B.大数据相关应用场景一览

1 风险控制

整合内外部数据，依靠前沿数据挖掘技术，打造全方位风险控制能力。

2 欺诈识别

利用行业反欺诈数据，基于设备指纹、决策引擎等技术，形成完善的反欺诈体系。

3 客户价值分析

综合全方位信息对客户信息进行评估，甄别核心价值客户，并制定发展策略。

4 商圈分析

基于地理数据剖析不同网点的流量表现，挖掘有潜力的商圈作为区域营销重点。

5 产品分析

对不同贷款、理财产品表现进行分析，评估产品表现，优化产品策略。

6 渠道优化

对门店、网站、电销等不同渠道进行效率分析，为产品推广选择合适的渠道。

7 网站及APP优化

对客户在网页及APP的使用行为进行分析，指导网站设计，提高客户转化率。

8 精准营销

基于用户画像，根据用户特征及用户偏好针对性地提供产品，提高营销成功率。

9 人员效能分析

收集员工效能、考勤、展业等数据，分析员工表现，并找到影响员工表现的关键因素。

10 敏捷BI

通过丰富的图表来展示业务表现，让决策层能以直观的形式实时了解业务表现。

11 智能投资顾问

基于投资组合理论，为不同风险偏好的客户选择合适的产品组合，实现一键理财。

12 智能客服

利用知识图谱技术，将客服知识托管给计算机，实现7*24小时在线智能客服。

AB.大数据场景示例—精准制导的营销平台

AB.信用评分体系

社保公积金　交易信息　社交信息　资产信息　人行信用

身份信息

AB.大数据助力风控全流程—构建立体化风险监控

事前

新技术&大数据，构建事前欺诈防线。

事中

传统数大数据分析，模型和规则并重精准识别伪冒。

事后

聚类排查、链式分析提前挖掘关联欺诈。


谢谢
沈百军
平安银行零售大数据技术总监

Alibaba Group
阿里巴巴集团

容量规划和流量管控

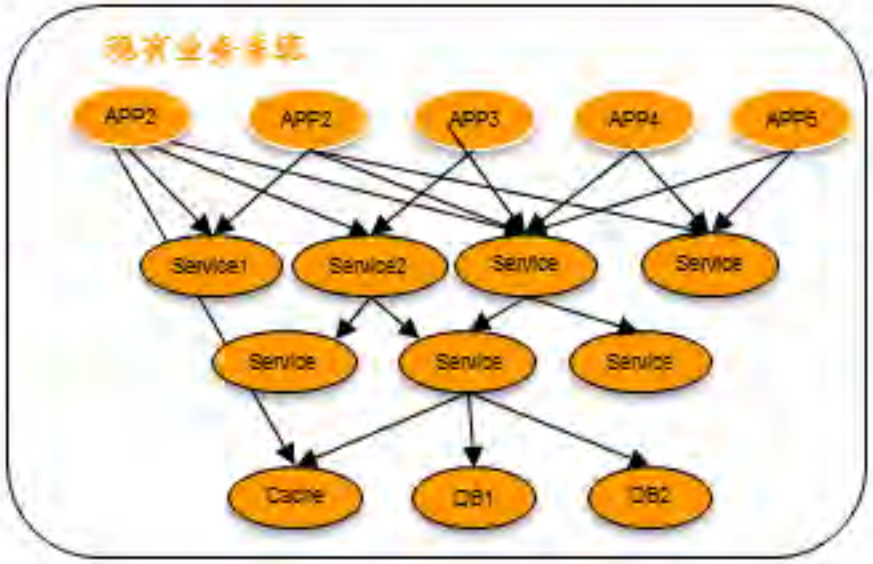
阿里巴巴高可用架构团队
张军,林佳梁

Alibaba Group

容量规划

业务需求

- 业务增长
- 新业务
- 大促



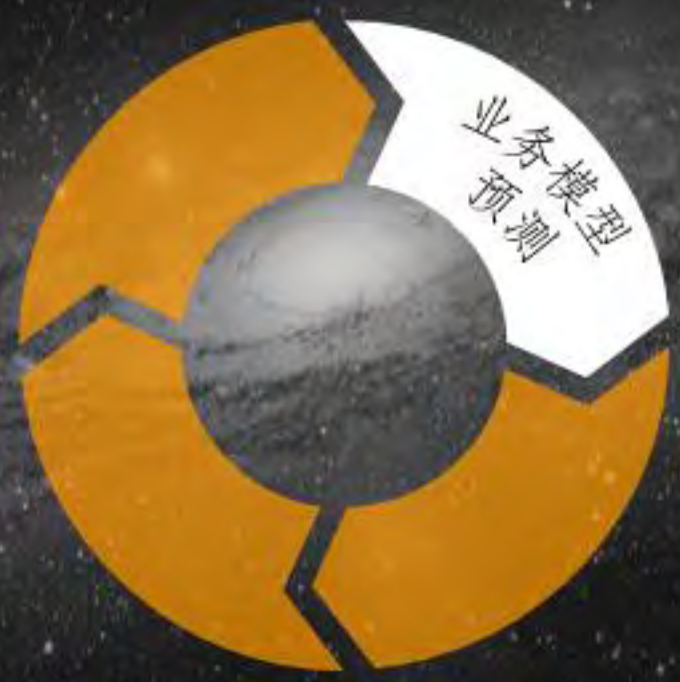
问题

- 什么时候需要扩容?
- 需要多少机器?
- 不同应用的比例?

Capacity planning is the process of determining the capacity needed by a complexity distributed system to meet the workload with guarantee on certain level of performance

Alibaba Group
阿里巴巴集团

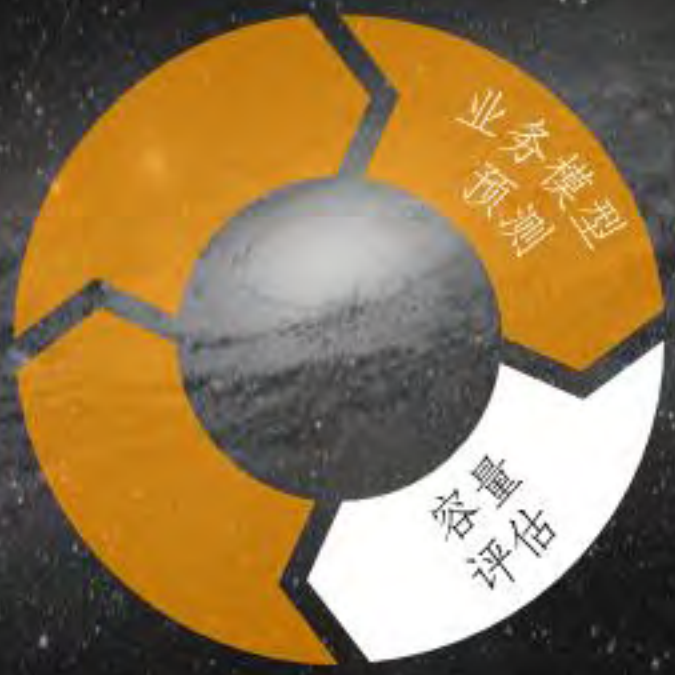
容量规划



- 流量模型
- 历史数据
- 预测算法

Alibaba Group
阿里巴巴集团

容量规划



- 流量模型
- 历史数据
- 预测算法

- 单机容量
- 应用模型

Alibaba Group
阿里巴巴集团

单机容量评估的四种方式

模拟

复制

重定向

Load Balance

Alibaba Group
阿里巴巴集团

模拟

生产环境

测试环境

优点

- 容易实现
- 适合新应用

缺点

- 请求不够逼真
- 额外的脏数据

Alibaba Group
阿里巴巴集团

复制

生产环境

测试环境

优点

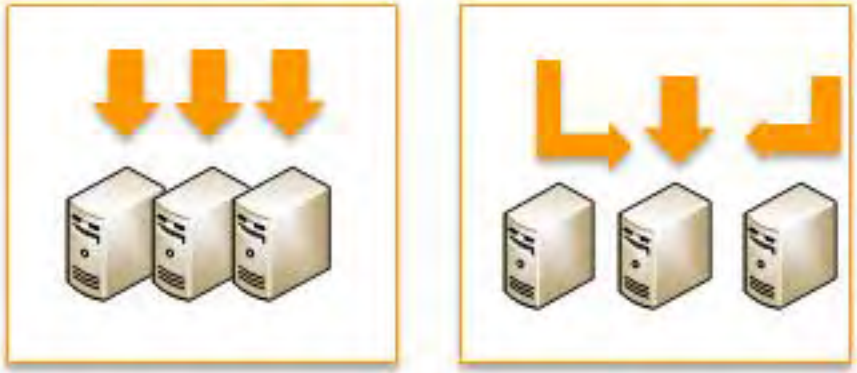
- 贴近生产环境
- 对于流量较少的应用,可以通过复制来扩大流量.

