Fate V1.0 部署指南

目录

1.安装准备

1.1 服务器配置

服务器	
数量	>1(根据实际提供的服务器分配模块)
配置	16 core /32 memory / 300GB 硬盘/50M 带宽
操作系统	CentOS linux 7.2
依赖包	yum source gcc gcc-c ++ make autoconfig openssl-devel supervisor gmp-devel mpfr-devel libmpc-devel libaio numactl autoconf automake libtool libffi-dev（可以使用初始化脚本 env.sh 安装它们）
用户	用户：app owner：apps（app 用户可以 sudo su root 而无需密码）
文件系统	1. 300G 硬盘安装在/ data 目录下 2.创建/ data / projects 目录，项目目录属于 app:apps

1.2 安装包准备

wget https://webank-ai-1251170195.cos.ap-guangzhou.myqcloud.com/fate-base.tar.gz

tar -xzvf fate-base.tar.gz

2.集群规划

节点名称	主机名	IP 地址	操作系统
Host-PartyA	VM_0_1_centos	192.168.0.1	CentOS 7.2
Guest-PartyB	VM_0_2_centos	192.168.0.2	CentOS 7.2

3.基础环境配置

3.1 hostname 配置

1）两台机器分别修改主机名

hostnamectl set-hostname VM_0_1_centos

hostnamectl set-hostname VM_0_2_centos

2）两台机器分别加入主机映射

vim /etc/hosts

192.168.0.1　VM_0_1_centos

192.168.0.2　VM_0_2_centos

3.2 关闭 selinux

sed -i　'/^SELINUX/s/=.*/=disabled/' /etc/selinux/config

setenforce 0

3.3 修改 Linux 最大打开文件数

vim /etc/security/limits.conf

* soft nofile　65536

* hard nofile 65536

3.4 关闭防火墙

systemctl disable firewalld.service

systemctl stop firewalld.service

systemctl status firewalld.service

3.5 软件环境初始化

上传完毕后，可将上述带依赖包的 fate-base 目录打成 fate-base .tar 包放到目标服务器的/data/app(可自选)目录下，然后进行解压操作:

cd /data/app

tar -xf fate-base .tar

cd fate-base

3.5.1 初始化服务器

1）初始化服务器，root 用户下执行：

sh env.sh

2）配置 sudo

vim /etc/sudoers.d/app

app ALL=(ALL) ALL

app ALL=(ALL) NOPASSWD: ALL

Defaults !env_reset

3）配置 ssh 无密登录

a.两台机器分别执行

su app

ssh-keygen -t rsa

cat ~/.ssh/id_rsa.pub >> /home/app/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys

b.合并 id_rsa_pub 文件

拷贝 192.168.0.1 的 authorized_keys 到 192.168.0.2 ~/.ssh 目录下,追加到 192.168.0.2 的 id_rsa_pub 到 authorized_keys，然后再拷贝到 192.168.0.1

scp ~/.ssh/authorized_keys app@192.168.0.2:/home/app/.ssh

输入密码：fate_dev

在 192.168.0.2 执行

cat ~/.ssh/id_rsa.pub >> /home/app/.ssh/authorized_keys

scp ~/.ssh/authorized_keys app@192.168.0.1:/home/app/.ssh

覆盖之前的文件

c.ssh 测试

ssh app@192.168.0.1

ssh app@192.168.0.2

3.5.2 配置 JDK

以下操作步骤在 app 用户下执行

检查 jdk1.8 是否安装，若未安装则执行 install_java.sh 脚本进行安装：

sh install_java.sh

3.5.3 配置 redis

如果需要安装在线模块，请检查是否安装了 redis 5.0.2，如果没有安装，请执行 install_redis.sh 脚本进行安装：

sh install_redis.sh

3.5.4 配置 Python

检查 Python3.6 及虚拟化环境，若未安装则执行 install_py3.sh 脚本进行安装：

sh install_py3.sh

3.5.5 配置 mysql

检查 mysql8.0 是否安装，若未安装则执行 install_mysql.sh 脚本进行安装：

sh install_mysql.sh

安装完 mysql 之后，将 mysql 密码修改为“fate_dev”并创建数据库用户“fate”(根据实际需要修改)

mysql 安装后需要在安装 mysql 的节点上使用以下语句对 party 内所有 ip 赋权

$/data/projects/common/mysql/mysql-8.0.13/bin/mysql -uroot -p –S /data/projects/common/mysql/mysql-8.0.13/mysql.sock

