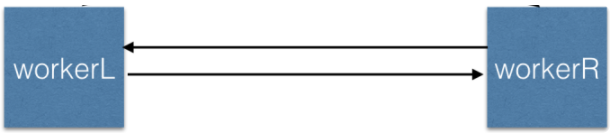
Tutorials

AntChain-MPC是一个一个隐私保护计算系统。其中的
MORSE-STF模块是一个基于tensorflow的隐私计算系统，它可以在保护用户私有数据的前提下对多方数据进行联合计算。包括算术运算，逻辑运算，序运算等基础运算，以及基于基础运算功能之上的机器学习。

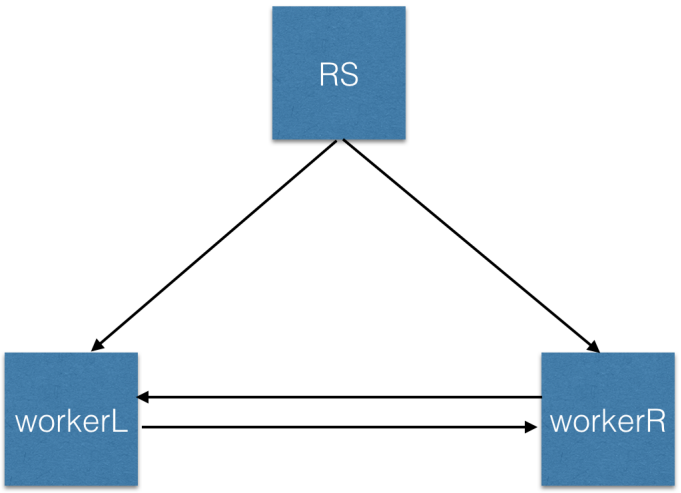
网络拓扑

MORSE-STF可以支持三方协议或双方协议。当使用双方协议时，网络拓扑如下：



由workerL和workerR进行密态运算。

当使用三方协议时，网络拓扑如下：



其中，RS (Random Service) 生成伪随机数并发送给workerL和workerR，由workerL和workerR进行密态运算。

使用双方协议或双方协议，由`\conf\config.json`文件中的parties字段指定，当使用双方协议时，parties字段应当设定为2；当使用三方协议时，parties字段应设定为3.

无论使用双方协议或三方协议，机器的ip:host由`\conf\config.json`文件中的hosts字段指定。
而提交python脚本的机器称为clinet，client可以是workerL，也可以是workerR.

系统要求

MORSE-STF可运行于MAC OS 11.1+ 或 Linux version 3.10.0+

安装

MORSE-STF的安装很方便，只需要

```yaml
pip install morse-stf -i https://pypi.antfin-inc.com/simple/
```

即可完成安装。

运行

在本Tutorials中，以三方协议作为例子讲解如何运行：

local运行

建立一个工作目录 morse-stf ，并将开源代码中conf, examples, dataset目录copy到morse-stf下。

然后配置好`morse-stf\conf\config.json`文件:

JSON 复制代码

```
{
    "parties": 3,
    "hosts": {
        "workerL": "0.0.0.0:8886",
        "workerR": "0.0.0.0:8887",
        "RS": "0.0.0.0:8888"
    },
    "stf_home": "/Users/qizhi.zqz/projects/morse-stf/morse-stf",
    "prf_flag": true,
    "compress_flag": true,
    "default_fixed_point": 14,
    "ml": {
        "dataset_train": {
            "L": "dataset/xindai_xx_train.csv",
            "R": "dataset/xindai_xy_train.csv"
        },
        "dataset_predict": {
            "L": "dataset/xindai_xx_test.csv",
            "R": "dataset/xindai_xy_test.csv"
        },
        "predict_to_file": "output/predict"
    },
    "protocols":
    {
        "drelu": "log"
    }
}
```