缺点

- 需要额外的机器
- 产生脏数据

Alibaba Group
阿里巴巴集团

重定向



生产环境　　生产环境

优点

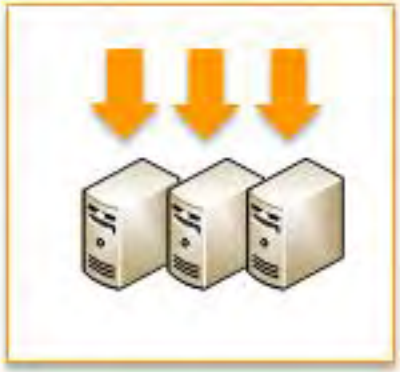
- 所有的数据都是来源真实
- 无脏数据

缺点

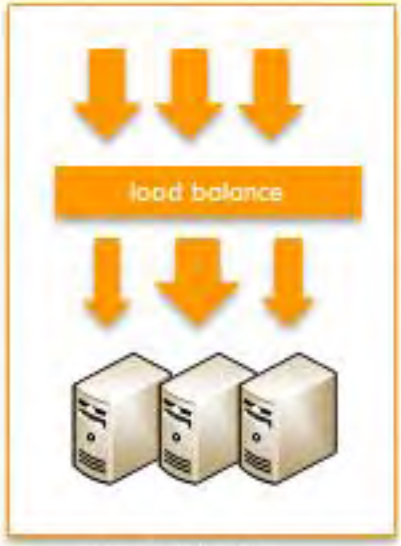
- 无法实施于新应用或者流量较少的应用

Alibaba Group
阿里巴巴集团

Load Balance



生产环境



生产环境

优点

- 所有的数据都是来源真实
- 无脏数据

缺点

- 无法实施于新应用或者流量较少的应用

Alibaba Group

单机压测平台的架构

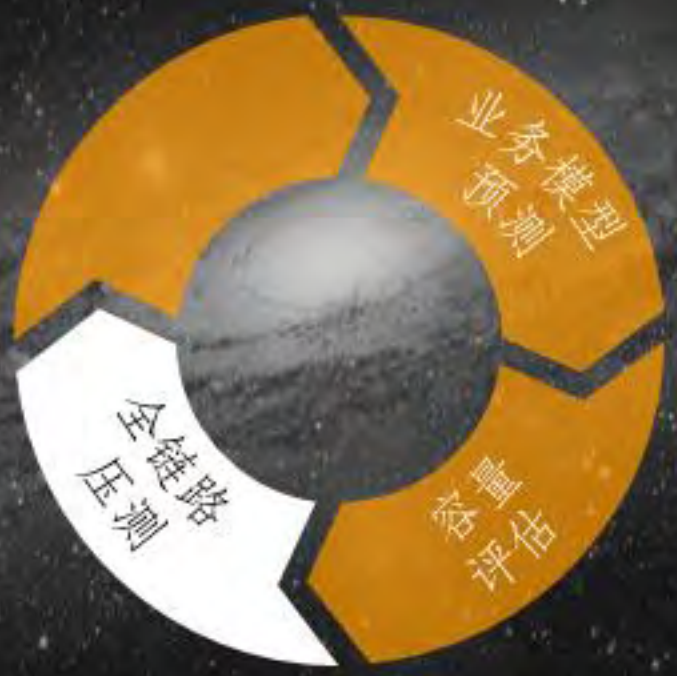
- 自动执行/停止压测
- 产生容量水位及压测报告
- 每个月承载5000+次基线维护

Alibaba Group
阿里巴巴集团

需要的机器数目=
业务估算/单机容量+冗余

Alibaba Group
阿里巴巴集团

容量规划



- 流量模型
- 历史数据
- 预测算法

- 单机容量
- 应用模型

- 验证
- 微调

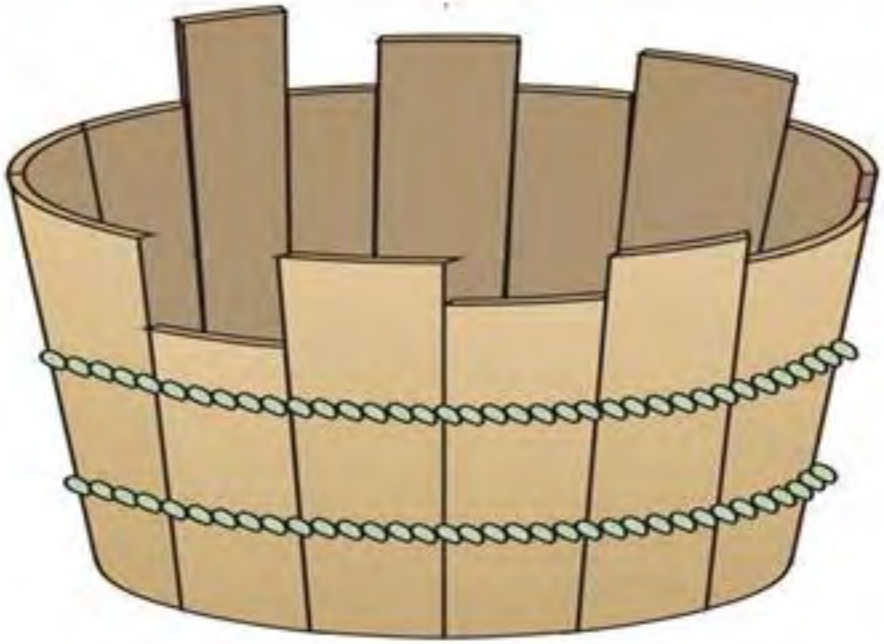
Alibaba Group

为什么会有全链路压测

"Single point" approach is totally different from "scenarios"

