

2017
SA-Summit

全球软件架构 技术大会

演讲稿合辑

www.sa-summit.org

《首席技术官/架构师》

风靡美国技术界的CTO必修课，揭开技术管理迷雾

主讲老师：



Martin Abbott

全球软件架构大师，《架构即未来》作者

Martin 从 eBay 创业初期直到发展为全球500强企业，担任 eBay 高级技术副总裁以及首席技术执行官。在加入eBay之前，Martin 曾在 Gateway 以及 Motorola 担任技术、管理等重要职位。

面向技术人员和技术管理人员: CTO、技术VP、首席架构师、技术总监、开发主管、研发经理。

- 提升自身领导力及架构能力
- 解决技术团队及架构现存问题
- 提升团队凝聚力及创新力
- 10+小时课程时长 丰富准则、工具、经验案例
- 凝聚Martin Abbott 20年一线实操精粹，历经 350家企业实践检验。
- 历经数十年精心提炼打磨，专业在线课程团队全心策划制作。



2018 震撼推出！课程限量体验名额招募中。
扫码填写登记表。



SCALING TECHNOLOGY AND ORGANIZATIONS

MARTY ABBOTT
CEO
AKF PARTNERS

AKF Positioning Statement



For companies dissatisfied with or challenged to meet the demands of their growth, we are growth problem solvers. Unlike technology consulting firms, our approach identifies the root causes from all sources including architecture, process, organization, product and strategy.

Most consulting firms fix the symptoms of your problem. We treat both the symptoms and the causes. We are the world's first growth consulting firm.

We differentiate ourselves through superior leadership and fast time to client returns. We bridge the gap between technology, product and strategy.

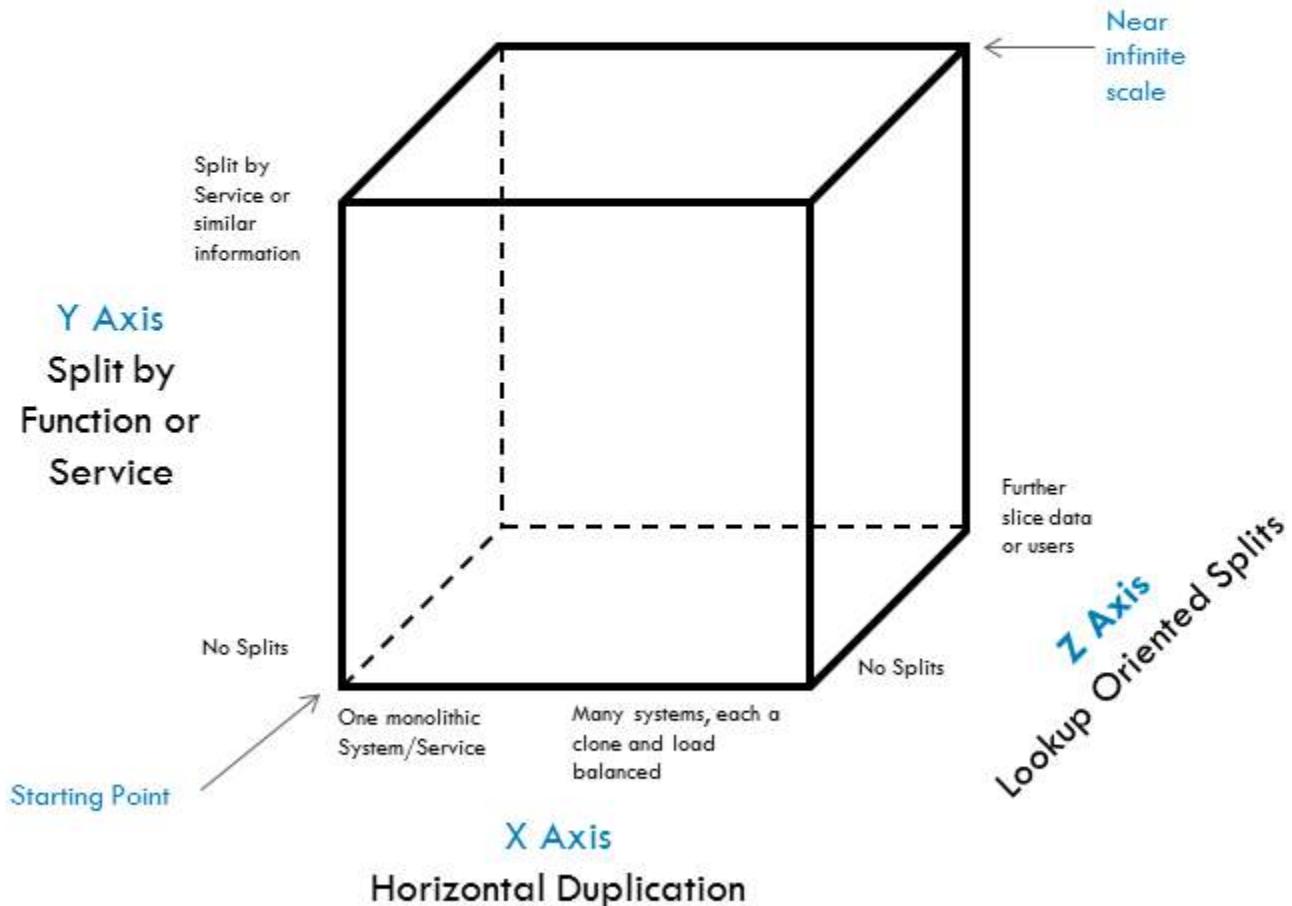
Where we exist



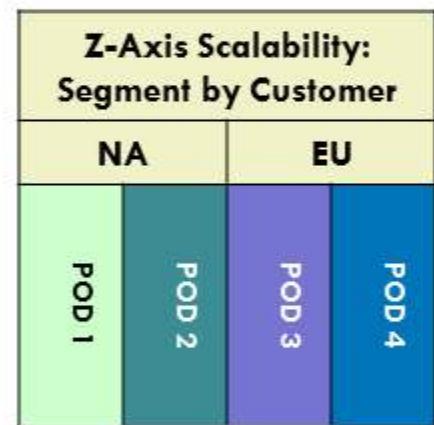
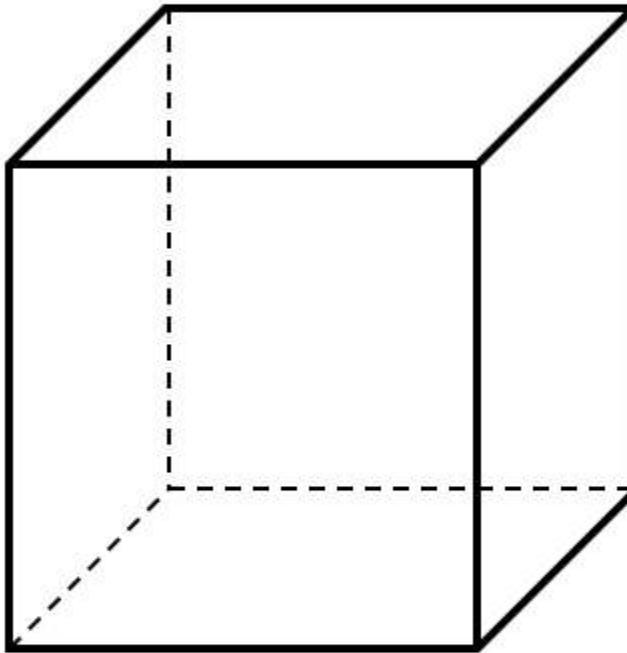
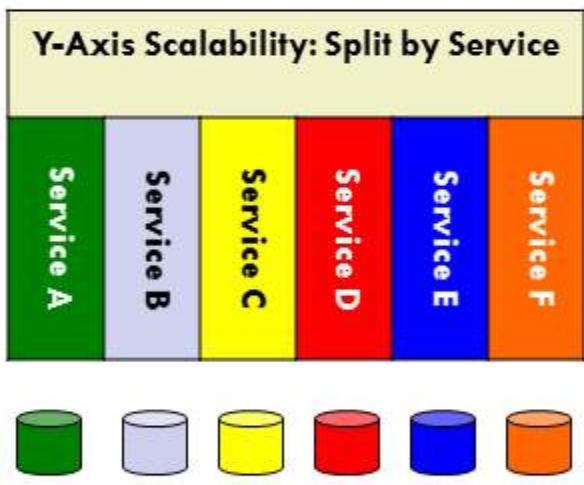


AKF SCALE CUBE

AKF Scale Cube



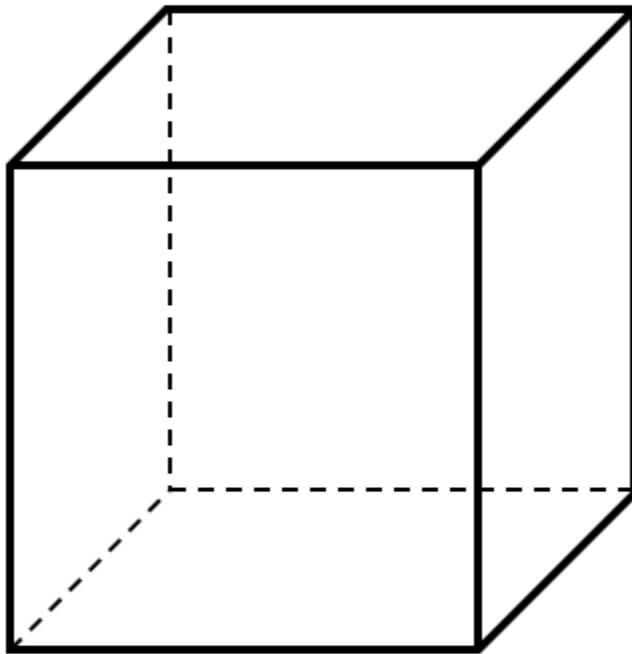
AKF Scale Cube



X-Axis Scalability: Replicate & LB

Web Tier	Replicate Web Servers & Load Balance
App Tier	Store Session in browser or separated Object Cache to horizontally scale app tier independent of web tier
DB Tier	Use Read-Replicas for read-only use cases like reporting, search, etc.

AKF Scale Cube

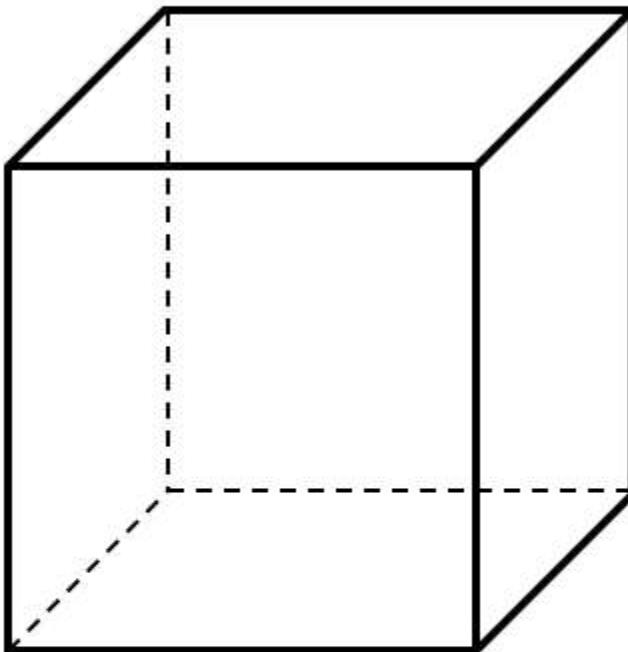


Pros	
Easy	Simple to implement
Fast	Typically very fast to implement
Transaction Scalability	Scales transactions well

Cons	
Storage	Can be costly
Cost Scalability	Multiple data sets
Cacheability	Doesn't address caching

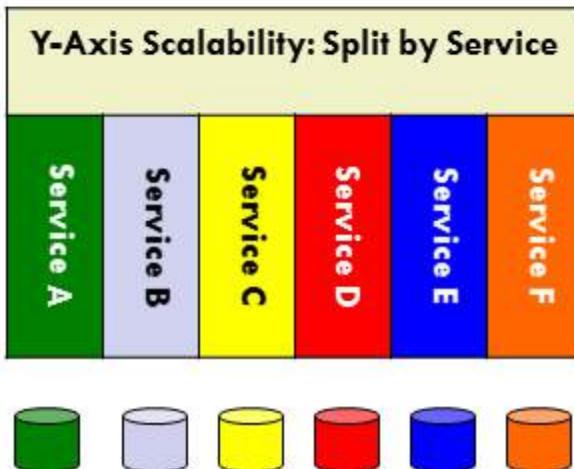
X-Axis Scalability: Replicate & LB	
Web Tier	Replicate Web Servers & Load Balance
App Tier	Store Session in browser or separated Object Cache to horizontally scale app tier independent of web tier
DB Tier	Use Read-Replicas for read-only use cases like reporting, search, etc.

AKF Scale Cube

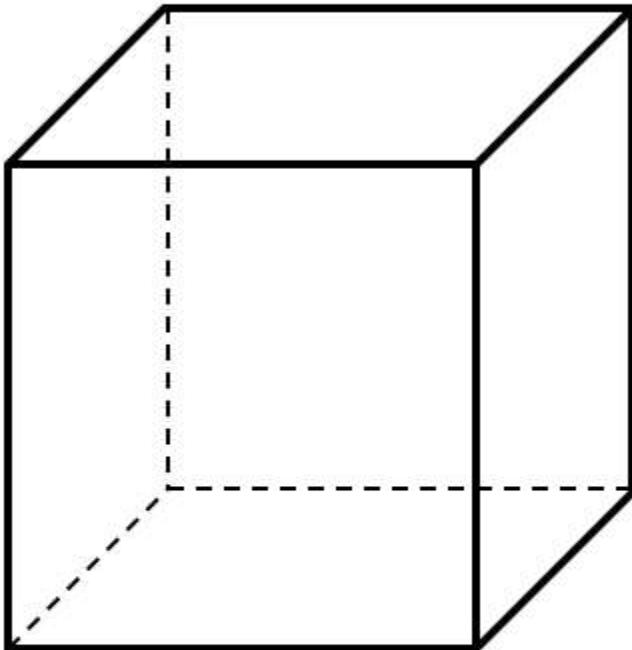


Pros	
Transaction Scalability	Scales transactions well
Cost	Independent scale of services
Org Scale	Allows for Org Scale
Cache	Increases cache hit rate
Availability	Can have fault isolation

Cons	
Difficult	Can take time to architect
Complexity	More services



AKF Scale Cube



Pros	
Transaction Scalability	Scales transactions well
Latency	Reduces response times
Cache	Increases cache hit rate
Availability	Can have fault isolation

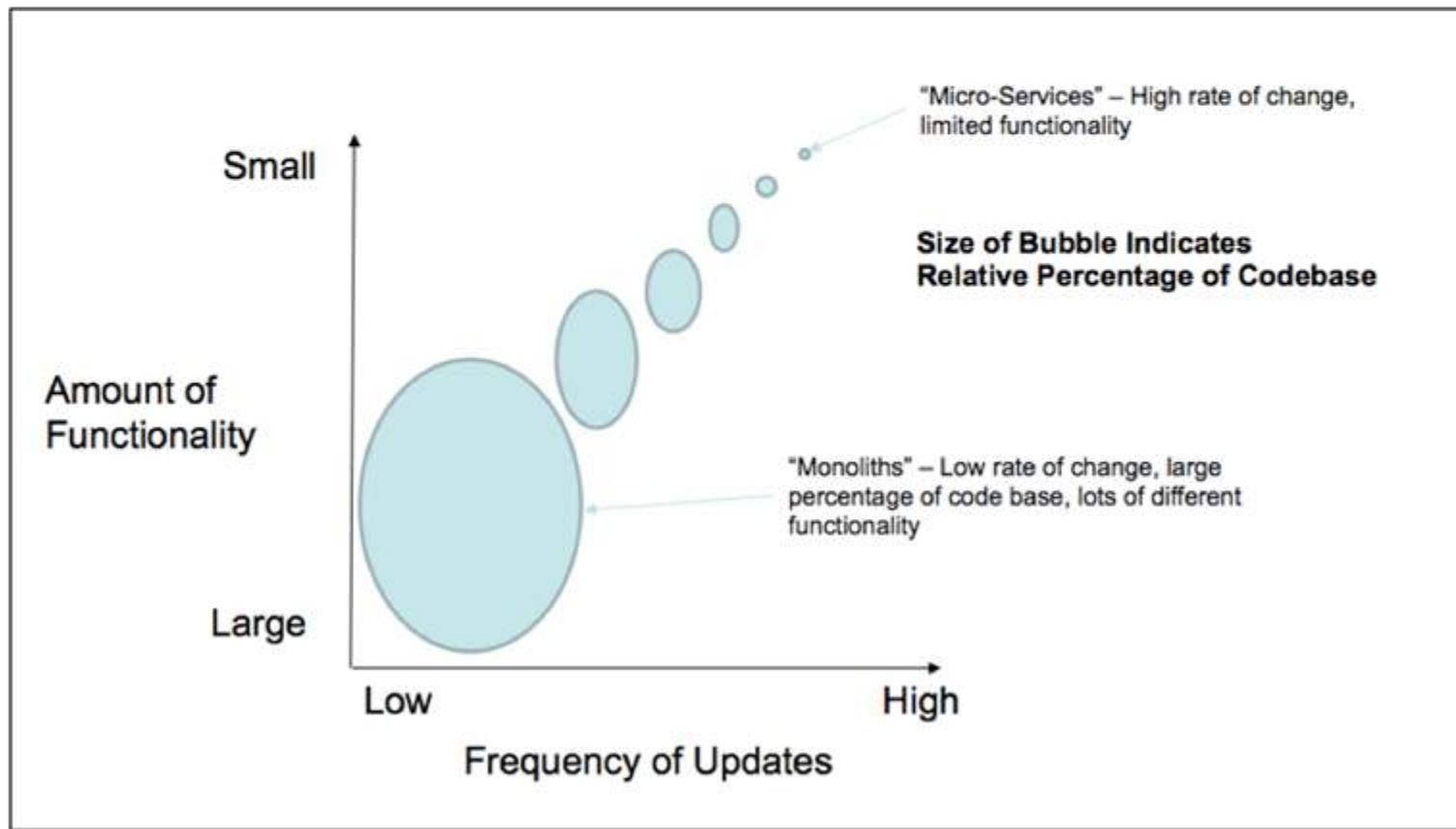
Cons	
Difficult	Can take time to architect
Complexity	More things to manage

Z-Axis Scalability: Segment by Customer			
NA		EU	
POD 1	POD 2	POD 3	POD 4

When splitting services, for the Y axis,
how far should one go?

Determining Service Size

The diagram represents the relationship between functionality, frequency of changes, and the relative percentage of the codebase across your system.



Determining Service Size



Common Questions and Considerations When Splitting

How Granular Should The Split Services Be? How Many Swimlanes Should We Split Out? Which Services Should Be Grouped Together?

There is a point where splitting too many of your services will yield little return or even cost you.

Developer Throughput

- Frequency of Change?
- Degree of Reuse?
- Team Size
- Specialized Skills?

Availability and Fault Tolerance

- Desired Reliability and Dependency on Other Functions?
- Criticality to the Business?
- Risk of Failure?

Cost

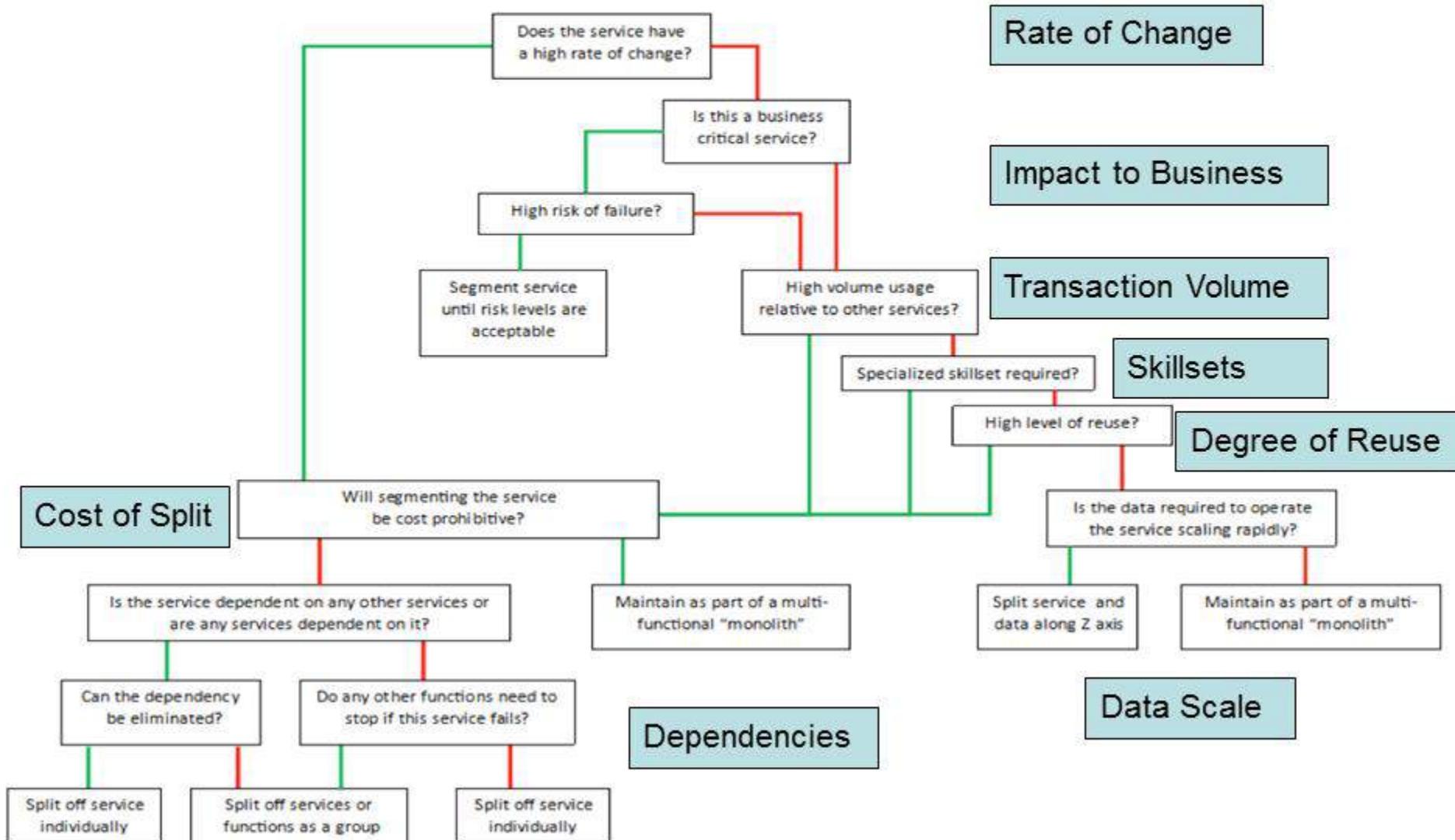
- Effort To Split Code?
- Shared Persistent Storage Tier?
- Network Configuration?

Scalability

- Scalability of Data?
- Scalability of Services?
- Dependency on Other Service's Data?

Determining Service Size

Decision Tree for Applying Splits to your Design



ARCHITECTURE PRINCIPLES FOR SCALE AND AVAILABILITY – TOP 10

Architecture Principles

The Top 10

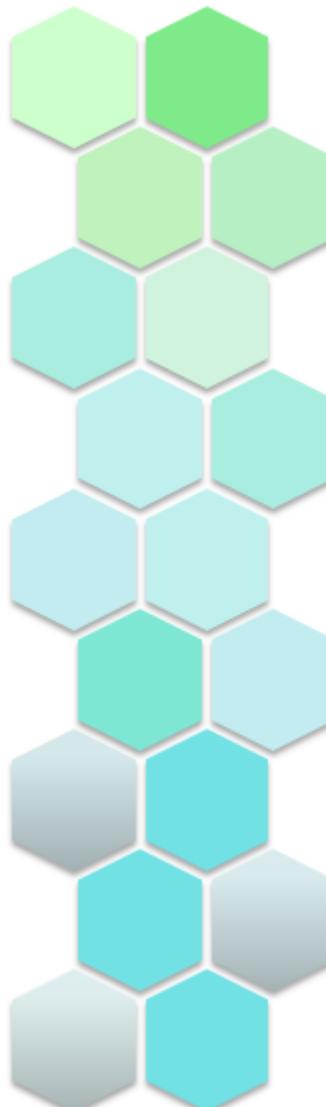
Design for Rollback. Ensure you can roll back any release of functionality.