Enter password:(please input the original password)

>set password='fate_dev';

>CREATE USER 'root'@'192.168.0.1' IDENTIFIED BY 'fate_dev';

>CREATE USER 'root'@'192.168.0.2' IDENTIFIED BY 'fate_dev';

>GRANT ALL ON *.* TO 'root'@'192.168.0.1';

>GRANT ALL ON *.* TO 'root'@'192.168.0.2';

>flush privileges;

检查完毕，回到执行节点进行项目部署。

4.项目部署

注：此指导安装目录默认为/data/projects/，执行用户为 app，安装时根据具体实际情况修改。

4.1 获取项目

进入执行节点的/data/projects/目录，执行 git 命令从 github 上拉取项目：

cd /data/projects/

git clone https://github.com/WeBankFinTech/FATE.git

4.2 Maven 打包

进入项目的 arch 目录，进行依赖打包：

cd FATE/arch

mvn clean package -DskipTests

cd FATE/fate-serving

mvn clean package -DskipTests

cd FATE/fateboard

mvn clean package -DskipTests

wget https://webank-ai-1251170195.cos.ap-guangzhou.myqcloud.com/third_party.tar.gz

tar -xzvf third_party.tar.gz -C FATE/arch/eggroll/storage-service-cxx/third_party

4.3 修改配置文件

进入到 FATE 目录下的 FATE/cluster-deploy/scripts 目录下，修改配置文件 configurations.sh

配置文件 configurations.sh 说明:

配置项	配置项意义	配置项值	说明
user	服务器操作用户名	默认为 app	使用默认值即可
dir	fate 安装路径	默认为/data/projects/fate	使用默认值即可
mysqldir	mysql 安装目录	默认为 /data/projects/common/mysql/mysql-8.0	Mysql 安装路径
javadir	JAVA_HOME	默认为 /data/projects/common/jdk/jdk1.8	jdk 安装路径
venvdir	python virtualenv 安装目录	默认为/data/projects/fate/venv	python virtualenv 安装目录
partylist	party.id 数组	每一个数组元素代表一个 partyid	根据 partyid 进行修改
JDBC0	对应 party 所在 jdbc 配置	每一个 party 对应的 jdbc 配置：从左到右依次是 ip dbname username password（此 user 需要有 create database 权限）	若有多个 party 则依次为 JDBC0， JDBC1… 顺序与 partid 对应 根据 jdbc 配置进行填充
iplist	各 party 包含服务器列表	表示每个 party 包含的服务器 ip 列表(除 exchange 角色外)	所有 party 涉及的 ip 都放到这个 list 中，重复 ip 一次即可
fedlist0	federation 角色 ip 列表	表示 party 内包含 federation 角色的服务器列表(当前版本只有一个)	若有多个 party 则依次为 fedlist0， fedlist1… 顺序与 partid 对应
meta0	meta-service 角色 ip 列表	表示 party 内包含 meta-service 角色的服务器列表(当前版本只有一个)	若有多个 party 则依次为 meta0， meta1… 顺序与 partid 对应

proxy0	proxy 角色 ip 列表	表示 party 内包含 proxy 角色的服务器列表(当前版本只有一个)	若有多个 party 则依次为 proxylist0，proxylist1…顺序与 partid 对应
roll0	roll 角色 ip 列表	表示 party 内包含 roll 角色的服务器列表(当前版本只有一个)	若有多个 party 则依次为 rolist0， rolist1… 顺序与 partid 对应
egglist0	egg 角色列表	表示 party 内包含 egg 角色的服务器列表	若有多个 party 则依次为 egglist0，egglist1…顺序与 partid 对应
exchangeip	exchange 角色 ip	exchange 角色 ip	双边部署时若不存在 exchange 角色，则可为空，此时双方直连； 单边部署时，exchange 值可为对方 proxy 或 exchange 角色，必须提供。
tmlist0	task_manager 角色 ip 列表	表示 party 内包含 task_manager 角色的服务器列表(当前版本只有一个)	若有多个 party 则依次为 tmlist0， tmlist1… 顺序与 partid 对应
serving0	Serving-server 角色所在 ip 列表	每个 party 中包含 Serving-server 角色 ip 的列表(当前版本只有一个)	若有多个 party 则依次为 serving0，serving1…顺序与 partid 对应