Alibaba Group
阿里巴巴集团

木桶理论

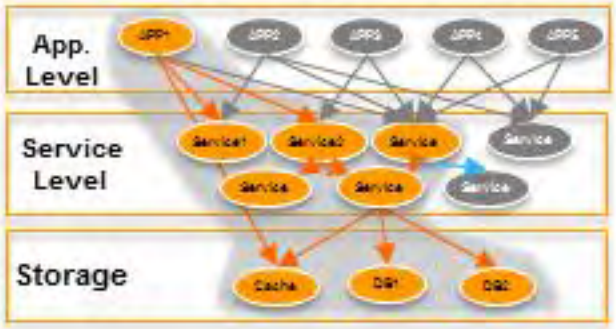


- 木桶最短的板决定站点能力
- 探测系统瓶颈点，进行针对性优化，提升站点性能

Alibaba Group
阿里巴巴集团

全链路压测

通过模拟大促的所有场景,验证我们所做的规划



目的

- 校验我们的规划
- 找出链路的薄弱点
- 微调容量配比
- 为 11/11作演练

Alibaba Group
阿里巴巴集团

困难点

请求规模大

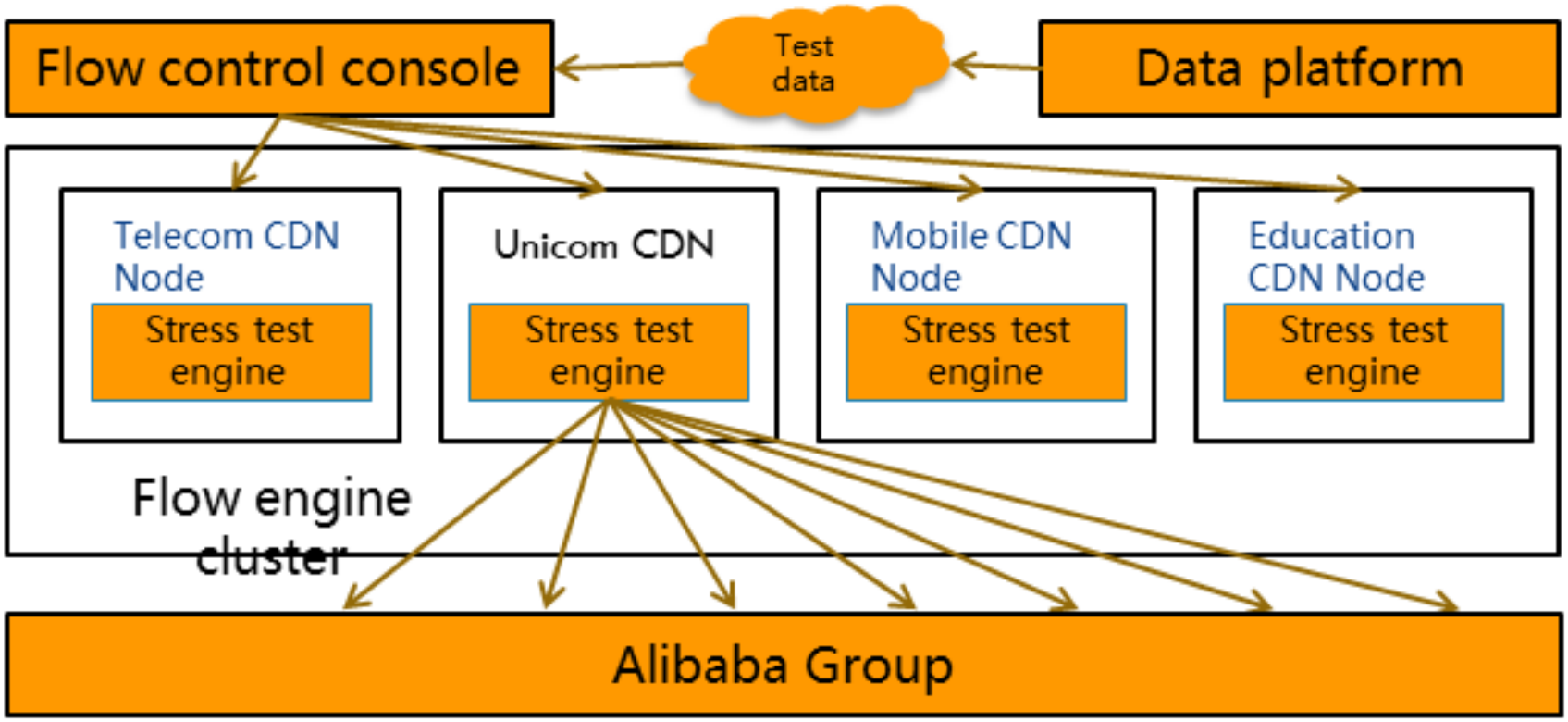
- 10,000,000 requests/sec

用户行为复杂

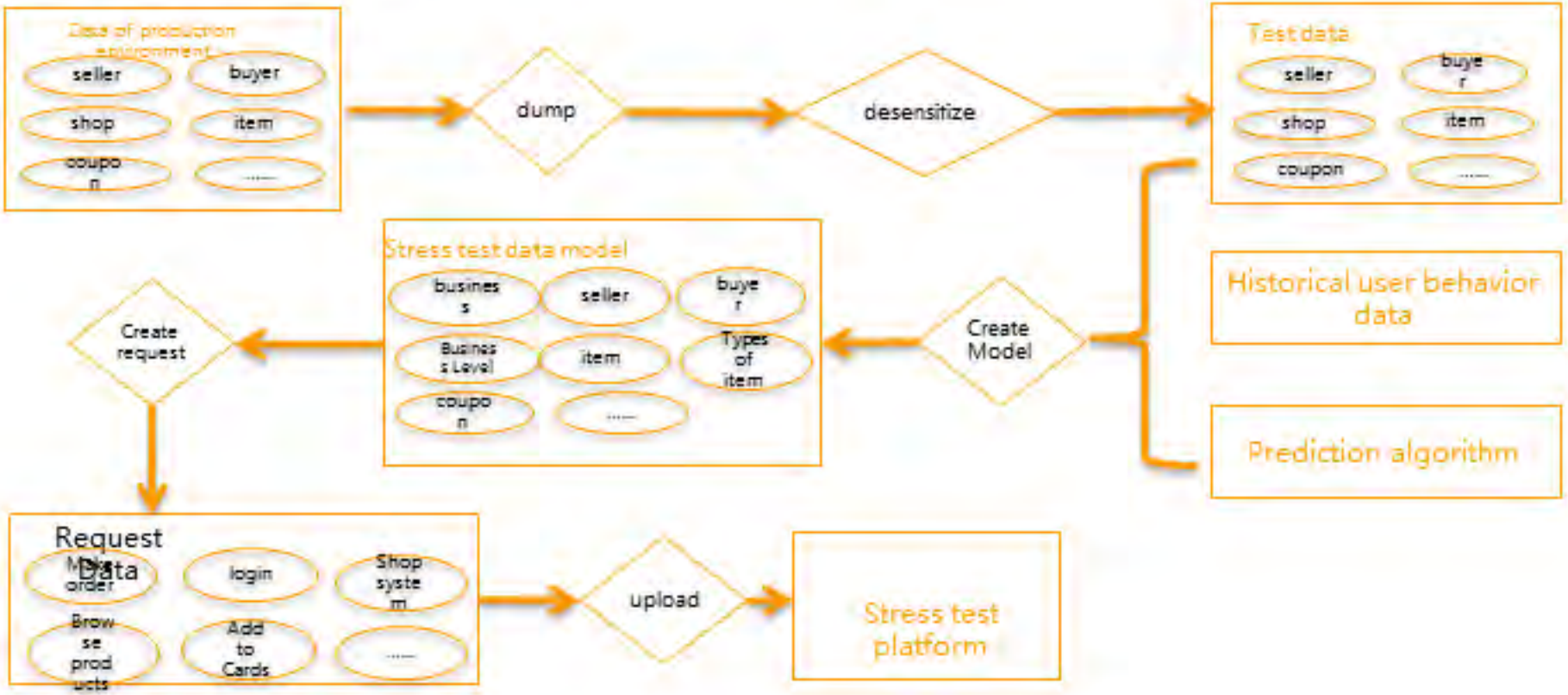
大促场景特有

不能对生产环境有影响

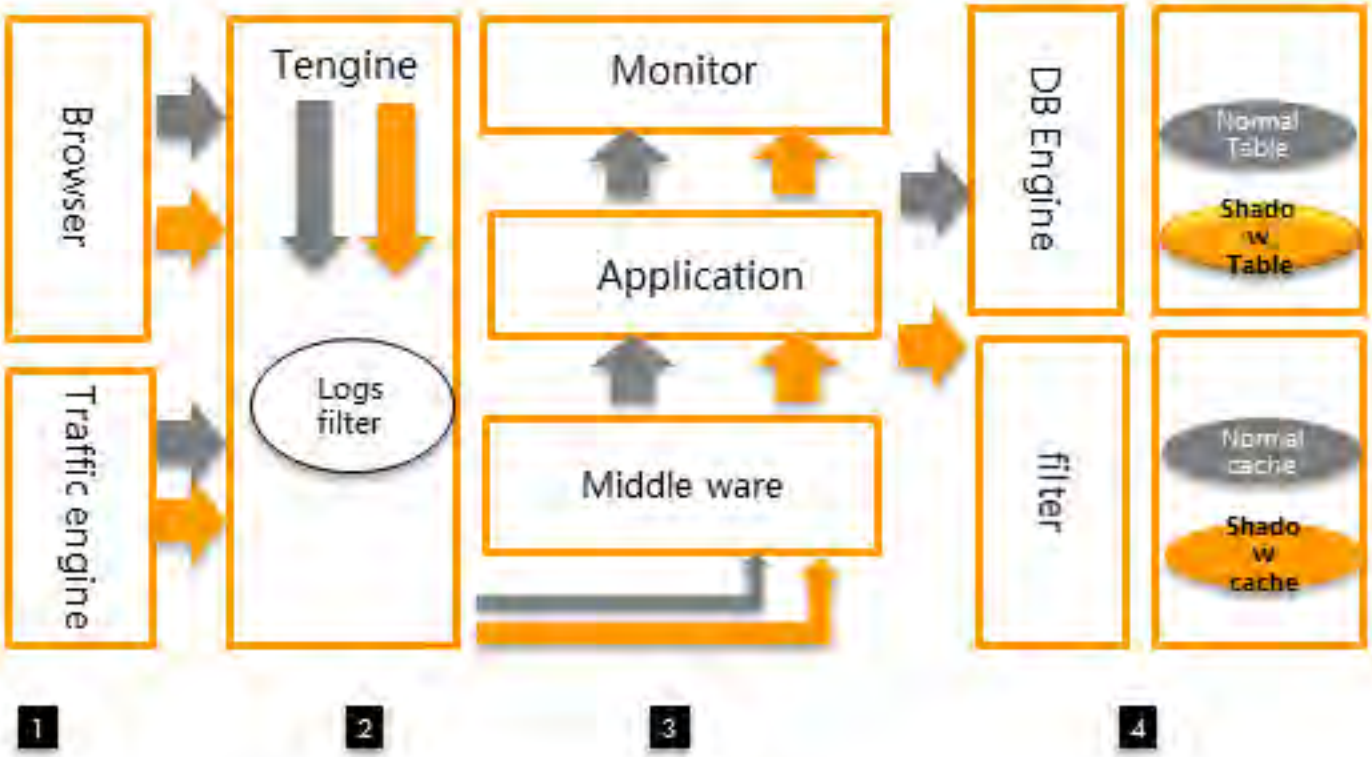
模拟用户请求


Flow control console
Test data
Data platform
Telecom CDN Node
Stress test engine
Unicom CDN
Stress test engine
Mobile CDN Node
Stress test engine
Education CDN Node
Stress test engine
Flow engine cluster
Alibaba Group

