WeBank

# 构建端到端的联邦学习Pipeline生产服务

曾纪策
微众银行人工智能系统架构师
jarviszeng@webank.com

FedAI Ecosystem

https://www.fedai.org/
https://github.com/WeBankFinTech/FATE

# 目录
CONTENTS


1 研发背景介绍

2 高弹性联邦学习Pipeline调度

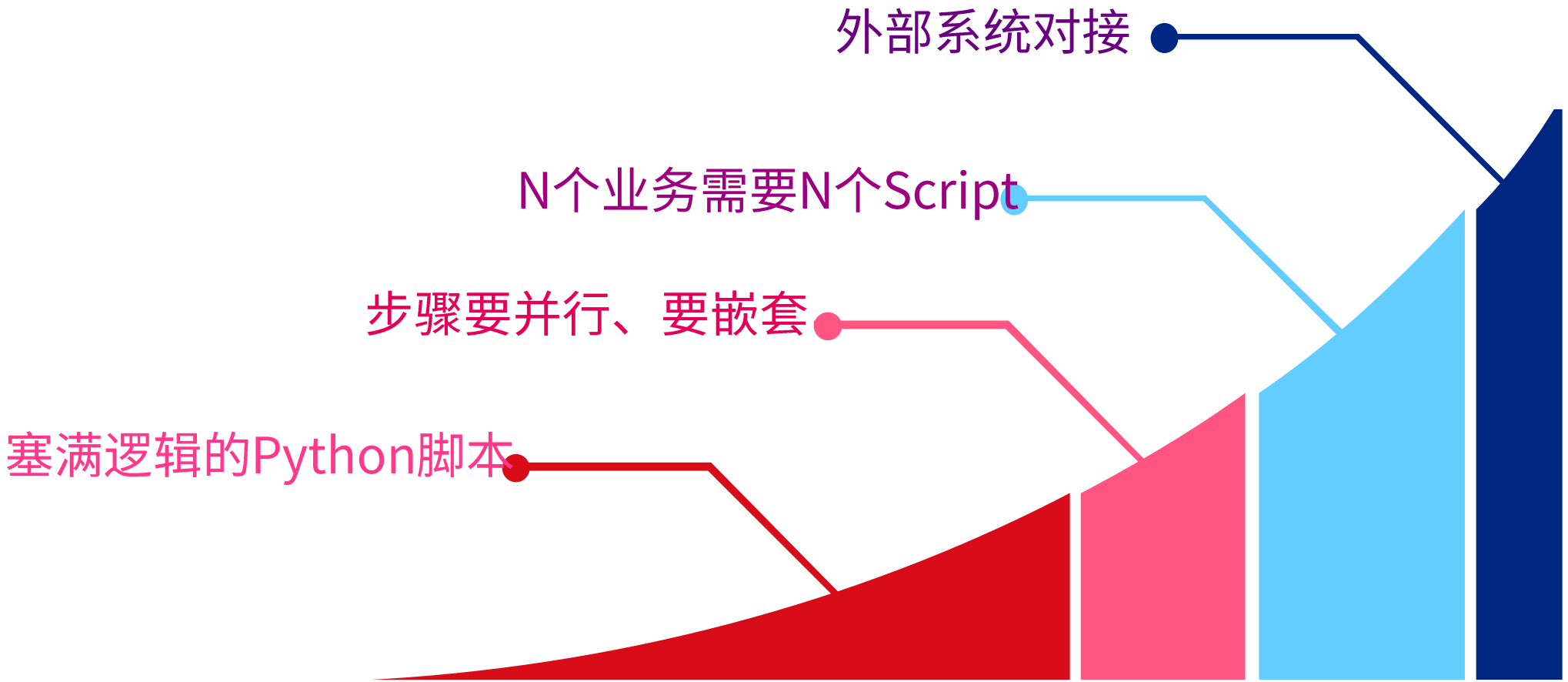
3 联邦学习任务可视化

4 高性能联邦学习在线推理服务

# 01

# 背景介绍

# 机器学习任务编排

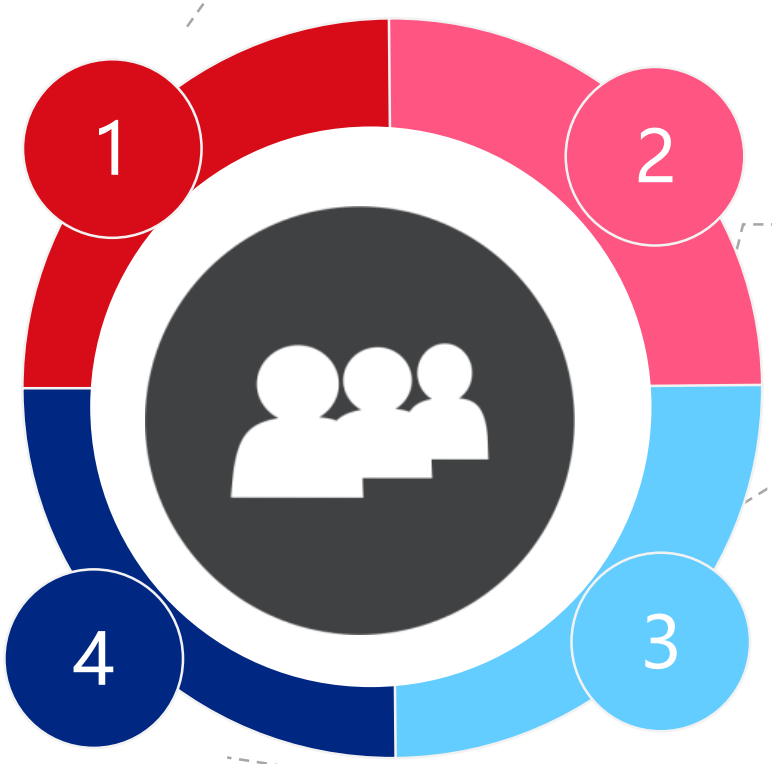


# 机器学习任务状况观察

# 有没有需求是看日志不能解决的?

## 如果有，那就加几行日志!

# 联邦学习任务协同建模




启动多方任务

各方的运行状况

