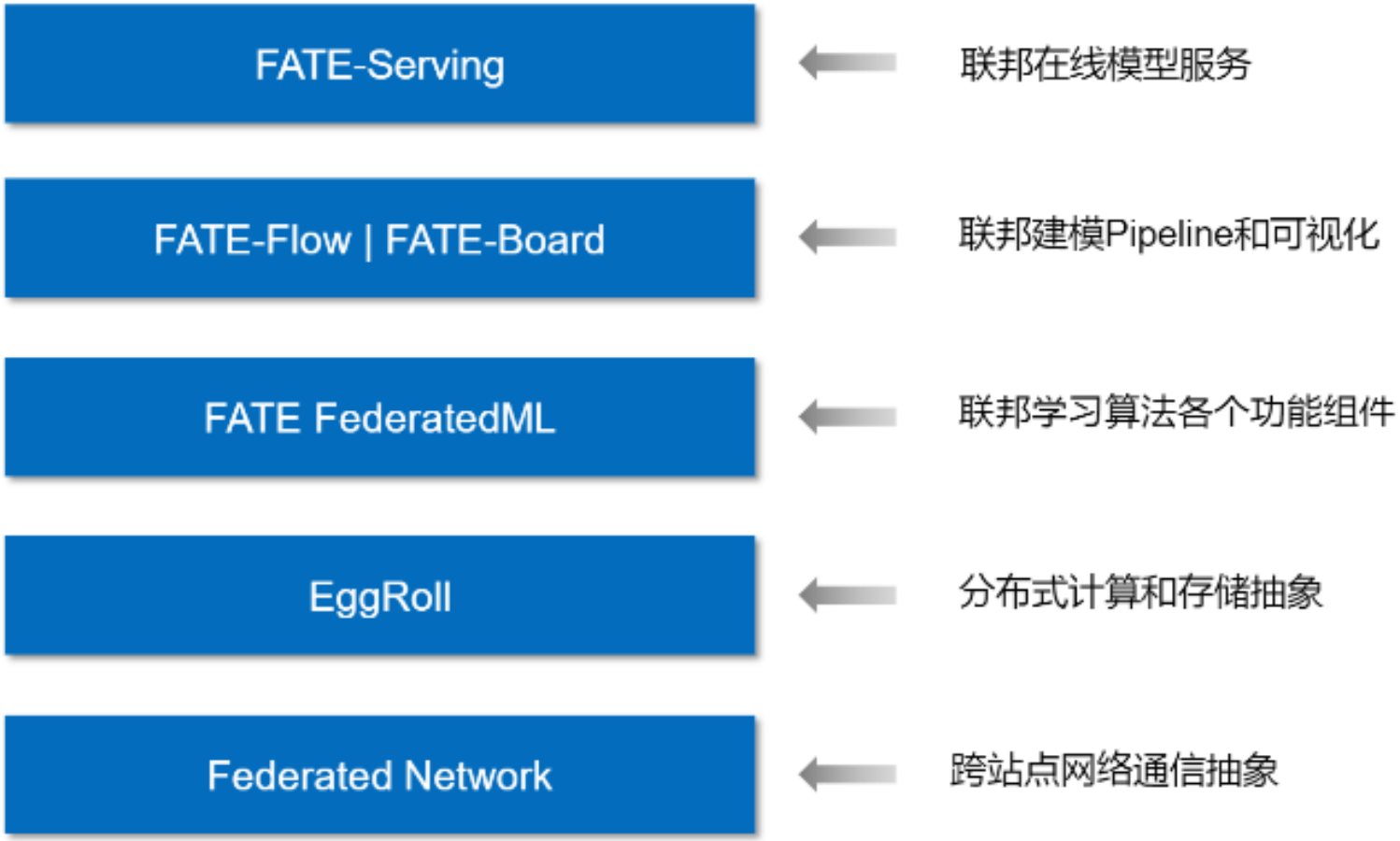
Fate

赵春玲

目录

CONTENTS

Fate框架结构


FATE-Serving
联邦在线模型服务
FATE-Flow | FATE-Board
联邦建模Pipeline和可视化
FATE FederatedML
联邦学习算法各个功能组件
EggRoll
分布式计算和存储抽象
Federated Network
跨站点网络通信抽象