如何构造用户请求模型

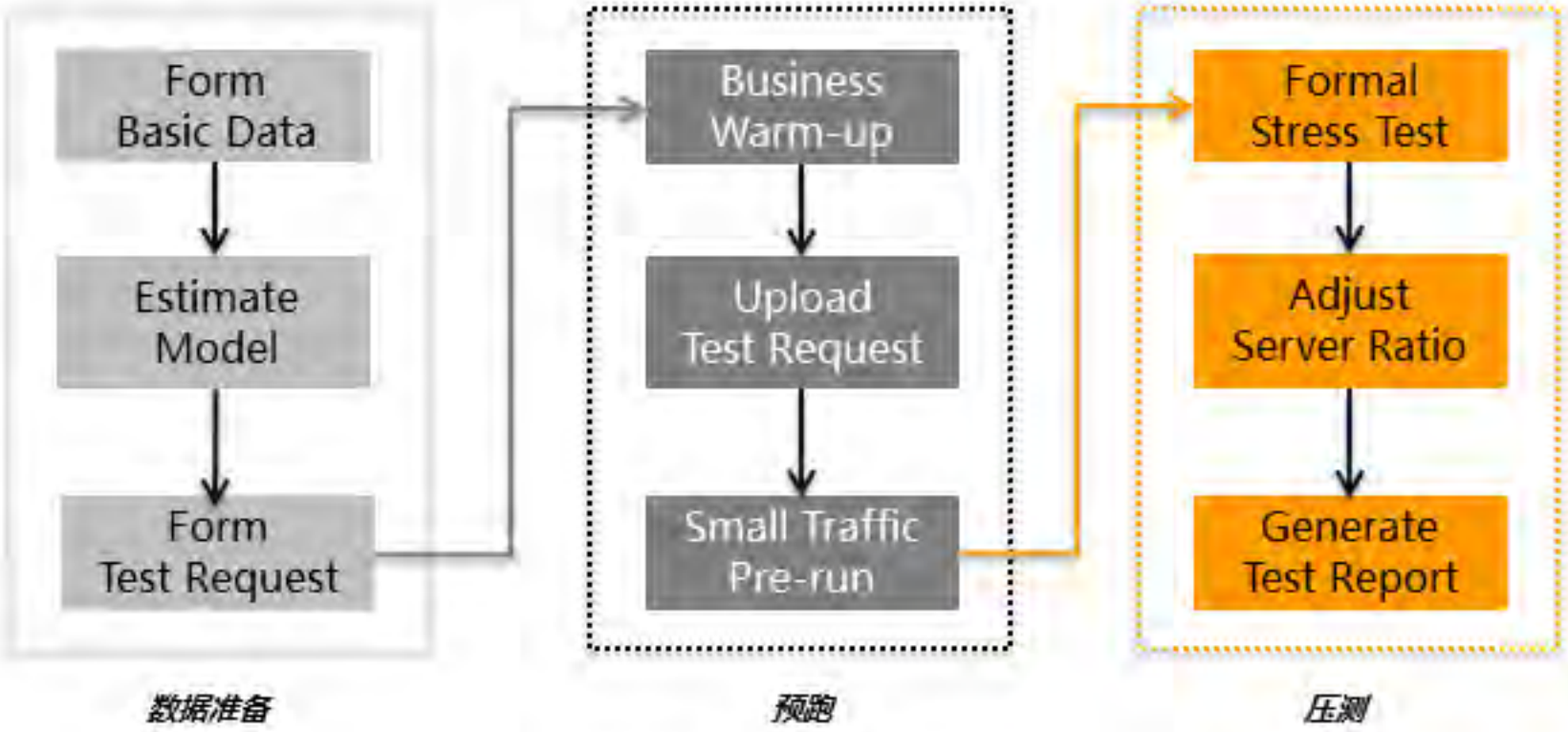

Data of production environment
seller
buyer
shop
item
coupon
dump
desensitize
Test data
seller
buyer
shop
item
coupon
Historical user behavior data
Prediction algorithm
Create Model
Stress test data model
business
seller
buyer
Business Level
item
Types of item
coupon
Create request
Request Data
login
Shop system
Browse products
Add to Cards
upload
Stress test platform

隔离测试数据


Browser
Traffic engine
Tengine
Logs filter
Monitor
Application
Middle ware
DB Engine
filter
Normal Table
Shadow Table
Normal cache
Shadow cache
1
2
3
4
1 测试数据携带一个压测标
2 标会被一层层传下去
3 每一层都有处理自己标的逻辑
4 最后到达DB,Cache等会将存放在影子区域

Alibaba Group

关键步骤



Alibaba Group
阿里巴巴集团

成果

2013 第一次全链路压测

- 3.8 大促
- 6.18 大促
- 9.9 大促
- 双十一 (5 次模拟)

2013 to 2015 平台化

2016 的能力

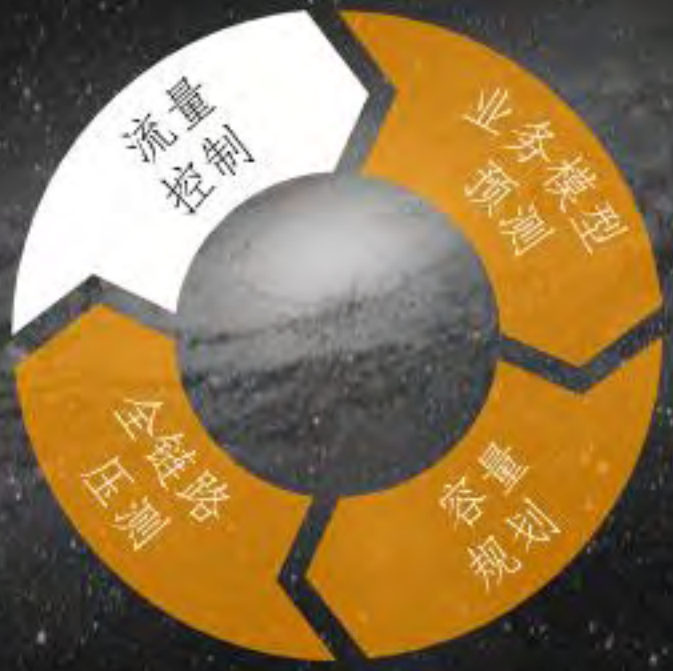
- 4000+ 链路压测
- 兼容优酷，土豆等子公司模式

Alibaba Group
阿里巴巴集团

容量规划

- 流控

- 验证
- 微调



- 流量模型
- 历史数据
- 预测算法

- 单机容量
- 应用模型

Alibaba Group
阿里巴巴集团

流量控制的重要性

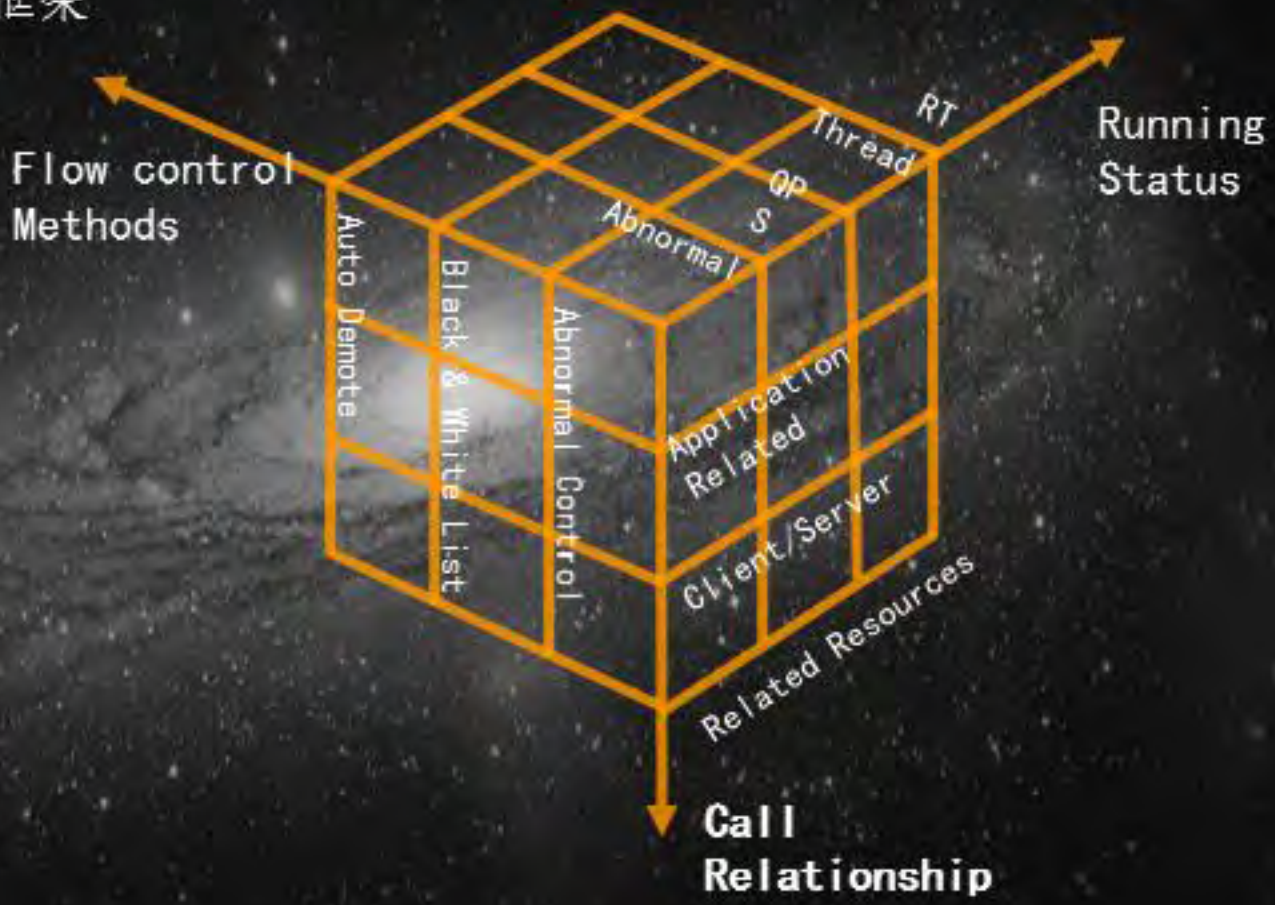
超过容量的流量会造成

- 影响服务器的性能
- 拉长相应时间
- 影响用户体验.