Design to Be Monitored. Think about monitoring during design, not after.

Isolate Faults. Practice fault isolative design – implement circuit breakers to keep failures from propagating.

Scale Out Not Up. Never rely on bigger, faster systems.

Asynchronous Design. Communicate synchronously only when absolutely necessary.



Stateless Systems. Use state only when the business return justifies it.

Build Small, Release Small, Fail Fast. Build everything small and in iterations to grow.

Automation Over People. Build everything to be automated – never rely on people to do something that a robot can do.

Buy When Non-Core. If you aren't the best at building it and it doesn't offer competitive differentiation, buy it.

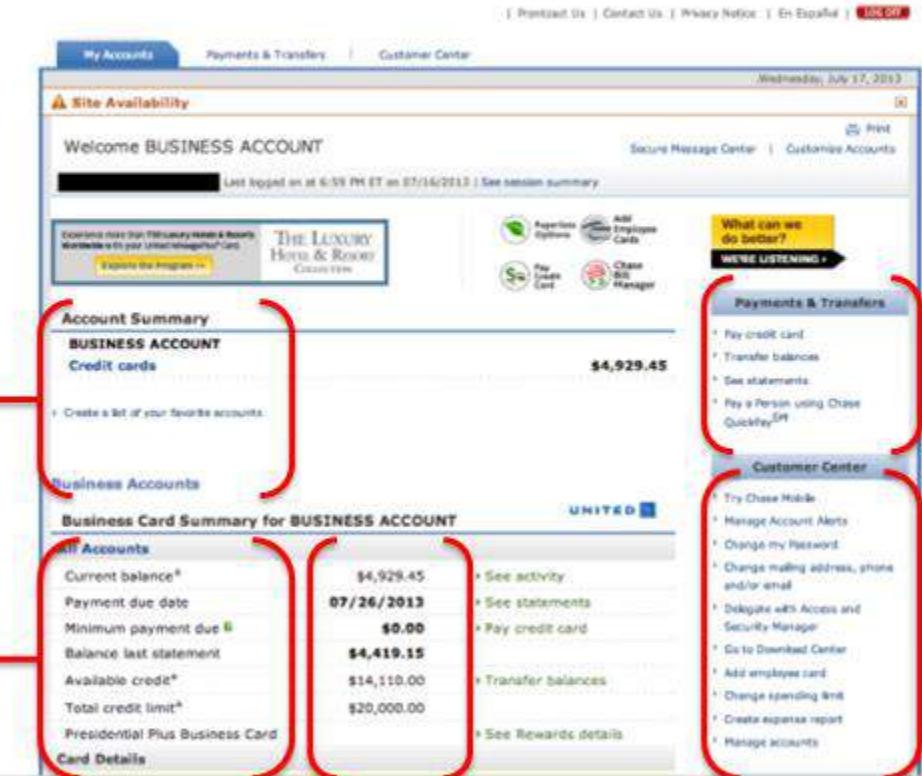
Design for at Least Two Axes. Think one step ahead of your scale needs.

Fault Isolation and Y Axis

Everything Fails.

We Must Implement Circuit Breakers to Contain Failures.

Y Axis Scalability:
Functionally Split Services & Fault Isolate. Portions of a page delivered from separate services.

The screenshot shows a complex web page with the following sections highlighted by red brackets:

- Site Availability:** Top banner area.
- Account Summary:** Shows a BUSINESS ACCOUNT with a credit card balance of \$4,929.45.
- Business Accounts:** A table showing a single account with a current balance of \$4,929.45, payment due date of 07/26/2013, and a total credit limit of \$20,000.00.
- Business Card Summary for BUSINESS ACCOUNT:** Shows a Presidential Plus Business Card with a balance of \$4,419.15.
- Payments & Transfers:** Sidebar menu with options like Pay credit card, Transfer balances, and See statements.
- Customer Center:** Sidebar menu with options like Try Chase Mobile, Manage Account Alerts, and Change my Password.

Each bracket represents a component of the page that fails or succeeds independently. A failure of that element does not bring down the entire page.

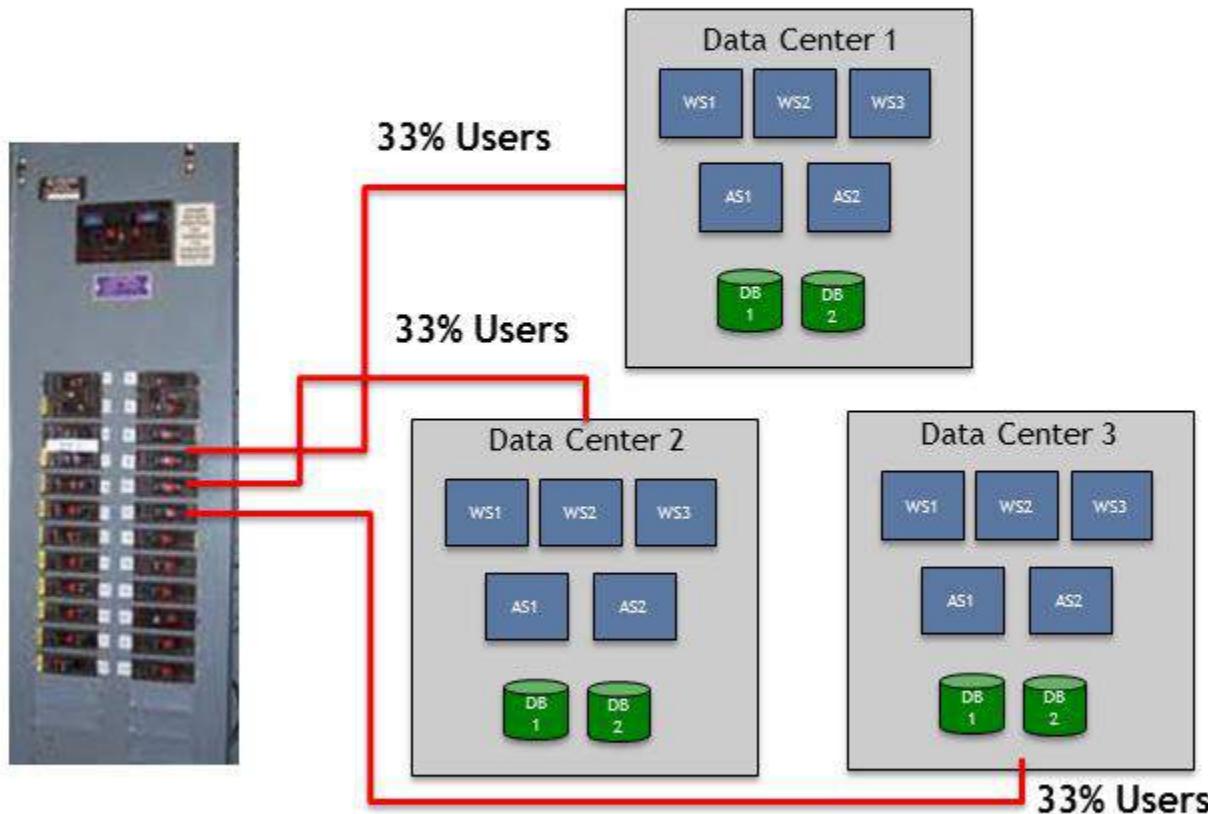
Fault Isolation and Z Axis

Everything Fails.

We Must Implement Circuit Breakers to Contain Failures.

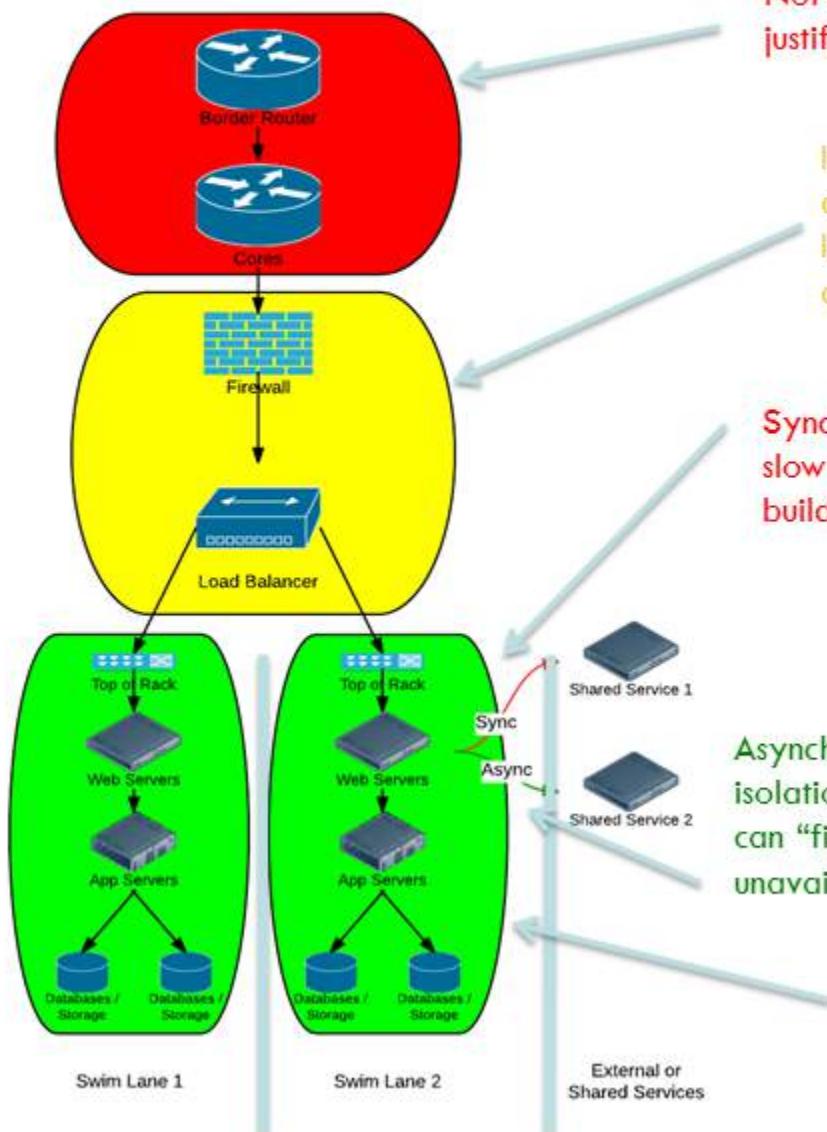
Z Axis Scalability:

Split Customers and assign them to data centers for fault isolation.



Separate customers into partitions. Make changes independently.

What constitutes a fault isolated swim lane?



Not Isolated – generally cost prohibitive to justify full isolation.

Isolate if Practical – firewalls and load balancers are common failure points, but are expensive. Ideally each swim lane will have its own firewalls and load balancers.

Synchronous calls to outside services cause problems when they are slow or unavailable. If you must make a synchronous call, consider building a copy of the service within the swim lane.

Asynchronous calls to outside apps and services allow for fault isolation – an app or service is designed for fault isolation when they can “fire and forget” or deal with the outside service being slow or unavailable.

Always isolate – Top of Rack Switches, Compute, and Storage should always be fault isolated within a swim lane.



SCALING ORGANIZATIONS

**DESIGNING FOR EMPOWERMENT,
ACCOUNTABILITY AND INNOVATION**

40+ Years of Research

Management and Technology researchers, theorists, professors and practitioners have known for more than 40 years that developing software based solutions in functional organizations is hazardous.

And yet, we have largely done nothing about it.

That is, until recently.

Drucker: Why is it that large companies can't seem to innovate (one exception)?

Drucker: Most innovation and most employment in the US from the SMB space (<500 employees).

Innovation and Entrepreneurship, 1986

Mel Conway (1968): Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

Conway's Law (*Popularized by Fred Brooks*)

Brooks' Corollary (1975): Because the design that occurs first is almost never the best possible, the prevailing system concept may need to change. Therefore, flexibility of organization is important to effective design.

Mythical Man Month

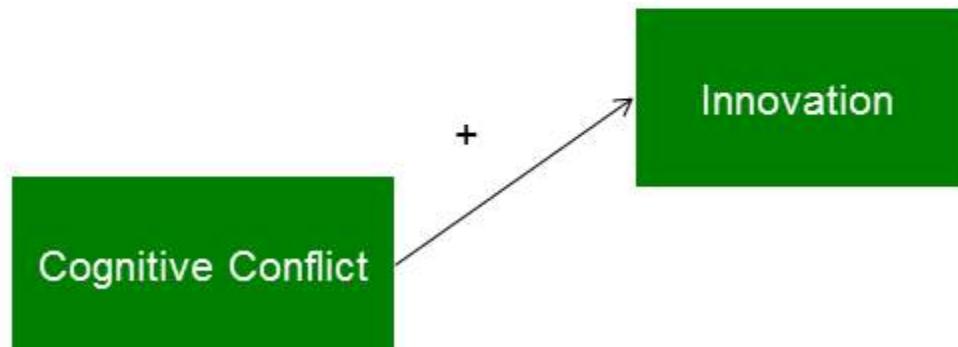
Steve Blank: Innovation comes from opportunity, chaos and rapid experimentation.

Why Innovation Dies

How does the structure of your organization
impact its ability to innovate?

Cognitive Conflict

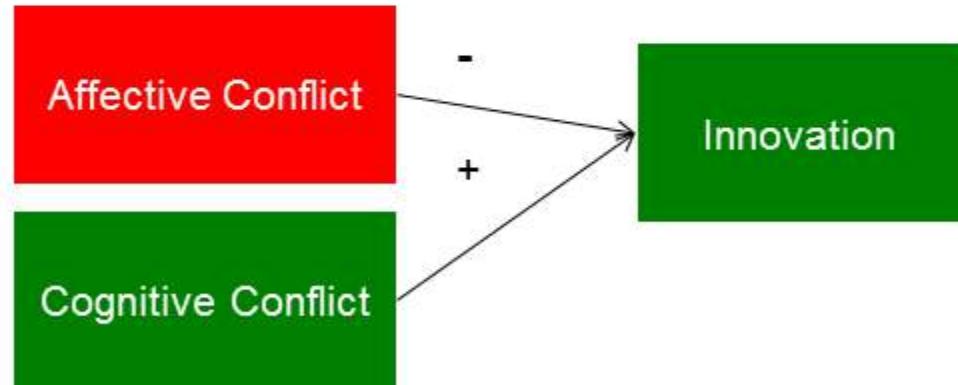
As cognitive conflict increases (what and why), so does innovation



**De Dreu and Weingart 2003

Affective Conflict

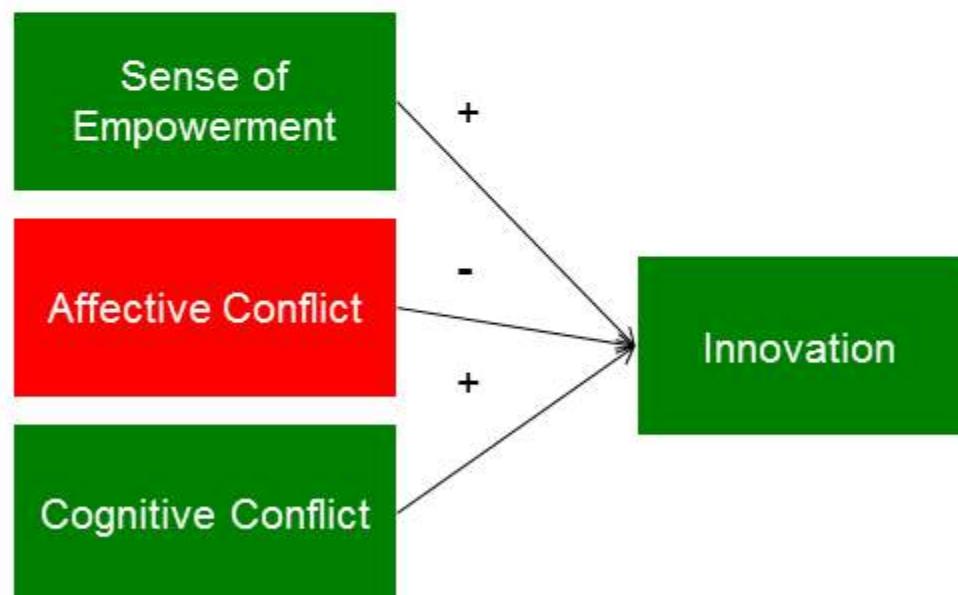
As affective conflict increases (who and how), innovation decreases



**Amazon and Mooney, 1999; De Dreu and Weingart 2003

Empowerment

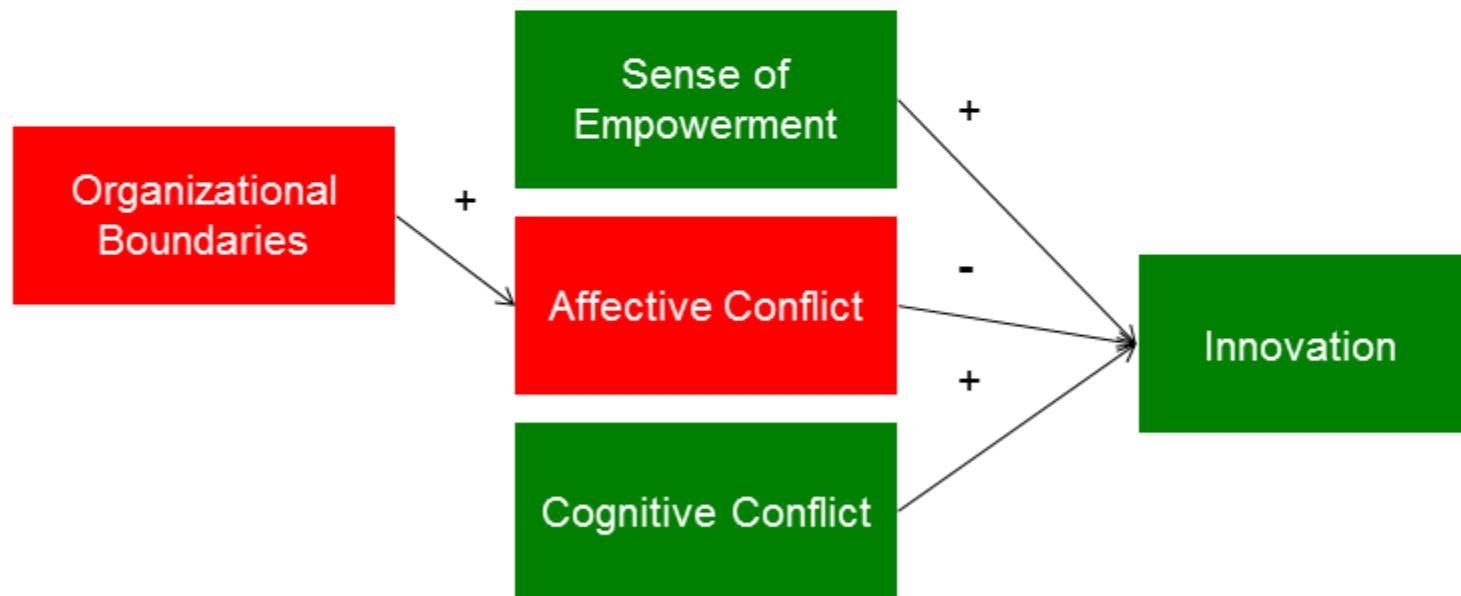
Feelings of empowerment increase innovation



**Spreitzer, De Janasz, 1999

Org Boundaries

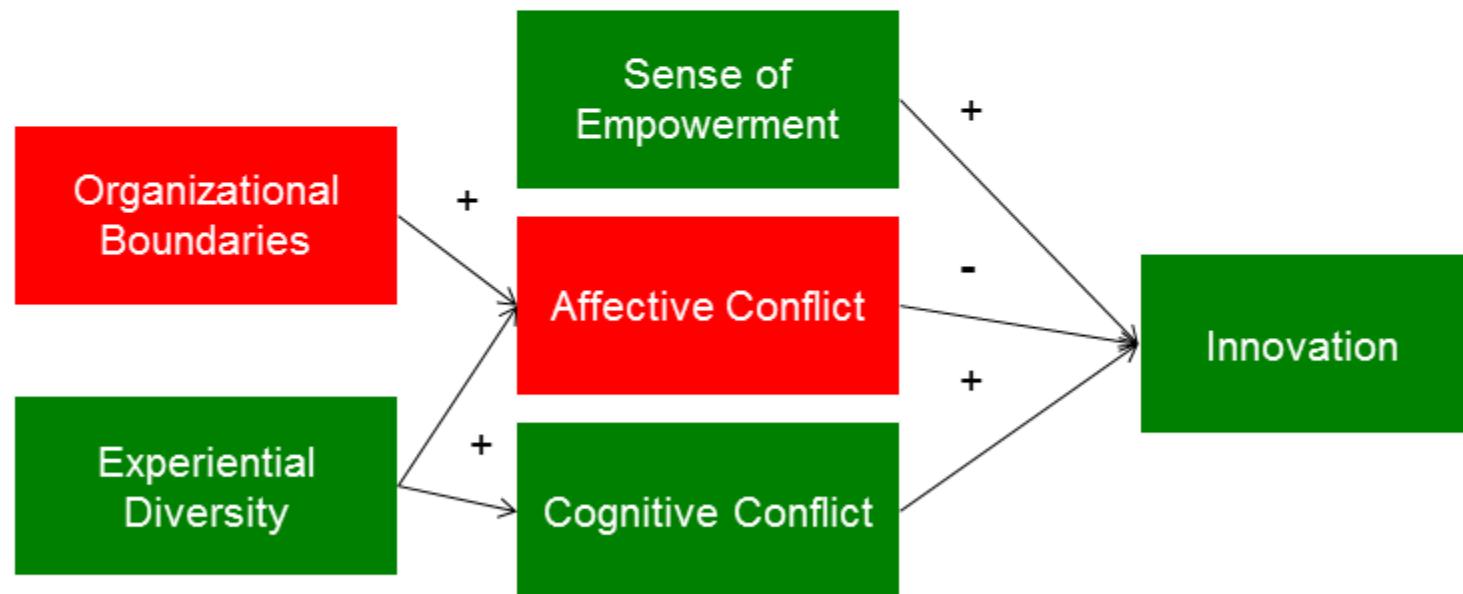
Organizational boundaries across which collaboration must happen increase affective conflict



** Tajfel 1974; Hogg and Terry 2000; Cummings and Kiesler 2005

Diverse Experience

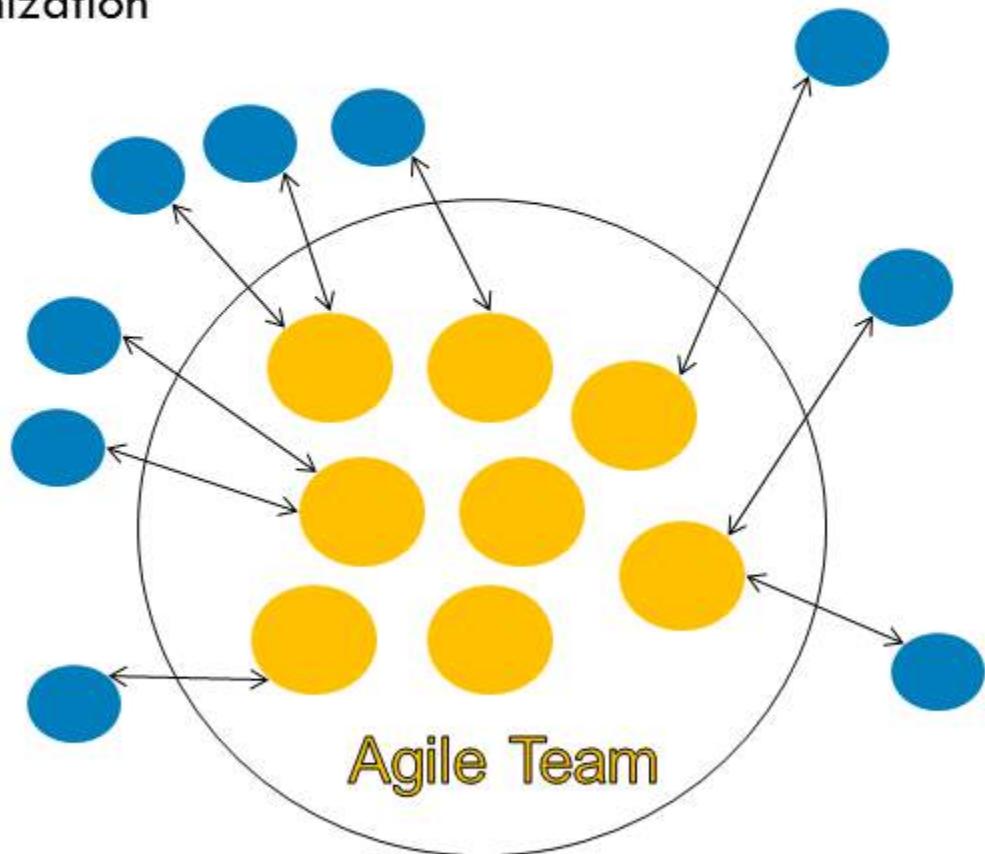
Both cognitive and affective conflict driven by diversity in experiences