注：tmlist0 和 serving0 只有在需要在线部署时才需要配置，仅离线部署时不需要配置。

4.4 例子

1）单 party 部署

Party A（configurations.sh）：

```
#!/bin/bash
user=app
dir=/data/projects/fate
mysqldir=/data/projects/common/mysql/mysql-8.0
javadir=/data/projects/common/jdk/jdk1.8
venvdir=/data/projects/fate/venv
redisip=(192.168.0.1)
redispass=fate_dev
partylist=(10000)
JDBC0=(192.168.0.1 eggroll_meta root fate_dev)
fateflowdb0=(192.168.0.1 fate_flow root fate_dev)
```

```
iplist=(192.168.0.1)
iplist0=(192.168.0.1)
fateboard0=(192.168.0.1)
eggautocompile=true

fedlist0=(192.168.0.1)
meta0=(192.168.0.1)
proxy0=(192.168.0.1)
roll0=(192.168.0.1)
egglist0=(192.168.0.1)
tmlist0=(192.168.0.1)
fllist0=(192.168.0.1)

serving0=(192.168.0.1)
exchangeip=
```

Party B（configurations.sh）：

```
#!/bin/bash
user=app
dir=/data/projects/fate
mysqldir=/data/projects/common/mysql/mysql-8.0
javadir=/data/projects/common/jdk/jdk1.8
venvdir=/data/projects/fate/venv
redisip=(192.168.0.2)
redispass=fate_dev
partylist=(9999)
JDBC0=(192.168.0.2 eggroll_meta root fate_dev)
fateflowdb0=(192.168.0.2 fate_flow root fate_dev)

iplist=(192.168.0.2)
iplist0=(192.168.0.2)
fateboard0=(192.168.0.2)
eggautocompile=true

fedlist0=(192.168.0.2)
meta0=(192.168.0.2)
proxy0=(192.168.0.2)
roll0=(192.168.0.2)
egglist0=(192.168.0.2)
tmlist0=(192.168.0.2)
fllist0=(192.168.0.2)

serving0=(192.168.0.2)
```

```
exchangeip=
```

2）partyA+partyB 同时部署

```
#!/bin/bash
user=app
dir=/data/projects/fate
mysqldir=/data/projects/common/mysql/mysql-8.0
javadir=/data/projects/common/jdk/jdk1.8
venvdir=/data/projects/fate/venv
redisip=(192.168.0.1 192.168.0.2)
redispass=fate_dev
partylist=(10000 9999)
JDBC0=(192.168.0.1 eggroll_meta root fate_dev)
JDBC1=(192.168.0.2 eggroll_meta root fate_dev)
fateflowdb0=(192.168.0.1 fate_flow root fate_dev)
fateflowdb1=(192.168.0.1 fate_flow root fate_dev)

iplist=(192.168.0.1 192.168.0.2)
iplist0=(192.168.0.1 192.168.0.2)
fateboard0=(192.168.0.1)
fateboard1=(192.168.0.2)
eggautocompile=true

fedlist0=(192.168.0.1)
fedlist1=(192.168.0.2)
meta0=(192.168.0.1)
meta1=(192.168.0.2)
proxy0=(192.168.0.1)
proxy1=(192.168.0.2)
roll0=(192.168.0.1)
roll1=(192.168.0.2)
egglist0=(192.168.0.1)
egglist1=(192.168.0.2)
tmlist0=(192.168.0.1)
tmlist1=(192.168.0.2)
fllist0=(192.168.0.1)
fllist1=(192.168.0.2)

serving0=(192.168.0.1)
serving1=(192.168.0.2)
exchangeip=
```