雪崩效应

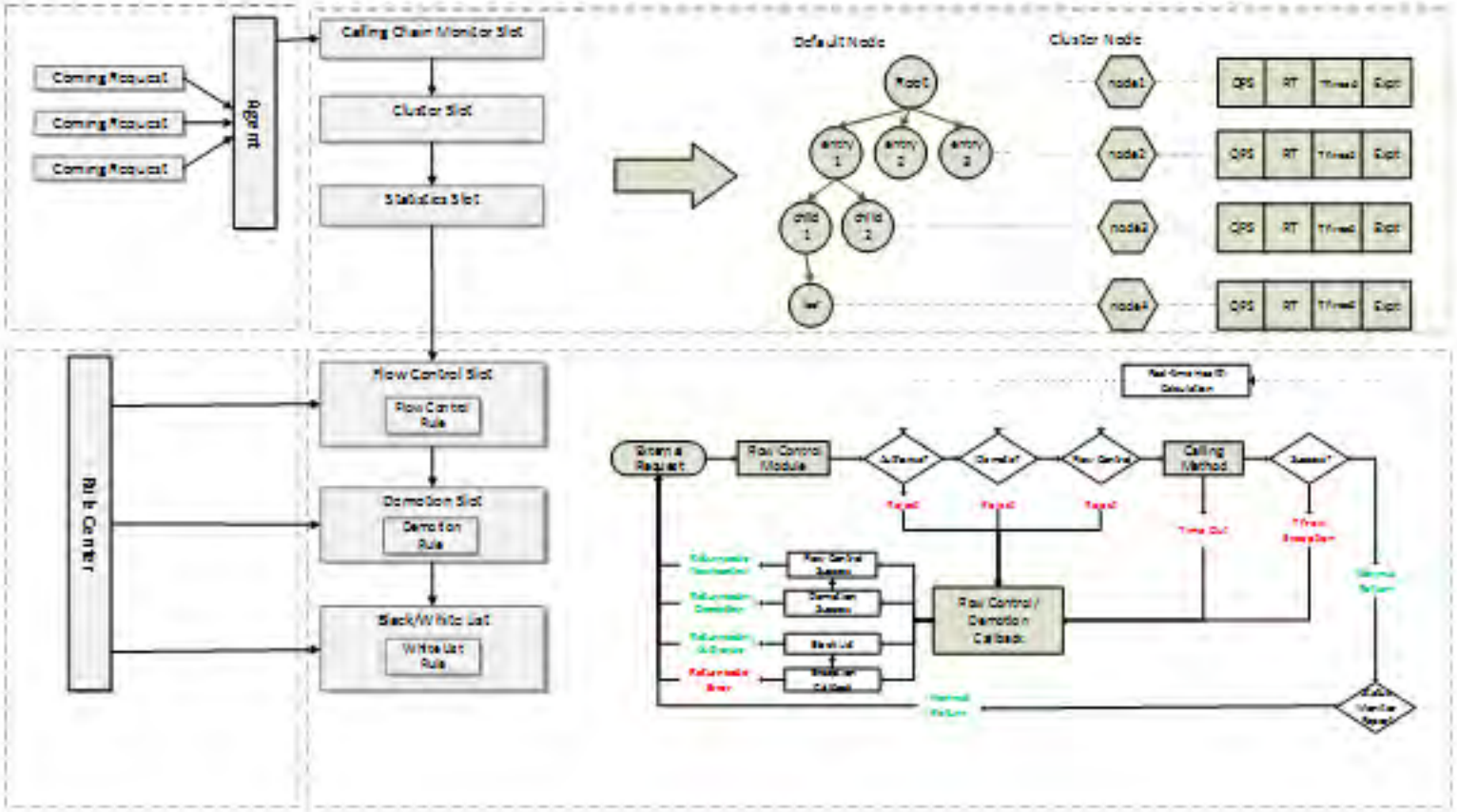
Alibaba Group

流量控制模型的框架



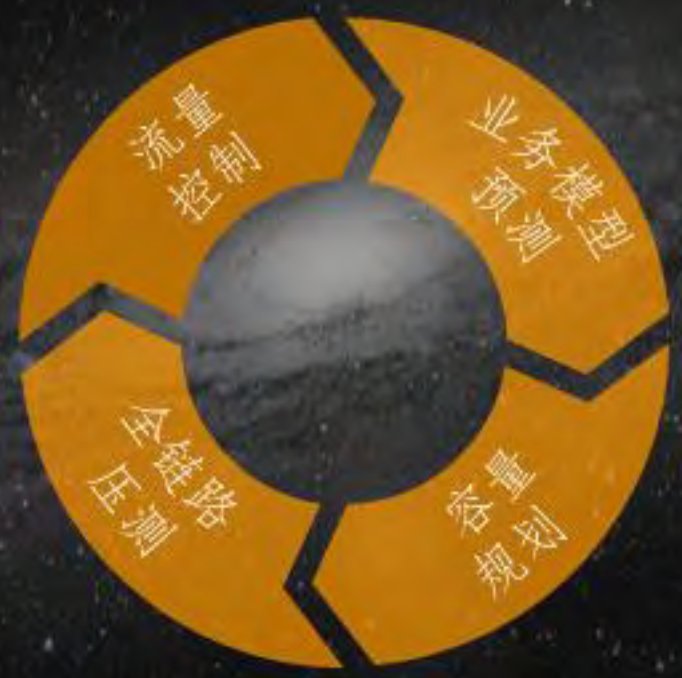
Alibaba Group

流量控制的工作流



Alibaba Group
阿里巴巴集团

容量规划

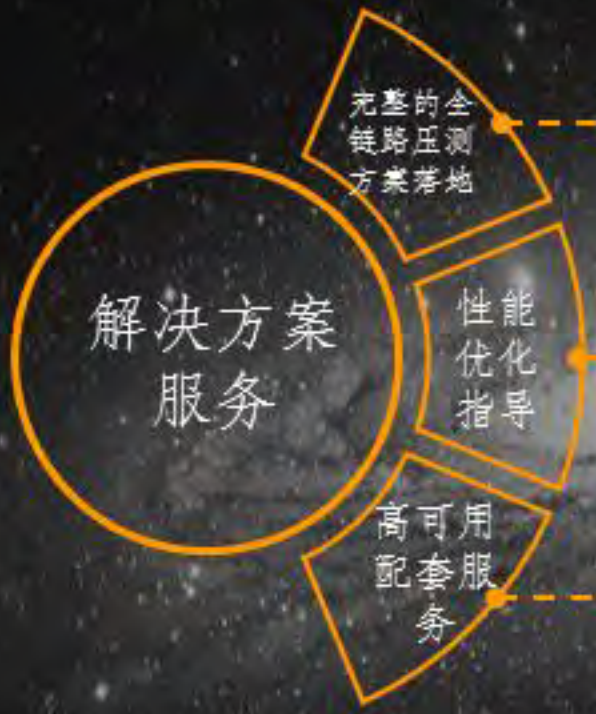


Alibaba Group
阿里巴巴集团

https://www.aliyun.com/product/pts

Alibaba Group
阿里巴巴集团

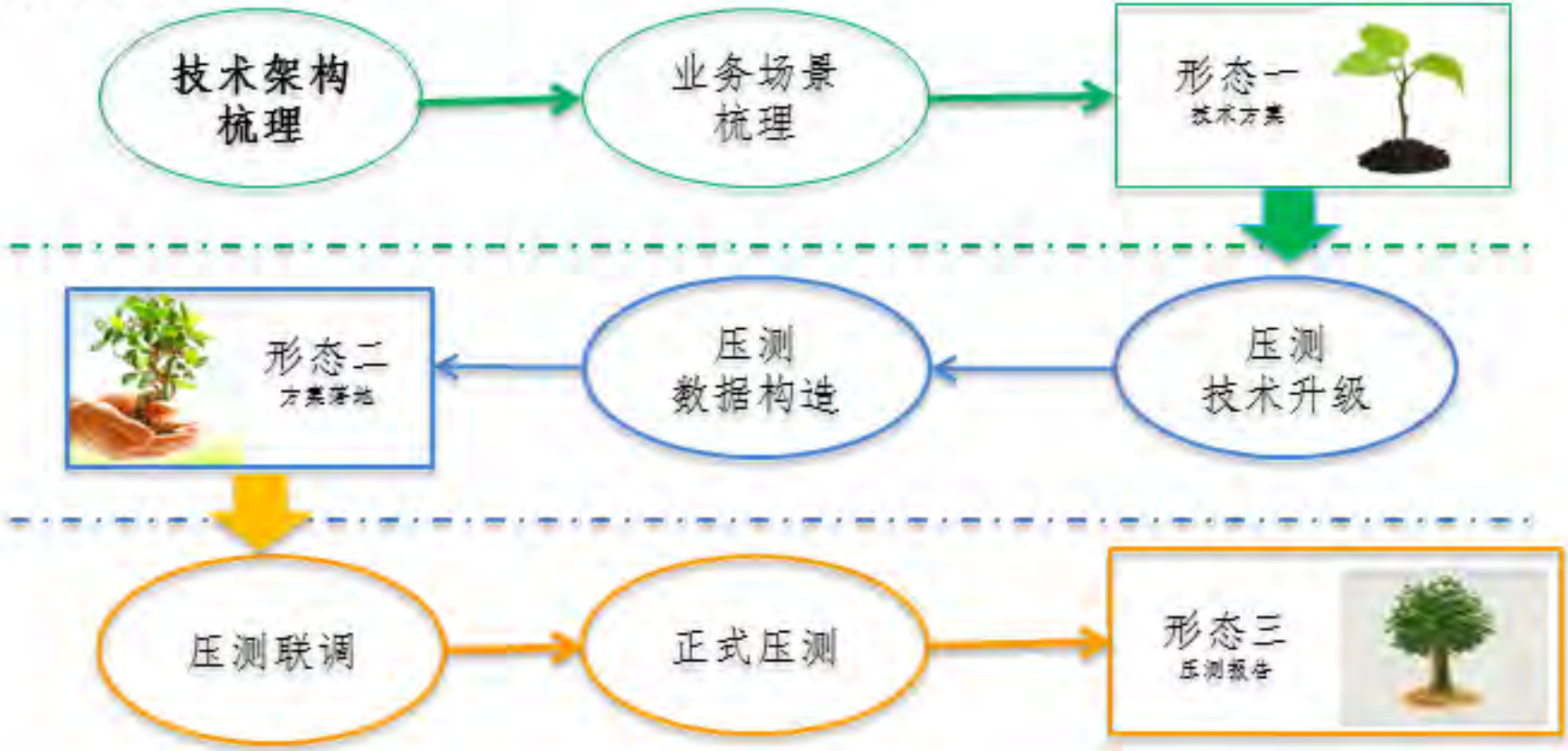
全链路压测解决方案服务



- 架构与业务梳理、数据隔离、技术升级、压测方案制定，压测联调与实施
- 压测问题的分析与优化指导
- 保障站点高可用其它手段的配套服务

Alibaba Group

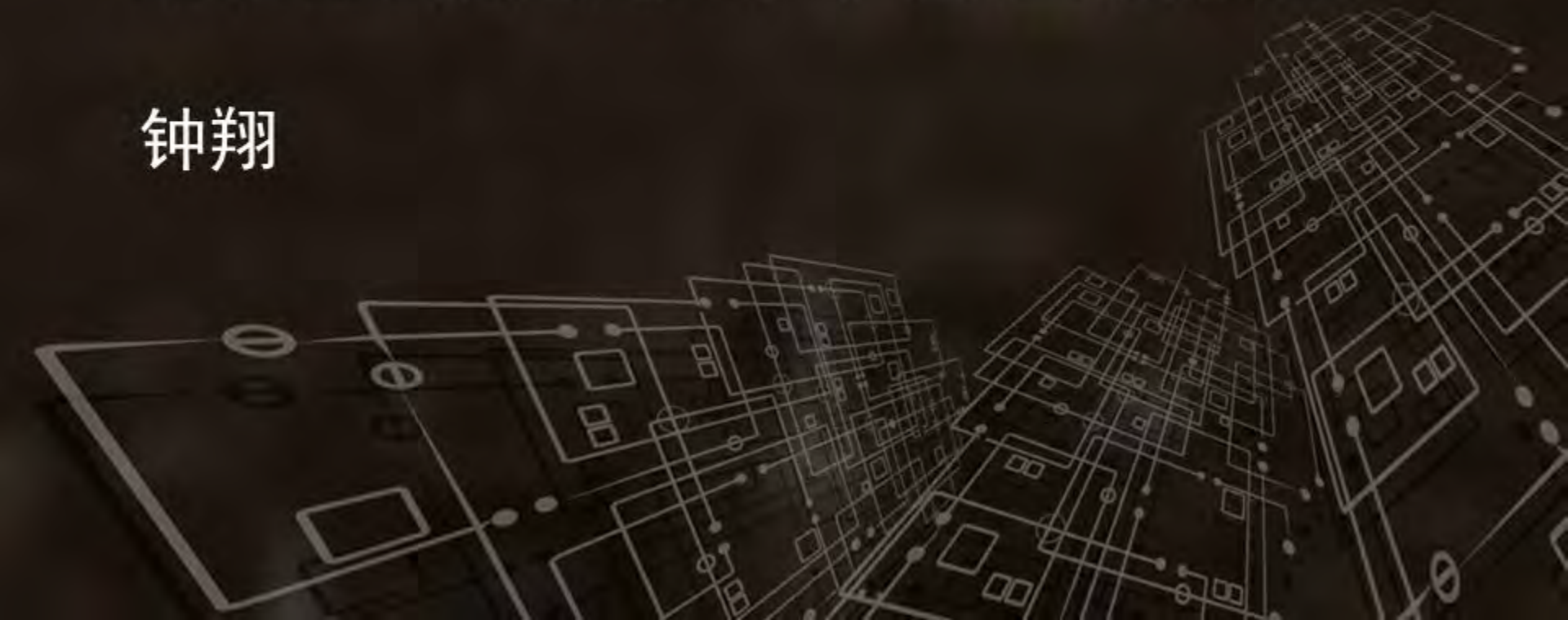
解决方案流程



2017 Software Architecture Summit