** Burt 2001; Bassett Jones 2005; Abbott, Lyytinen et al. 2010

Network Diversity

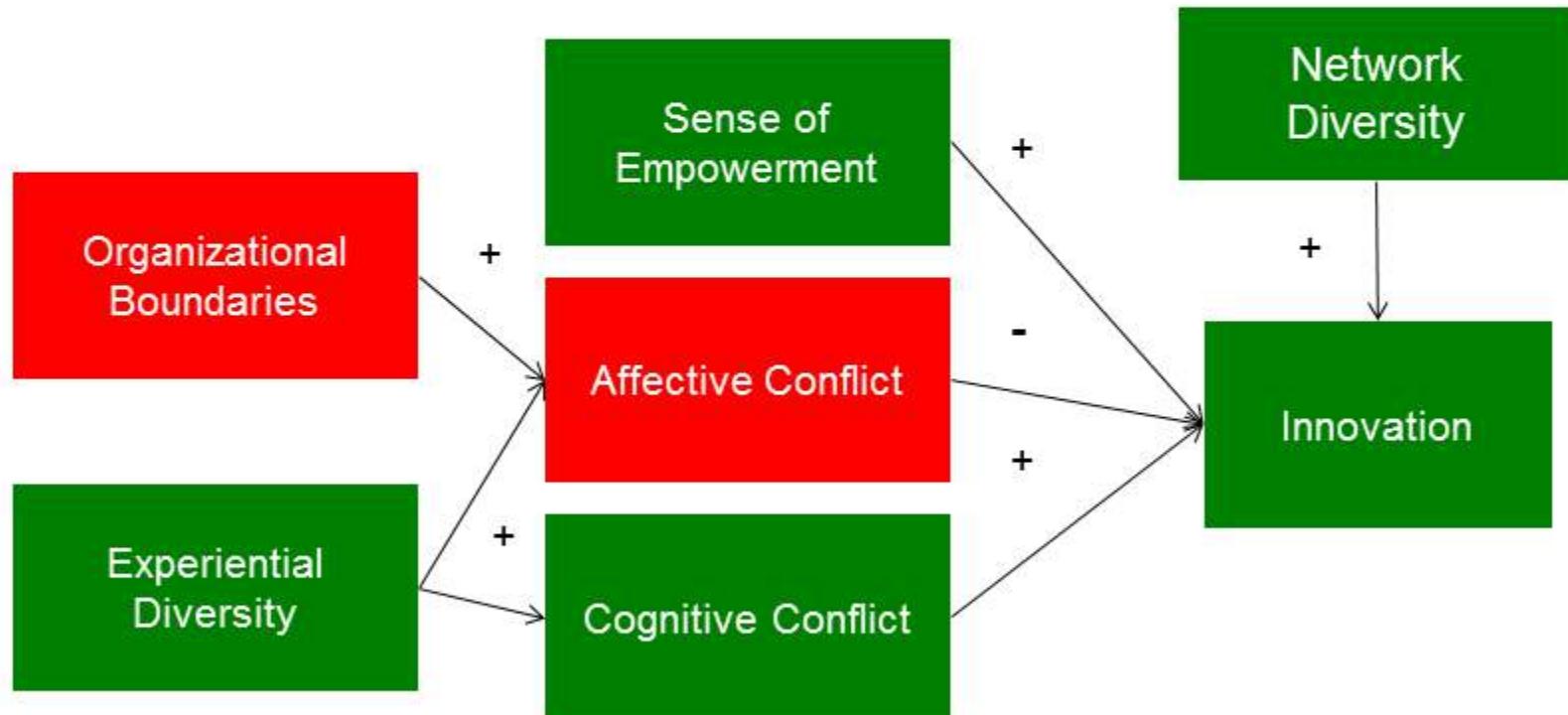
Best when high number of non-overlapping network links (loose acquaintances) outside of the organization



Network
Diversity

** Burt 2001; Bassett Jones 2005; Abbott, Lyytinen et al. 2010

Driving Innovation



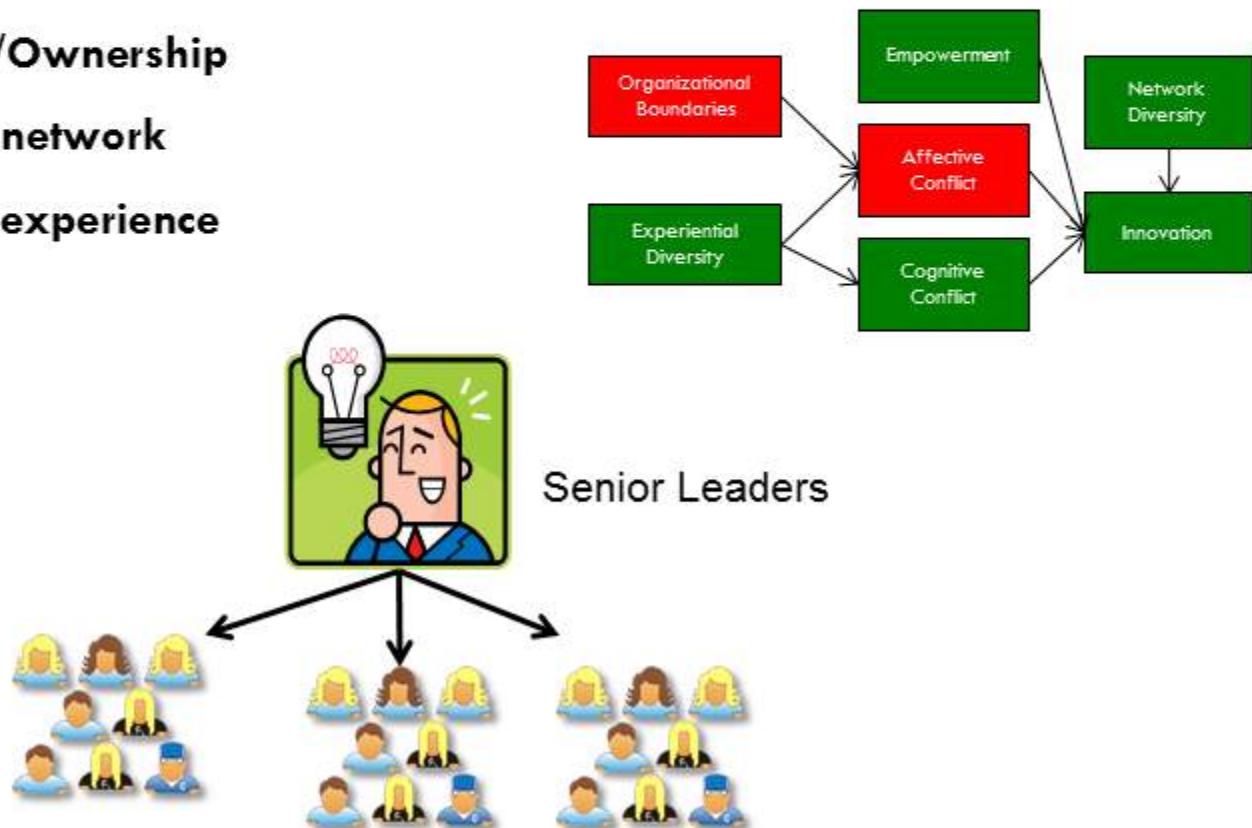
Barriers to Innovation

Top Down Innovation

This type of innovation should be avoided in favor of team driven innovation.

Empowering teams with KPIs/Goals as well as with all the necessary/diverse skillsets to achieve the goals will help drive innovation.

- No Empowerment/Ownership
- Does not leverage network
- Does not leverage experience

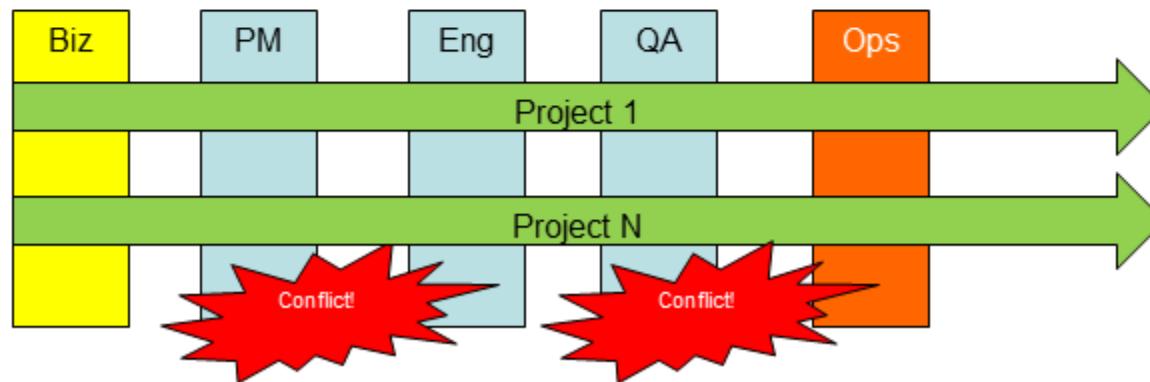
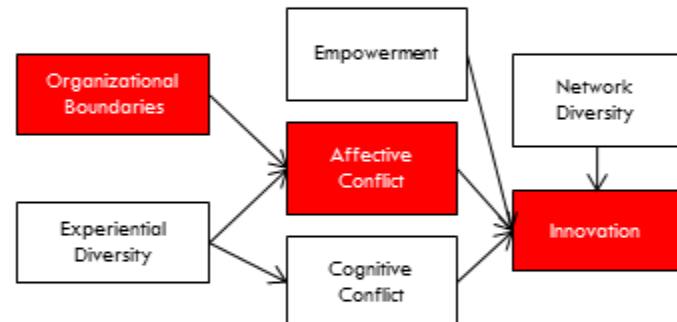


Barriers to Innovation

Functional Organizations

This type of organization structure promotes skill-based efficiency over innovation and empowerment.

- Organizational boundaries create conflict
- “Who and How” dominate instead of “What and Why” which slows innovation.

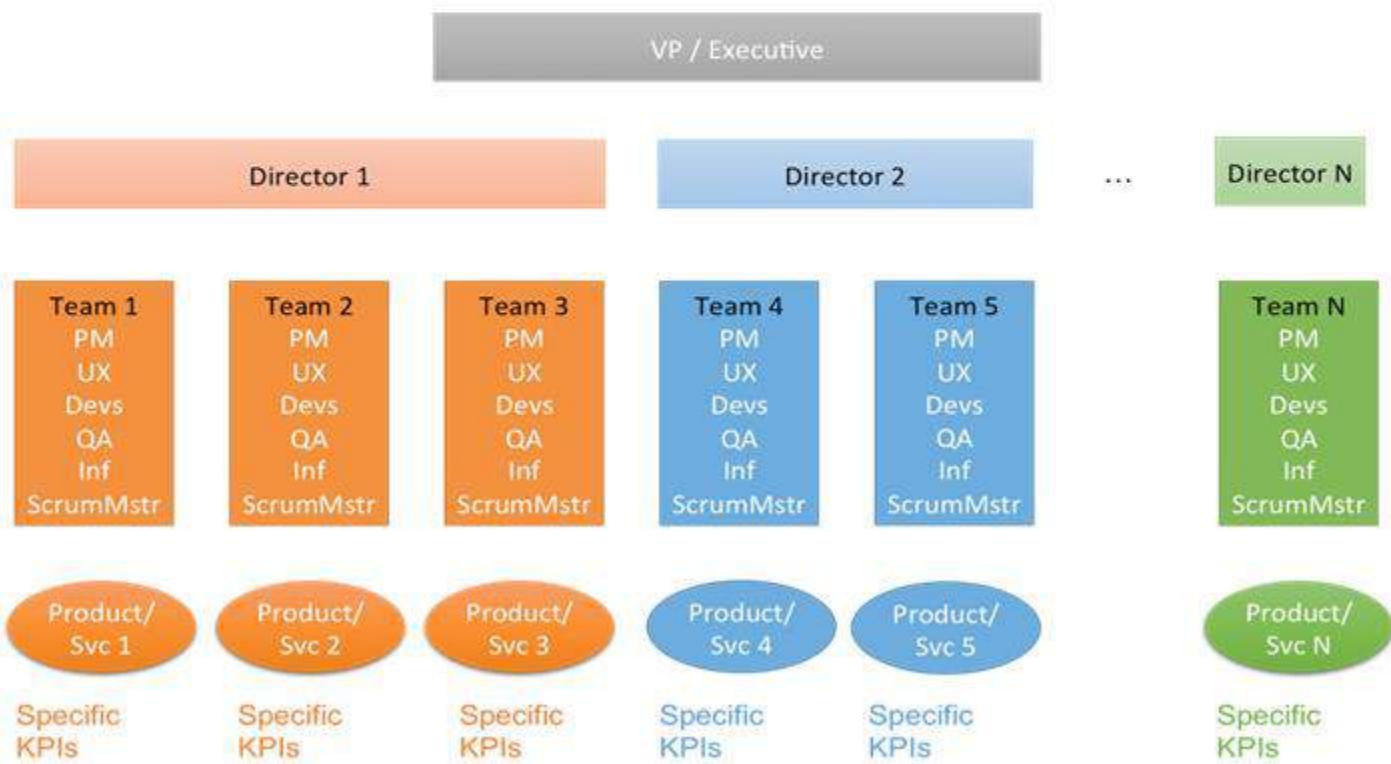


MULTIDISCIPLINARY AGILE TEAMS

Organization Options

Multidisciplinary Agile Teams

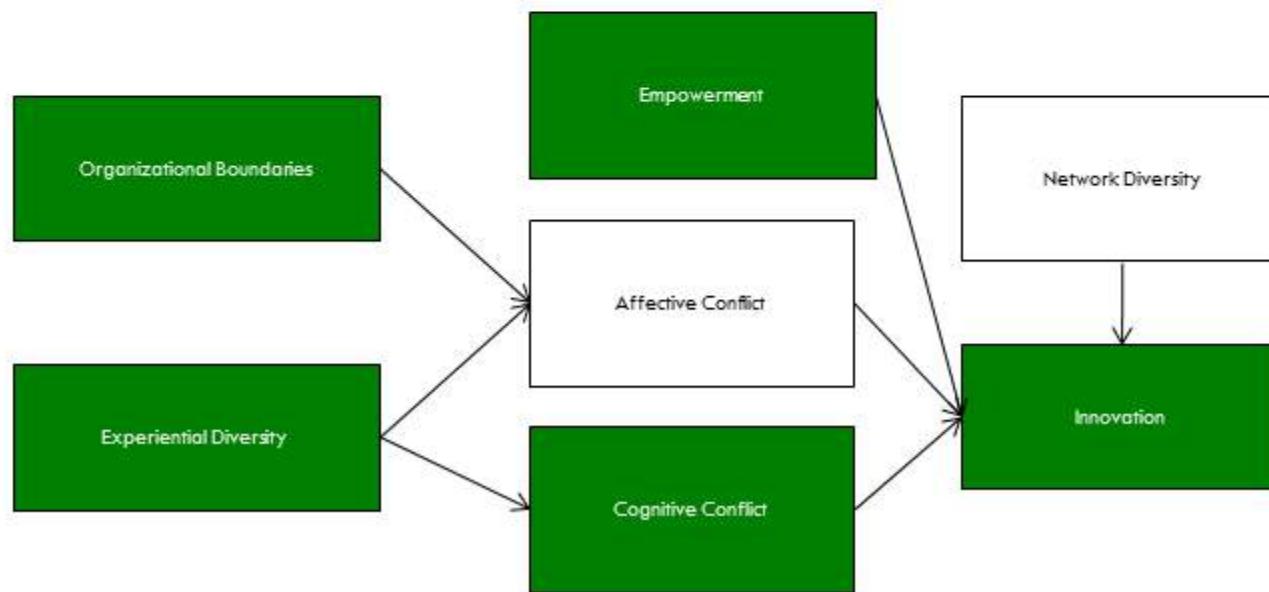
This type of organization encourages empowerment and innovation through diversity of experience in teams and lack of unnecessary organization boundaries.



Organization Options

Multidisciplinary Agile Teams

- Teams have all the skills necessary to achieve their goals and feel empowered
- Cognitive or “good” conflict increases



Organization Options

There are multiple options for organizations that we have seen in practice or worked with clients to leverage for their organization:

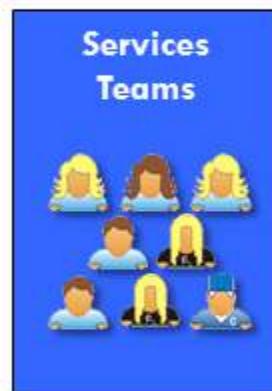
- “[NoOps](#)” Model which promotes automation heavily
- “[Full Stack](#)” Model which promotes end-to-end capability within an Agile team to delivery a service
- “[Squads, Chapters & Guilds](#)” Model which promotes autonomy between teams as if they were mini startups while encouraging communication and knowledge sharing across the various groups

No one model normally fits our clients exactly and there are pros and cons to each that need to be considered when thinking of the optimal structure.

Model selection is [not an “only choose one” decision](#). You can utilize elements of all 3 to find the right organization structure for your team.

“NoOps” Model

- The “NoOps” model is most often associated with the Netflix streaming service and its **near 100% use of the AWS Cloud**.
- High Scalability and fast **Time to Market** are primary goals of this model. Cost management is secondary.
- Budget and responsibility of figuring out how to use public cloud IaaS was given directly to the Development organization after failed attempts to quickly scale out data centers during periods of extreme growth.
- Relies heavily on **near 100% automation** for all critical development and releases processes.
- Netflix has a developer-oriented culture starting with the CEO Reed Hastings who was a developer.



CORE Team (Cloud Operations Reliability Engineering)

Responsible for:

- Monitoring
- Application Performance Management



“NoOps” Model

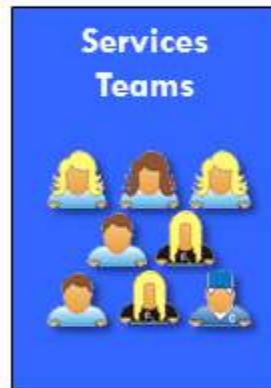


“A handful of DevOps engineers ...are coding and running the build tools and bakery, and updating the base AMI from time to time. Several hundred development engineers use these tools to build code, run it in a test account in AWS, then deploy it to production themselves. They never have to have a meeting with ITops, or file a ticket asking someone from ITops to make a change to a production system, or request extra capacity in advance... This is part of what we call NoOps. The developers used to spend hours a week in meetings with Ops discussing what they needed, figuring out capacity forecasts and writing tickets to request changes for the datacenter. Now they spend seconds doing it themselves in the cloud... ”

There is no central control, the teams are responsible for figuring out their own dependencies and managing AWS security groups that restrict who can talk to who.”

— Adrian Cockcroft

<http://perfcap.blogspot.com/2012/03/ops-devops-and-noops-at-netflix.html>



CORE Team (Cloud Operations Reliability Engineering)

Responsible for:

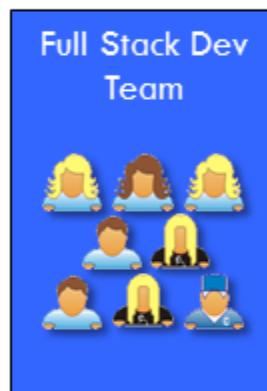
- Monitoring
- Application Performance Management



“Full Stack” Model



- Minimizing risk and moving fast are primary goals of this model.
- Teams responsible for services have all of the necessary skill to be innovative and release frequently. They have knowledge of UI/UX, Mid-Tier services, Databases, Hosting environments, Networks, etc. (i.e. “Full Stack” development teams)
- Facebook also has a sizeable Release Engineering team that assists with releases multiple times daily and weekly.
- Facebook Infrastructure Engineering is a separate organization that is responsible for data centers, capacity planning, etc.
- Facebook has a developer-oriented culture starting with the CEO who was a developer.



Full Stack Dev Team

Release Engineering

Responsible for:

- Controlling releases
- Daily pushes
- Weekly pushes

Infrastructure Engineering

- Data Centers
- Server build out
- Capacity
- Performance
- Incident Management

“Full Stack” Model



“None of the previous principles work **without operations and engineering teams that work together seamlessly**, and can handle problems together as a team. The person responsible for something must have control over it. At Facebook we push code to the site every day, and **the person who wrote that code is there to take responsibility for it**.

.. We have three people working on photos. Each of those three people knows photos inside and out, and can make decisions about it. So when something needs to change in photos they get it done quickly and correctly.“

– Peter Deng

https://www.facebook.com/note.php?note_id=409881258919

Full Stack Dev Team



Release Engineering

Responsible for:

- Controlling releases
- Daily pushes
- Weekly pushes

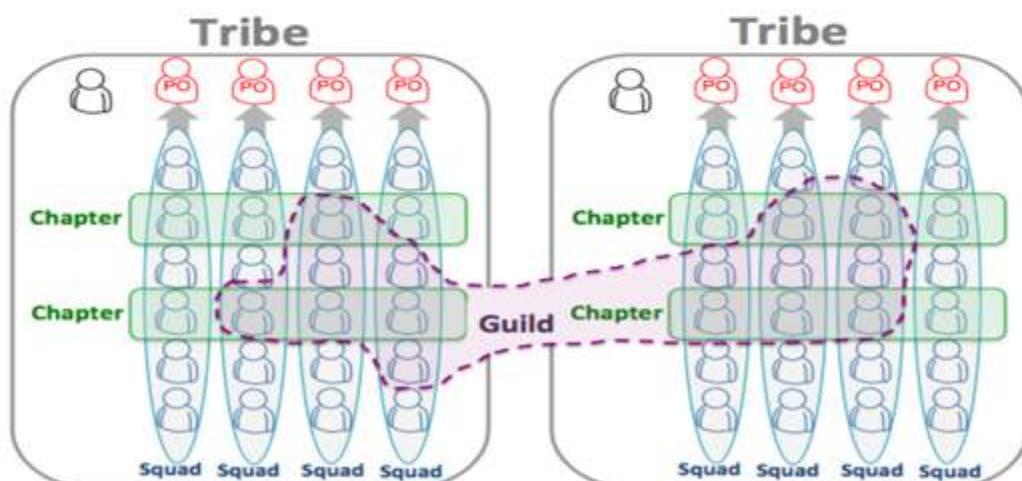
Infrastructure Engineering

- Data Centers
- Server build out
- Capacity
- Performance
- Incident Management

“Squads, Chapters & Guilds”



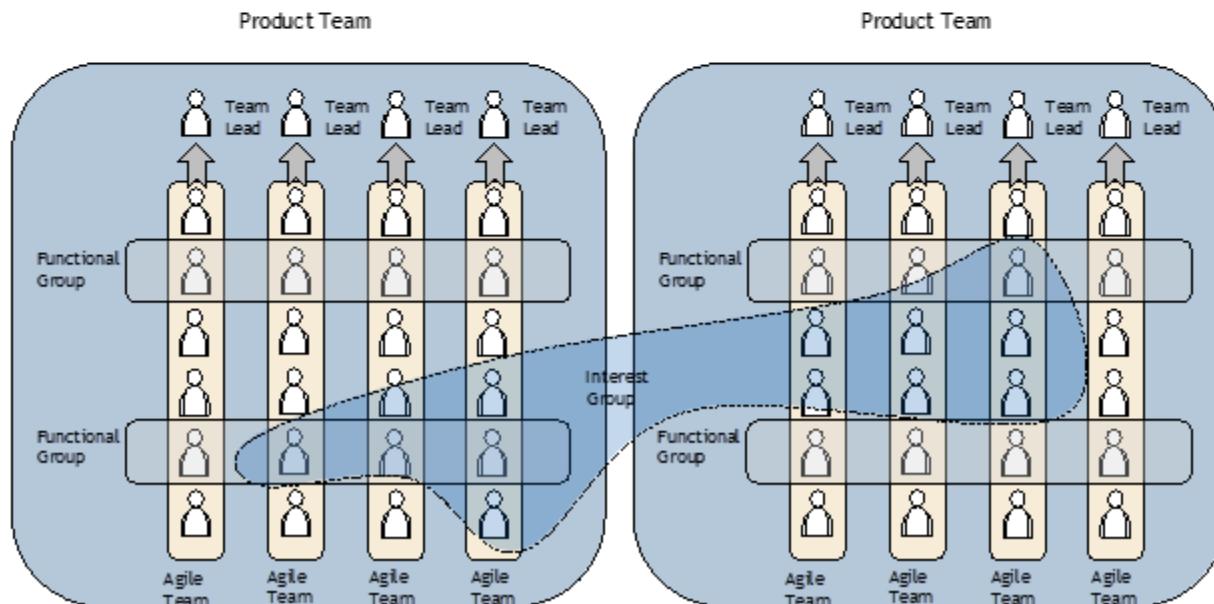
Spotify Organizational Structure – A real-world example of a service oriented, cross-functional team can be found at Spotify. Their organizational structure is not functionally aligned but rather is organized into small Agile teams, which they refer to as Squads. These are self-contained teams organized around a deliverable or service which the business provides. The following image and descriptions are taken from Henrik Kniberg and Anders Ivarsson's October 2012 paper “Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds.”