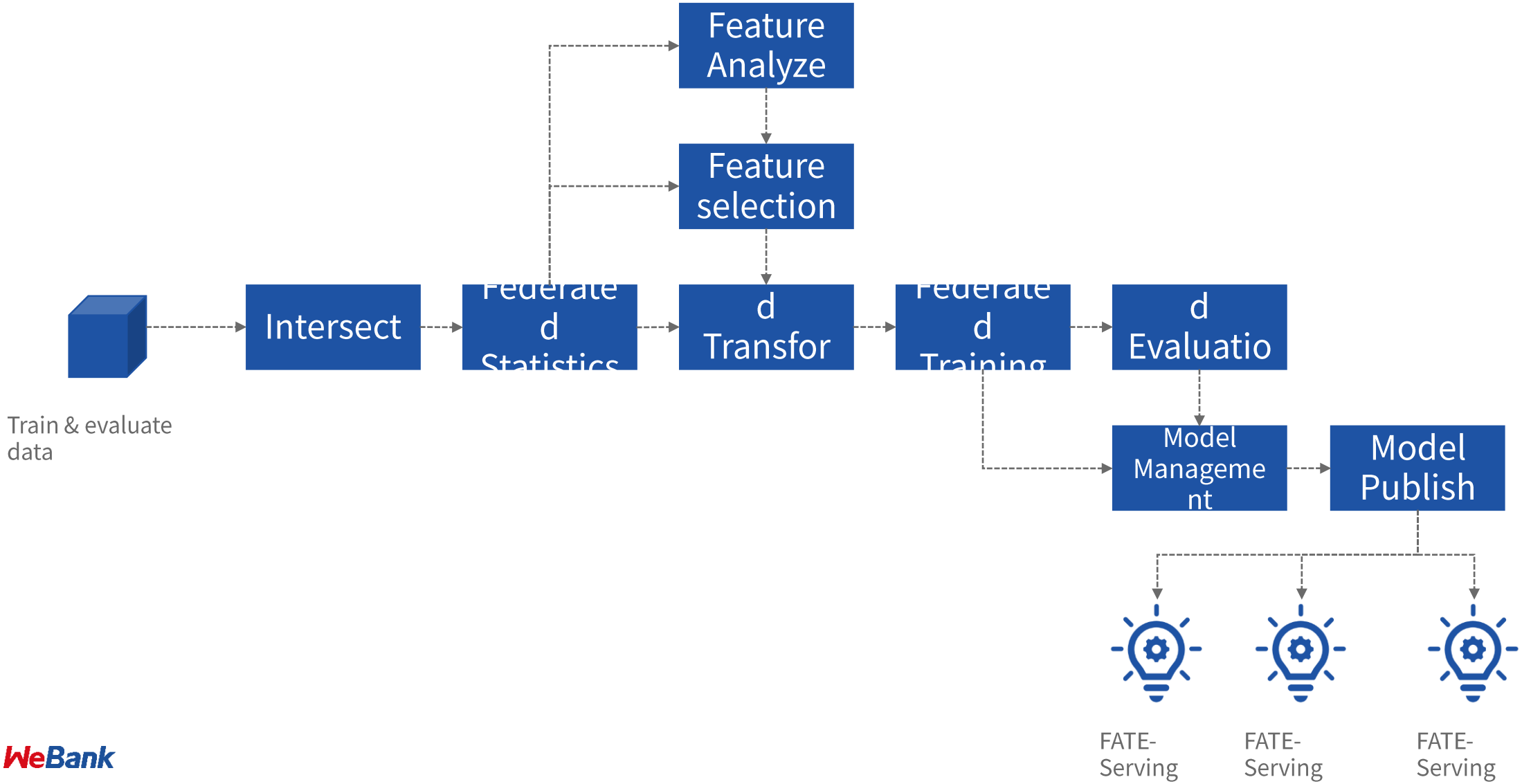
多方日志、排查问题

多方任务管理

# 02

# 高弹性联邦学习Pipeline调度

# 端到端的联邦学习Pipeline



# FATE-Flow:端到端的联邦学习Pipeline调度平台

## 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

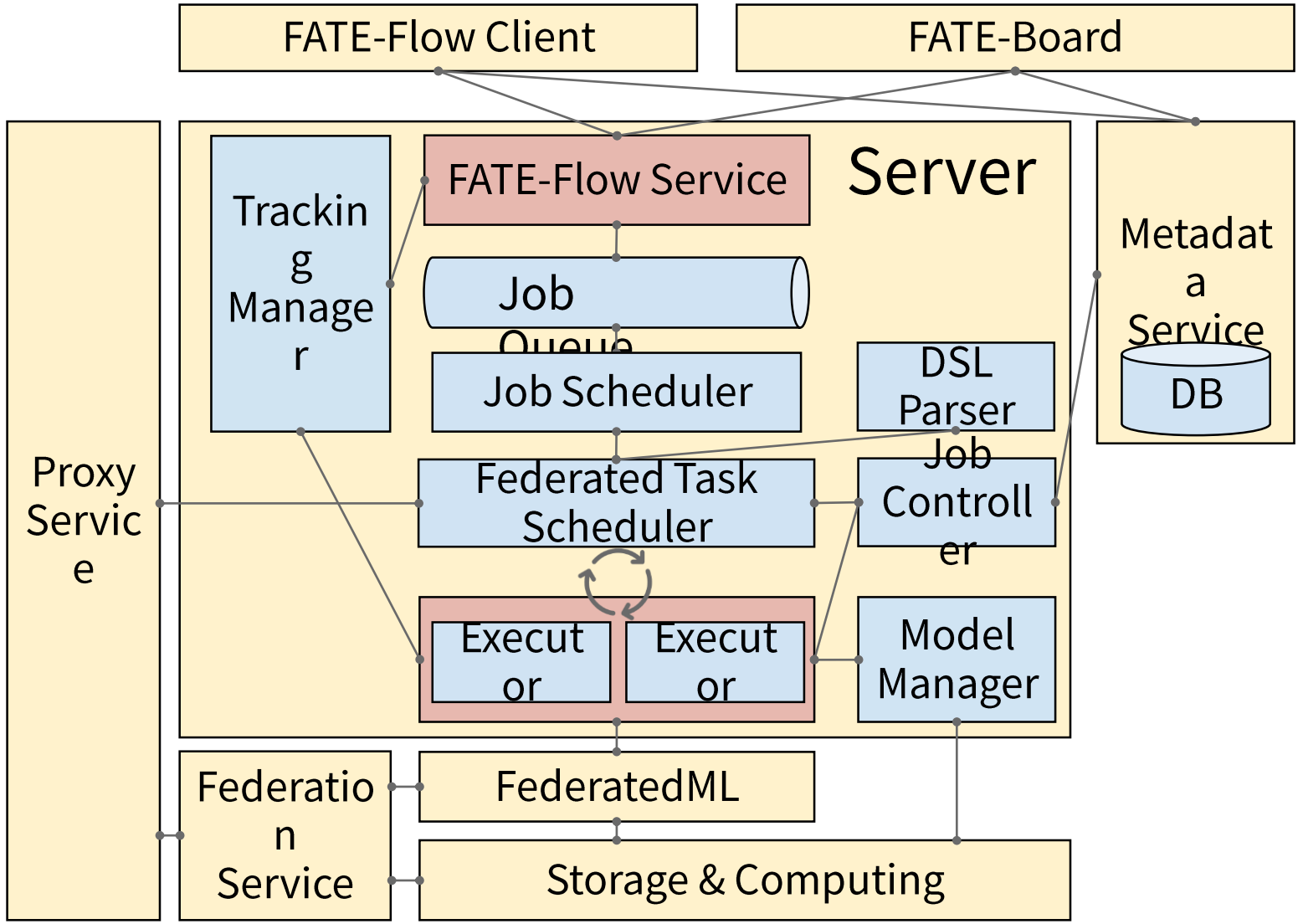
多方启停、状态检测等

## 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

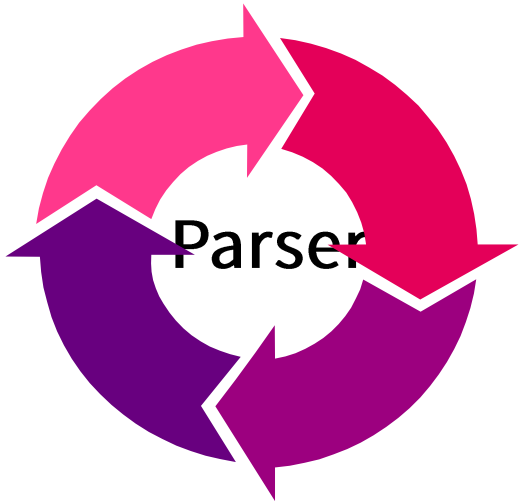
# FATE-Flow架构

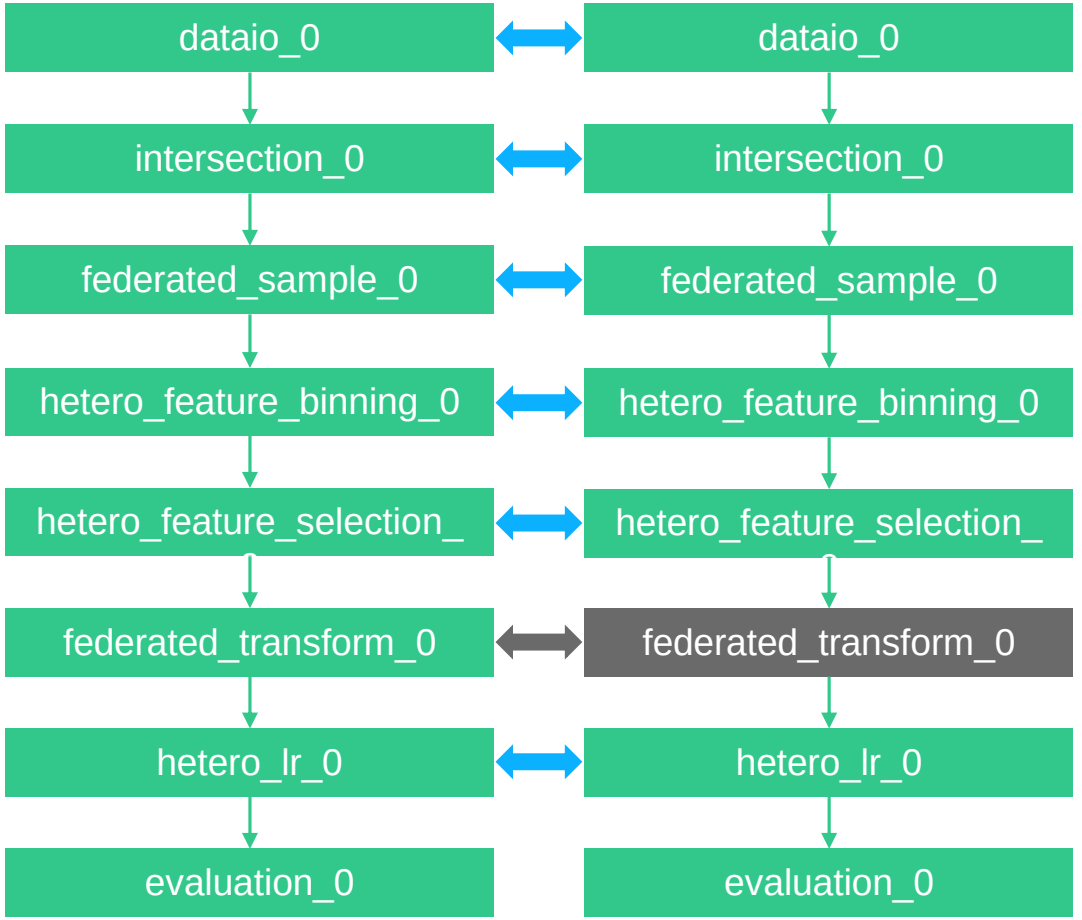
- DSL Parser: 非对称DAG DSL解析器
- Job Scheduler: 联邦任务DAG调度器
- Federated Task Scheduler: 联邦任务DAG节点协同调度中心
- Job Controller: 联邦任务控制器
- Executor: 联邦任务执行节点，Operator容器
- Tracking Manager: 任务输入输出实时追踪器
- Model Manager: 联邦模型管理器



# 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"]
```



# Runtime DSL

配置运行DSL，只需三步

- module: 模型组件，FATE当前支持11个模型组件
- Input:
  - data: 数据输入
  - model: 模型输入
  - isometric_model: 异构模型，当前只用于Feature Selection
- Output
  - data: 数据输出
  - model: 模型输出

