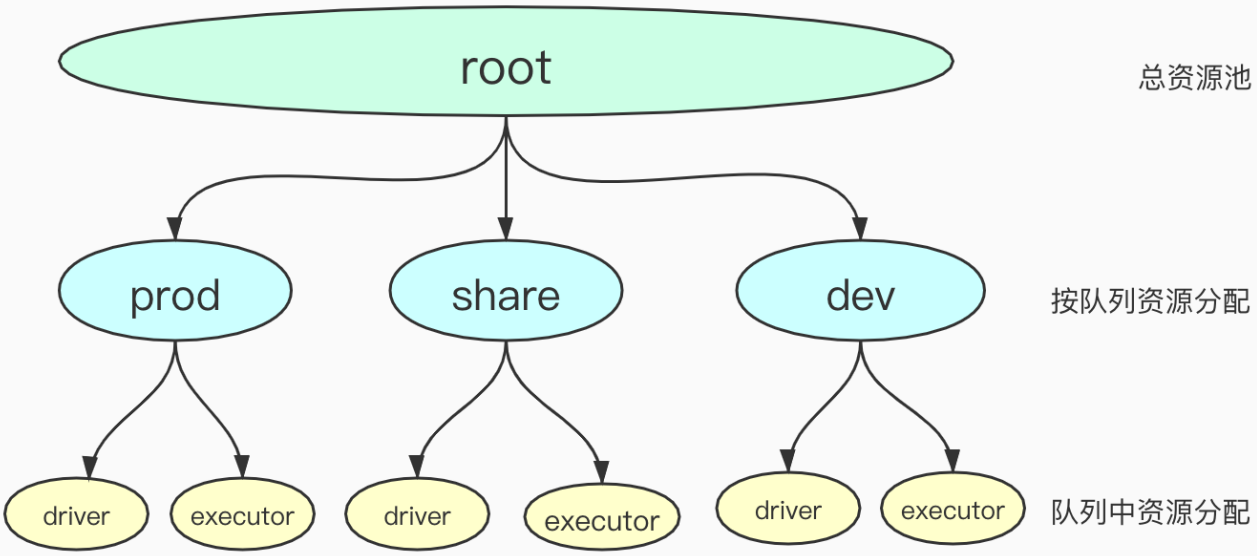
- Dashboard地址: http://spark-k8s.aipp.io
- Spark K8S负责人：吴俊超

  环境设置：

  Spark-Prod: 用于Pony work任务　　链接

  Spark-Dev: 用于Pony 实验任务　　链接

1.2.3 资源配置设计



i. 资源节点区分使用环境通过K8S节点打不同的标签：

   1. share环境：spark-dev: spark-dev, spark-prod: spark-prod
   2. dev环境：spark-dev: spark-dev
   3. prod环境：spark-prod: spark-prod
   4. 目前节点配置share:prod:dev=6:8:0

ii. 考虑Driver任务先启动，Executor在Driver启动之后由Driver向k8s申请资源。在离线调度任务请求非常多时极限情况下生成大量的Driver任务占用集群全部资源，而Executor没有资源可以申请，Driver任务因Executor没有资源超时失败。Driver任务使用独立的节点，资源节点按Pod跑Driver类型任务使用`driver:driver`标记专用节点。

iii. 资源节点链接

iv. driver yaml:

```
nodeSelector:
  role: {{ service_account_name }}
  {{ namespace }}: {{ namespace }}
  driver: driver
```

v: executor yaml:

```
nodeSelector:
  role: {{ service_account_name }}
  {{ namespace }}: {{ namespace }}
```

vi: 资源监控

driver 节点内存监控 http://monitor.aipp.io/d/6kcRgvxZz/jie-dian-jian-kong-xin-xi-spark-kubernetes?orgId=1&from=now-24h&to=now&fullscreen&panelId=63

executor 节点内存监控 http://monitor.aipp.io/d/6kcRgvxZz/jie-dian-jian-kong-xin-xi-spark-kubernetes?orgId=1&from=now-12h&to=now&fullscreen&panelId=69