按照上述配置含义修改 configurations.sh 文件对应的配置项后，然

后执行 auto-packaging.sh 脚本：

cd /data/projects/FATE/cluster-deploy/scripts

bash auto-packaging.sh

继续在 FATE/cluster-deploy/scripts 目录下执行部署脚本：

cd /data/projects/FATE/cluster-deploy/scripts

bash auto-deploy.sh

5.配置检查

执行后可到各个目标服务器上进行检查对应模块的配置是否准确，每个模块的对应配置文件所在路径可在此配置文件下查看 cluster-deploy/doc。

6.启动和停止服务

6.1 启动服务

ssh 登录各个节点 app 用户下，进入/data/projects/fate 目录下执行以下命令启动所有服务：

cd /data/projects/fate

sh services.sh all start

如果该服务器是 serving-server 节点，则还需要：

cd /data/projects/fate/serving-server

sh service.sh start

如果该服务器是 fate_flow 节点，则还需要：

cd /data/projects/fate/python/fate_flow

sh service.sh start

说明：若目标环境无法安装 c++环境，则可将 services.sh 文件中的 storage-serivice-cxx 替换为 storage-serivice 再启动即可使用 java 版本的 storage-service 模块。

6.2 检查服务状态

查看各个服务进程是否启动成功:

cd /data/projects/fate

sh services.sh all status

如果该服务器是 serving-server 节点，则还需要:

cd /data/projects/fate/serving-server

sh service.sh status

如果该服务器是 fate_flow 节点，则还需要:

cd /data/projects/fate/python/fate_flow

sh service.sh status

6.3 关机服务

若要关闭服务则使用:

cd /data/projects/fate

sh services.sh all stop

如果该服务器是 serving-server 节点，则还需要:

cd /data/projects/fate/serving-server

sh service.sh stop

如果该服务器是 fate_flow 节点，则还需要:

cd /data/projects/fate/python/fate_flow

sh service.sh stop

注：若有对单个服务进行启停操作则将上述命令中的 all 替换为相应的模块名称

即可，若要启动 fate_flow 和 Serving-server 两个服务则需要到对应目录/data/projects/fate/python/fate_flow 和/data/projects/fate/serving-server 下执行 sh service.sh start。

7.测试

7.1 单机测试

使用 ssh 登录到每个节点 app 用户，输入/data/projects/fate 目录来执行：

source /data/projects/fate/venv/bin/activate

export PYTHONPATH=/data/projects/fate/python

cd $PYTHONPATH

sh ./federatedml/test/run_test.sh

请参阅“确定”字段以指示操作成功。在其他情况下，如果失败或卡住，则表示失败，程序应在一分钟内生成结果。

7.2 Toy_example 部署验证

要运行测试，您需要设置 3 个参数：guest_partyid，host_partyid，work_mode。对于独立版本：

work_mode 为 0. guest_partyid 和 host_partyid 应该相同，并且对应于运行测试的 partyid。

对于分布式版本：

work_mode 为 1，guest_partyid 和 host_partyid 应对应于您的分布式版本设置。

请注意分发版测试只在 guest 9999:192.168.0.2 进行

将不同版本的正确值传递给以下命令，然后运行：

export PYTHONPATH = /data /projects/fat /python

source /data/projects/fate/venv/bin/activate

cd /data/projects/fate/python/examples/toy_example/

python run_toy_example.py 9999 10000 1

测试结果将显示在屏幕上。

7.3 最小化测试

快速模式

在 guest 和 host 部分的节点中，根据需要在 run_task.py 中设置字段：guest_id，host_id，arbiter_id。

该文件位于/data/projects/fate/python/examples/min_test_task /中。

在 Host 的节点 192.168.0.1 中，运行：

export PYTHONPATH = /data/projects/fate/python

source /data/projects/fate/venv/bin/activate

cd /data/projects/fate/python/examples/min_test_task /

sh run.sh host fast

从测试结果中获取“host_table”和“host_namespace”的值，并将它们传递给以下命令。

在 Guest 的节点：192.168.0.2 中，运行：

export PYTHONPATH = /data/projects/fate/python

source /data/projects/fate/venv/bin/activate

cd /data/projects/fate/python/examples/min_test_task/

sh run.sh guest fast $ {host_table} $ {host_namespace}

等待几分钟，看到结果显示“成功”字段，表明操作成功。在其他情况下，如果失败或卡住，则表示失败。

正常模式

只需在所有命令中将“fast”替换为“normal”，其余部分与快速模式相同。

7.4. Fateboard testing

Fateboard 是一项 Web 服务。 获取 fateboard 的 ip。 如果成功启动了 fateboard 服务，则可以通过访问 http://${fateboard-ip}:8080 来查看任务信息。