```
"hetero_lr_0": {
    "module": "HeteroLR",
    "input": {
        "data": {
            "train_data": ["hetero_feature_selection_0.train"]
        }
    },
    "output": {
        "data": ["train"],
        "model": ["hetero_lr"]
    }
},
```

# DAG DSL Parser

1. 根据DSL定义和任务配置，解析每个Component运行参数
2. 分析DSL定义data、model输入输出，提取依赖关系

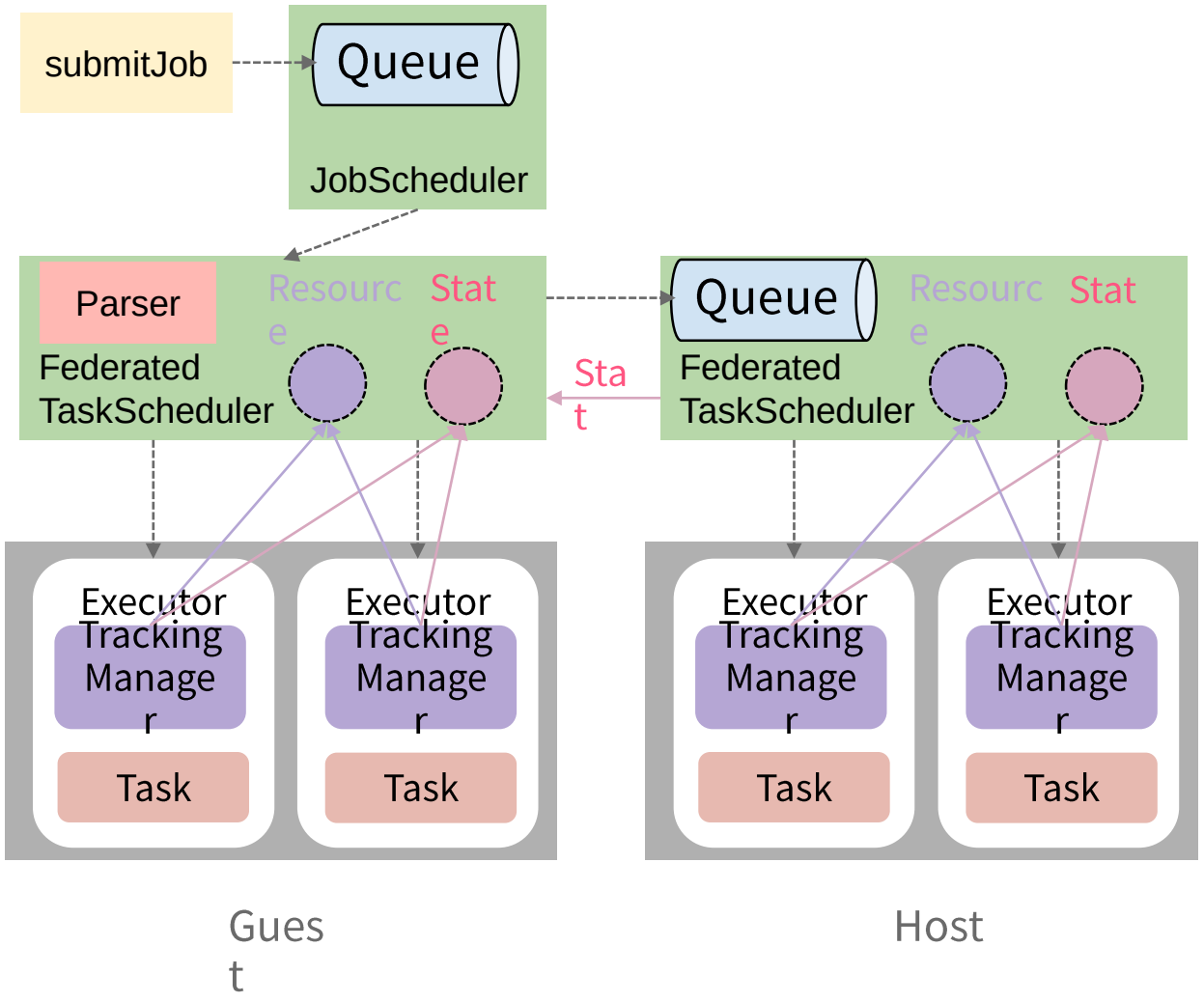
1. 构建依赖关系邻接表
2. 拓扑排序进行DAG依赖检测

1. 实时输出Component无依赖上下游
2. Component依赖度自动递减

1. 剔除预测阶段无用Component数据，模型依赖传递，推导预测DSL

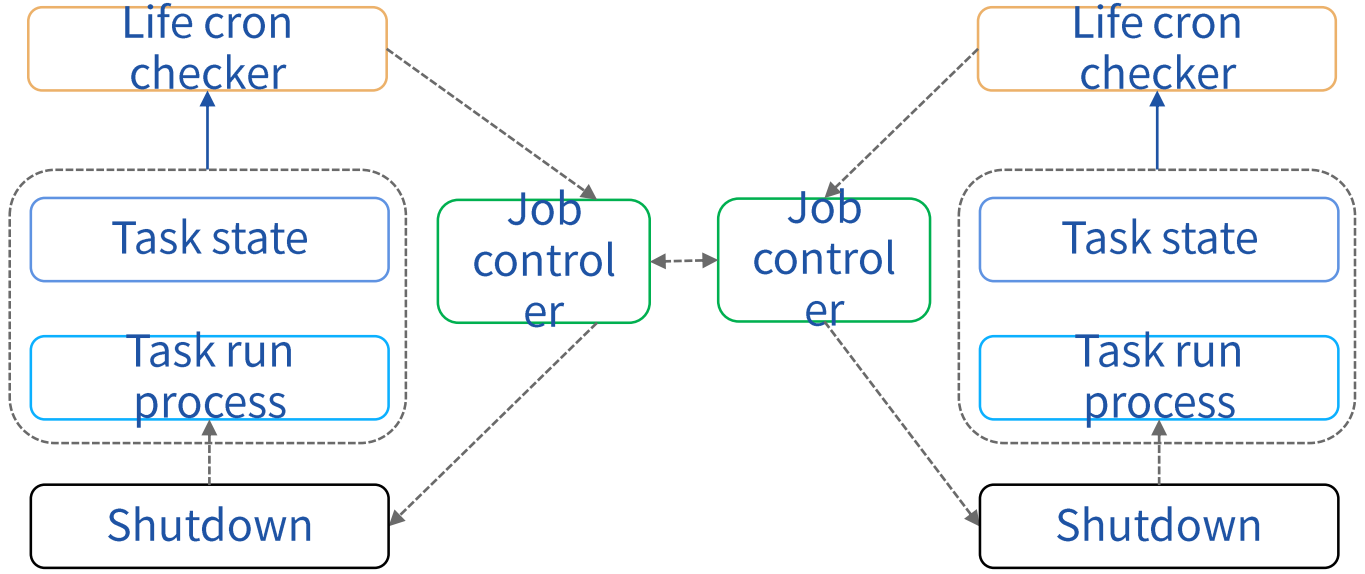
# 联邦学习任务多方协同调度

- 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



# 联邦任务多方生命周期管理

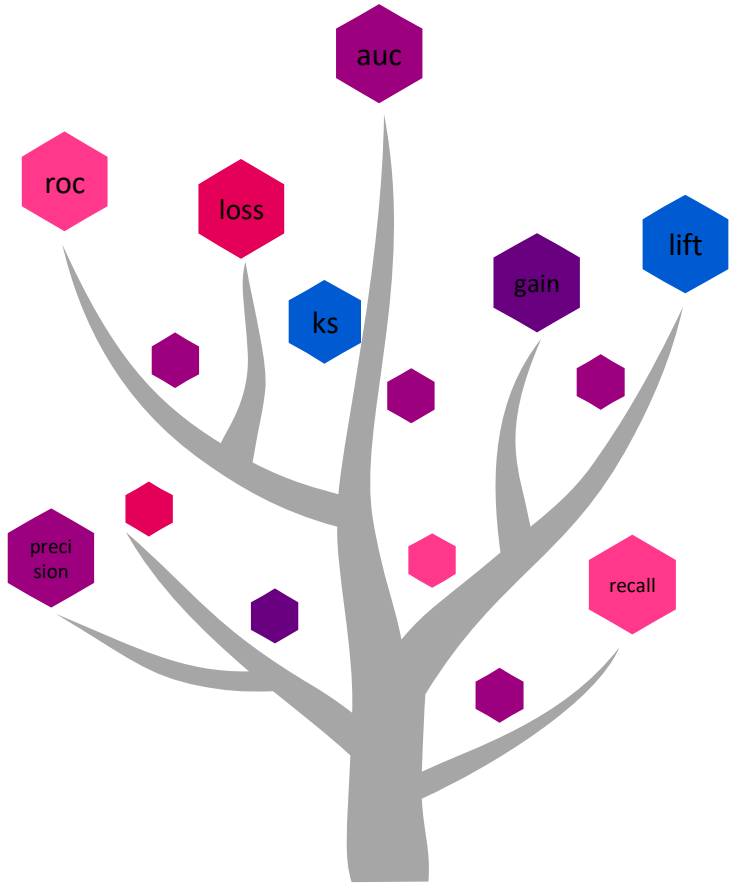
- Task stat: Task状态信息，如启动时间、运行状态、结束时间、超时时间等
- Task run process: Task运行进程
- Life cron checker: Task生命周期定时检测
- Job Controller: 联邦任务控制器
- 若Task运行时间超过配置超时时间或默认超时时间(一般较长)，启动Shutdown
- 若Task运行进程异常终止，启动Shutdown
- 若Task正常运行终止，启动Shutdown