Product & Agile Teams

Product Teams (“Tribes”) and Agile Teams (“Squads”)

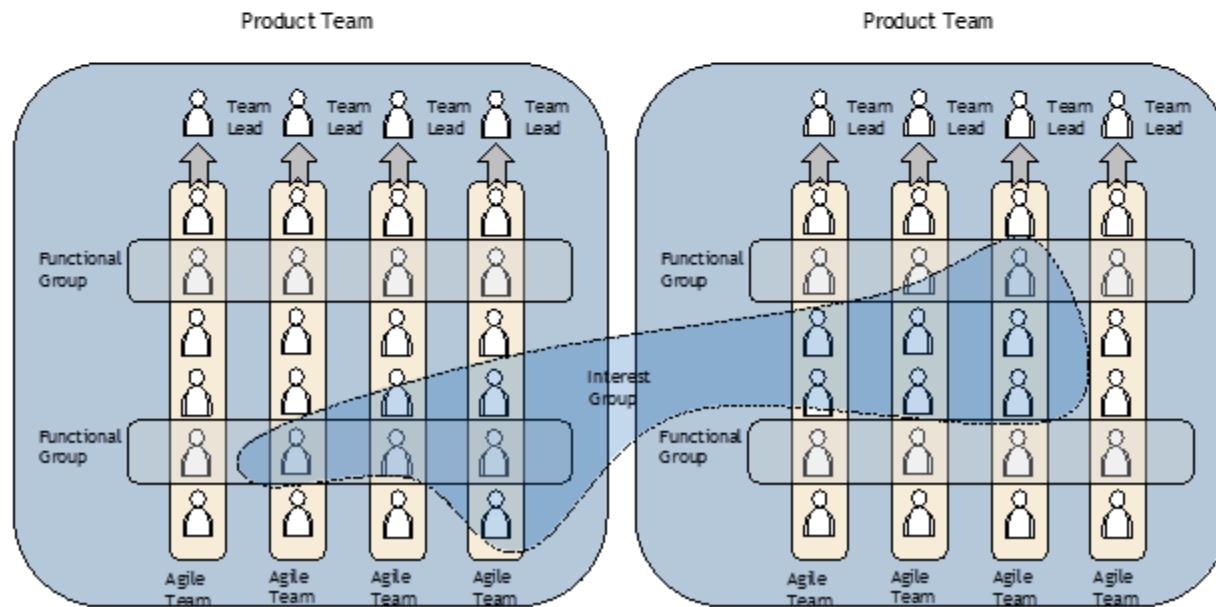
- A Product Team is a collection of Agile Teams that work in related areas. The Product Team can be seen as the “incubator” for the mini-startups focused on services.
- Each Product Team has a lead who is responsible for providing the best possible habitat for the Agile Teams within that Product Team.
- Agile Teams within a Product Team are all physically in the same office, normally right next to each other.
- Product Teams are less than 100 people in total.



Functional Groups

Functional Groups (“Chapters”)

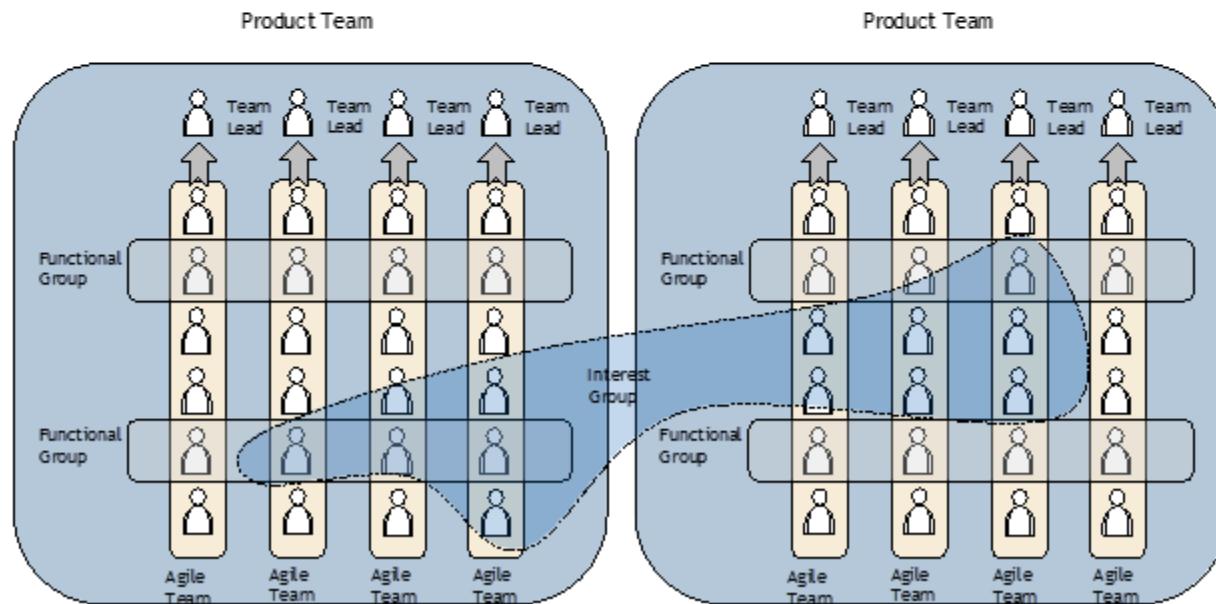
- Small group of people having similar skills and working within the same general competency area, within the same Product Team.
- Each Functional Group meets regularly to discuss their area of expertise and their specific challenges.
- The Functional Group lead is a line manager for her Functional Group members, with all the traditional responsibilities such as developing people, setting salaries, etc.
- However, the Functional Group lead is also part of a Agile Team and is involved in the day-to-day work, which helps her stay in touch with reality.



Interest Groups

Interest Groups (“Guilds”)

- An Interest Group is a more organic and wide reaching “community of interest”, a group of people that want to share knowledge, tools, code, and practices.
- Functional Groups are always local to a Product Team, while an Interest Group usually cuts across the whole organization. Some examples are: the web technology guild, the tester guild, the agile coach guild, etc.



Key Takeaways

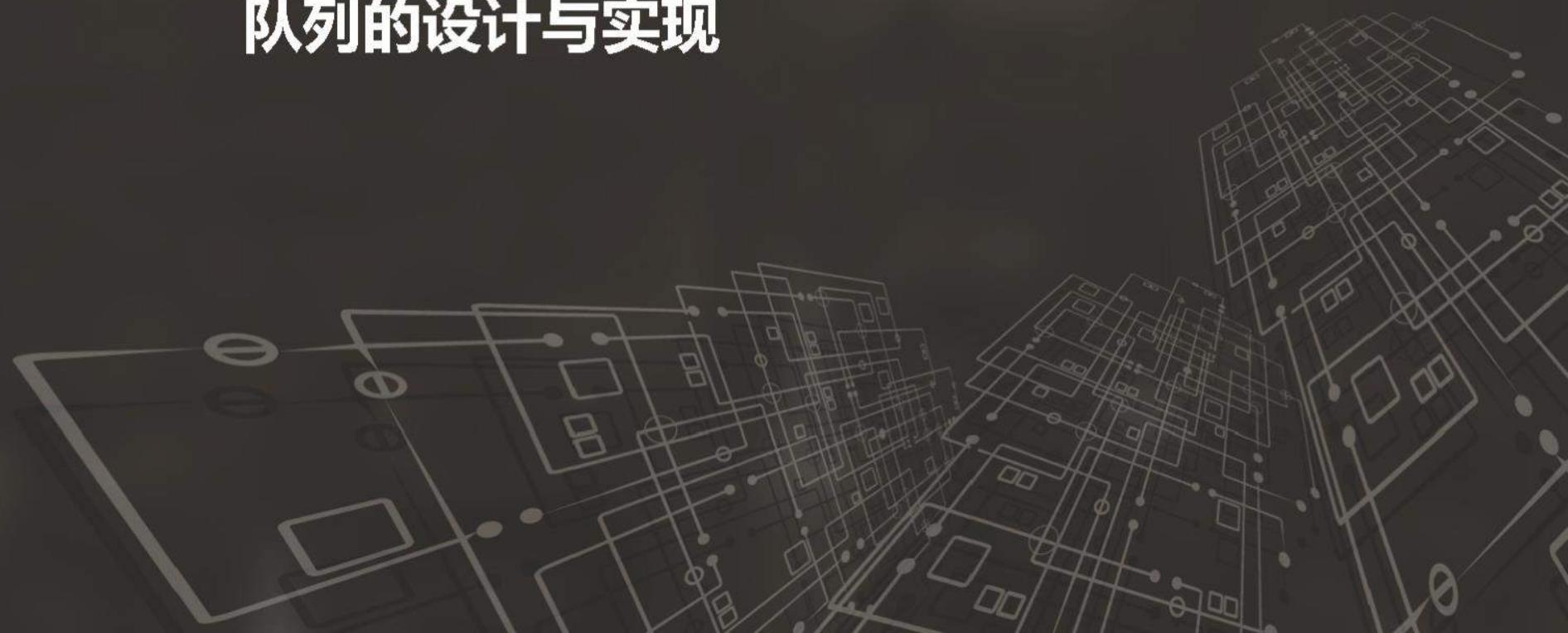
For fastest Time to Market, Best Quality, Best Organizational Scale and Highest Levels of Innovation, teams must:

- Cross functional – have all the skills embedded within the team to do the job they are asked to do.
- Need very little communication with other teams to do their job.
- Work against outcomes – not requirements or tasks
- Own the outcomes AND the solutions to achieve them – no shared ownership of solutions (software and hardware)

2017 Software Architecture Summit

PhxQueue

微信开源高可用强一致分布式 队列的设计与实现



自我介绍

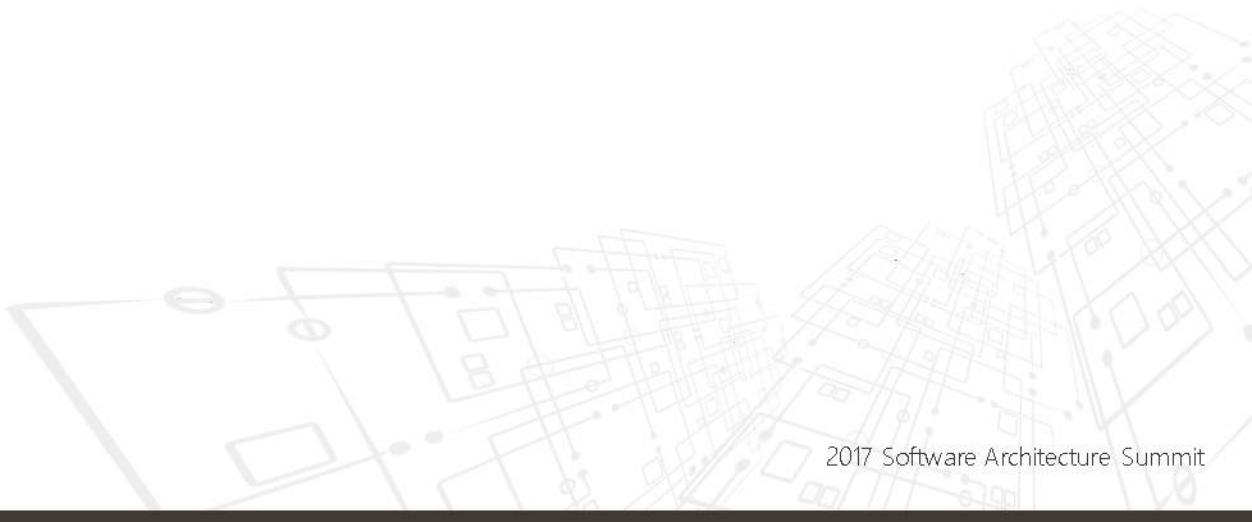
- 梁俊杰 Unixliang
- 微信 基础消息组
- 2011年11月加入腾讯
 - 曾负责微博
 - 私信
 - 反垃圾系统
 - 现负责微信
 - 消息系统
 - 跨域同步组件
 - 队列中间件

听众受益

- 了解微信后台分布式消息队列的架构演进
- 了解 PhxQueue 的功能特性
- 了解 PhxQueue 设计与实现细节
- 了解微信后台消息队列最佳实践

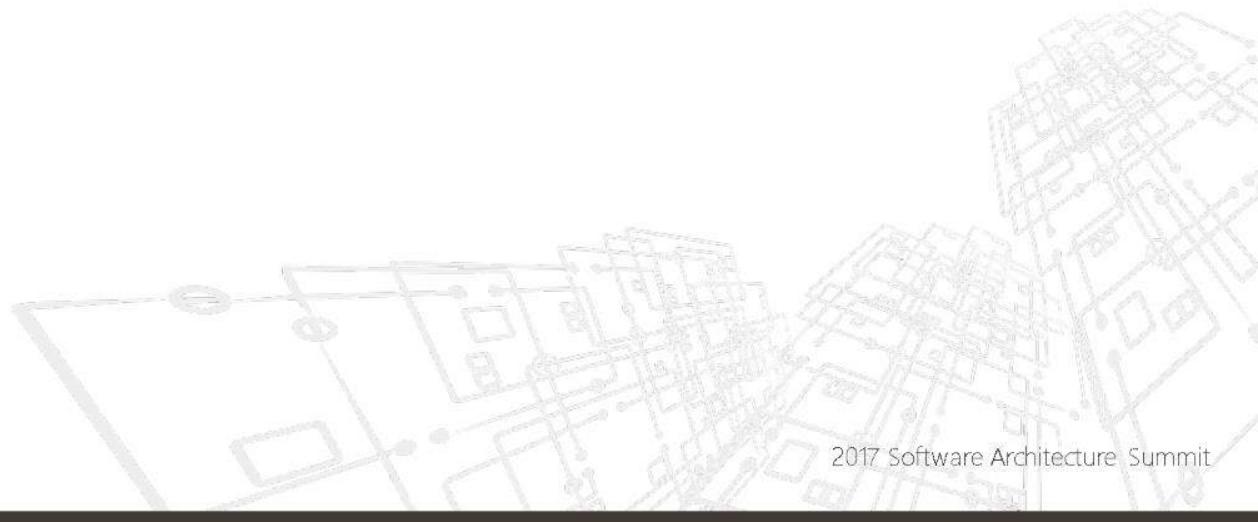
目录

- 背景
- 设计
- 实现
- 最佳实践
- 效果展示
- 总结



目录

- 背景
 - 设计
 - 实现
 - 最佳实践
 - 效果展示
 - 总结
- 微信内部队列使用场景
 - 旧队列介绍
 - 旧队列问题



微信内部队列使用场景

业务解耦

- 发布订阅模式
- 消息总线

分布式事务

- 拆分为多个本地事务
- 可靠消息传递

削峰和流控

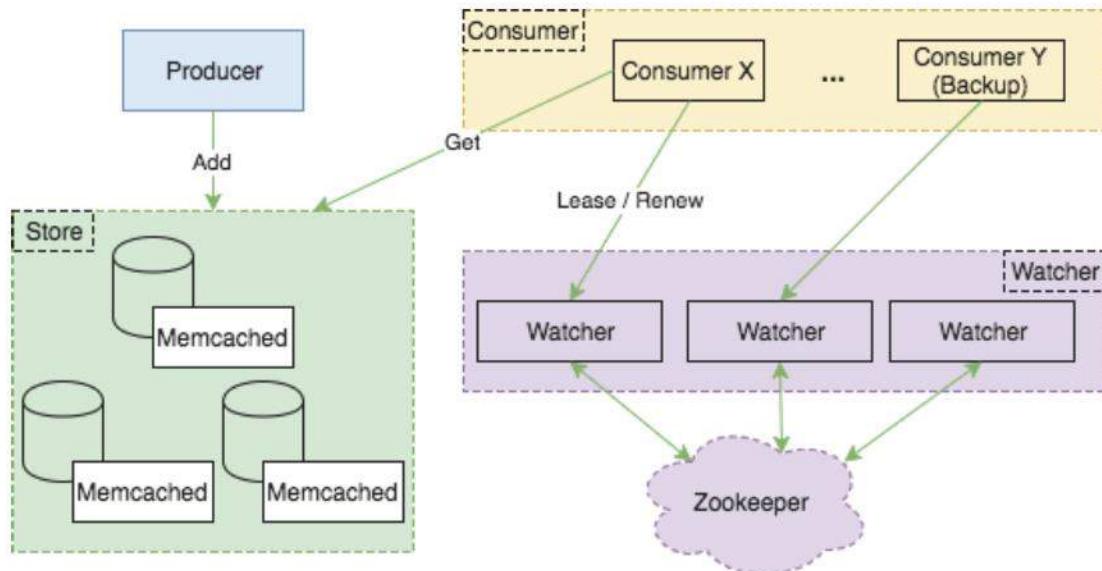
- 缓存突发消息
- 按能力消费

延迟消费

- 定时任务 - 延后推
- 离线任务 - 随时拉

目前PhxQueue广泛支持微信支付、公众平台等多个重要业务。
日均入队达千亿，分钟入队峰值达一亿。

旧架构及存在问题



典型分布式队列架构

- Producer – 生产者
- Store – 存储(NRW)
- Consumer – 消费者
- Zookeeper – 分布式锁

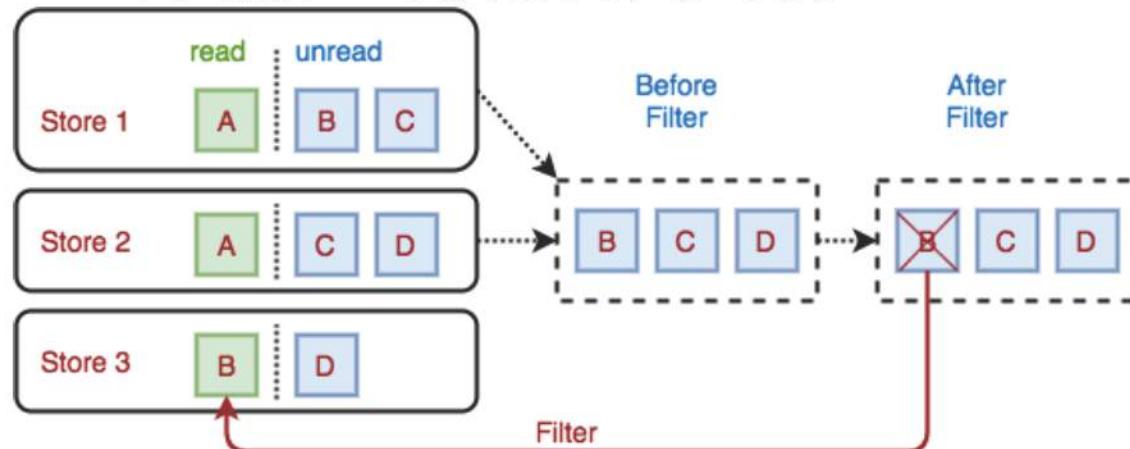
旧架构及存在问题

Store 读写

- N=3, W=2, R=2

Store 读去重

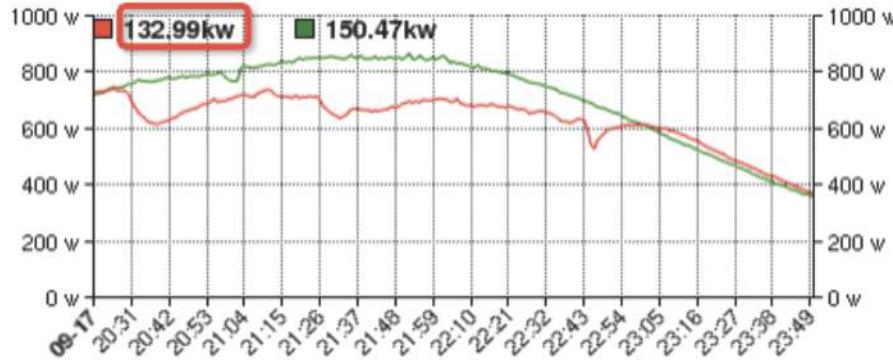
- Filter=3
- Memcached 维护数据MD5到其偏移的对应关系



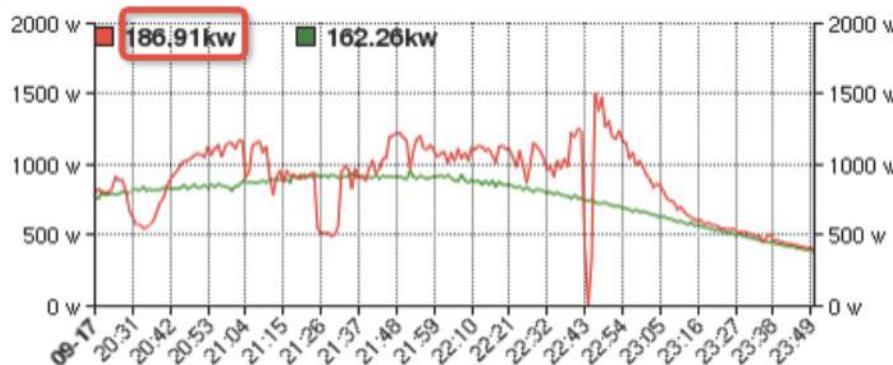
旧架构及存在问题

1. 积压下去重失效

- 重复消费率达**40.9%**
- 原因：
 - NRW需依赖Memcached去重
 - Memcached满容量则去重失效
- 影响：
 - 降低积压清理效率
 - 部分业务幂等失效