1.3 Spark History Server

1.3.1 交接说明

Spark History Server提供Spark任务的WEBUI界面列出运行时信息，目前最多显示的作业数是500。目前处在维护中。还存在的问题是实时流任务因为运行时间长，保存了任务的大量Events信息，在Spark UI上展示解析要很久，基本解析不出来。

修改Spark History Server流程，修改eigen-spark2里对应的配置，push到仓库，触发Jenkins部署构建。

交接接收人：金鸿彬

1.3.2 地址

https://spark.aidigger.com/

1.3.3 配置

https://git.aipp.io/EigenLab/eigen-spark2/blob/branch_2.4.3/eigen_spark_conf/spark-defaults.conf

1.3.4 部署

https://jenkins.aidigger.com/job/Spark-history-Prod-Deploy/

1.4 Spark Log Server

1.4.1 交接说明

Log Server是@倪哲鸣部署用来查看日志的服务。现在使用稳定，使用上也没有用户反馈问题。

交接接收人：金鸿彬

1.4.2 地址

https://logs.aidigger.com/files/

1.4.3 Spark任务日志

python进程日志配置：https://git.aipp.io/EigenLab/etlsdk/blob/master/etlsdk/__init__.py

java进程日志配置：https://git.aipp.io/EigenLab/eigen-spark2/blob/branch_2.4.3/eigen_spark_conf/log4j.properties

1.4.4 NFS上Spark日志

查看Spark日志在NFS目录：

dev: /srv/nas-logs/spark-dev/spark-logs/{pony-execute-id}

prod: /srv/nas-logs/spark-prod/spark-logs/{pony-execute-id}-exec-{exec_num}

删除策略：

最后修改时间15天前或文件大小大于64M

日志被异常删除或未被删除联系@王飞

1.5 构建 PySpark 基础镜像

1.5.1 交接说明

PySpark镜像容器，目前在维护阶段，之前做过的修改是，Spark版本升级，修改Spark UI日志链接改成公司的Log Server地址，修改Spark源码给Executor增加Ptrace权限。这块平时是金鸿彬在维护。

存在的问题：

a. 类似于spark-defaults.conf这样的配置文件经常修改，每次修改要通过编译新的镜像生效。之前想到通过修改Ambari上Spark配置，保持Hadoop这块配置文件修改统一入口。

下面是鸿彬提的改进：

a. 将 pyspark-official 的 OpenJDK 环境抽离成单独的镜像

b. 在 pyspark-official 中缓存 zinc

c. 在 pyspark-official 中缓存 Scala

d. 合并两个构建流程 pyspark-official 和 pyspark-eigen

e. 将三个构建流程（合并后变为两个）自动化

开发流程：

Repo地址：https://git.aipp.io/EigenLab/eigen-spark2/tree/branch_2.4.3

在开发机上创建新分支，做对应代码修改，完成测试后，推到远程仓库。PySpark 基础镜像的构建是手动触发 Jenkins job，分为三步：spark-official （基础源码构建），pyspark-official（安装python环境，将pyspark加到PYTHONPATH，安装pyspark shell） 和 pyspark-eigen（使用公司hadoop配置）. 根据代码修改选择需要构建的镜像，用构建完的新镜像做功能测试，然后merge到主分支。之前做版本升级，做了批量复制Pony任务修改写出表（修改不了用户hardcode在代码里的写出表），批量测试脚本。

交接接收人：金鸿彬

1.5.2 spark-official 构建流程

spark-official 是在 eigen 基础镜像中构建 Spark 源代码，生成 Spark jar 文件（虽然有 PySpark 的文件，但没有配置 Python 环境）.