- Shutdown流程: kill process、清理任务以及同步指令到所有联邦参与方，保证联邦任务状态一致性



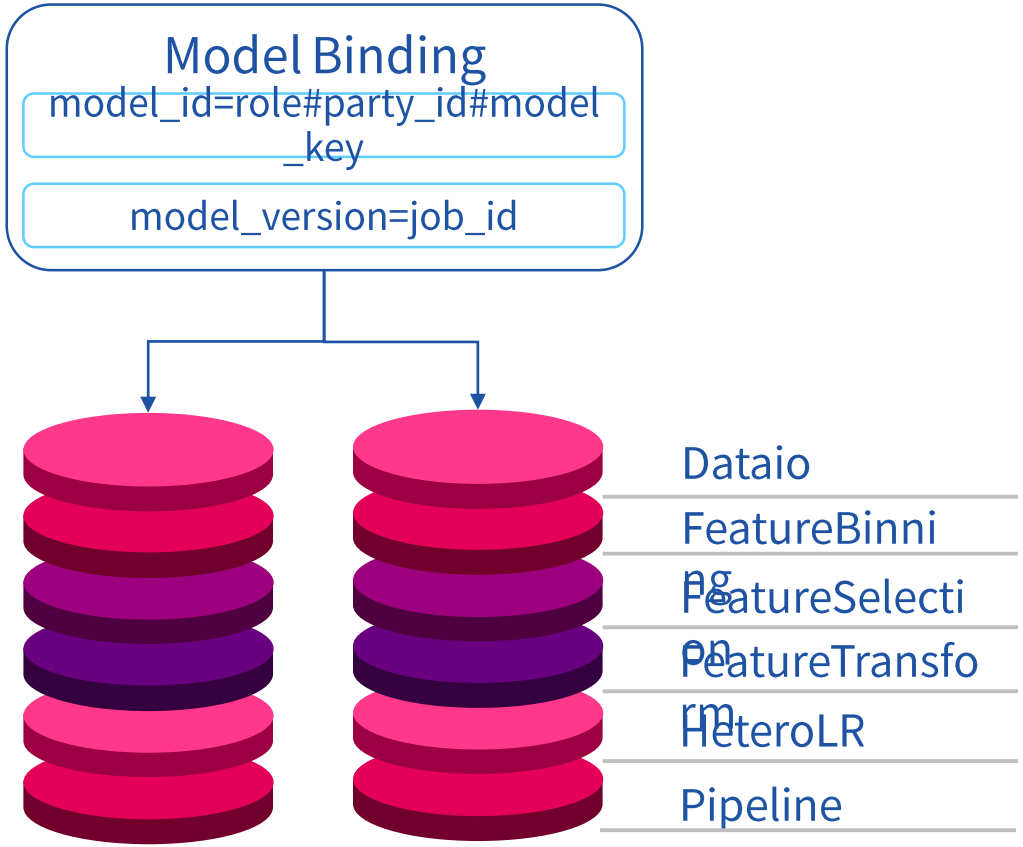
# 联邦任务输入输出实时追踪

- Definition
  - metric type: 指标类型，如auc, loss, ks等等
  - metric namespace: 自定义指标命名空间，如train, predict
  - metric name: 自定义指标名称，如auc0，hetero_lr_auc0
  - metric data: key-value形式的指标数据
  - metric meta: key-value形式的指标元信息，支持灵活画图
- API
  - log_metric_data(metric_namespace, metric_name, metrics)
  - set_metric_meta(metric_namespace, metric_name, metric_meta)
  - get_metric_data(metric_namespace, metric_name)
  - get_metric_meta(metric_namespace, metric_name)

VS



# 联邦模型管理



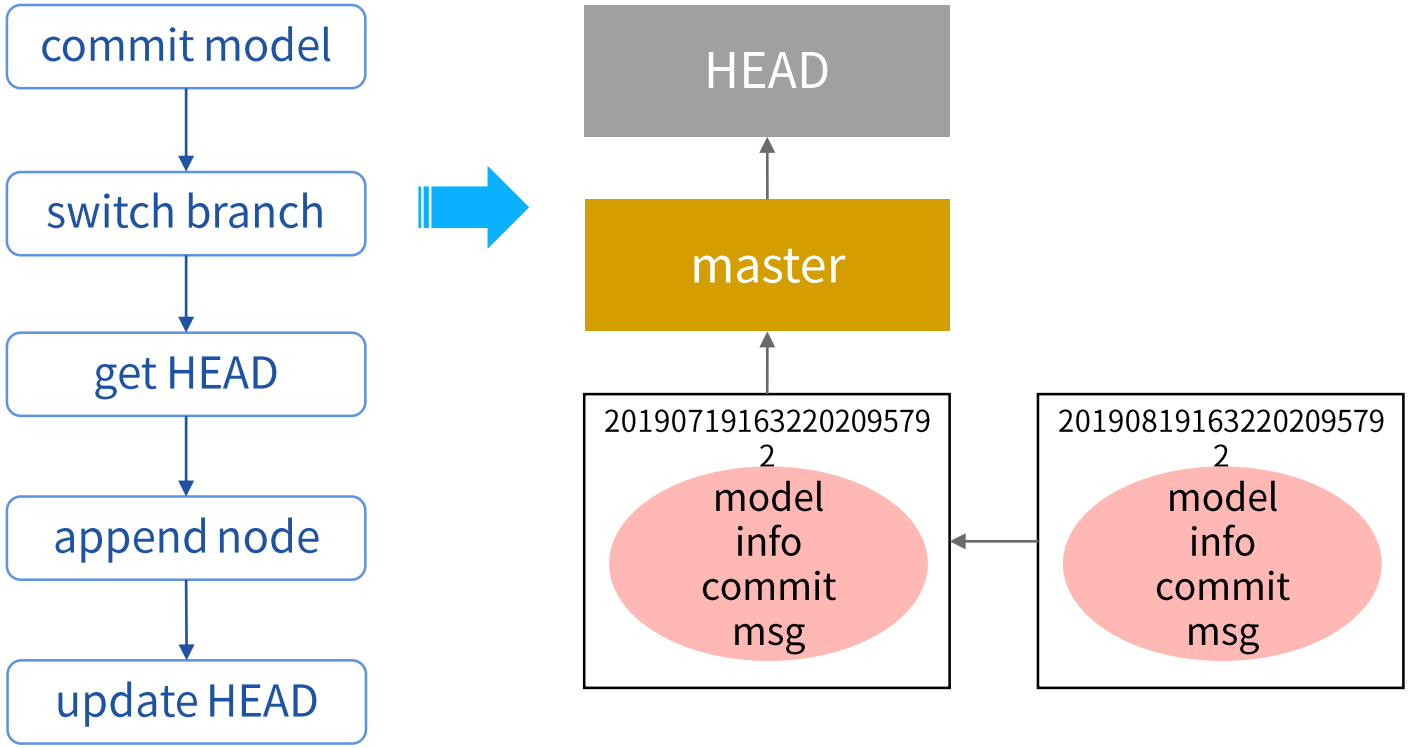

- 使用Google Protocol Buffer作为模型存储协议，利用跨语言共享
- 每个算法模型由两部分组成：ModelParam & ModelMeta
- 一个Pipeline产生一系列算法模型
- 命名为Pipeline的模型存储Pipeline建模DSL及在线推理DSL
- 联邦学习下，需要保证所有参与方模型一致性，即模型绑定
- model_key为用户提交任务时定义的模型标识
- 联邦各方的模型ID由本方标识信息role、party_id，加model_key
- 联邦各方的模型版本必须唯一且保持一致，FATE-Flow直接设置为job_id