(图1) 入队数

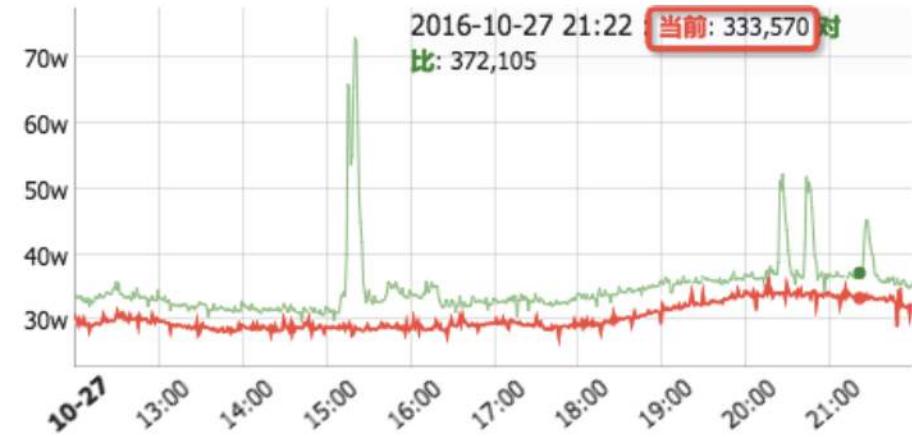


(图2) 出队数

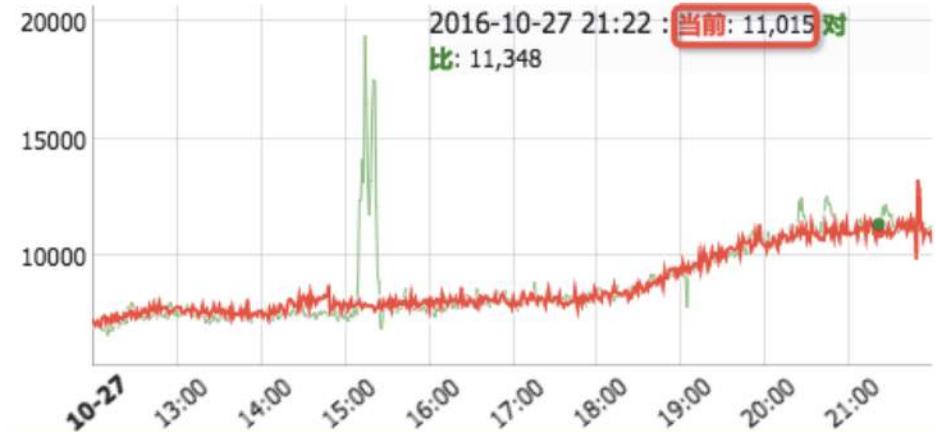
旧架构及存在问题

2. 出队乱序

- 出队乱序率达3.37%
- 原因：NRW天然乱序缺陷
- 影响：影响用户体验



(图1) 出队数



(图2) 出队乱序数

旧架构及存在问题

3. 负载均衡效果差

- 原因：
 - 负载均衡与分布式锁逻辑耦合
 - 主动变更锁会导致一段时间重复出队，不可取，从而限制了均衡时机
- 影响：Consumer单机负载过高影响可用

4. 丢数据风险增加

- 原因：异步刷盘
- 故障概率增加：
 - 机器过保
 - 部分地区无法三DC部署

最高负载Consumer:

服务器名	load	总使用率
mmpayhbastrosched18	3.06	12%
mmpayhbastrosched20	2.83	12%

最低负载Consumer:

mmpayhbastrosched26	0.17	1%
mmpayhbastrosched28	0.19	0%

(图) Consumer负载不均衡

目录

- 背景
- 设计
 - 拥抱Paxos
 - 拆分分布式锁、负载均衡服务
- 实现
- 最佳实践
- 效果展示
- 总结

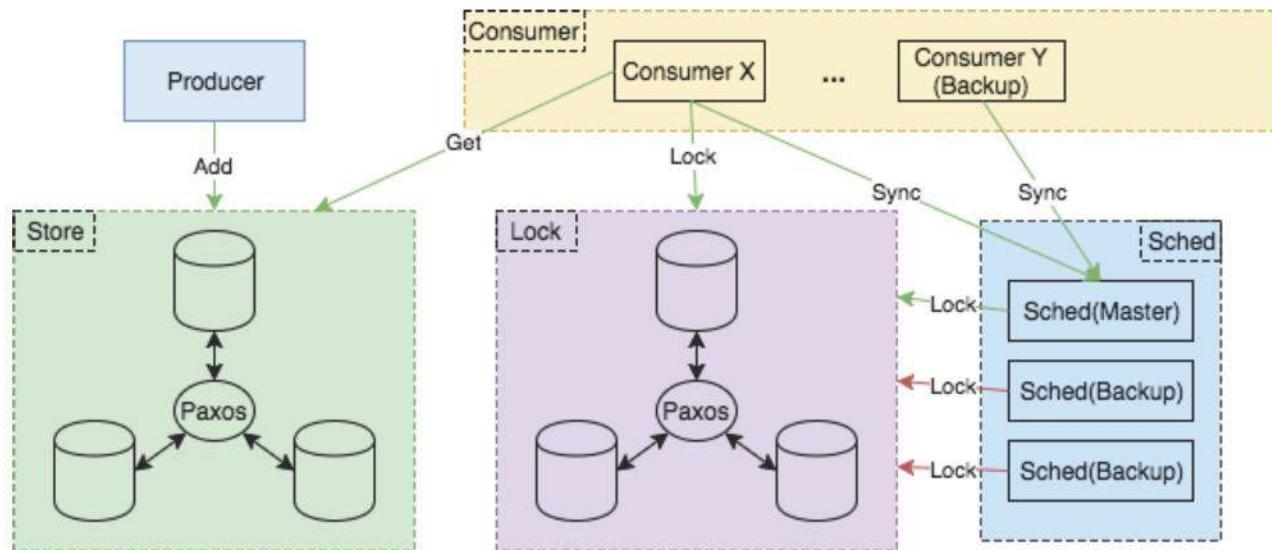


新架构(PhxQueue)设计

主要特性

- 同步刷盘，入队数据绝对不丢，自带内部实时对账
- 出入队严格有序
- 多订阅
- 出队限速
- 出队重放
- 所有模块均可平行扩展
- 存储层批量刷盘、同步，保证高吞吐
- 存储层支持同城多中心部署
- 存储层自动容灾/接入均衡
- 消费者自动容灾/负载均衡

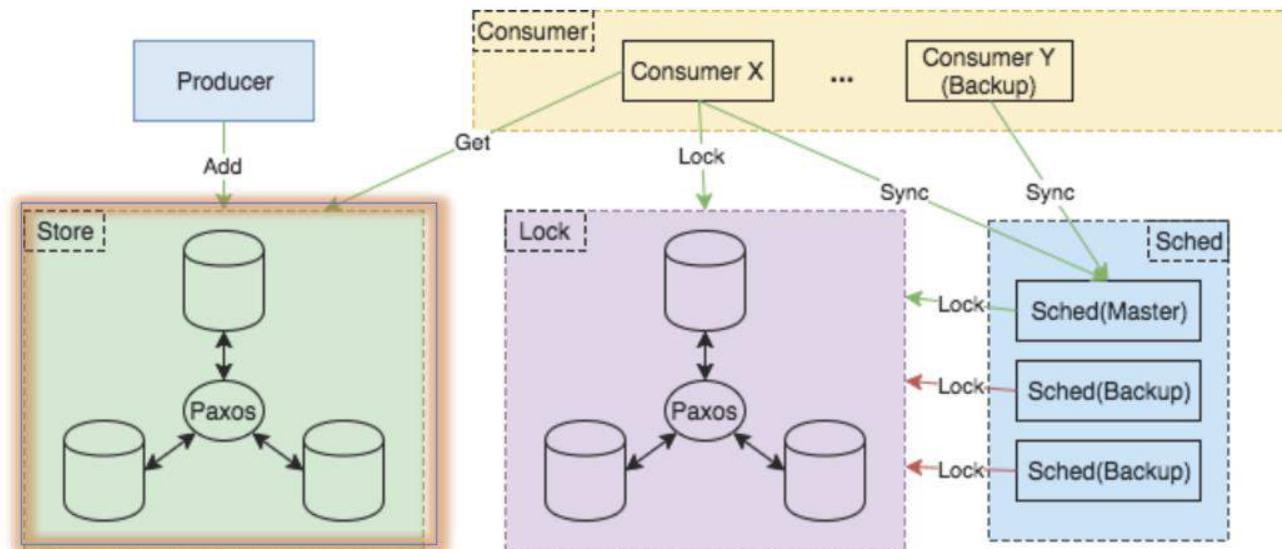
新架构(PhxQueue)设计



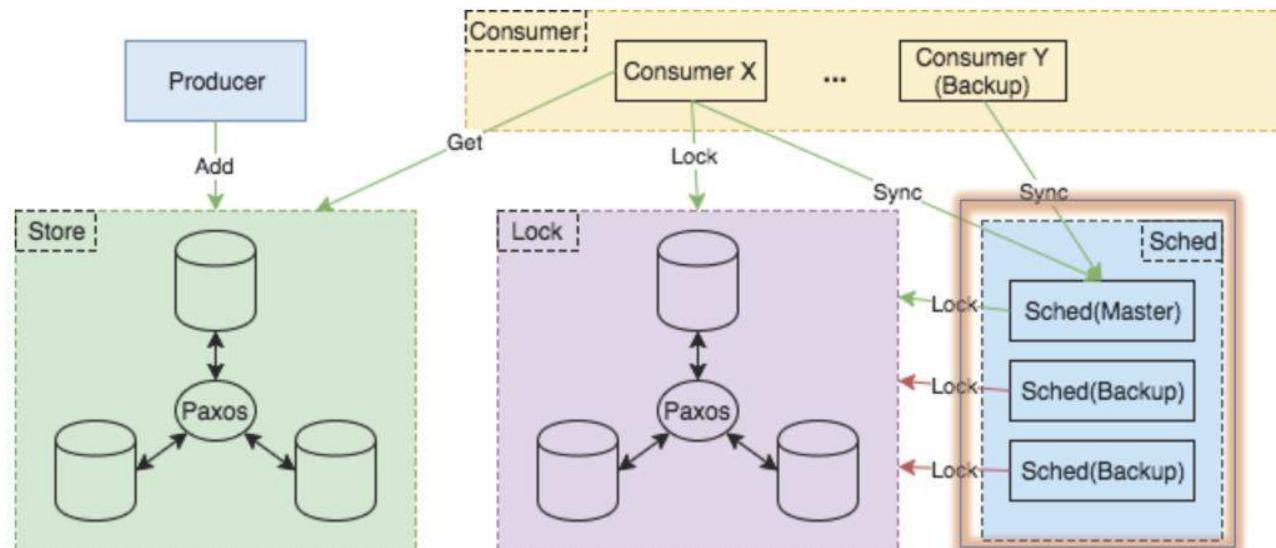
新架构(PhxQueue)设计

Store – 队列存储

- 引入 Paxos
- 同步刷盘
- 对外强一致
- 无乱序，免去重



新架构(PhxQueue)设计



Store – 队列存储

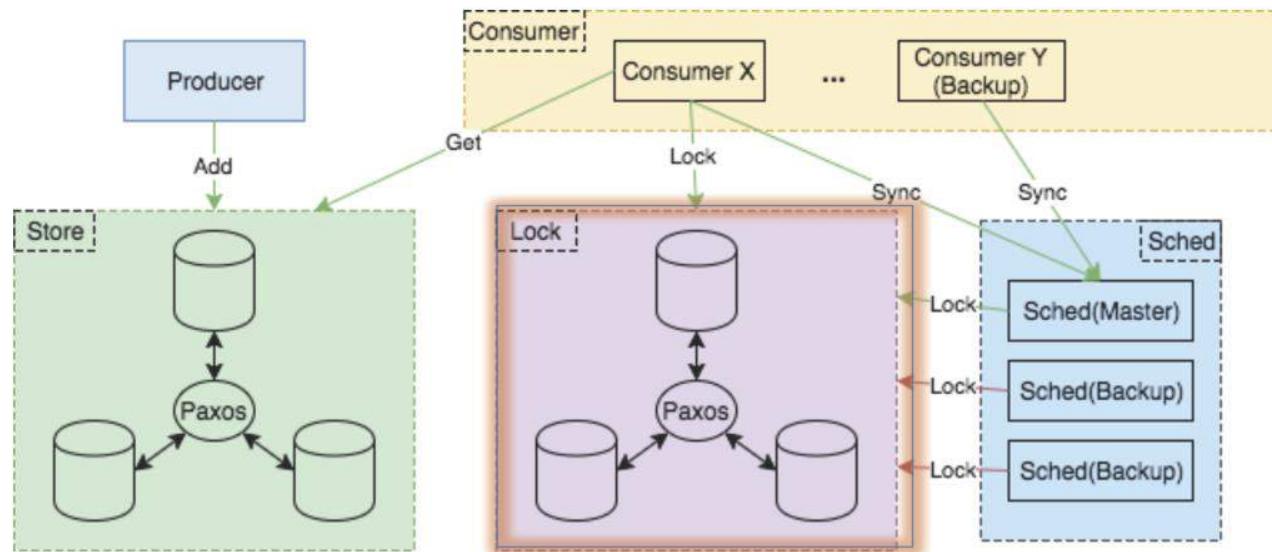
- 引入Paxos
- 同步刷盘
- 对外强一致
- 无乱序，免去重

Sched – 消费者管理

- 消费者容灾
- 消费者负载均衡
- 单机(master)提供服务

1. master挂掉可自行切换
2. master选举过程中仅负载均衡功能失效, 整个系统不对Sched强依赖.

新架构(PhxQueue)设计



Store – 队列存储

- 引入Paxos
- 同步刷盘
- 对外强一致
- 无乱序，免去重

Sched – 消费者管理

- 消费者容灾
- 消费者负载均衡
- 单机(master)提供服务

Lock – 分布式锁

- 引入Paxos
- 通用分布式锁服务
- 负责Sched master选举
- 规避队列重复出队

目录

- 背景
- 设计
- 实现
- 最佳实践
- 效果展示
- 总结

- 高可靠
 - 队列状态机设计
- 高性能
 - Plog as queue
 - Group Commit
- 高可用
 - Store租约型读写
 - Store接入均衡
 - 轻量级租约维护
 - Consumer负载均衡

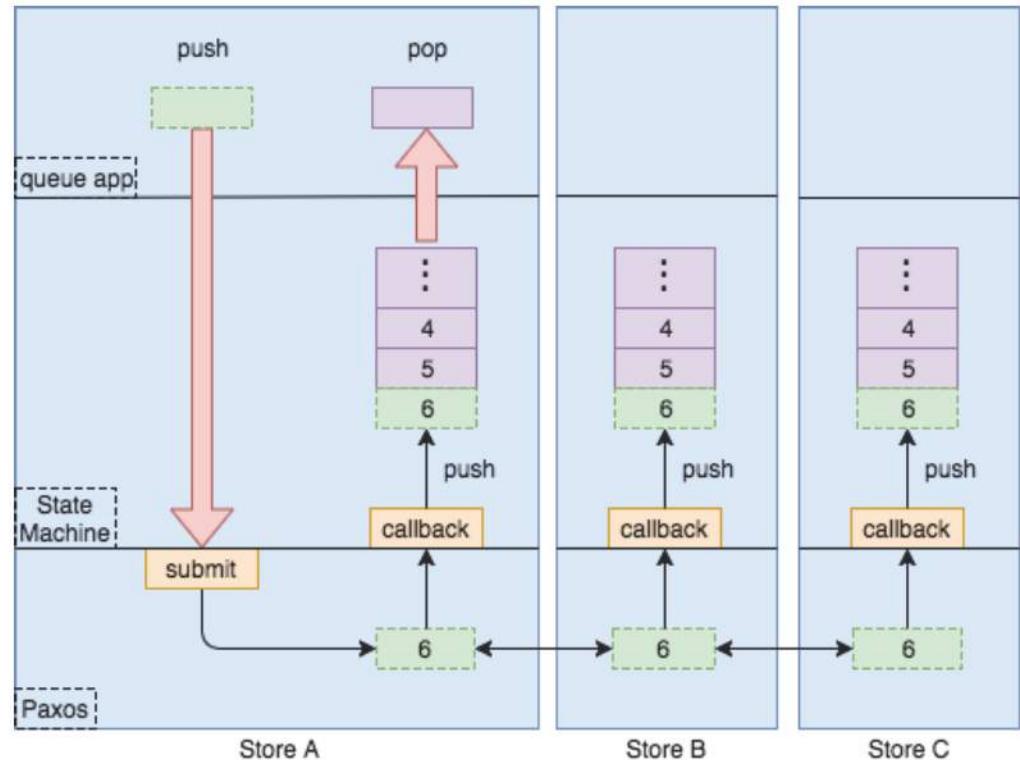
高可靠 - 队列状态机设计

概念映射

队列概念	PhxPaxos 概念
队列数据	paxos log
队列偏移	instance id
队列最小读偏移	check point

极端状况下数据不丢

- Paxos协议的正确运行
- 简单拜占庭检测手段



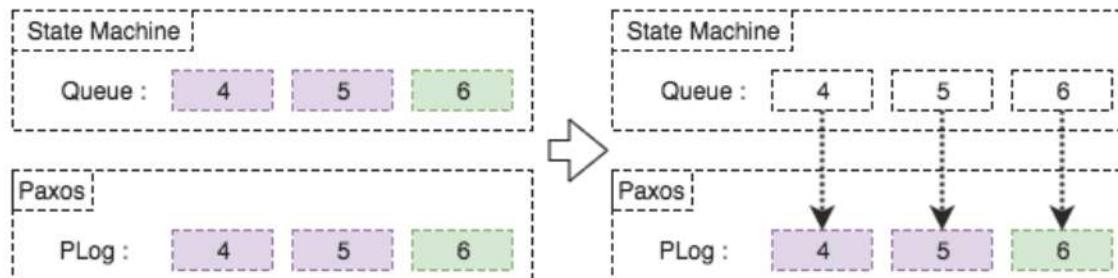
高性能(1/2) – Plog as queue

问题: 副本数从2提高至6

(写量大, 存储成本高)

解决方案: PLog as Queue

效果: 存储减半, 写带宽减半



高性能(2/2) – Group commit

问题1: 同步刷盘写放大 (1.5k -> 4k)

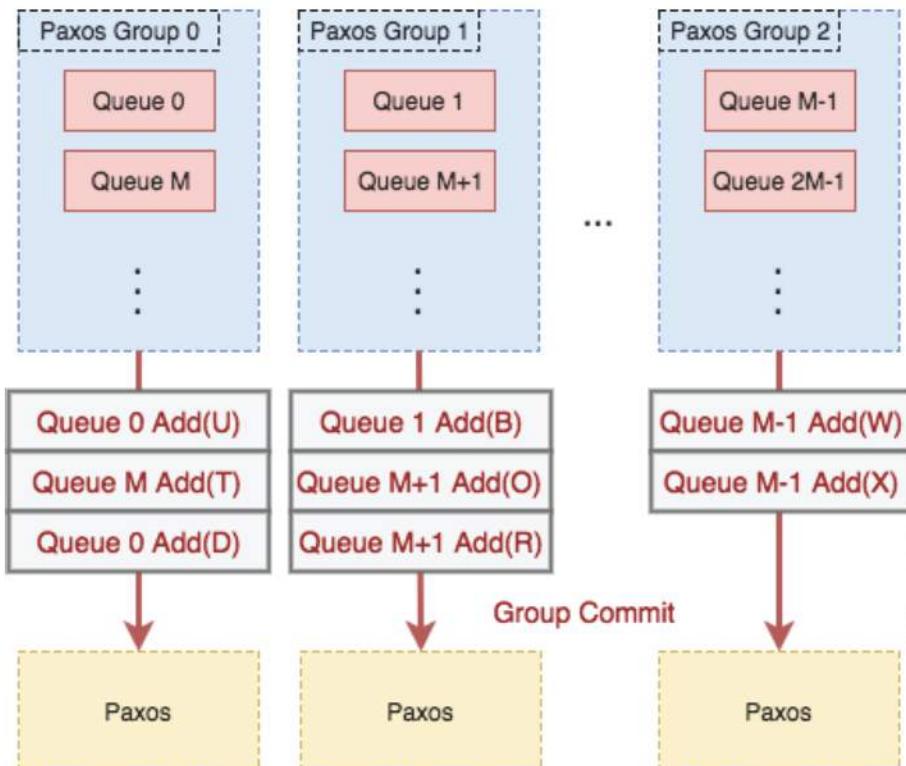
问题2: DC间RTT 4ms, 吞吐低

解决方案: 批量同步刷盘

- Paxos写时机:
5ms超时 或 请求数达到20
- N Queue : 1 Paxos group
- Paxos group太少 – 并发不足
- Paxos group太多 – 批量效果不足
- 经实测, Paxos group数为100较合适

效果:

- 入队QPS上限50w
- 写放大降低20倍



高可用 (1/4) – Store租约型读写

无租约 or 有租约?

	优点	缺点
无租约	<ul style="list-style-type: none">可用性高	<ul style="list-style-type: none">读: 需要访问三机写: 失败时, 需要随机避让避免活锁
有租约	<ul style="list-style-type: none">读: 只读master写: 只写master, 无活锁问题	<ul style="list-style-type: none">master切换有不可用时长

为Store引入租约型读写, 考量如下:

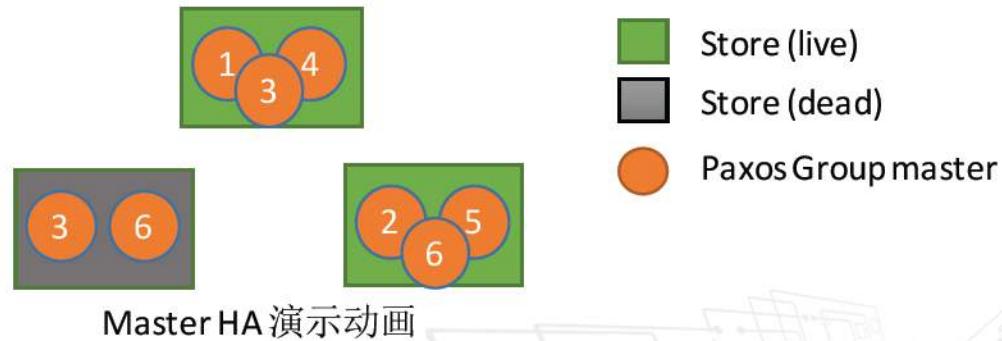
- 即使一个Paxos实例只处理一个queue, 写量也很大, 随机避让严重影响吞吐.
- 在非宕机情况下的master切换, 可通过主动结束租约避免不可用时长

高可用 (2/4) – Store接入均衡

问题: 各机请求不均

解决方案: master调度

- 按模摊分master到各机
- 出灾时: 其它节点随机避让抢master
- 恢复时: 按成功率收回自己负责的master
- 屏蔽时: 主动结束master租约



高可用 (3/4) – 轻量级租约维护

租约作用: Sched选master / 保证队列消费权唯一
代替Zookeeper的好处

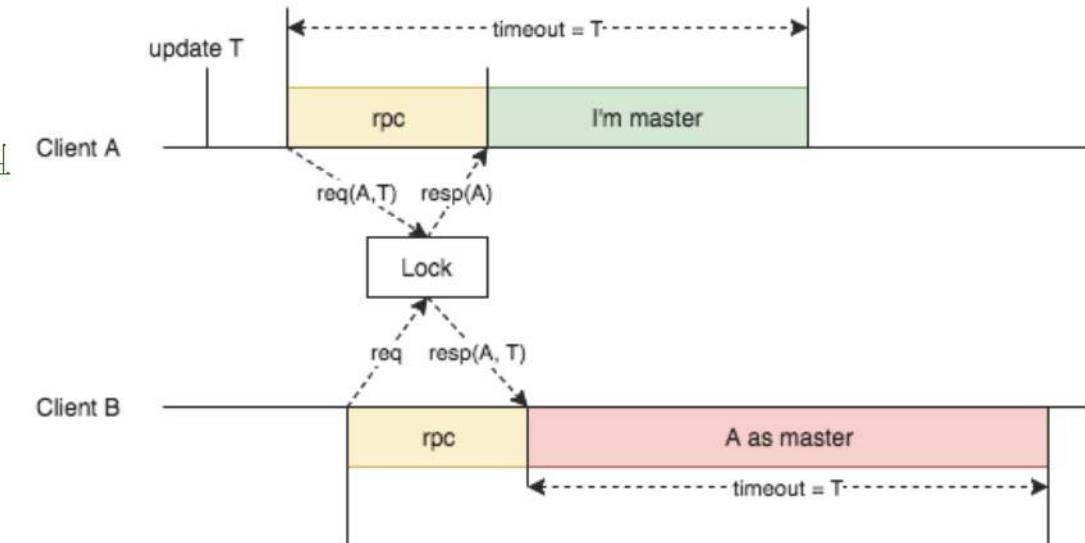
- 简单方案代替复杂黑盒 (伪代码不足20行)
- 减低运维、使用成本

难点: 协议设计

- master: 租约时长内续租
- 非master: 避让租约
- 乐观锁

特点: 无死锁

- 可重入
- 过期自动失效



高可用 (4/4) – Consumer负载均衡

Consumer权重式负载均衡

- Sched收集Consumer负载更新权重
- Consumer按权重计算目标处理队列

For each consumer:

$$\text{new_weight} \downarrow = \text{default_weight} * \text{mean_load} / \text{load} \uparrow;$$