其中，`morse-stf\conf\config.json`文件中的hosts字段需要配置本机的三个端口，而stf_home字段需要填写morse-stf工作目录的绝对路径，`ml`字段下的所有路径填写morse-stf下的相对路径。

distributed运行

distributed运行较local运行需要多一个步骤，需要在workerL, workerR, RS每台机器上先启动服务。

```
Plain Text | 复制代码
(在workerL上运行) morse-stf-server --player=workerL  --config_file=.\conf\config.json
(在workerR上运行) morse-stf-server --player=workerR --config_file=.\conf\config.json
(在RS上运行)      morse-stf-server --player=RS --config_file=.\conf\config.json
```

其中，`.\conf\config.json`文件中的hosts字段需要配置三台机器的ip:host，而stf_home_workerL, stf_home_workerR, stf_home_RS字段需要分别填写三台机器上morse-stf工作目录的绝对路径，`ml`字段下的所有路径填写morse-stf下的相对路径：

JSON 复制代码

```json
{
    "parties": 3,
    "hosts": {
        "workerL": "0.0.0.0:8886",
        "workerR": "0.0.0.0:8887",
        "RS": "0.0.0.0:8888"
    },
    "stf_home_workerL": "/Users/qizhi.zqz/projects/morse-stf/morse-stf",
    "stf_home_workerR": "/Users/qizhi.zqz/projects/morse-stf/morse-stf",
    "stf_home_RS": "/Users/qizhi.zqz/projects/morse-stf/morse-stf",
    "prf_flag": true,
    "compress_flag": true,
    "default_fixed_point": 14,
    "ml": {
        "dataset_train": {
            "L": "dataset/xindai_xx_train.csv",
            "R": "dataset/xindai_xy_train.csv"
        },
        "dataset_predict": {
            "L": "dataset/xindai_xx_test.csv",
            "R": "dataset/xindai_xy_test.csv"
        },
        "predict_to_file": "output/predict"
    },
    "protocols":
    {
        "drelu": "log"
    }
}
```

在服务正确启动时屏幕会打印类似下面的信息：（该信息为workerL上打印出来的，因此workerL的ip:hosts会显示`localhost:8886`）

```
2021-06-15 14:59:17.439153: I tensorflow/core/platform/cpu_feature_guard.cc:143] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2021-06-15 14:59:17.456491: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x7fe5b8476590 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2021-06-15 14:59:17.456525: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
2021-06-15 14:59:17.461589: I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:300] Initialize GrpcChannelCache for job RS -> {0 -> xxx.xxx.xxx.003:8886}
2021-06-15 14:59:17.461634: I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:300] Initialize GrpcChannelCache for job workerL -> {0 -> localhost:8886}
2021-06-15 14:59:17.461645: I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:300] Initialize GrpcChannelCache for job workerR -> {0 -> xxx.xxx.xxx.002:8886}
2021-06-15 14:59:17.462864: I tensorflow/core/distributed_runtime/rpc/grpc_server_lib.cc:390] Started server with target: grpc://localhost:8886
```

三台机器的服务均正确启动后，在要运行的脚本上方加入

```
from stensorflow.engine.start_server import start_clinet
start_clinet(config_file="../conf/config.json", job_name="workerR") //运行脚本从workerR上提交
```

示例

示例：双方私有矩阵乘法

我们给出一个利用stf计算L方私有矩阵x和R方私有矩阵y的矩阵乘法的例子。简明起见，这里的矩阵都由明文形式给出；实际应用时矩阵可分别从双方的磁盘上读出。

```
import tensorflow as tf
from stensorflow.basic.basic_class.private import PrivateTensor
from stensorflow.global_var import StfConfig
from stensorflow.engine.start_server import start_local_server, start_clinet
from stensorflow.random.random import random_init

start_local_server(config_file="../conf/config.json")   # local运行时
# or
# start_clinet(config_file="../conf/config.json", job_name="workerR")  # distributed运行时

x = PrivateTensor(owner='L')          #声明一个L方私有Tensor
x.load_from_tf_tensor(tf.constant([[0.1, 0.2], [-1.1, 0.9]]))   #读入
y = PrivateTensor(owner="R")           # 声明一个R方私有Tensor
y.load_from_tf_tensor(tf.constant([[1.0, 2.1], [-1.4, -2.5]]))   #读入

z = x @ y                        # 计算矩阵乘法，背后将运行一个MPC协议
tf_z = z.to_tf_tensor("R")       # 将计算结果转化为R方机器上的 普通tf.Tensor

with tf.compat.v1.Session(StfConfig.target) as sess:
    random_init(sess)                # 运行MPC协议所使用的的伪随机数生成器的初始化
    print(sess.run(tf_z))            # 打印运算结果
```