参考官方的构建文档，官方的 Dockerfile 是假设已经通过 `./dev/make-distribution.sh`，在 `dist` 目录下生成了运行 Spark 需要的文件，然后将 `dist` 下的文件放入镜像中. 对此的修改是，完全将 Spark 的构建放在有 Java 环境的 Docker 镜像中. 同时，使用 multi-stage build 降低镜像大小. 需要说明的是，spark-official 构建流程几乎是完全参照官方的 Spark 构建流程. 如果使用单个镜像构建，其中必定会包含源代码和中间文件（删除也不会降低镜像的大小，参考 Before multi-stage builds）. 而运行 Spark 应用只需要 JDK（也许只需要 JRE），Spark jar 文件和相应的配置. 因此引入 multi-stage build，在中间镜像中生成 jar 文件，然后把文件复制到 Spark 镜像中.

此外，对 Spark 构建流程做了一些优化. 分析 `./dev/make-distribution.sh` 发现，构建逻辑主要在 `./build/mvn` 文件中. 分析可知，构建主要是以下工作：zinc，Scala 和 Maven 的下载，以及 Maven 构建 Spark. 对此作了两处修改：使用阿里云的 Apache 镜像下载 Maven（设置 `APACHE_MIRROR` 环境变量）；使用阿里云的 Maven repository（设置 `setting.xml` ）. 下载 zinc 和 Scala 还是使用官方的 URL，可能会比较慢（需要设置 `TYPESAFE_MIRROR` 环境变量，但是没有找到合适的镜像）.

在这个 merge request 生效之后（目前还没有生效），手动触发 spark-official 构建时，设置 `buildtag=2.4.3` 和 `branch_name=origin/branch_2.4.3` 即可.

1.5.3 pyspark-official 构建流程

pyspark-official 是在 spark-official 的基础上，安装 Python 环境和配置 `PYTHONPATH` 环境变量. 配置 `PYTHONPATH` 环境变量的目的是为了让 Python 解释器找到 PySpark 的代码. 因此，使用 pip 安装 PySpark 是多余的. 但是，由于在 PySpark 中引入了 gevent 和 eigenlog，需要在镜像中额外安装这两个依赖.

手动构建 pyspark-official 时，指定 `python_minor_version`，`buildtag` 和 `branch_name`. 例如构建支持 Python 3.6 的镜像时，设置 `python_minor_version=6`，`buildtag=2.4.3-py36`

和`branch_name=origin/branch_2.4.3`.

1.5.4 pyspark-eigen 构建流程

pyspark-eigen 是在 pyspark-official 镜像中加入了 Kubernetes 和 Hadoop 相关的配置. *用户应该使用这个构建产生的镜像, 不应该直接使用前面的两个镜像.*

手动构建 pyspark-eigen 时, 指定 `basetag`, `buildtag` 和 `branch_name`. 例如构建支持 Python3.6 的镜像时, 设置 `basetag=2.4.3-py36`, `buildtag=2.4.3-py36` 和 `branch_name=origin/branch_2.4.3`.

1.5.5 pyspark-eigen镜像测试

```
git clone https://git.aipp.io/EigenLab/etlsdk
cd etlsdk/
pip3 install -r tests/requirements.txt
py.test --junitxml results.xml --cov etlsdk --cov-report=xml
```

2.函数编程范式数据清洗工具-ETLSDK

2.1 交接说明

ETLSDK让用户使用函数的方式编写数据清洗过程，提供DatasourceFactory根据数据表的大禹信息简化用户读写的数据源过程。现在基本维护状态，现在大部分需求是在原有的功能做些修改，比如修复写出HIVE时使用Dynamic Partitions报错，增加写出到MYSQL时不检查表开关，解决df写出的MYSQL表不在大禹上时验证不通过问题。ETLSDK的使用文档是ETLSDK最详细的文档了，基本可以解决用户问题。

存在的问题：

a. PySpark Worker进程日志有EOF Error报错。

b. 实时流读写Hbase。

开发流程：

在开发机上克隆代码后创建新分支，做完对应代码修改（一般一并修改代码中文档描述，代码合并到主分支后readthedocs自动构建更新使用文档），加上UT测试，跑通UT测试后，推到到远程分支，通过pr_ci后，合并到主分支。在主分支上打tag（并修改setup中版本号，保持和tag一致），推送到远程。在Jarxis上提发布审批。