唯品会机器学习平台建设实践

钟翔



机器学习平台MLP

- **MLP**: **M**achine **L**earning **P**ipeline

议程

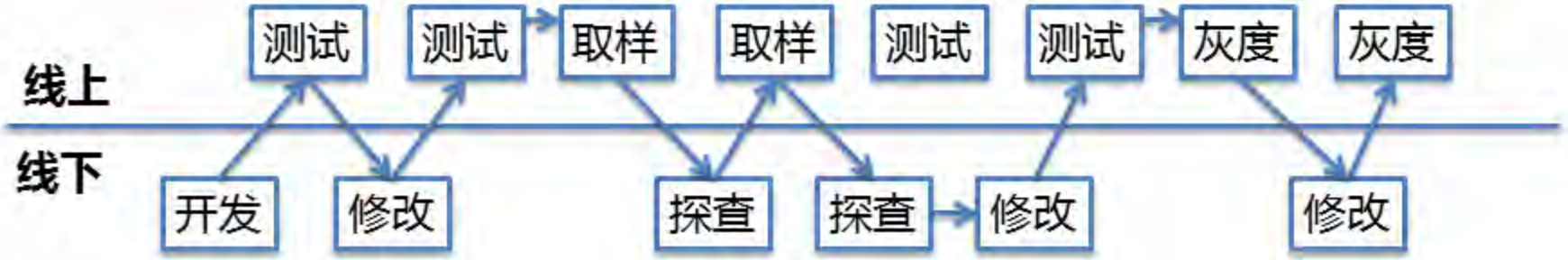
我们要解决什么问题?

问题1：共享协助的问题

很多人做一件**共同**的事情，如何站在别人的肩膀上把事情做到**最**好？

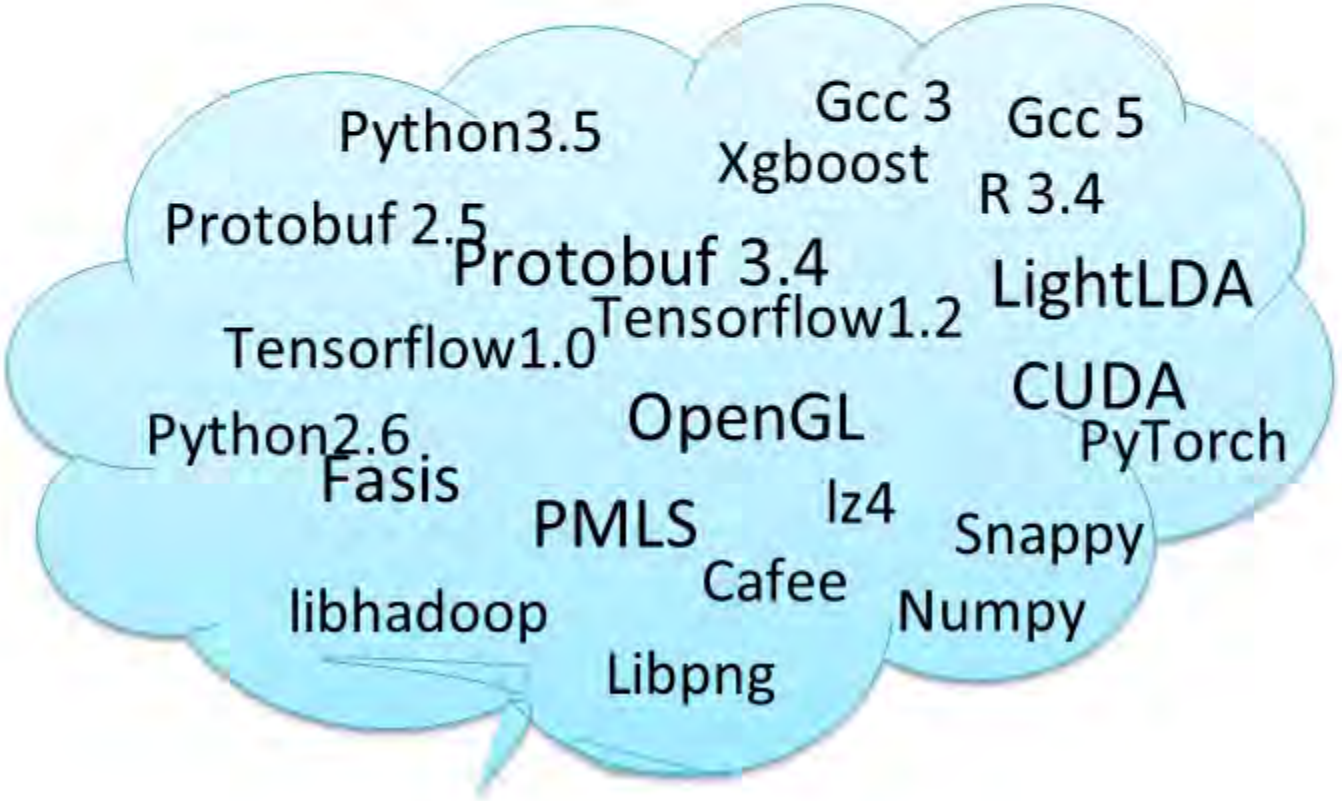
问题2：开发模型周期长，时间成本高的问题

- 流程长，由于线上手段的缺乏，需要频繁在线上线下切换。影响开发效率



问题3：运行环境多样，系统维护成本高的问题

- 版本依赖，native lib依赖，kernel依赖，跨平台移植性等问题



问题4：实时分布式计算能力的问题

简而言之，我们的目标是：

- 设计一个系统，
  - **解放生产力**，机器学习现代化。
  - 在**实时数据，长周期数据，图像数据上具**备处理能力，满足在线学习深度学习需求；
  - **鼓励共享协作。**

那我们的解决思路是什么呢?