FederatedML

机器学习算法

DataIO

Intersect

Federated Sampling

Feature Scale

Hetero Feature Binning

OneHot Encoder

Hetero Feature Selection

Hetero LR

Homo LR

Hetero Secure Boosting

Evalution

工具

支持联合学习的工具，如加密工具、统计模块、参数定义和传输变量自动生成器等

Fate-Flow

DAG定义联邦学习Pipeline

多方非对称Pipeline DAG、通用json格式DAG DSL、DSL-Parser

联邦任务协同调度

多方任务队列管理、协同分发任务、任务一致性保证、多方状态同步等

联邦模型管理

联邦模型存取、联邦模型一致性、版本管理、发布管理等

联邦任务生命周期管理

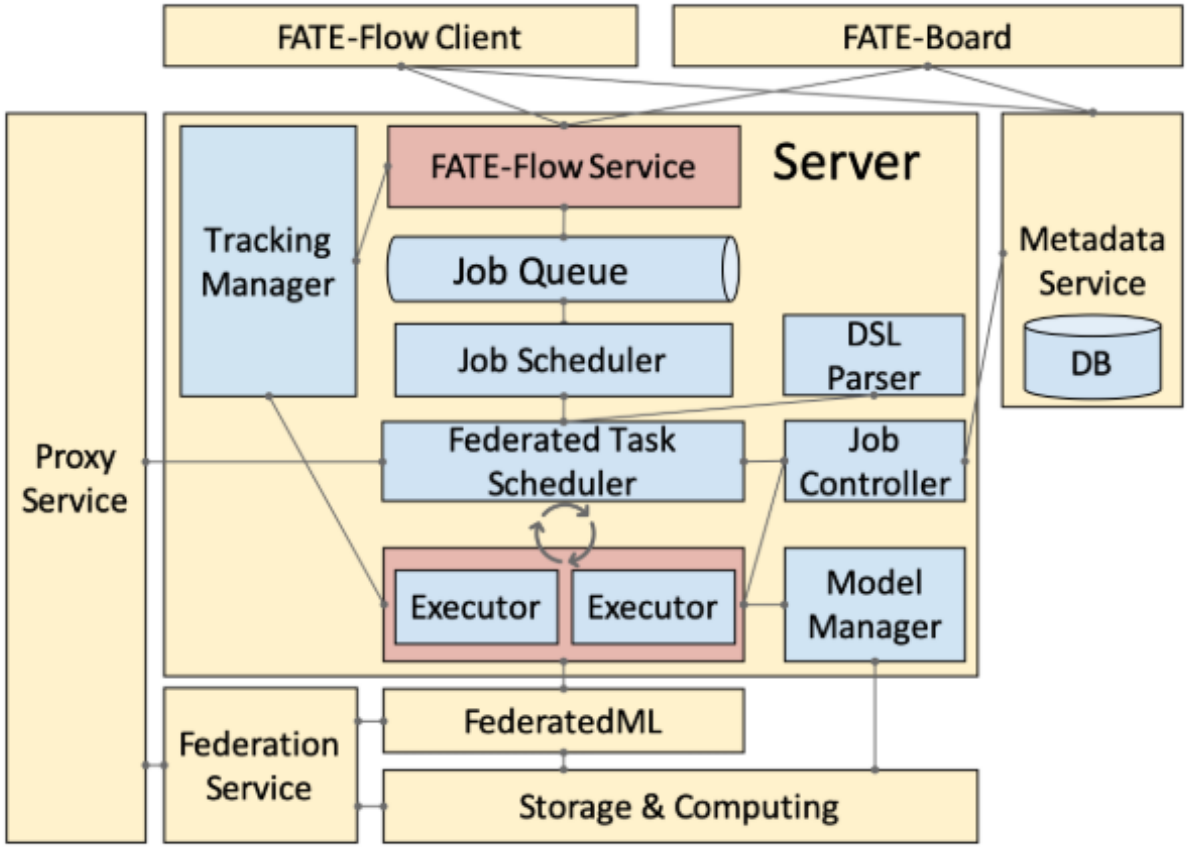
多方启停、状态检测等

联邦任务输入输出实时追踪

数据、模型、自定义指标、日志等实时记录存储

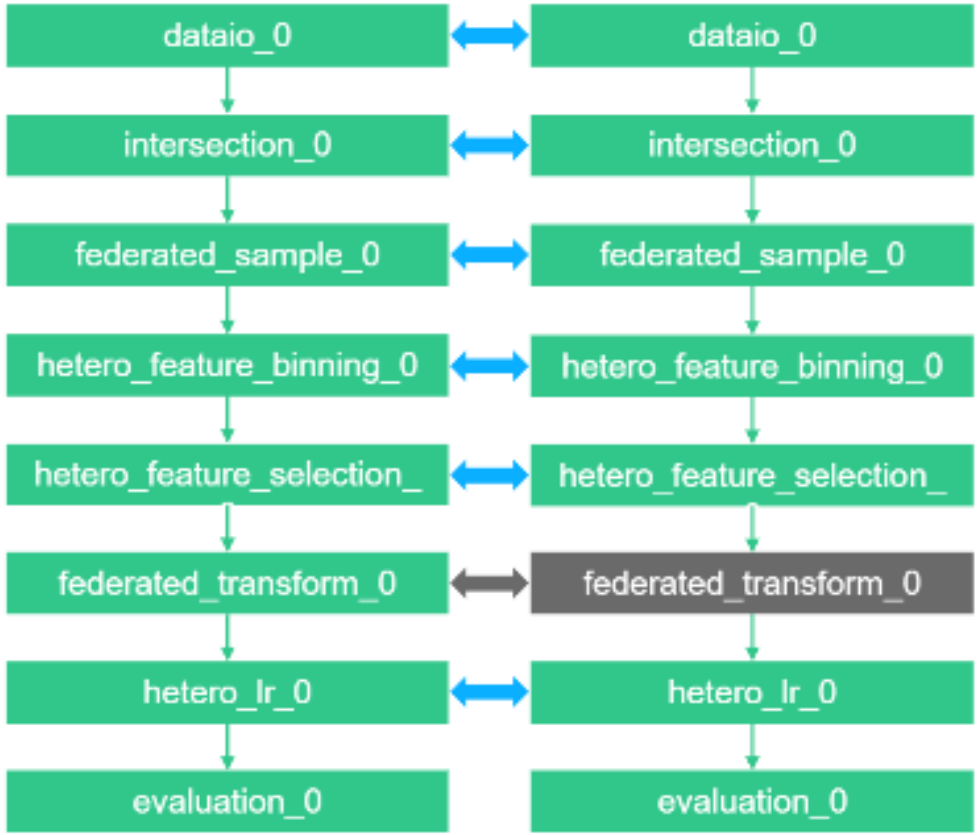
FATE-Flow架构

DSL Parser：调度的核心，包含上下游关系、依赖等
Job Scheduler：DAG作为一个 Job，DAG 里面的节点 run 起来后形成一个 task。
Federated Task Scheduler：task是最小调度粒度，需要调度多方运行同一个组件但参数算法不同的 task
Job Controller：联邦任务控制器
Tracking Manager：任务输入输出的实时追踪，包括每个 task 输出的 data 和 model
Model Manager：联邦模型管理器



DAG 定义联邦学习 Pipeline

```
{
  "components" : {
    "dataio_0": {
        "module": "DataIO",
        "input": {
            "data": {
                "data": [
                    "args.train_data"
                ]
            }
        },
        "output": {
            "data": ["train"],
            "model": ["dataio"]
        },
        "need_deploy": true
    },
    "intersection_0": {
        "module": "Intersection",
        "input": {
            "data": {
                "data": [
                    "dataio_0.train"
                ]
            }
        },
        "output": {
            "data": ["train"]
```