```
syntax = "proto3";

package com.webank.ai.fate.common.mlmodel.buffer;
option java_outer_classname = "FeatureSelectionParamProto";


message FeatureValue {
    map<string, double> feature_values = 1;
}

message LeftCols {
    repeated string original_cols = 1;
    map<string, bool> left_cols = 2;
}

message FeatureSelectionFilterParam{
    map<string, double> feature_values = 1;
    map<string, FeatureValue> host_feature_values = 2;
    LeftCols left_cols = 3;
    map<string, LeftCols> host_left_cols = 4;
    string filter_name = 5;
}

message FeatureSelectionParam{
    repeated FeatureSelectionFilterParam results = 1;
    LeftCols final_left_cols = 2;
}
```

```
syntax = "proto3";

package com.webank.ai.fate.common.mlmodel.buffer;
option java_outer_classname = "FeatureSelectionMetaProto";

message FeatureSelectionMeta {
    repeated string filter_methods = 1;
    bool local_only = 2;
    repeated string cols = 3;
    UniqueValueMeta unique_meta = 4;
    IVValueSelectionMeta iv_value_meta = 5;
    IVPercentileSelectionMeta iv_percentile_meta = 6;
    VarianceOfCoeSelectionMeta variance_coe_meta = 7;
    OutlierColsSelectionMeta outlier_meta = 8;
    bool need_run = 9;
}

message UniqueValueMeta {
    double eps = 1;
}

message IVValueSelectionMeta{
    double value_threshold = 1;
}

message IVPercentileSelectionMeta{
    double percentile_threshold = 1;
}

message VarianceOfCoeSelectionMeta {
    double value_threshold = 1;
}

message OutlierColsSelectionMeta {
    double percentile = 1;
    double upper_threshold = 2;
}
```

# 联邦模型版本管理

- 基于多叉树的版本记录
- 支持commit message
- 支持分支功能，如experiment, product, release
- 支持tag，如release
- 支持history查看
- 支持版本回溯



# 使用样例

```
python fate_flow_client.py -f submitJob -d examples/test_hetero_lr_job_dsl.json -c
examples/test_hetero_lr_job_conf.json
python fate_flow_client.py -f jobStatus -j 20190720105134389931_1
python fate_flow_client.py -f kill -j 20190720105134389931_1
```

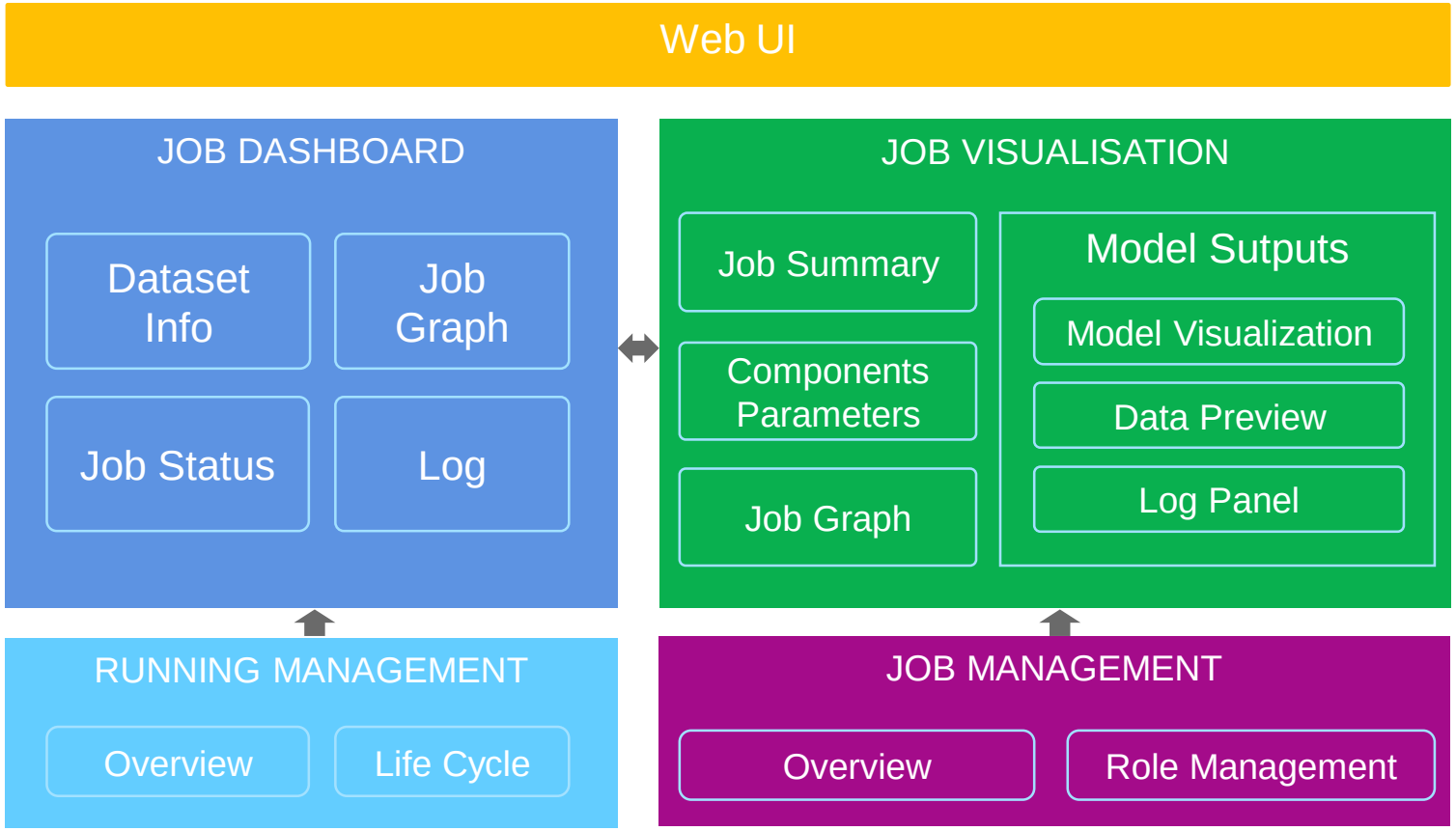