问题1: 轻微调整引起全部Consumer抖动

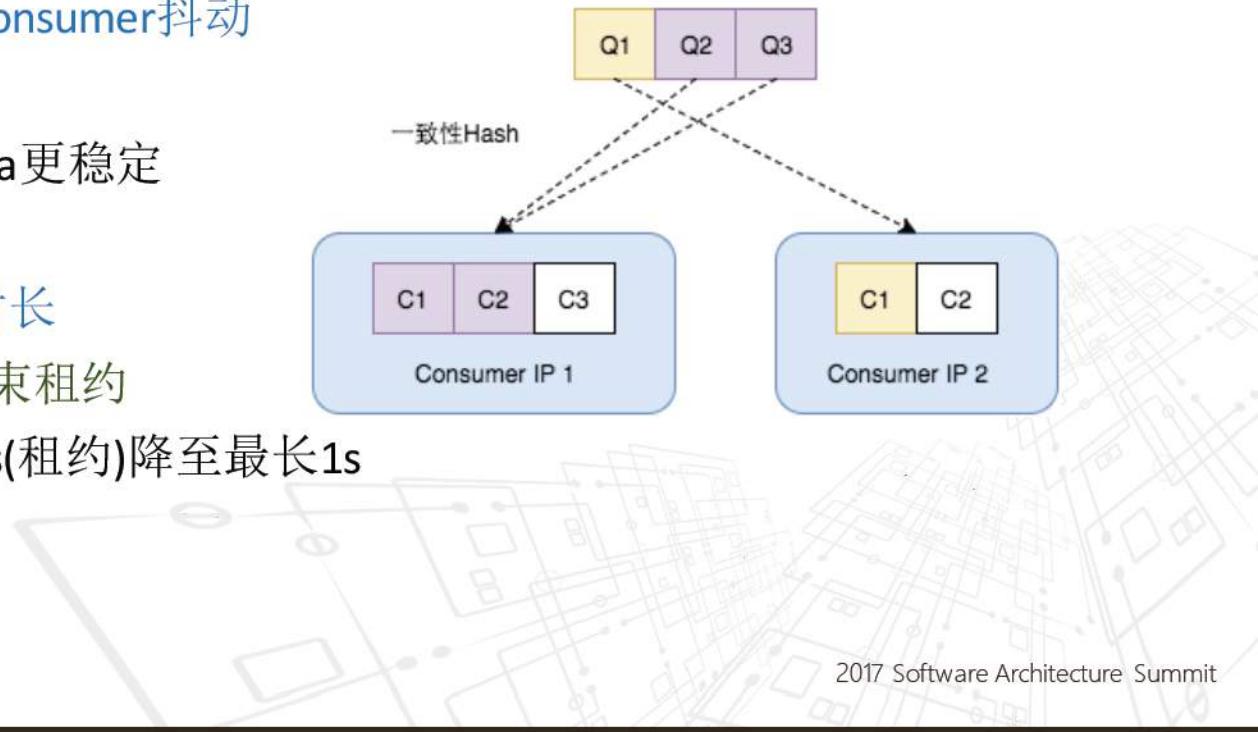
解决方案: 一致性hash

效果: 避免惊群效应, 比Kafka更稳定

问题2: 均衡时存在不可用时长

解决方案: 弃租锁时主动结束租约

效果: 不可用时长从最长20s(租约)降至最长1s



目录

- 背景
- 设计
- 实现
- 最佳实践
- 效果展示
- 总结

- 队列与业务的相互协调
- 主动屏蔽
- 基于接口调用统计的后向流控



队列与业务的相互协调

业务对队列的妥协

- 幂等 - 基于版本号的乐观锁
- 顺序性处理 - 打包有依赖关系的数据

队列对业务的定制化

- 可用性优化 - 换组/队列重试
- 热点处理 - 随机路由避免热点

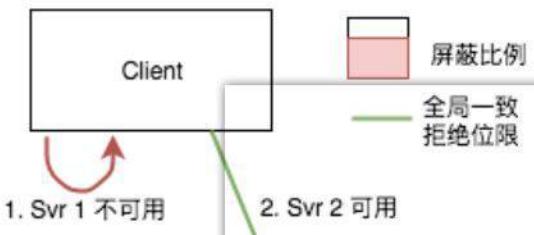
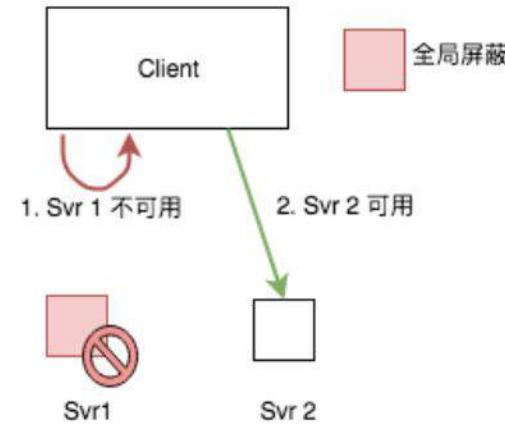
主动屏蔽

目的

- 应对雪崩、爆流量/CPU/内存等异常

手段

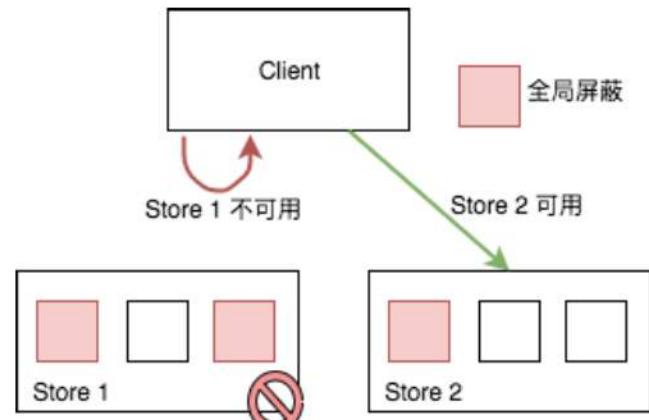
- 全局屏蔽
- 按比例屏蔽



主动屏蔽 - 以组为单位

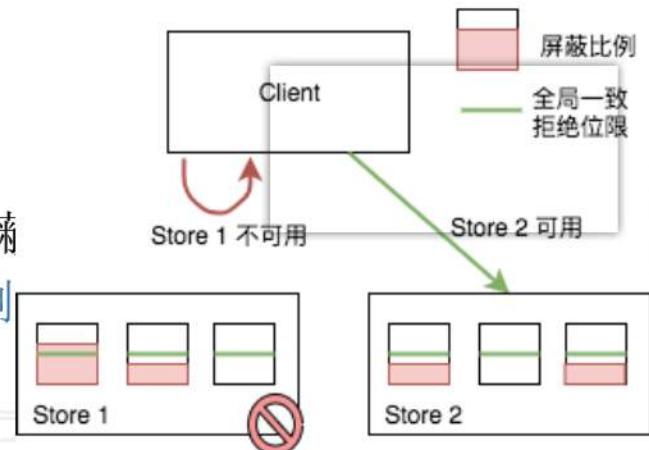
单组全局屏蔽

- 等价于 至少两个节点全局屏蔽



单组按比例屏蔽

- 等价于 该uin至少命中一个节点按比例屏蔽
- 单组屏蔽比例 = 组内最高单节点屏蔽比例



基于接口调用统计的后向流控

目的

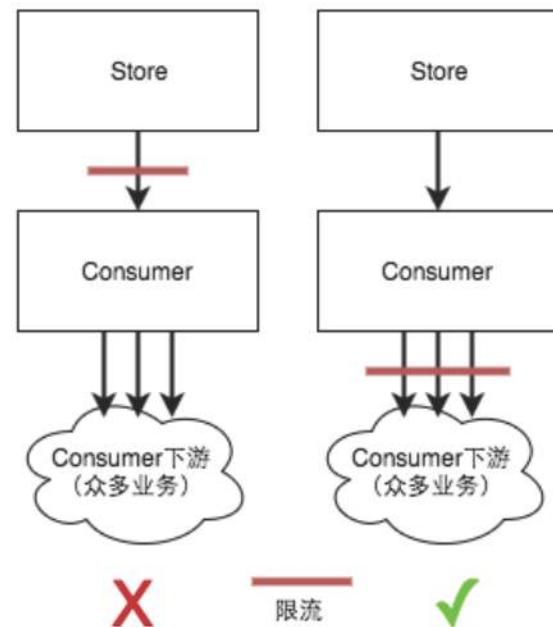
- 削峰，保护Consumer下游

控制拉取数量 X

- 消费一个请求被扩散成多个调用
- 扩散比不稳定

控制接口调用 ✓

- 粒度：业务 + 接口
- 借助RPC API统计接口调用数
- 设定周期内调用上限（天花板）
- 均匀周期内调用：计算拉取数量和间隔



目录

- 背景
- 设计
- 实现
- 最佳实践
- 效果展示
- 总结

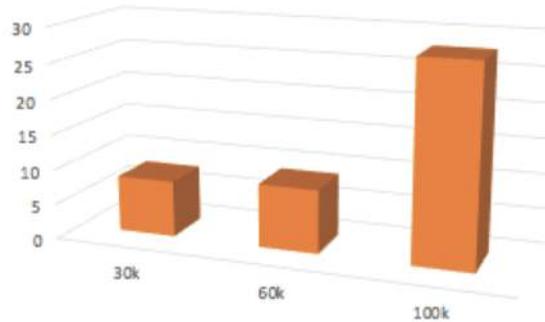
- 高性能 – 吞吐表现
- 高可用 – 负载均衡
- 顺序消费
- 基于接口调用统计的后向流控



效果展示 - 高性能 - 吞吐表现

副本数	3
刷盘模式	同步
机型	64*2.4GHz, SSD Raid 10
Buffer大小	2K
读写模型	写+读

写QPS与请求耗时(ms)关系



• 效果：耗时换吞吐

效果展示 - 高可用 - 负载均衡

旧队列

最高负载

服务器名	load	总使用率
mmpayhbastrosched18	3.06	12%
mmpayhbastrosched20	2.83	12%

最低负载

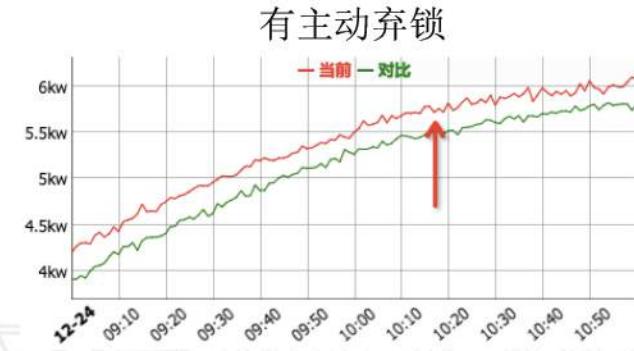
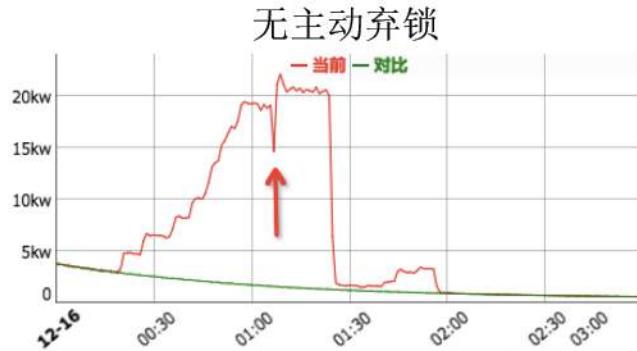
服务器名	load	总使用率
mmpayhbastrosched26	0.17	1%
mmpayhbastrosched28	0.19	0%

PHXQueue

服务器名	load	总使用率
mmphxqueuesnsconsumerml51	1.41	17%
mmphxqueuesnsconsumerml49	1.30	17%

服务器名	load	总使用率
mmphxqueuesnsconsumerml52	1.15	15%
mmphxqueuesnsconsumerml48	1.01	15%

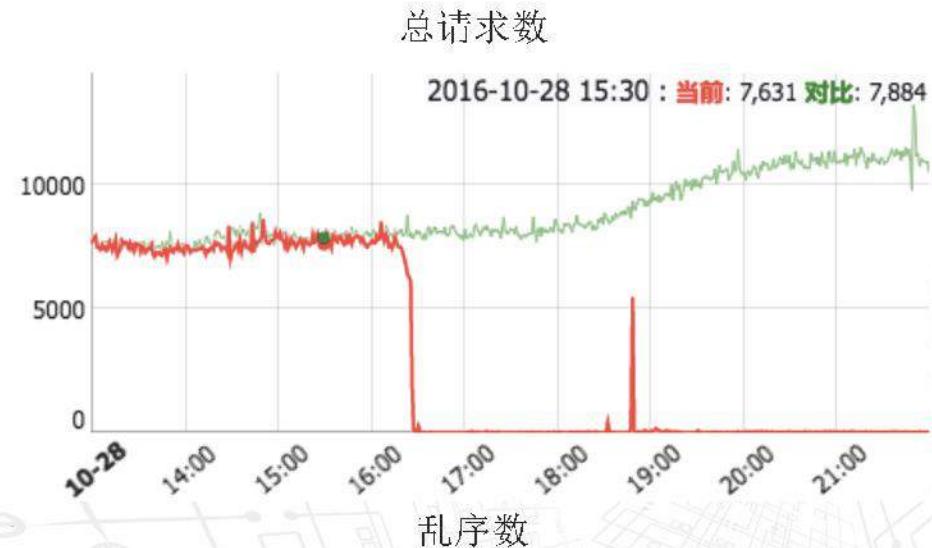
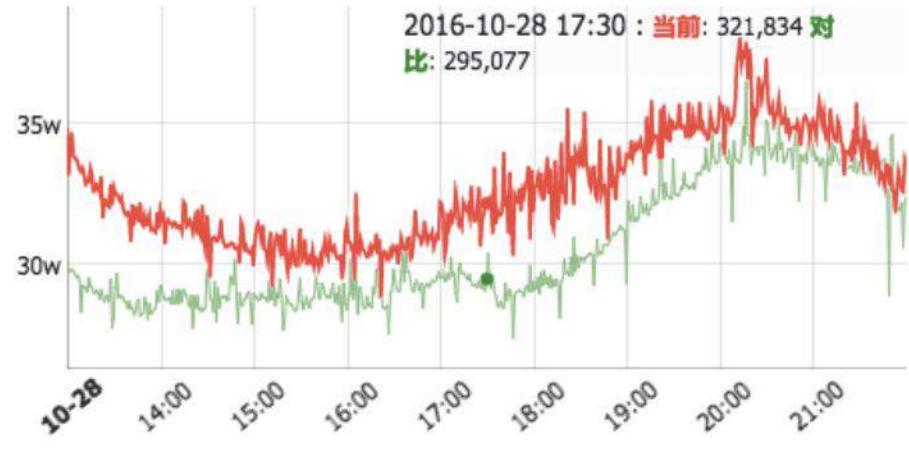
均衡对吞吐影响



效果展示 – 顺序消费

以某个读多写少的业务为例

- 旧架构乱序率达 **2.4%**
- 新架构将乱序率降低至 **0.006%**
(因重试队列导致乱序)
- 如业务需要绝对顺序消费, 可关闭重试队列



效果展示 - 基于接口调用统计的后向流控

流控效果精准

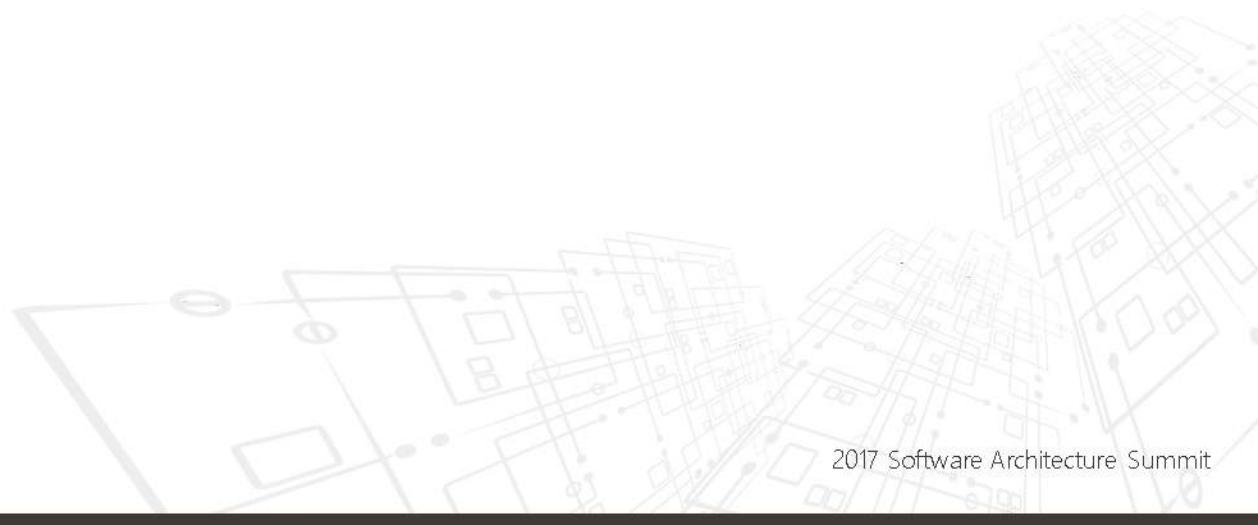
通过流控增强对业务系统的把控

- 压测确定容量上限和消费速率
- 常态配置流控策略



目录

- 背景
- 设计
- 实现
- 最佳实践
- 效果展示
- 总结



总结

- 引入Paxos能让事情变简单
 - 保证数据可靠性
 - 杜绝乱序、重复消费
- 良好设计解决其它核心问题
 - 高性能： Plog as queue、 Group Commit
 - 高可用： Store接入均衡、轻量级租约维护、 Consumer负载均衡
- 实现投产
 - 业务配合、热点处理、屏蔽策略、流控策略.....
- 展望： Paxos将会成为一种开发模式

PhxPaxos组件介绍

PhxPaxos是微信团队开源的Paxos类库

- 高性能
- 功能完善
- 接口方便易用

项目地址

- <https://github.com/tencent-wechat/phxpaxos>

腾讯云商用队列介绍

CKafka

- 场景：大数据分析，日志采集
- 一致性协议：ISR

CMQ

- 场景：金融级高可靠强一致
- 一致性协议：Raft

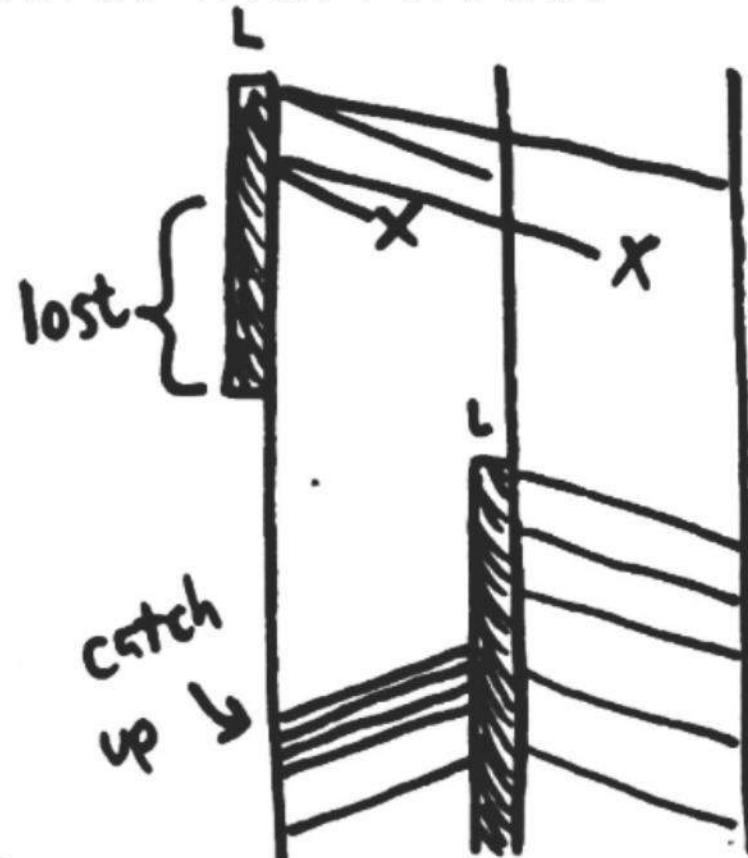
谢谢



微信后台团队公众号

About Kafka tolerate $f-1$ failures with f nodes

- ISR为空可能导致数据丢失
 - <https://aphyr.com/posts/293-jepsen-kafka>
- 进一步说明经逻辑证明的一致性模型才是可靠的



AGENDA

PayPal & PayPal Risk (Platform)

Risk DAL Service Challenge

Async Solution

Async Future Plan

AGENDA

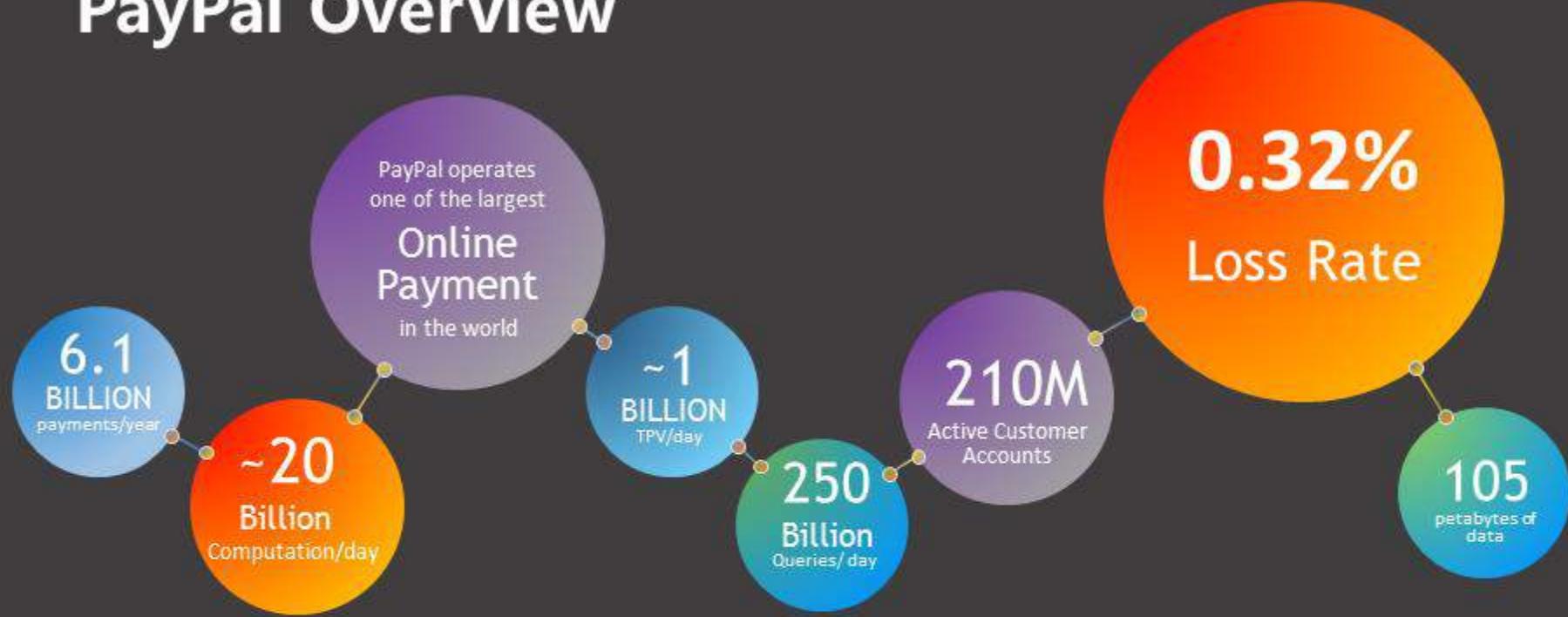
PayPal & PayPal Risk (Platform)