DSL定义

1. Module：选择模型组件，FATE当前支持11个模型组件。

2. Input：包括data、model和isometric_model(异构模型，目前只用于 Feature Selection)

3. Output：包括data和model

DSL Parser 解析过程

组件初始化：

根据 DSL 定义和任务配置，解析每个 Component 运行参数

分析 DSL 定义 data、model 输入输出，提取依赖关系

DAG图：

构建依赖关系邻接表

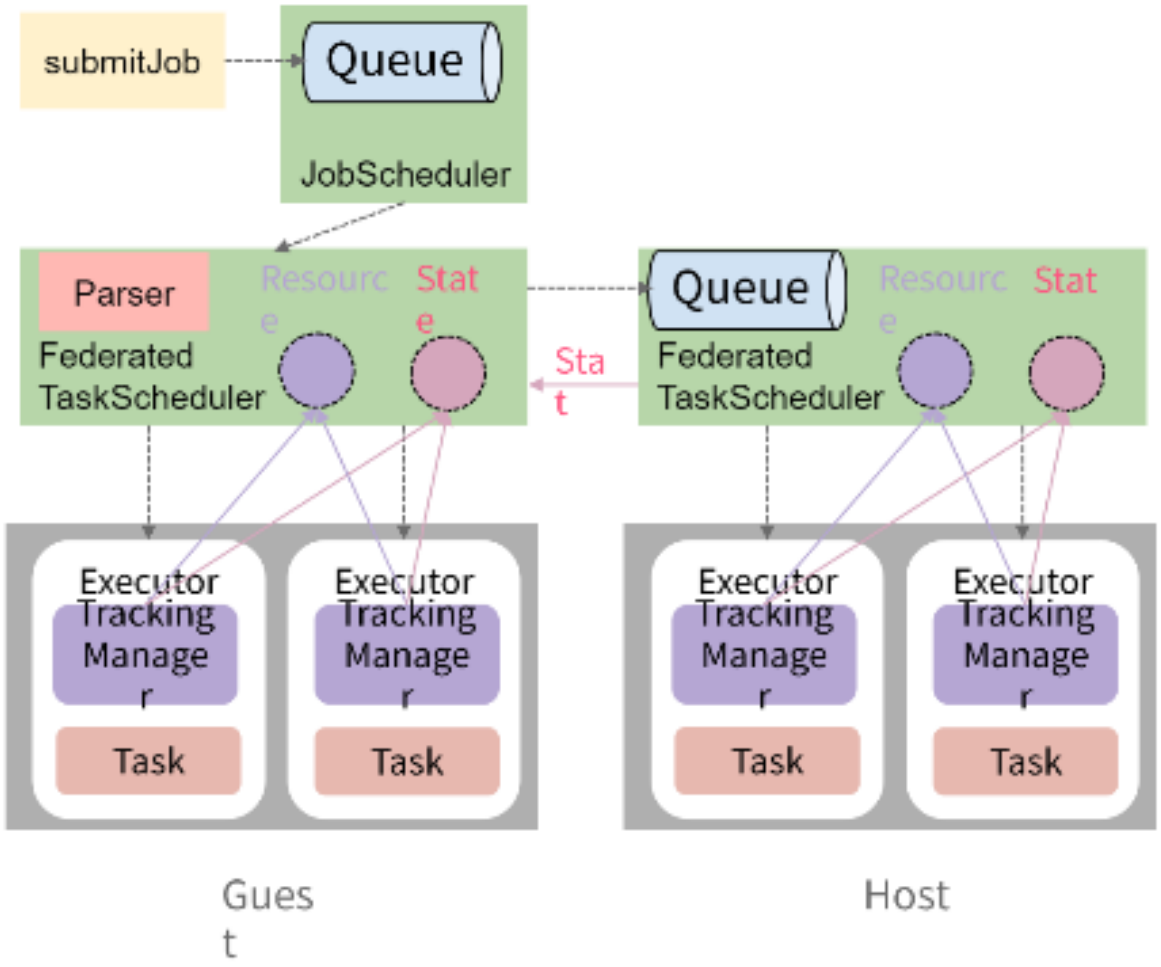
拓扑排序进行 DAG 依赖检测，排除无效的DSL

调度协作：

实时输出 Component 无依赖上下游

联邦学习任务多方协同调度

- Federated Scheduler
  - Run DAG as Job，Run Component as Task
  - Initiator，为调度控制方
  - 支持all_succss,all_done,one_success等策略
  - 支持rerun，specified_task_run等特定运行
- 一个Pipeline DAG Component Task为最小调度单位
  - Initiator JobScheduler从队列取出一个Pipeline DAG Job，分发到TaskScheduler.