```
{
    "data": {
        "job_dsl_path": "/data/projects/fate/python/jobs/20190720105134389931_1/job_dsl.json",
        "job_runtime_conf_path": "/data/projects/fate/python/jobs/20190720105134389931_1/job_runtime_conf.json",
        "job_url": {
            "arbiter": [
                "http://[redacted]:8080/#/details?job_id=20190720105134389931_1&role=arbiter&party_id=10000"
            ],
            "guest": [
                "http://[redacted]:8080/#/details?job_id=20190720105134389931_1&role=guest&party_id=10000"
            ],
            "host": [
                "http://[redacted]:8080/#/details?job_id=20190720105134389931_1&role=host&party_id=10000"
            ]
        },
        "model_id": {
            "arbiter": [
                "arbiter#10000#arbiter-10000#guest-10000#host-10000#model"
            ],
            "guest": [
                "guest#10000#arbiter-10000#guest-10000#host-10000#model"
            ],
            "host": [
                "host#10000#arbiter-10000#guest-10000#host-10000#model"
            ]
        },
        "model_version": "20190720105134389931_1"
    },
    "jobId": "20190720105134389931_1",
    "meta": null,
    "retcode": 0,
    "retmsg": "success",
    "created_at": "2019-07-20 10:51:34"
}
```

# 03

# 联邦学习建模可视化

# FATE-Board

FATE-Board作为FATE联邦建模的可视化工具，旨在跟踪和记录联邦建模全过程的信息，并通过可视化的方式呈现模型训练过程的变化以及模型训练结果，帮助用户简单而高效地深入探索模型与理解模型。



# 建模交互及可视化

1. 用户配置pipeline，建立graph、定义parameters等；
2. 用户提交job，返回job URL，同时启动job运行，进入web端查看fateboard；