Risk DAL Service Challenge

Async Solution

Async Future Plan

PayPal Overview



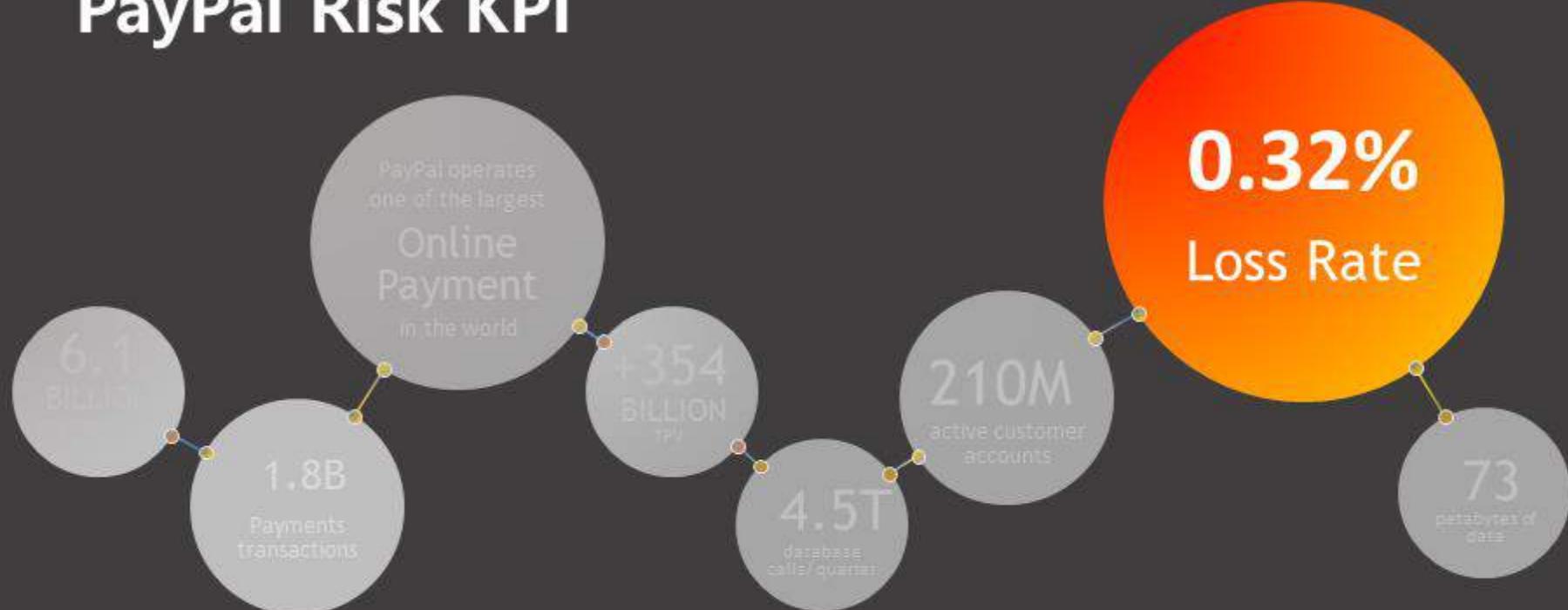
The power of our platform



Our technology transformation enables us to:

- Process payments at tremendous scale (**200+** countries & **25** currencies supported)
- Accelerate the innovation of new products
- Engage world-class developers & technologists

PayPal Risk KPI



The power of our platform



Our technology transformation enables us to:

- Process payments at tremendous scale (**200+** countries & **25** currencies supported)
- Accelerate the innovation of new products
- Engage world-class developers & technologists

Requirement for Risk Platform

Accuracy vs Latency

Need more Variables to Make Accurate Decision...

... But that impact decision Speed

Minimize Data Load Latency ...

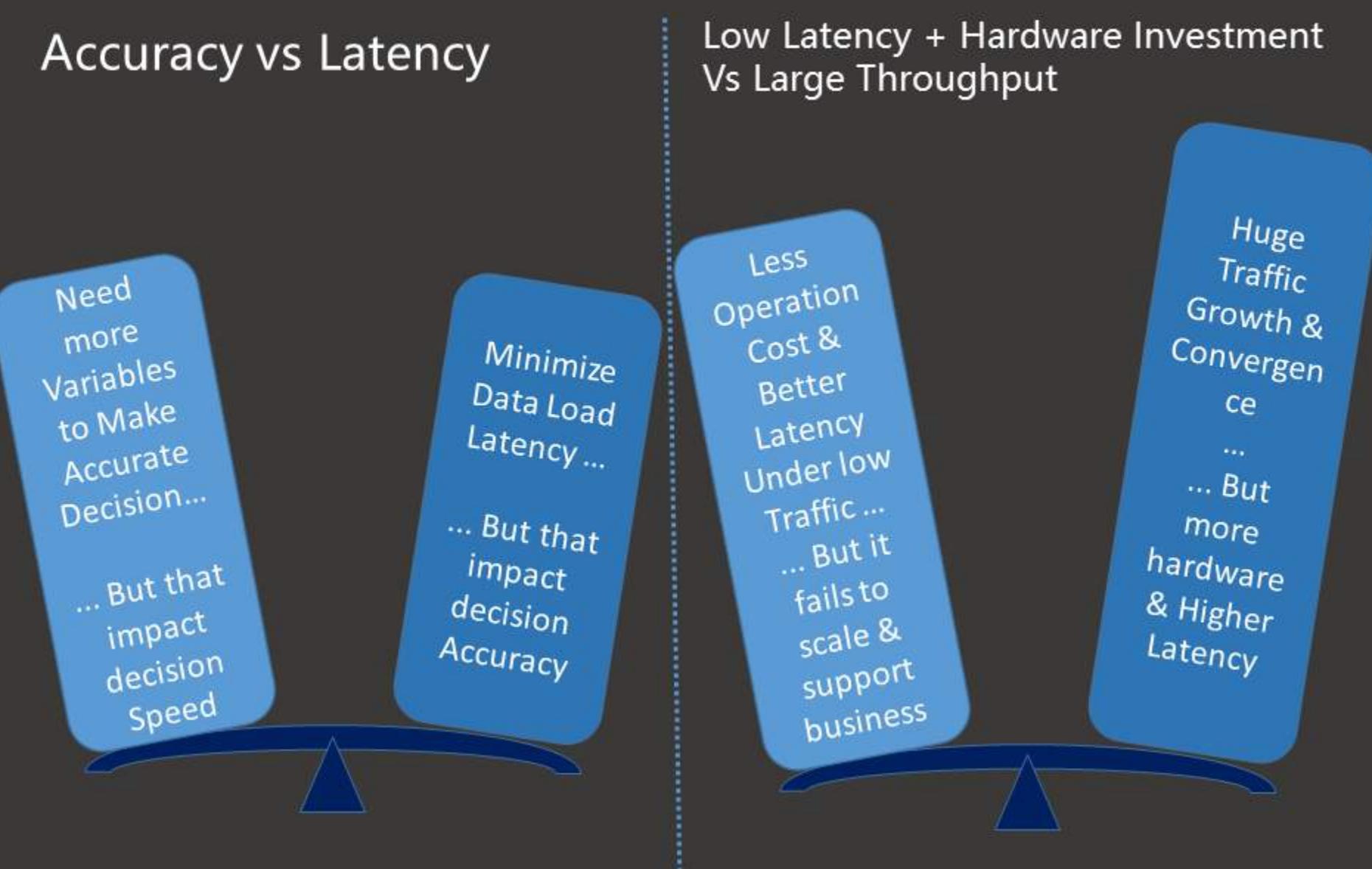
... But that impact decision Accuracy

Low Latency + Hardware Investment Vs Large Throughput

Less Operation Cost & Better Latency Under low Traffic ...
... But it fails to scale & support business

Huge Traffic Growth & Convergence ...

... But more hardware & Higher Latency



PayPal Risk Platform Architecture

Online

Read Path

Write Path

Offline

Gateway Service

Decision Service

Model + Variable Computation Service

DAL Service

Real-time Compute Data

Variable Rollup Service

Offline Generated Data

Logging System/ ETL

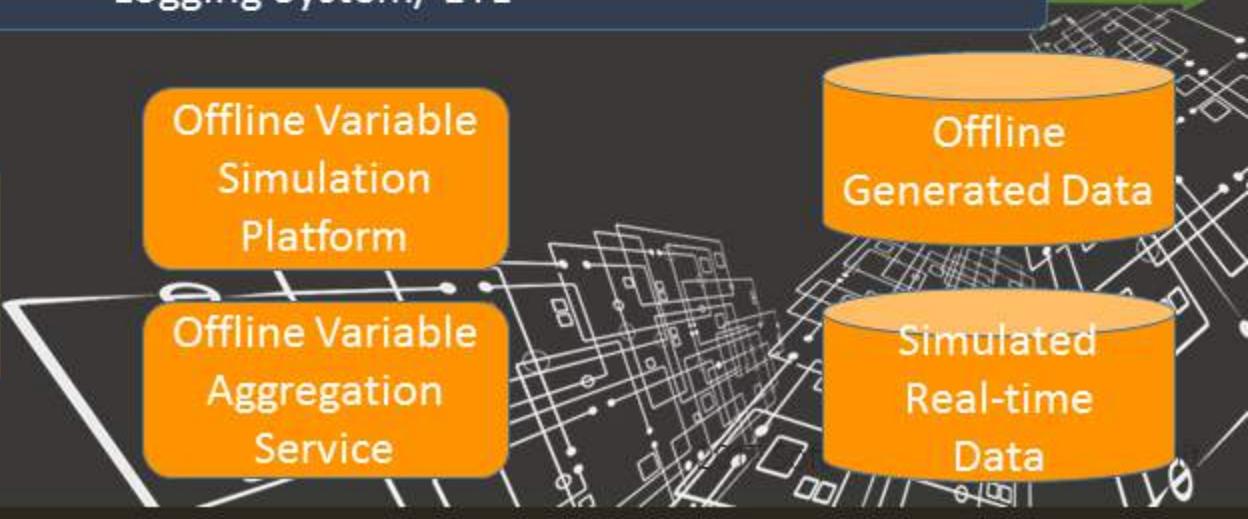
Model Training Platform

Offline Variable Simulation Platform

Offline Variable Aggregation Service

Offline Generated Data

Simulated Real-time Data



PayPal Risk Platform Architecture

Online

Read Path

Write Path

Offline

Gateway Service

Decision Service

Model + Variable Computation Service

Variable Aggregation Service

DAL Service

Offline Generated Data

Real-time Compute Data

Logging System/ ETL

Model Training Platform

Offline Variable Simulation Platform

Offline Variable Aggregation Service

Offline Generated Data

Simulated Real-time Data

AGENDA

PayPal & PayPal Risk (Platform)

Risk DAL Service Challenge

Async Solution

Async Future Plan

DAL Service Ultimate Questions



<100ms P99.99 Latency ??



For single instance, 20k-30k Peak TPS ??



99.99% Availability-To-Business??

JVM-Based High Performance & ATB DAL Service

DAL Service Technical Challenges

Customer Requirement

Req

- Adopt New Use Case
- Access behavior Differentiate per Colo
- Flexibility & Fast-evolving Use Case
 - Replication
 - Traffic Strategy

Budget Cost

- Align with traffic, Hardware investment Exponential Increase

Value

Cost

Performance Issue

- P99 Latency Significantly differentiate Avg latency
- Too Many Latency Spike under Traffic
- Storage Cluster Unavailability Impact Latency

Operational Cost

- Maintain too many Client with multiple versions
- Too Frequent Release tie to Biz Case
- Standby Storage Cluster switch-over

Tech

AGENDA

PayPal & PayPal Risk (Platform)

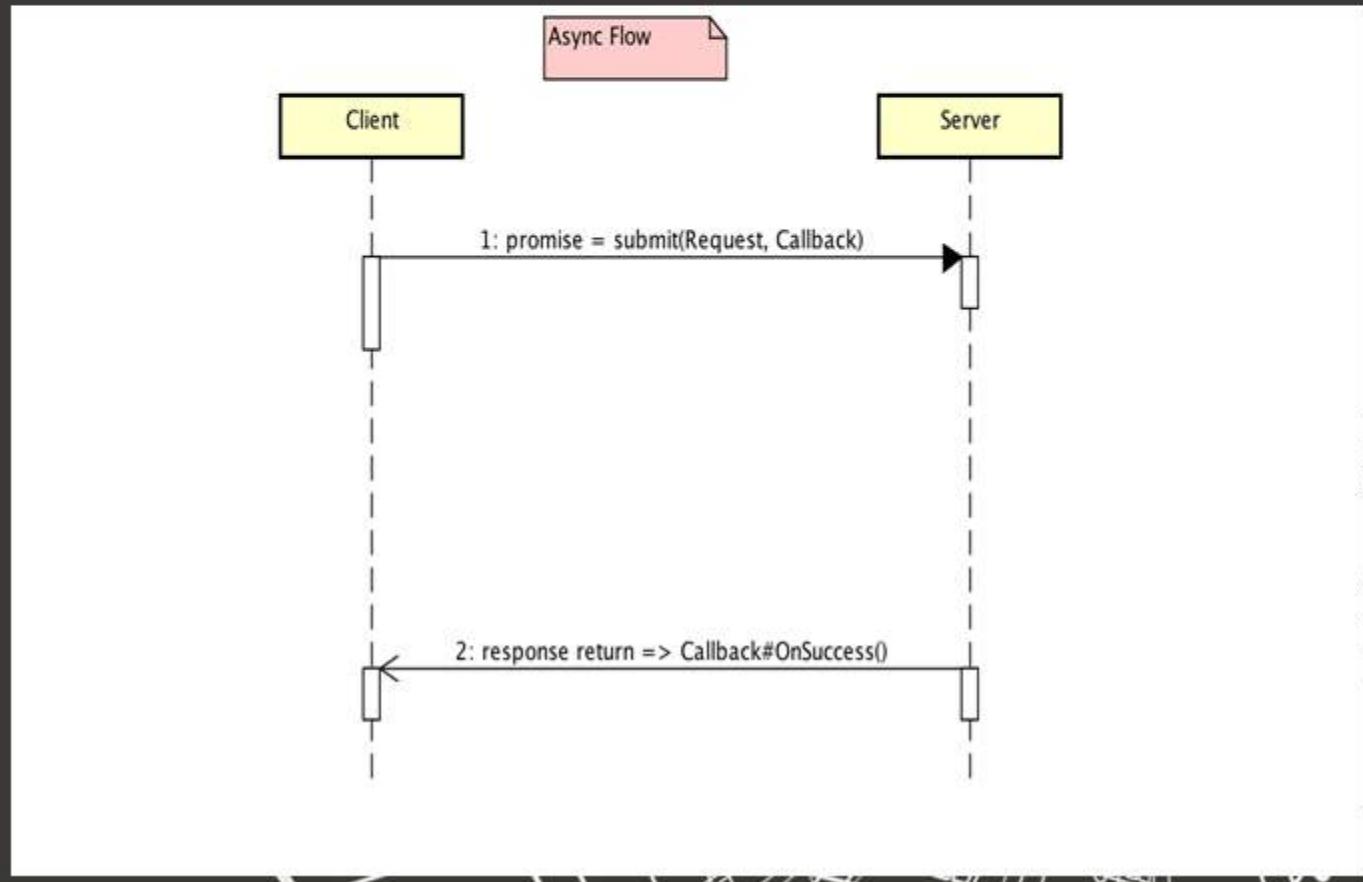
Risk DAL Service Challenge

Async Solution

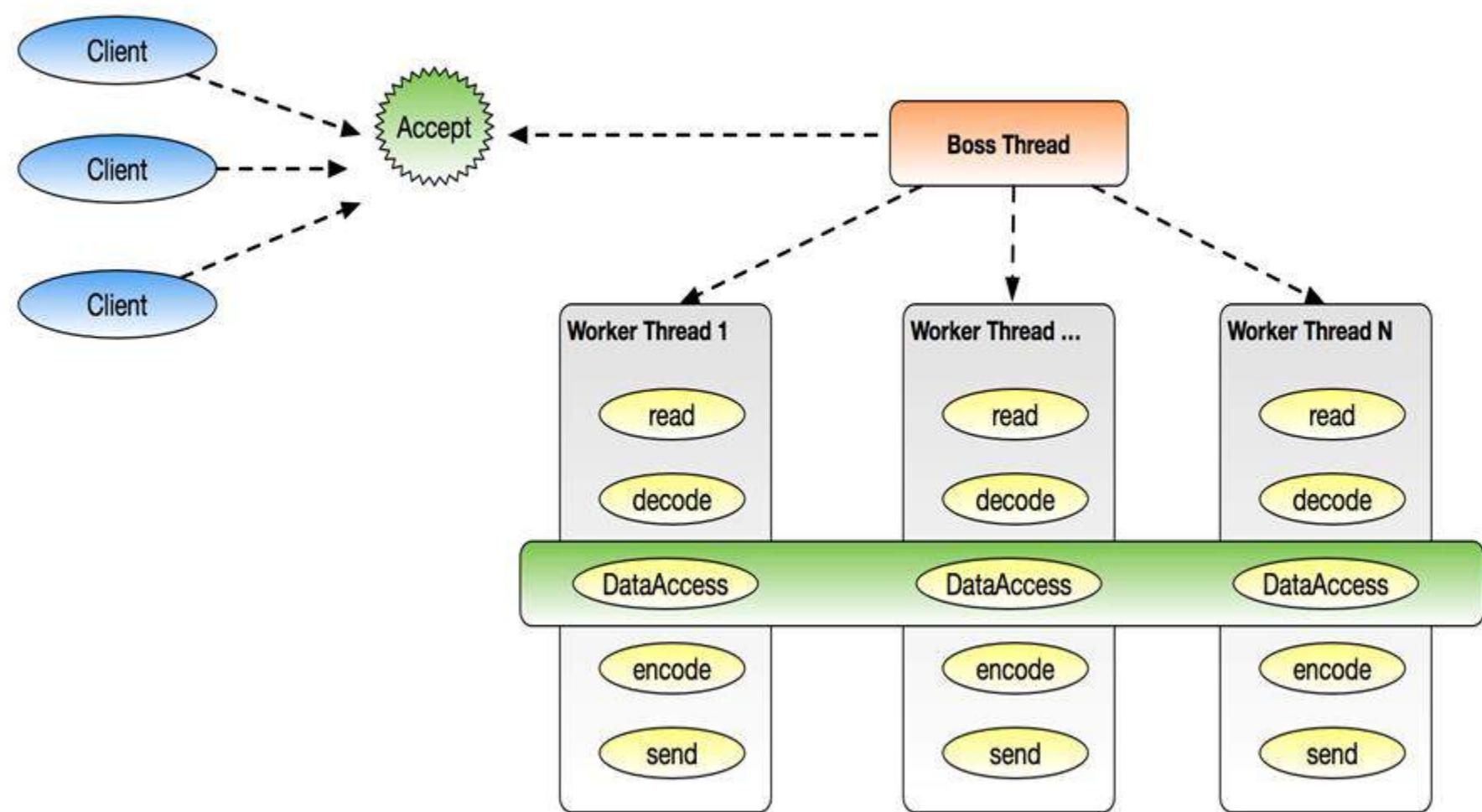
Async Future Plan

Async Original Benefit

- More Efficient Thread Scheduling
 - Non-blocking Call
 - Event-Driven Callback
- Less Context Switch
- Fault Isolation

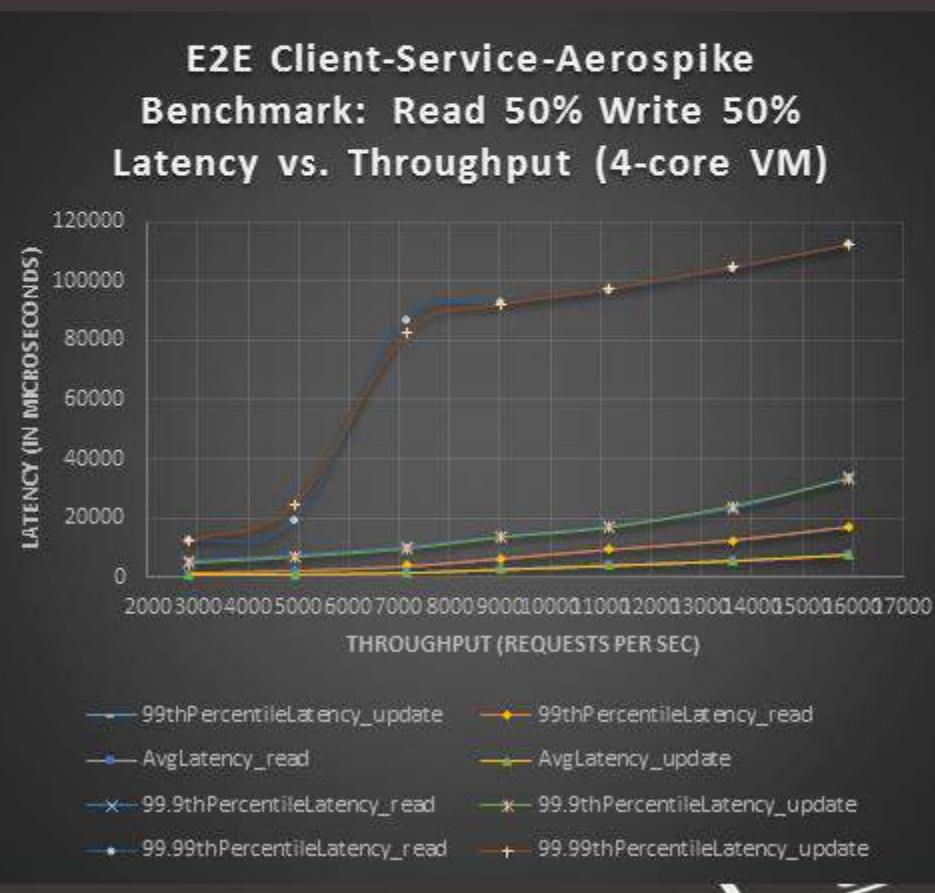


Reactor Pattern Threading Model



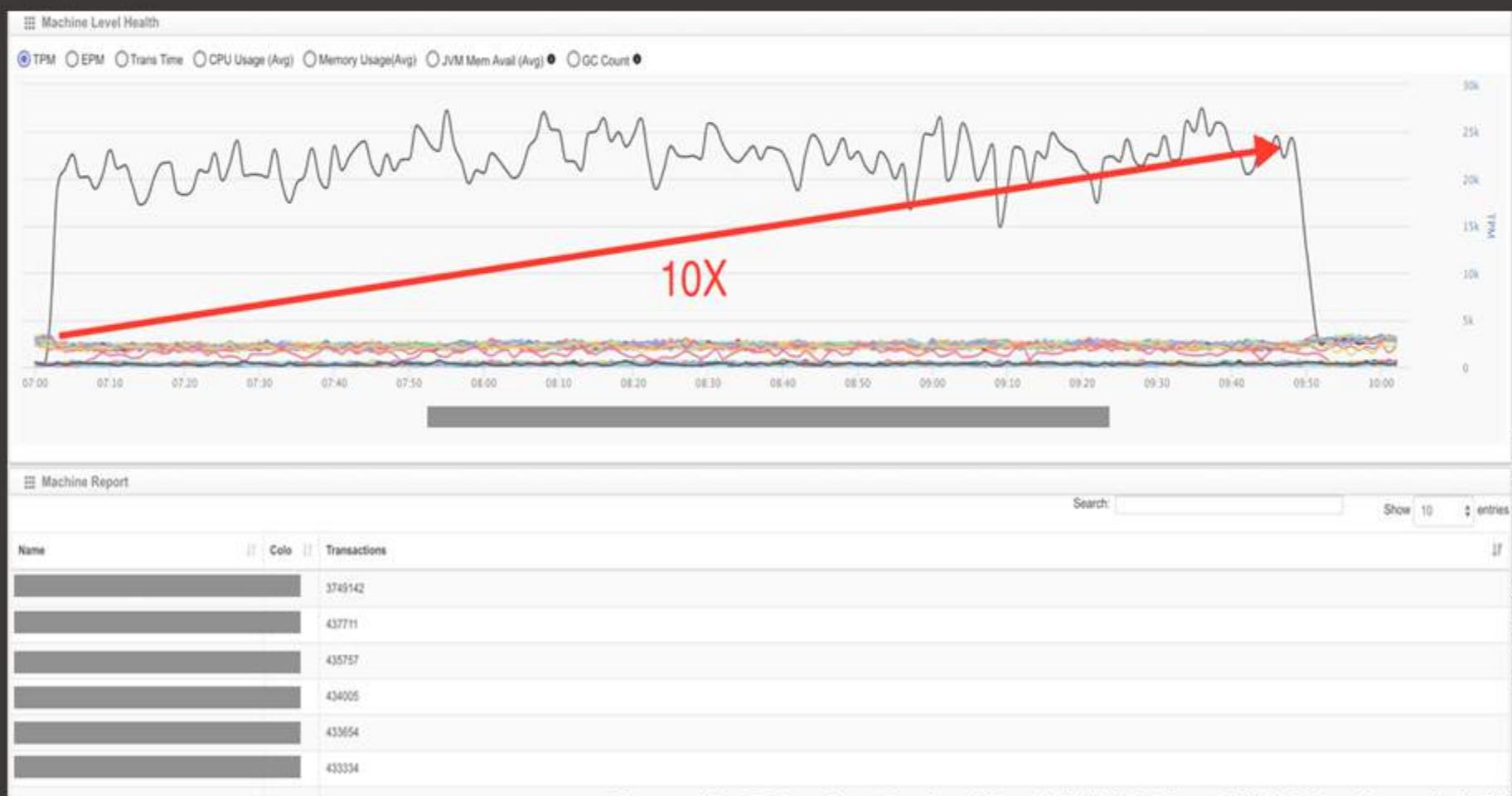
Async DAL Service KPI Comparison

- Low Latency
 - ~10-35% Reduction (Average/P99)