这个例子是在本地通过三个端口通信运行的。而对于distributed运行环境，在clinet上提交代码，提交的代码中把

```
start_local_server(config_file="../conf/config.json")
```

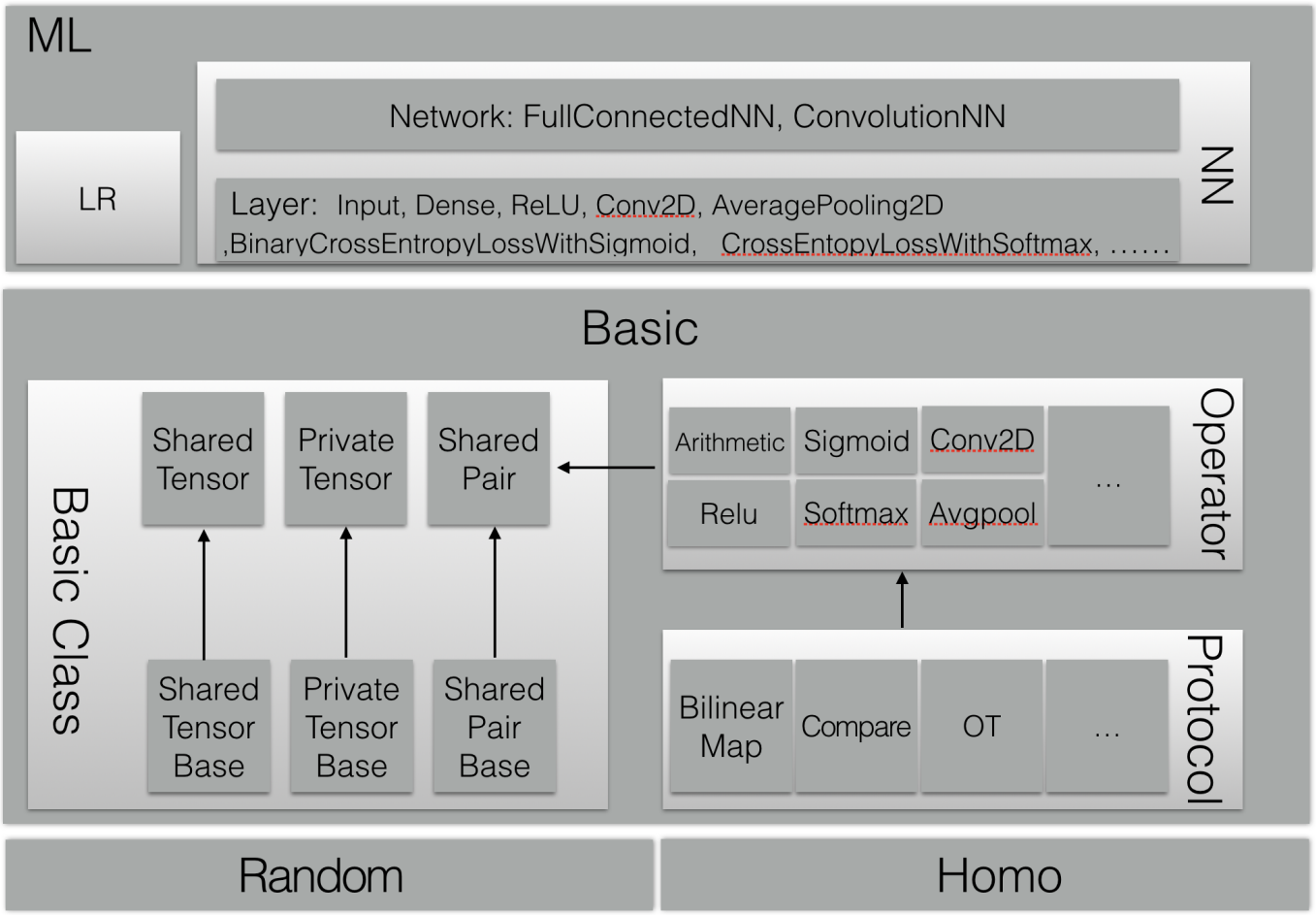
更改为

```
start_clinet(config_file="../conf/config.json", job_name="workerR")
```