3. 观察job运行状态，查看运行时的统计信息，包括运行进度、日志、模型过程输出等；
4. 查看job运行完成的结果，包括模型输出、模型评分、日志等内容及可视化结果；

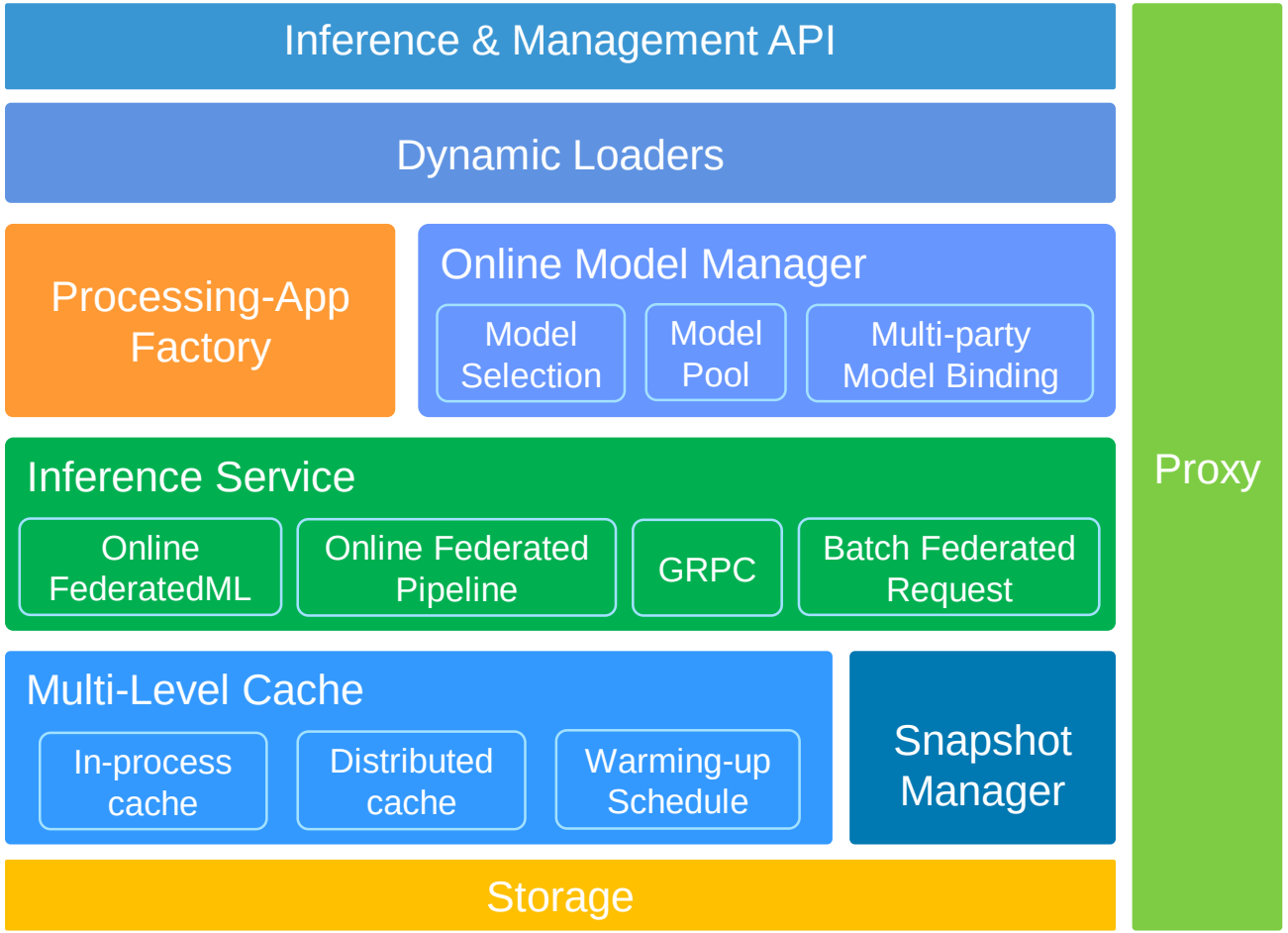
FATEBoard

# 04

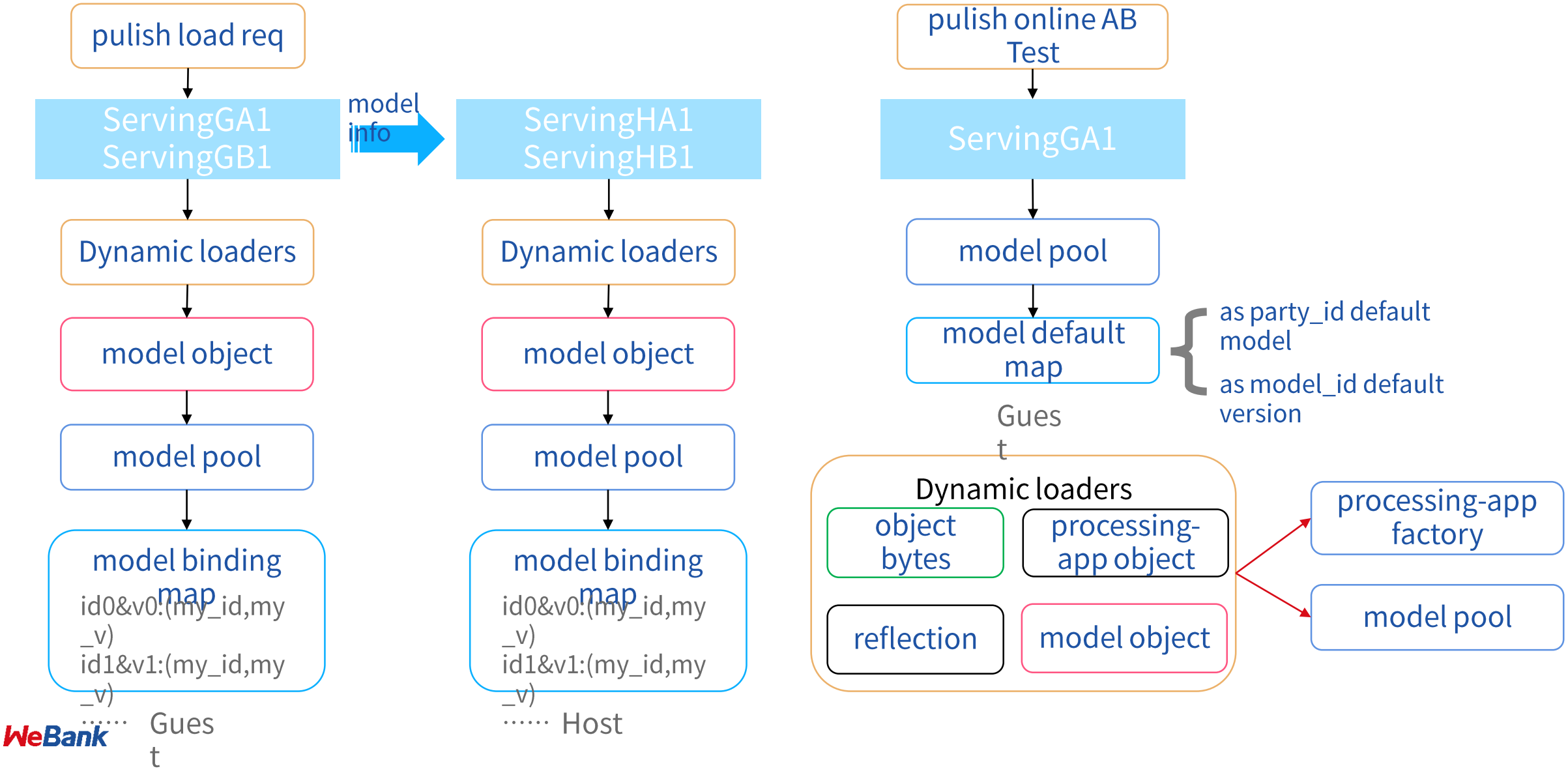
# 高性能联邦学习在线推理服务

# FATE-Serving

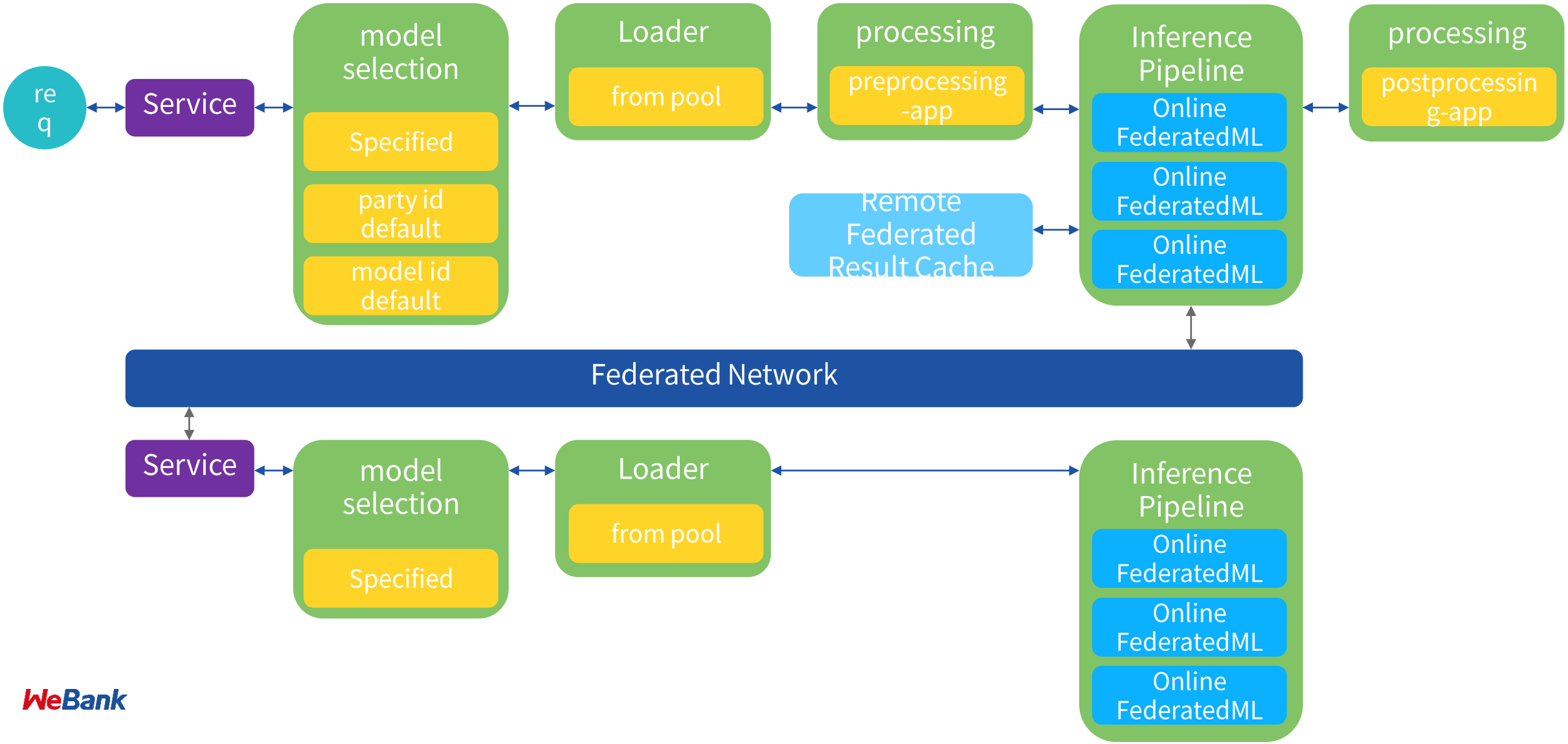
- 设计原则
  - 高性能，基于GRPC协议，批量联邦请求，联邦参与方模型结果多级缓存
  - 高可用，无状态设计，异常降级功能
  - 高弹性，模型&数据处理App动态加载
- 模块
  - Online FederatedML: 高性能在线联邦学习算法包
  - Online Federated Pipeline: 在线联邦推理Pipeline
  - Dynamic Loaders: 推理节点动态加载器，包括Model、Processing-App
  - Model Manager: 在线模型管理器
  - Processing-App Factory:数据处理节点工厂
  - Multi-Level Cache: 多级缓存管理器
  - Snapshot Manager: 快照管理，定期将在线模型、Processing-App信息落库



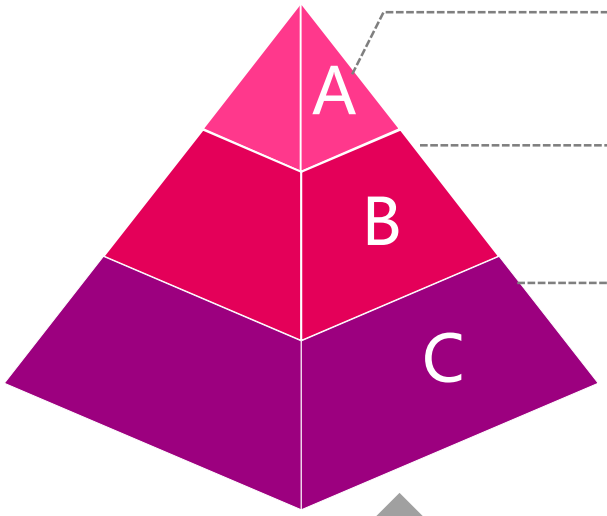
# 在线联邦模型管理


pulish load req
ServingGA1
ServingGB1
model info
ServingHA1
ServingHB1
Dynamic loaders
model object
model pool
model binding map
id0&v0:(my_id,my_v)
id1&v1:(my_id,my_v)
……
Gues t
Host
pulish online AB Test
ServingGA1
model pool
model default map
as party_id default model
as model_id default version
Gues t
Dynamic loaders
object bytes
processing-app object
reflection
model object
processing-app factory
model pool

# 在线联邦推理Pipeline



# 在线推理服务缓存




inference result in-process cache:
caseid，120s

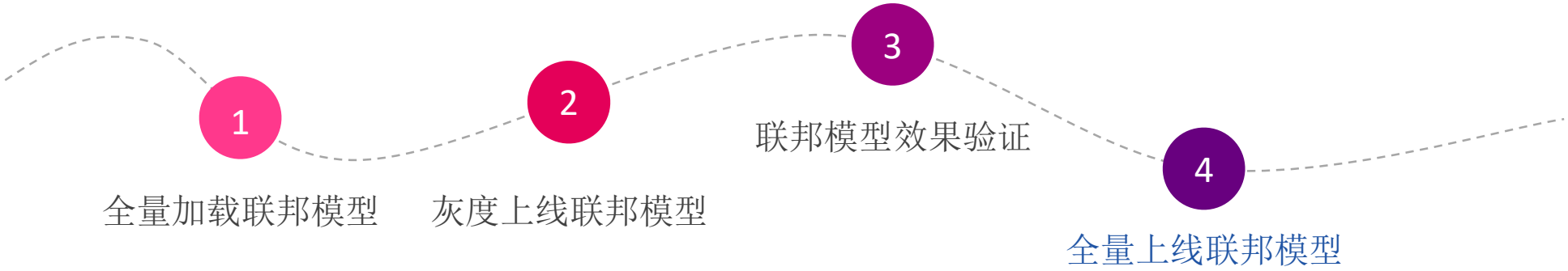
remote federated result in-process cache:
userid，600s

remote federated result distributed cache:
userid，24h


批次请求 → 用户提取 → 活跃度统计

Warming-up Schedule

# 联邦模型应用生产服务流程



# 关注FATE


FATE GitHub主页

FATE 微信小助手

Join FATE，Let’s Federated Everything!

官网：**https://www.fedai.org/**
邮箱：contact@fedai.org