交接接收人：马新民

2.2 相关链接及地址

设计文档： https://git.aipp.io/pub/wiki/blob/master/系统/Pony/ETL上线文档/ETLSDK/ETLSDK设计文档.md
使用文档： https://alpha-readthedocs.aidigger.com/docs/etlsdk/en/latest/
Readthedocs: https://alpha-readthedocs.aidigger.com/dashboard/ 用户名 eigen 密码 eigen
Repo地址: https://git.aipp.io/EigenLab/etlsdk/
ETLSDK Jenkins：
CI: https://jenkins.aidigger.com/job/etlsdk_ci/
https://jenkins.aidigger.com/job/etlsdk_pr_ci/
CD: https://jenkins.aidigger.com/job/etlsdk_cd/
项目贾维斯链接： https://jarvis.aidigger.com/app/78

---

3. 标准数据上传下载工具-data_connector

3.1 交接说明

这个工具用于标准数据上传下载，开发是按照ETLSDK的Plugin插件编程规范设计的(https://alpha-readthedocs.aidigger.com/docs/etlsdk/en/latest/quick_start.html#plugin)，2020-5-28之后再没有再修改过，下载数据的插件用户正在使用。数据插件平时是NLP算法那边用户使用的比较多。每个字段在Pony上有说明。

交接接收人：马新民

3.2 相关链接及地址

Repo: https://git.aipp.io/pony_plugins/data_connector
插件列表：

插件	SQL支持	是否支持调度	插件类型
hive2dataset_sql模式	支持sql所有语法	支持	下载
hive2dataset_表单模式	支持filter、select、limit	支持	下载
hive2json_sql模式	支持sql所有语法	支持	下载
hive2json_表单模式	支持filter、select、limit	支持	下载
mysql2json_sql模式	支持sql所有语法	不支持	下载
mysql2json_表单模式	支持filter、select、limit	不支持	下载
mysql2dataset_sql模式	支持sql所有语法	不支持	下载
mysql2dataset_表单模式	支持filter、select、limit	不支持	下载
根据json内容指定图片保存位置	不支持	不支持	下载

根据实验参数指定图片保存位置	不支持	不支持	下载
下载大禹数据集	支持limit	不支持	下载
下载 oss 数据	支持limit	不支持	下载
json上传至hive表	支持filter、select、limit	支持	上传
dataset上传至hive表	支持filter、select、limit	支持	上传

4. HADOOP组件使用和维护

4.1 交接说明

这块现在处于维护的状态，有些组件存在可做的feature会列在下面。报警触发过程，Ambari监控报警，Grafana上读取Ambari postgres报警信息并配置No Alert报警，Ambari监控报警通过No Alert发出。排查问题，我一般先看日志，日志路径一般Java进程上有，或者Ambari上有配置，遇到问题查问题。

交接接收人：金鸿彬

4.2 HDFS

存在的问题：

a. HDFS的Datanode存储大量小于64MB的小文件

1. 20200807Datanode-Heap报警排查
2. 20200728namenode异常

4.2 KAFKA

Kafka Bootstrap Server访问地址：kafka.aidigger.com:6667 ，使用负载均衡实例kafka.aidigger.com提供统一地址。

1. 20200827KafkaBroker和Namenode分离
2. 线上Ambari_Kafka暴露JMX及Burrow监控数据接口

4.3 HBASE

1. 20200831Hbase重启升级Worker3内核

4.4. EMR集群机器

1. 20200825header1机器加内存升级
2. emr2 部署使用 playbook 实现
3. emr2-worker-磁盘扩容

4.5 监控管理

ambari dashboard: http://ambari-server.ipa.aidigger.com:8080/#/login

user: admin

pwd: admin

5. Ray

5.1 交接说明

这个项目目前在调研阶段，之前把ETL的任务改写成Ray，执行结果不符合预期（更新在Ray框架调研与实践），下一步计划是分别给出“图片模型打标签”Ray和Spark版本耗时分析。

交接接收人：马新民

5.2 相关文档及链接

1. 用户使用文档
2. Ray框架调研与实践