  - Federated TaskScheduler 从Parser取得N个无依赖的Component，启动多个Executor 执行，并同步任务指令到联邦其他参与方
  - 联邦参与方取得任务，如果New Job，则放入队列；否则启动多个Executor执行
  - 联邦参与方定期调度队列中的Job，发起执行
  - Initiator TaskScheduler会等待收集该Task在所有方的运行状态



FATE-Flow 使用样例步骤

1、通过FATE-Flow CLI 上传数据

```
python fate_flow_client.py -f upload -c examples/upload_guest.json
python fate_flow_client.py -f upload -c examples/upload_host.json
```

2、通过 FATE-Flow CLI 提交一个 Job，需要提供 Job 的 DSL 描述以及配置文件

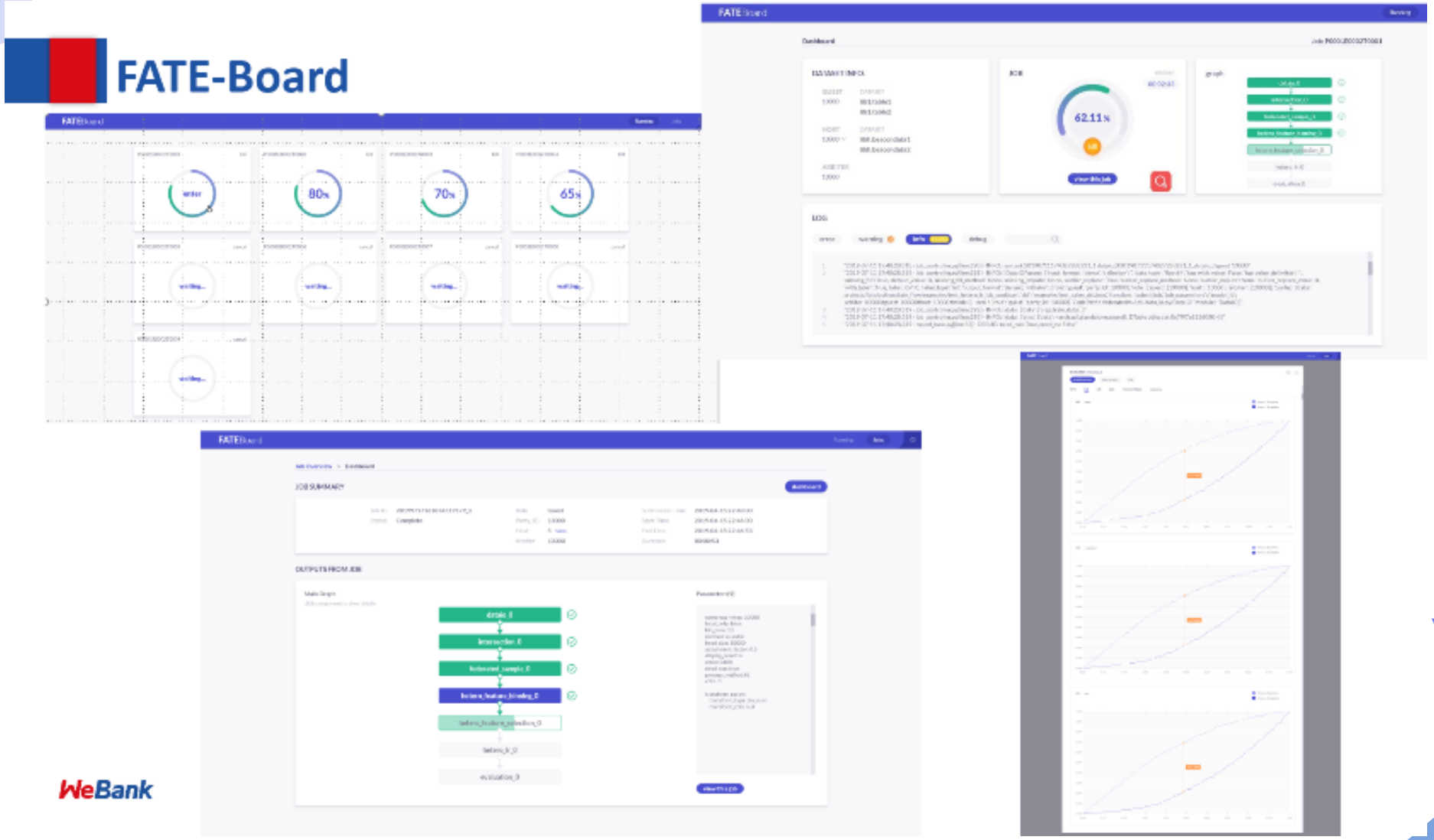
```
python fate_flow_client.py -f submit_job -d examples/test_hetero_lr_job_dsl.json -c examples/test_hetero_
```