思路1：端到端的一站式服务平台

- 在线服务，提高开发效率，促进共享协作。

思路2：容器化

多版本依赖

多租户

弹性计算

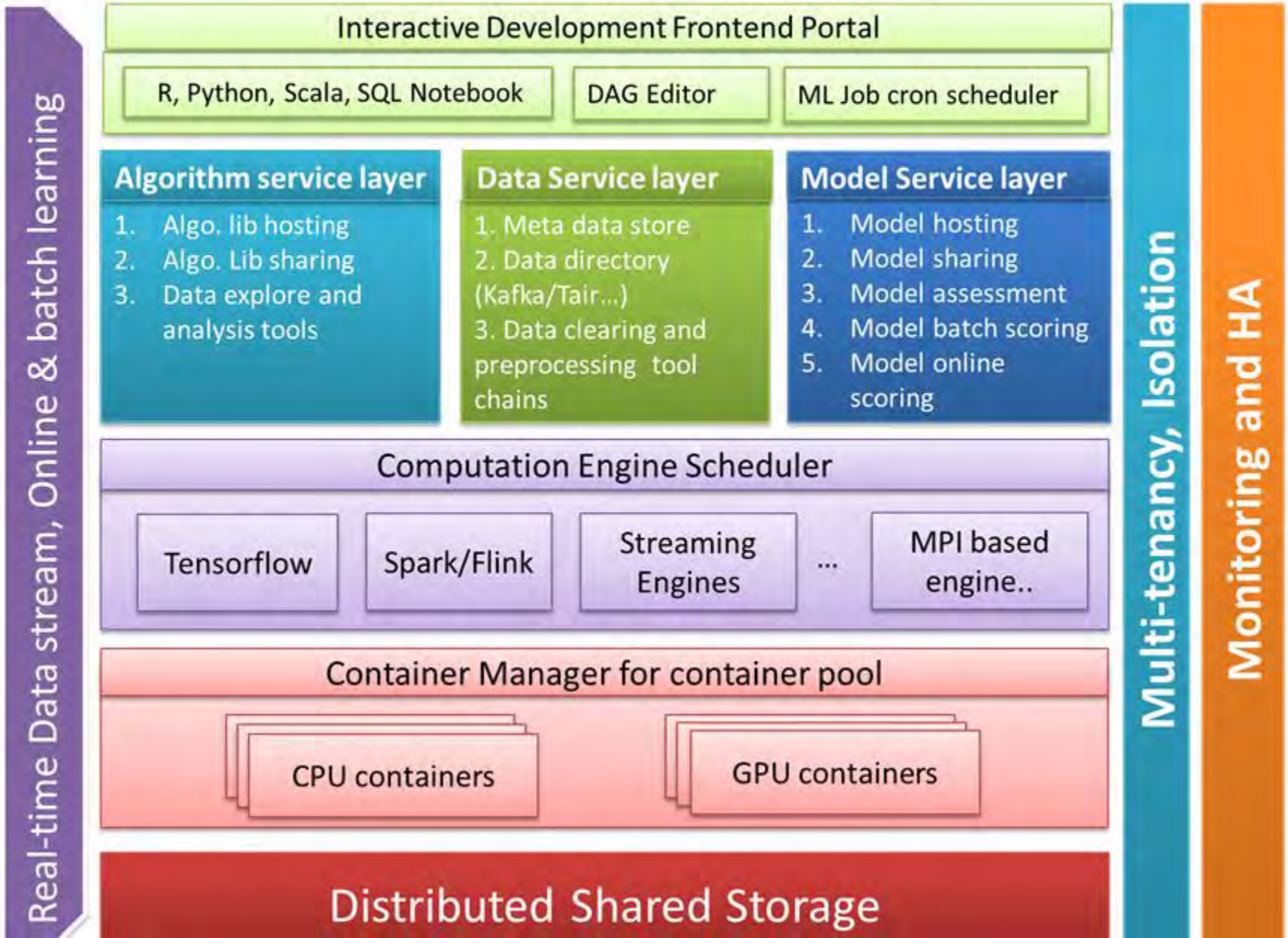
灵活部署

思路3:提供高性能高可靠的计算能力

对计算框架的选择技术上不做限定，按性能和稳定性作为ML引擎选择的标准

我们的技术方案是什么样的呢?

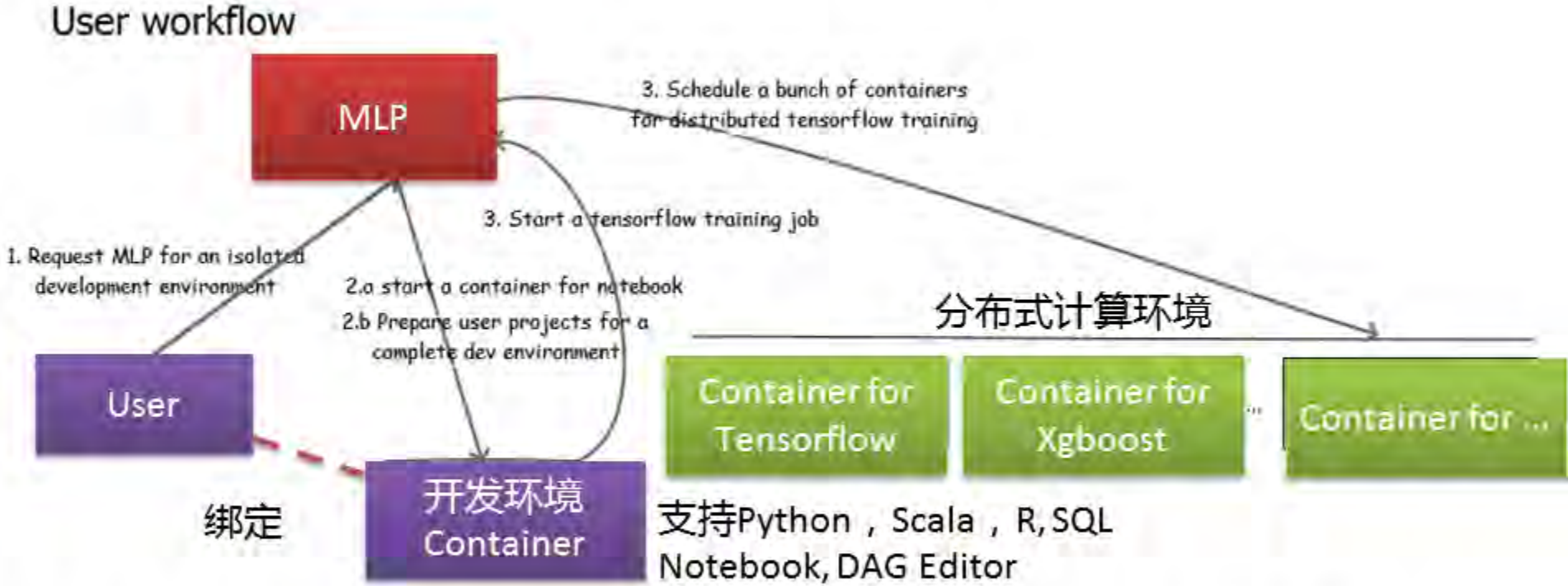
MLP平台架构图


Real-time Data stream, Online & batch learning
Interactive Development Frontend Portal
R, Python, Scala, SQL Notebook
DAG Editor
ML Job cron scheduler
Algorithm service layer
1. Algo. lib hosting
2. Algo. Lib sharing
3. Data explore and analysis tools
Data Service layer
1. Meta data store
2. Data directory (Kafka/Tair...)
3. Data clearing and preprocessing tool chains
Model Service layer
1. Model hosting
2. Model sharing
3. Model assessment
4. Model batch scoring
5. Model online scoring
Computation Engine Scheduler
Tensorflow
Spark/Flink
Streaming Engines
...
MPI based engine..
Container Manager for container pool
CPU containers
GPU containers
Distributed Shared Storage
Multi-tenancy, Isolation
Monitoring and HA

MLP是一个在线服务

- 在浏览器里直接访问
- 不需要用户装任何软件
- 包含受管理的集成开发环境，Remote IDE，支持Python，R，Scala，SQL。
- 包含可视化编程，缩短学习时间，快速上手。
- 包含受管理的可扩展的分布式计算集群。
- 包含工作流的调度。
- 包含支撑机器学习六个环节SEMMAS的各种工具链，支撑完成机器学习开发的全流程（从拿到数据到部署上线和模型验证）。