该示例收录在`./examples/private_matmul.py`

架构

我们的架构如图所示



其中Random模块为伪随机数生成模块。用作伪随机数的生成。Homo是同态加密模块，用作同态加/解密及密态运算。当使用3方协议时，Homo模块不起作用。

Basic模块包括Basic Class, Protocol和Operator三个子模块。

其中Basic Class子模块中实现了三个基本类：SharedTensor, PrivateTensor和 SharedPair. SharedTensor表示数据分片，PrivateTensor表示单方私有数据，SharedPair表示由双方各持一个数据分片表示的密文数据。SharedTensor, PrivateTensor和 SharedPair三个类的基类分别是SharedTensorBase, PrivateTensorBase和 SharedPairBase, 子类与基类的区别在于运算符重载：基类只实现了不需要MPC协议就可以实现的运算符，而子类实现了需要运行MPC协议才能实现的运算符。

Protocol子模块实现双线性映射，OT, 比较等MPC协议。

Operator子模块建立在Protocol子模块之上，利用Protocol子模中实现的MPC协议来构建Basic Class中运算符的重载函数。

ML模块基于Basic模块构建机器学习的训练、预测功能。其中

LR模块为逻辑回归，NN模块为神经网络。

NN模块中的Layer子模块实现了神经网络的各种Layer，如Input，Dense，ReLU等；而Network子模块实现了全连接神经网络和卷积神经网络。

常见问题

1. Too many open file错误

```
2021-09-30 10:31:18.166344: I tensorflow/core/distributed_runtime/rpc/grpc_server_lib.cc:390] Started server with target: grpc://localhos
t:8887
E0930 10:31:18.166532000 123145607254016 wakeup_fd_pipe.cc:40]         pipe creation failed (24): Too many open files
E0930 10:31:18.166617000 123145607254016 ev_poll_posix.cc:942]         pollset_work: {"created":"@1632969078.166589000","description":"To
o many open files","errno":24,"file":"external/com_github_grpc_grpc/src/core/lib/iomgr/wakeup_fd_pipe.cc","file_line":41,"os_error":"Too
many open files","syscall":"pipe"}
E0930 10:31:18.166641000 123145607254016 completion_queue.cc:1044]     Completion queue next failed: {"created":"@1632969078.166589000","
description":"Too many open files","errno":24,"file":"external/com_github_grpc_grpc/src/core/lib/iomgr/wakeup_fd_pipe.cc","file_line":41,
"os_error":"Too many open files","syscall":"pipe"}
E0930 10:31:18.166646000 123145607790592 wakeup_fd_pipe.cc:40]         pipe creation failed (24): Too many open files
E0930 10:31:18.166724000 123145607790592 ev_poll_posix.cc:942]         pollset_work: {"created":"@1632969078.166711000","description":"To
o many open files","errno":24,"file":"external/com_github_grpc_grpc/src/core/lib/iomgr/wakeup_fd_pipe.cc","file_line":41,"os_error":"Too
many open files","syscall":"pipe"}
zsh: segmentation fault  python examples/private_matmul.py
(base) fuchen@B-5CZXMD6M-0114 morse-stf %
```

可尝试增大系统最大打开文件数:

JSON | 复制代码

```
ulimit -n 10240
```