3、FATE-Flow Server 返回该 Job 的一些必要信息


```
{
    "data": {
        "board_url": "http://localhost:8080/index.html#/dashboard?job_id=201908171821197447191 2&role=gues
        "job_dsl_path": "xxx/jobs/201908171821197447191 2/job_dsl.json",
        "job_runtime_conf_path": "xxx/jobs/201908171821197447191 2/job_runtime_conf.json",
        "model_info": {
            "model_id": "arbiter-10000#guest-10000#host-10000#model",
            "model_version": "201908171821197447191 2"
        }
    },
    "jobId": "201908171821197447191 2",
    "meta": null,
    "retcode": 0,
    "retmsg": "success"
}
```

FATEBoard

联邦学习建模可视化工具，能够可视化和度量模型训练的全过程



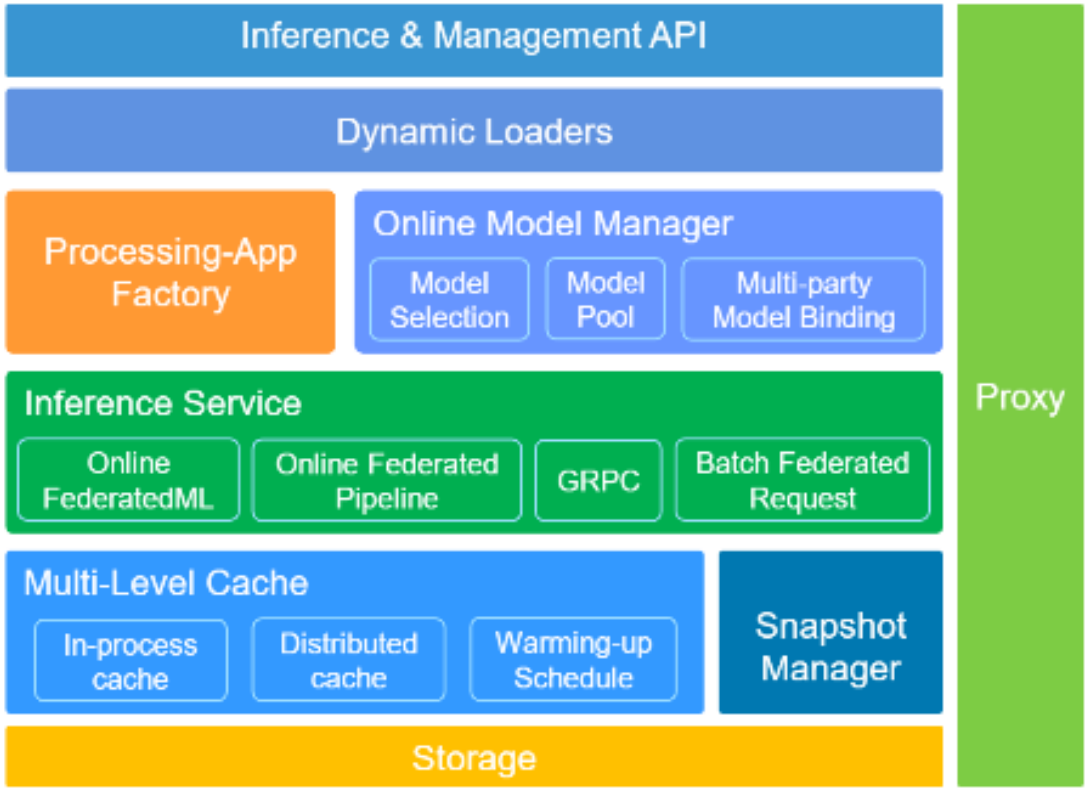
FATE-Serving

提供在线推理服务：

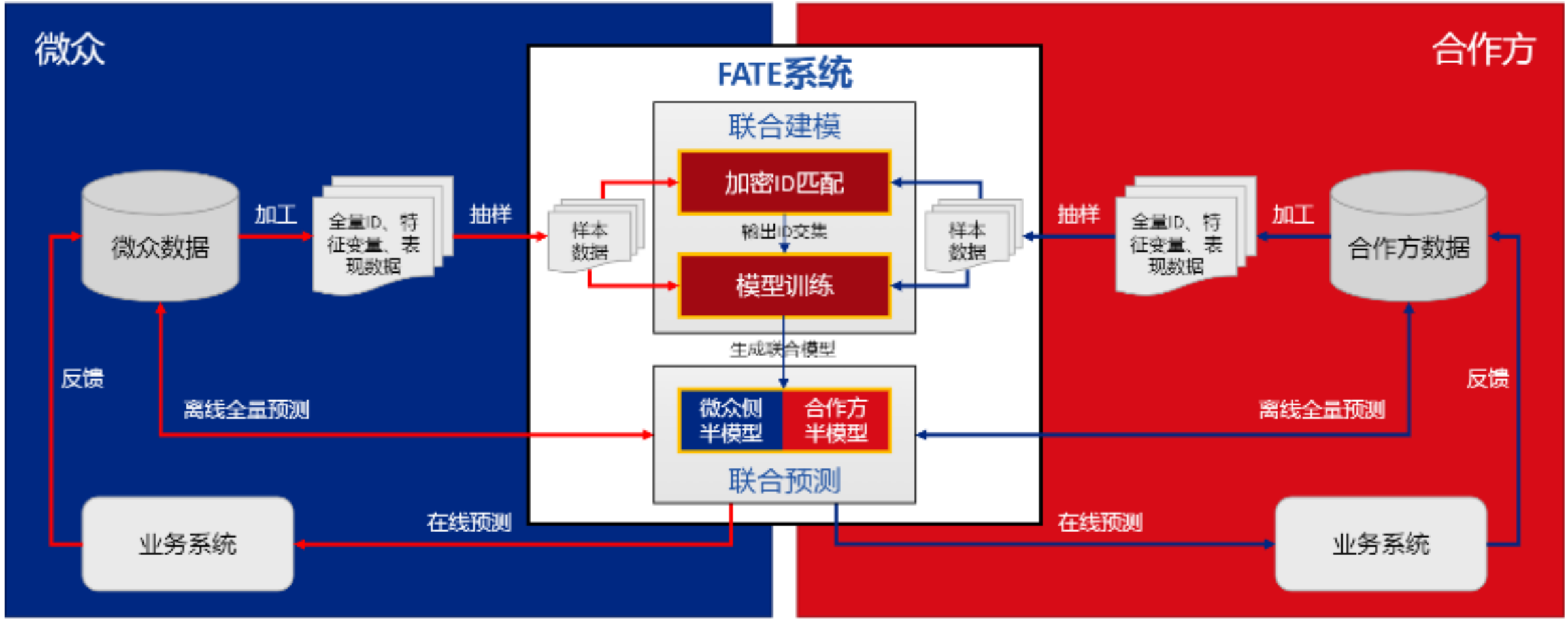
在线联邦模型管理

在线联邦推理 Pipeline

在线推理服务缓存



基于Fate的联合建模



自定义训练

1、配置上传数据配置文件

```
{
  "file": "examples/data/breast_b.csv",
  "head": 1,
  "partition": 10,
  "work_mode": 0,
  "table_name": "hetero_breast_b",
  "namespace": "hetero_guest_breast"
}
```