MLP用户工作流

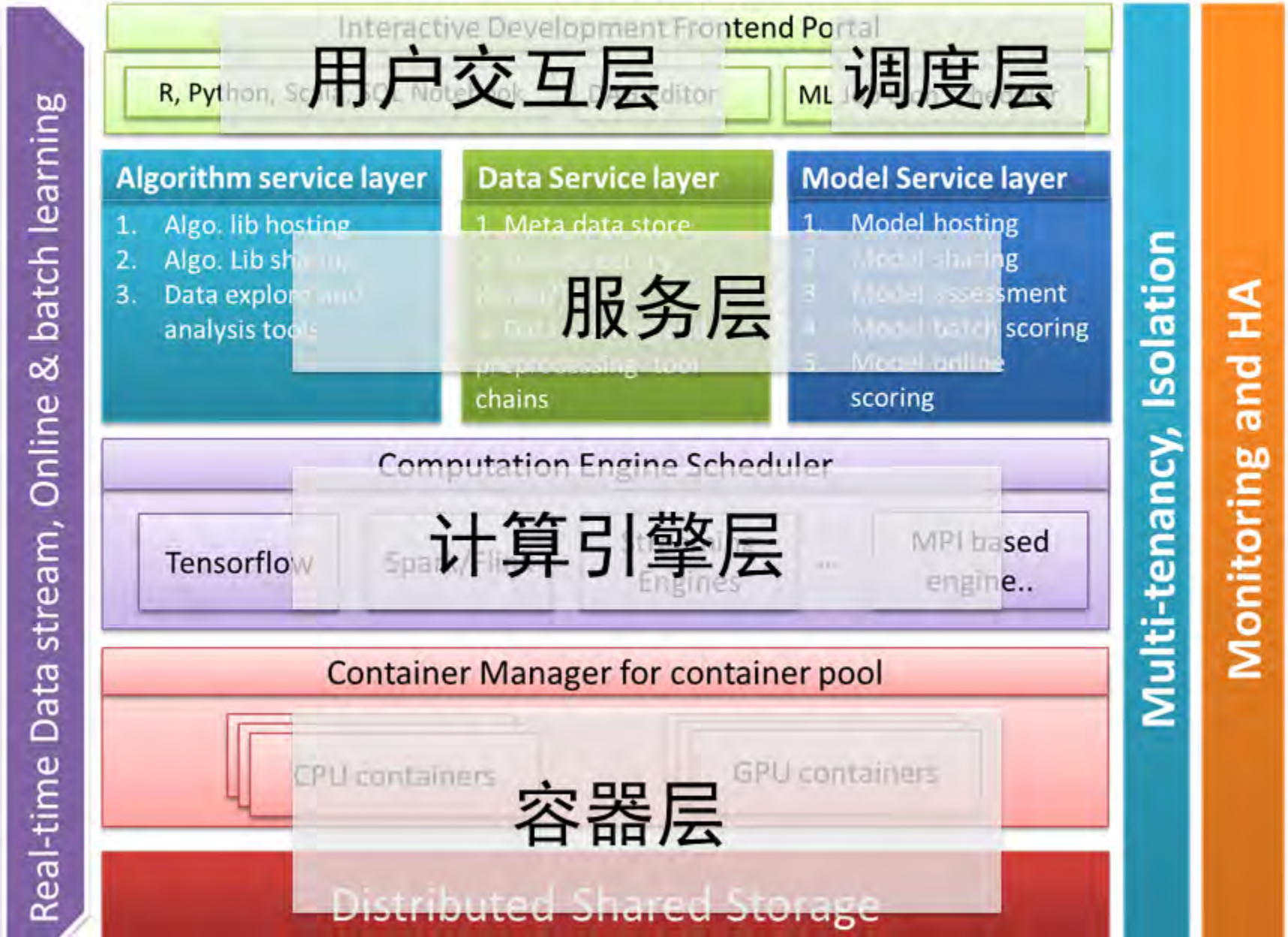


支持多租户，每个用户会拿到一个独有的，隔离的开发环境
成熟后开发中的Notebook/DAG直接转为生产，
添加到工作流调度器中.

期望这个系统可以做到:

1. 解放生产力，算法同学只需关注数据和算法上，从系统的负荷中解放出来
2. 支撑海量数据，长周期模型的系统需求
3. 支撑实时数据，在线学习的系统需求
4. 支撑深度学习模型的系统需求
5. 交互式迭代开发，提高开发效率
6. 促进算法共享，模型共享，数据共享。促进跨团队的快速协作
7. 标准化工具链，前置数据处理，数据探查，和后置数据评估.

MLP分层结构

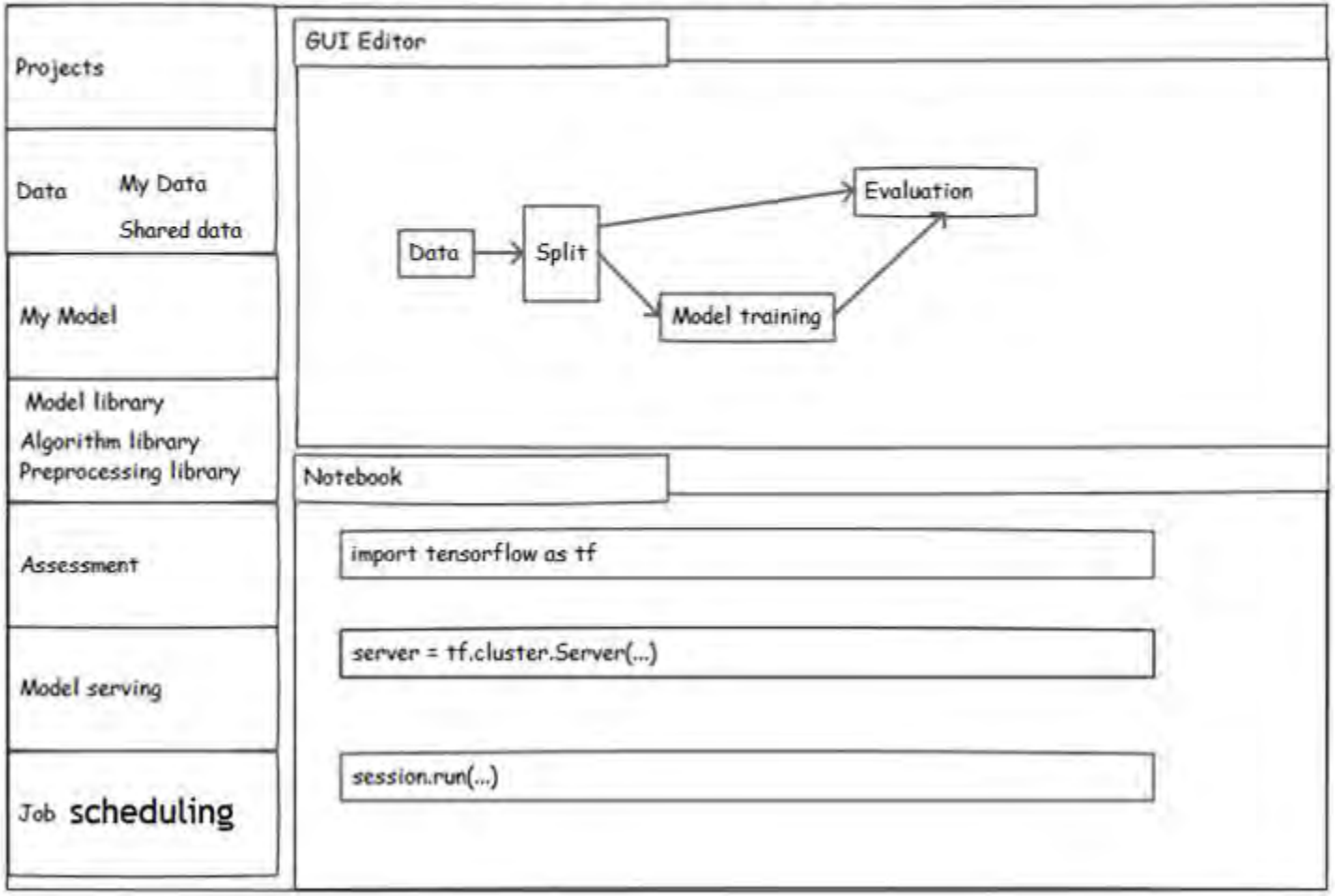

Real-time Data stream, Online & batch learning
Interactive Development Frontend Portal
用户交互层
调度层
Algorithm service layer
1. Algo. lib hosting
2. Algo. Lib sh
3. Data explor
analysis too
Data Service layer
1. Meta data store
chains
Model Service layer
1. Model hosting
scoring
服务层
Computation Engine Scheduler
Tensorflow
MPI based
engine..
计算引擎层
Container Manager for container pool
CPU containers
GPU containers
容器层
Distributed Shared Storage
Multi-tenancy, Isolation
Monitoring and HA

1. 用户交互层

- 开发环境IDE上网，扩展Notebook UI
- Notebook和DAG UI Editor相结合。
- 多租户，每个用户都有独立的容器作为开发环境。
- Git集成。
- 支持Python，R，Scala，SQL。
- 支持各种计算引擎的Kernel，比如Tensorflow等。

Notebook UI

- Notebook交互式开发环境



开发到线上部署的无缝转换

- 以Notebook为主要的“串联语言”
- 我们开发了一套工具链，Notebook可以直接转为在线调度作业。

2. 容器层


用户容器
多租户
计算容器
计算的横向扩展
卷存储
本地卷
日志数据
远程卷
用户数据，配置数据，
共享数据
Backed By Gluster

容器层特点

- **高可用：**定制开发了远程卷，一台机器offline，容器可以自动迁移，不影响业务。
- **日志放在本地卷：**通过日志收集系统统一管理。
- **横向扩展：**用户容器和计算容器都可以横向扩展。
- **隔离：**用户环境通过容器相互隔离。互不影响。
- **共享：**通过远程卷对共享协作提供底层支持。
- **监控:**定制开发容器监控，通过与cAdvisor集成监控集群性能。
- **经验：**1. 避免把高频访问的数据放在远程卷上。2. 增量升级

3. 计算引擎层

以Spark为主作为通用数据平台

支持多种异构的机器学习引擎
引擎选择更看中性能和稳定性，而不是平台引擎通用性。

Tensorflow分布式训练

- 我们的目标：稳定运行，线性Scale。
- 我们解决了:
  - HDFS性能问题
  - gRPC性能问题
  - 训练因为Queue异常不能启动的问题
  - 训练超时的问题
  - PS失败不能退出的问题
  - 作业分发分布式调度的问题
  - HDFS容错的问题
  - 增加Metrics，实现模型有效性监控和报警