2、定义模型任务结构

3、为每个组件定义运行时的配置

4、开始模型任务

5、检查输出结果

6、查看输出日志

开发新模块

1、定义模型使用的参数对象

```
class LogisticParam(BaseParam):
```

2、定义新模块的设置配置

```
{
    "module_path":  "federatedml/logistic_regression/hetero_logistic_regression",
    "default_runtime_conf": "logistic_regression_param.json",
    "param_class" : "federatedml/param/logistic_regression_param.py/LogisticParam",
    "role":
    {
        "guest":
        {
            "program": "hetero_lr_guest.py/HeteroLRGuest"
        },
        "host":
        {
            "program": "hetero_lr_host.py/HeteroLRHost"
        },
        "arbiter":
        {
            "program": "hetero_lr_arbiter.py/HeteroLRArbiter"
        }
    }
}
```

3、定义模块默认运行配置

4、定义模块的传输变量json并生成传输变量对象

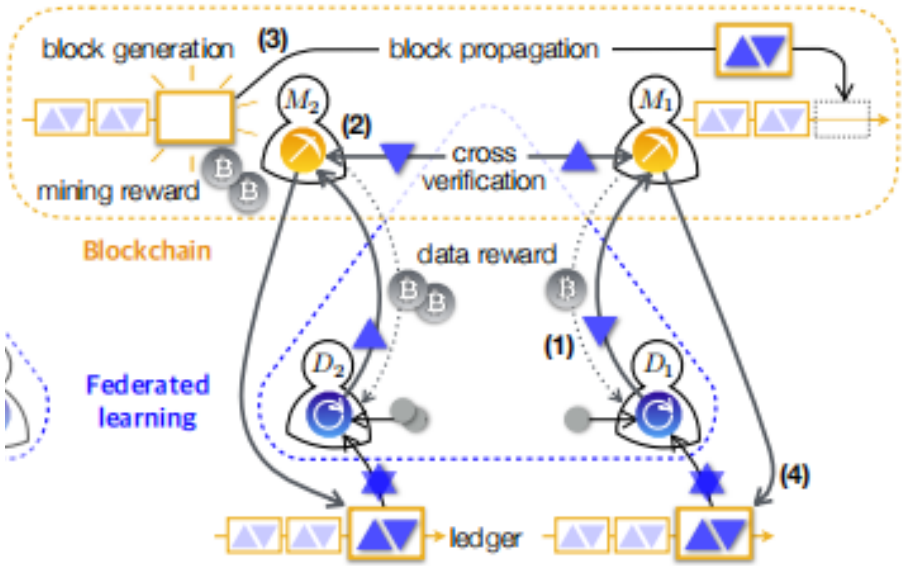
```
{
  "HeteroLRTransferVariable": {
    "paillier_pubkey": {
      "src": "arbiter",
      "dst": [
        "host",
        "guest"
      ]
    },
    "batch_data_index": {
      "src": "guest",
      "dst": [
        "host"
      ]
    },
    ...
  }
}
```

5、定义模块，要求继承model_base

联邦学习与区块链

论文：On-Device Federated Learning via Blockchain and its Latency Analysis

(1) Each device in BlockFL computes and uploads the local model update to its associated miner in the blockchain network, while in return receiving the data reward proportional to the number of its data samples from the miner.
(2) Miners exchange and verify all the local model updates, and then run the Proof-of-Work (PoW) [7].
(3) Once a miner completes the PoW, it generates a block where the verified local model updates are recorded, and receives the mining reward from the blockchain network.
(4) Finally, the generated block storing the aggregate local model updates is added to a blockchain, also known as distributed ledger, and is downloaded by the devices. Each device computes the global model update from the freshest block, which is an input of the next local model update.



Thanks