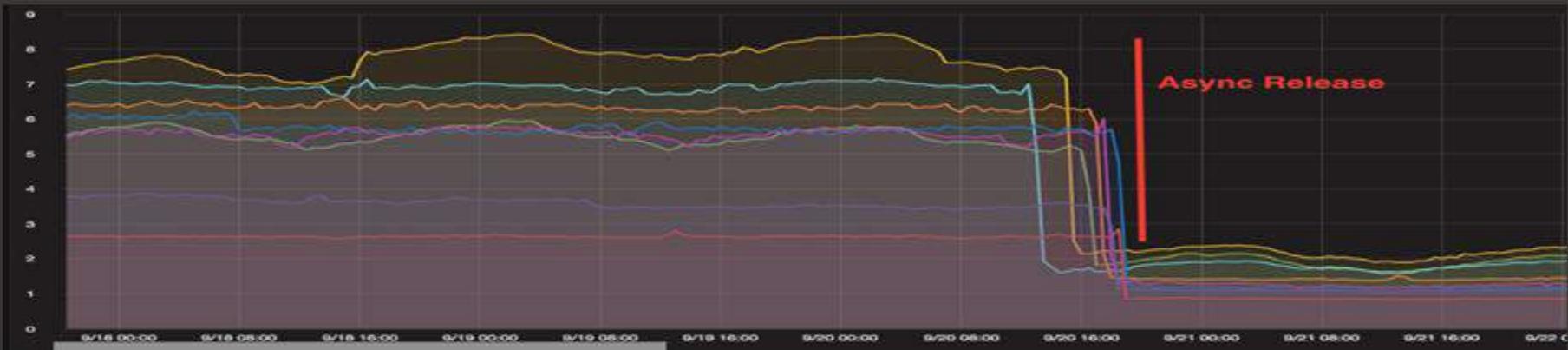
Async DAL Service KPI Comparison – Cont.

- High Throughput
 - 3-10X Increase (Single Instance Comparison)



Async DAL Service KPI Comparison – Cont.

- Less CPU Usage
 - 50% CPU Usage Reduction
 - 66%+ Reduction for Context Switch & System Interrupts



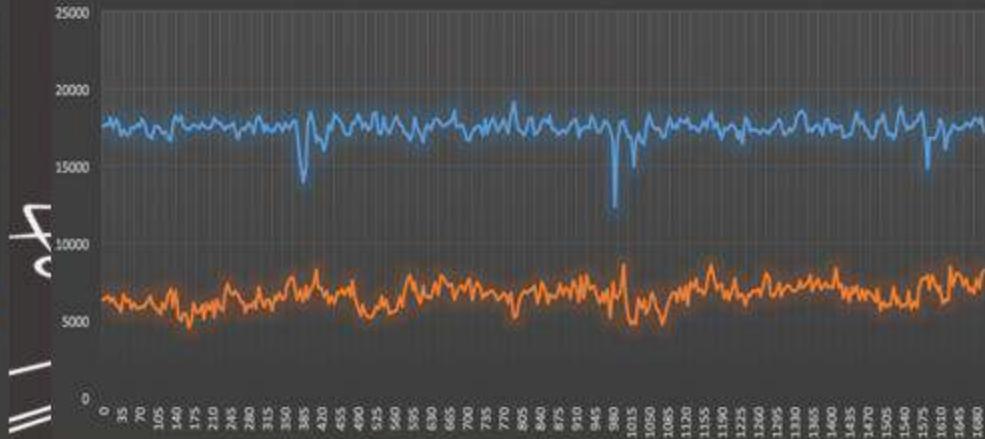
Async vs Sync System Interruption Comparison

Sync Interruption Async Interruption



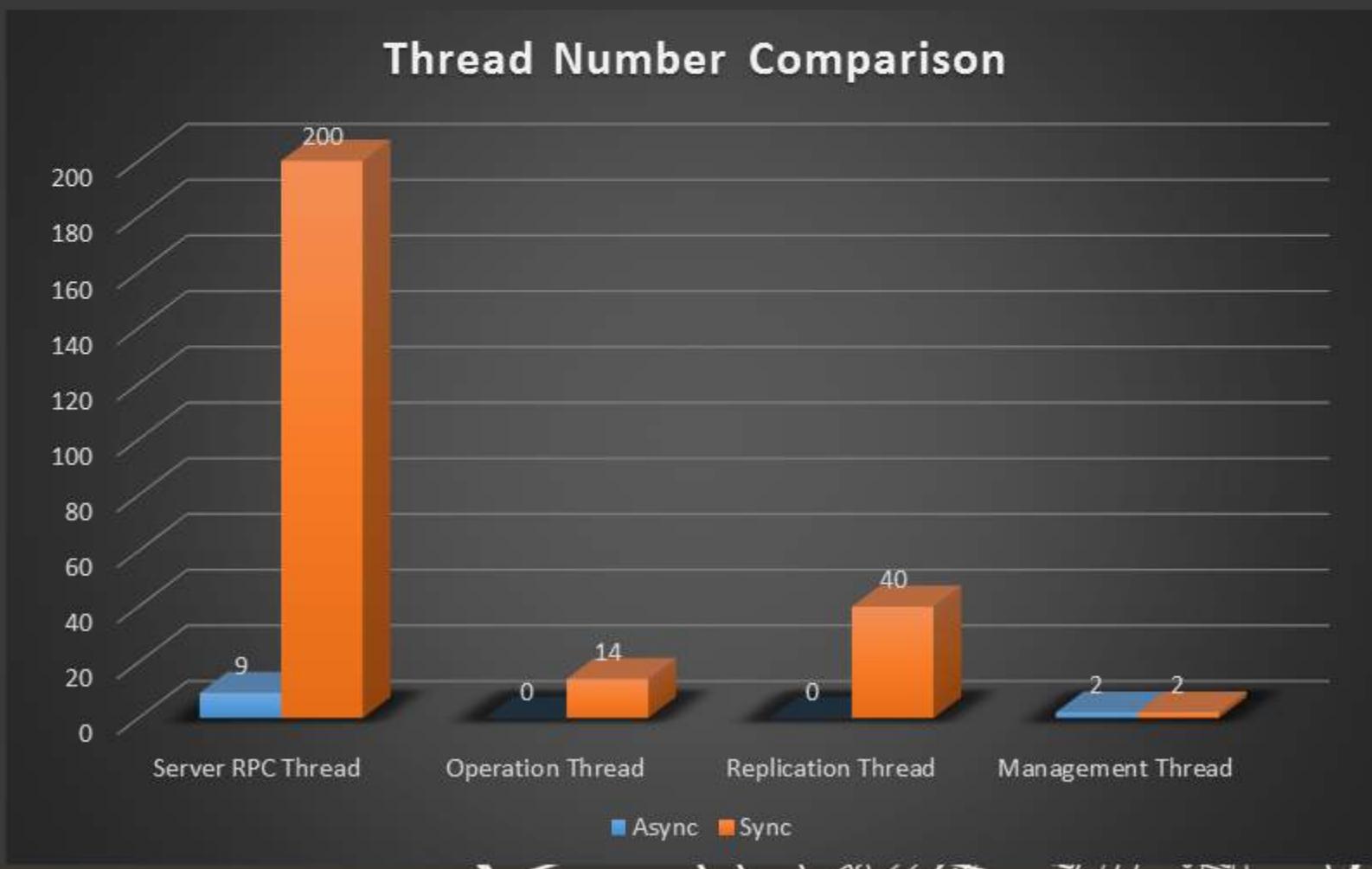
Async vs Sync Context Switch Comparison

Sync Context Switch Async Context Switch



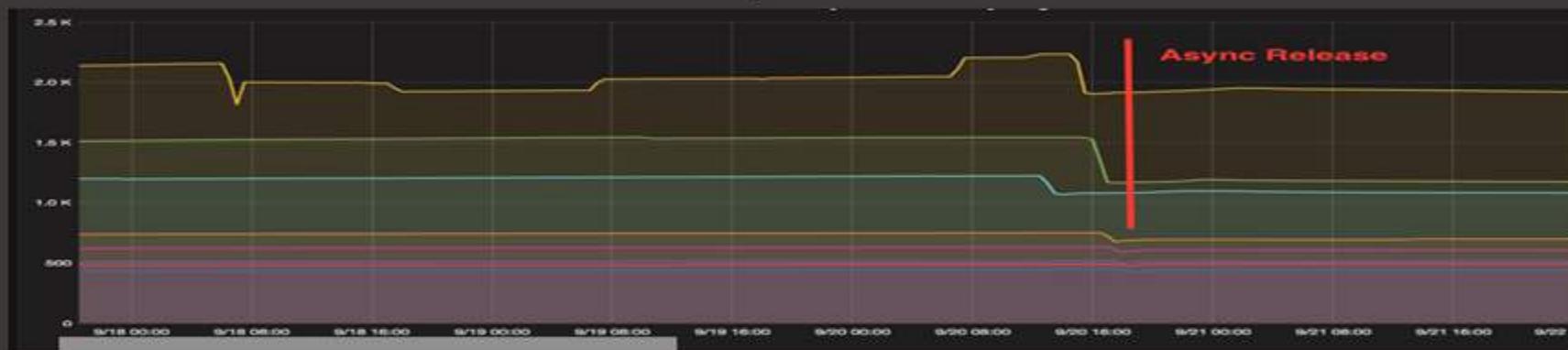
Async DAL Service KPI Comparison – Cont.

- Less Thread Pool
 - 90% Reduction for Thread pool number

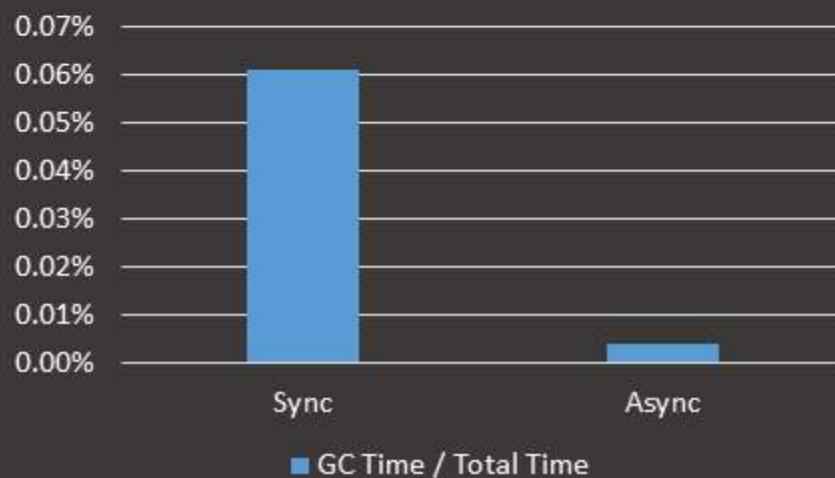


Async DAL Service KPI Comparison – Cont.

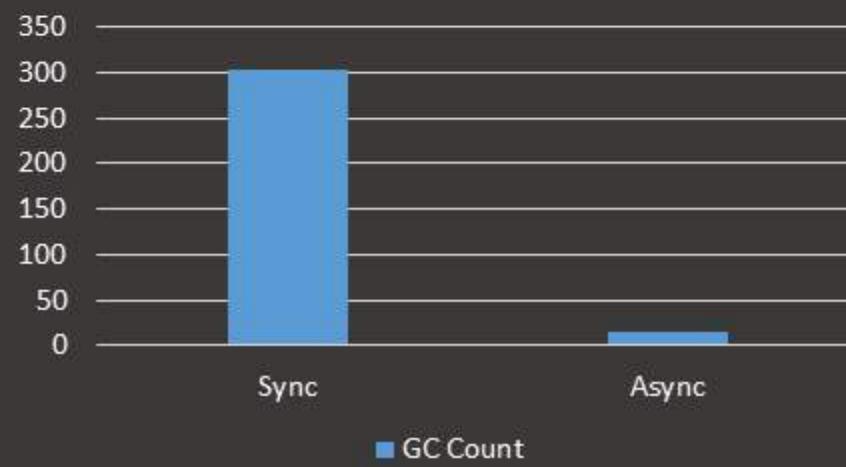
- Memory Friendly
 - 20% Reduction for Memory Allocation
 - 100+MB Young Generation after Young GC
 - 130+MB Pooled Off-heap



GC Time / Total Time

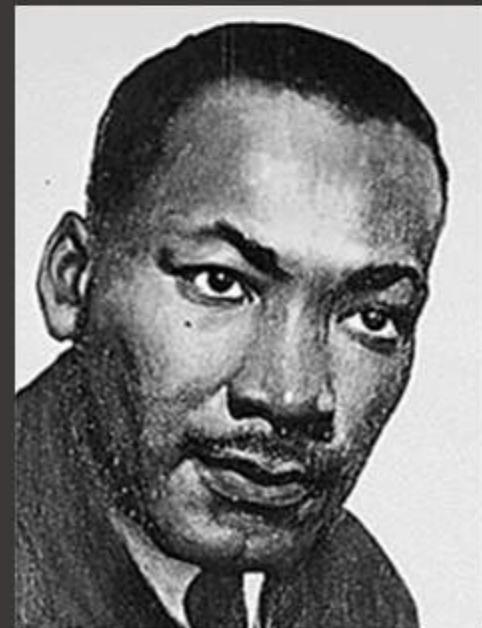


GC Count



We Have ONE Async Dream

- Reform Application Charter from CPU-bound Charter to IO-bound
- Traffic Throughput (non-)linear growth with CPU Usage
- By guarantee Low Latency, Taking 20-30K TPS with 500MB JVM Heap (After young GC)
- Cloud Friendly Application
 - Less Hardware Investment
 - Low Operational Cost
 - Easy Capacity Estimation



High Performance Design

E2E Async

- Non-blocking Pipeline: Async RPC + Async DataAccess
-

Less is More

- **Shared** ThreadPool OVER Separate ThreadPool
 - **Inline** Execution over Execution cross Multiple Thread Pool
-

Autonomous Memory Management

- Use Off-Heap as much as possible
(inbound/outbound & [de]serialization)
- Release Inbound Memory At earlier stage (submitRequest)

High Performance Good Practice

Inbound/Outbound Management

- Batch Consolidation
 - Order Management
 - Timeout Management
 - Retry Only Happen in Client Side
-

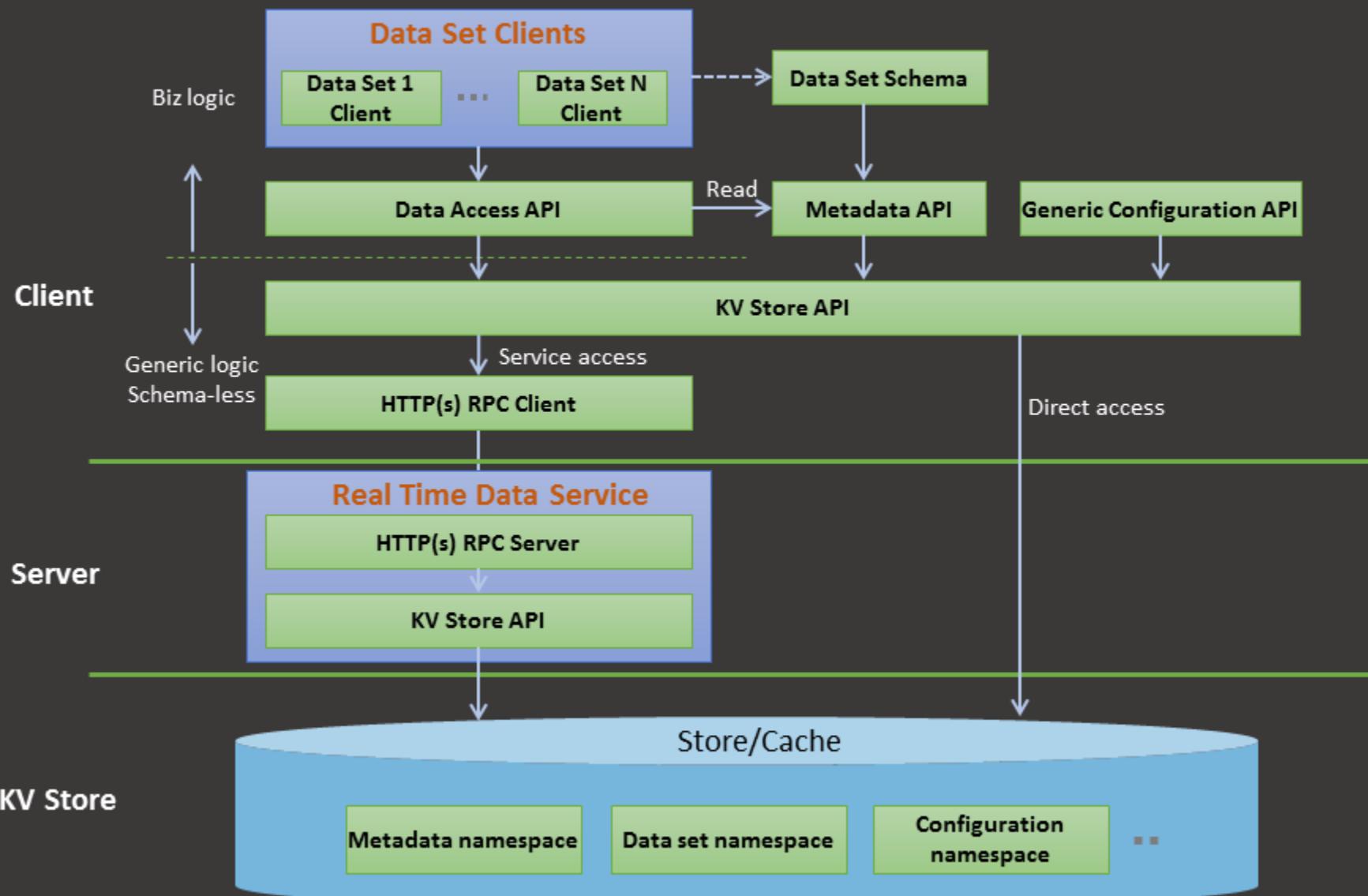
Programming Habit

- Fast Fail over Exception Thrown Cascading
 - Logging & Monitoring Matters
 - Thread-safe Write Operation In Control Plan while Exception-safe Read Operation In Data Plane
-

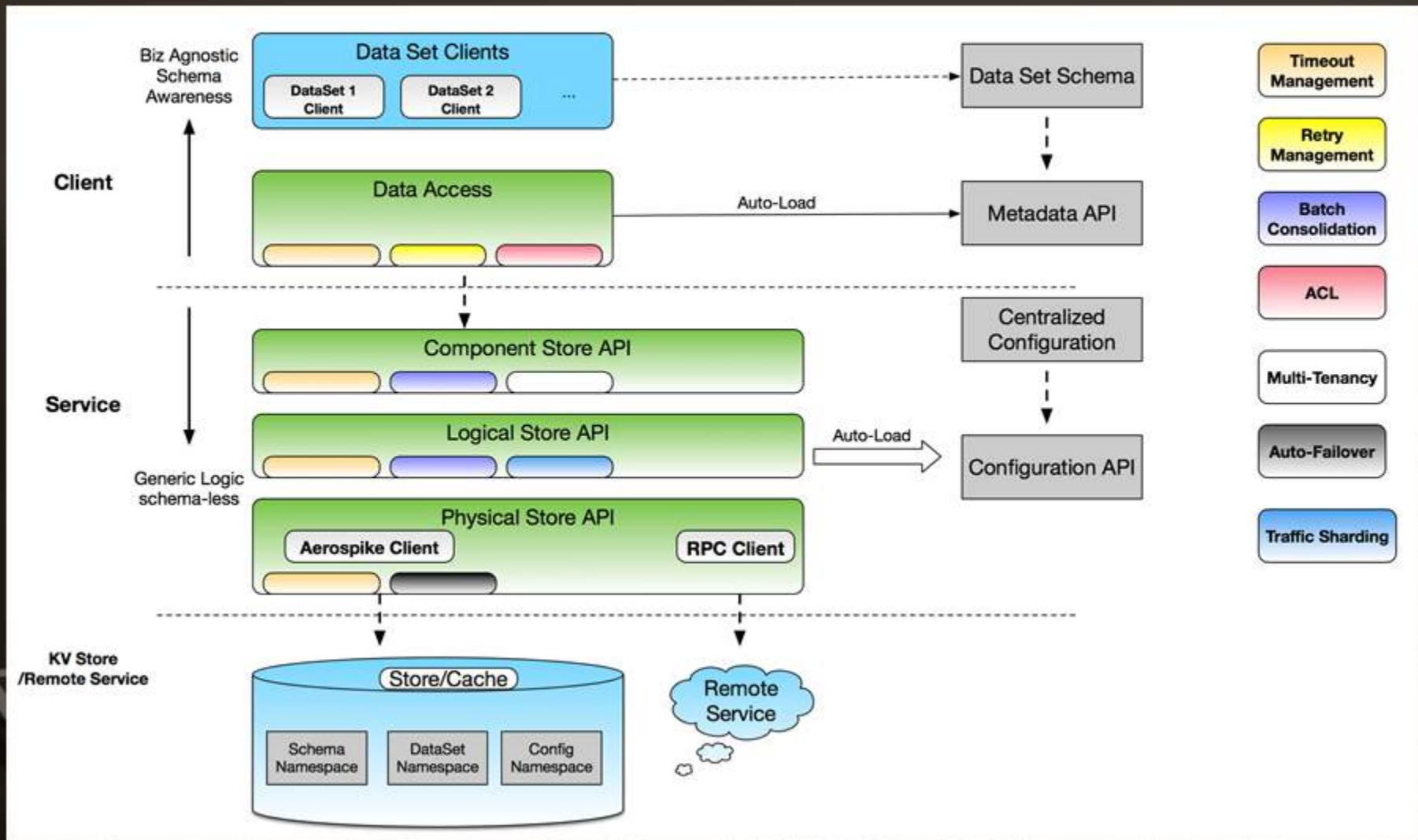
KPI Sign-Off

- Performance Test as Critical Path for Each Commit
- [Mandatory] Continuous Performance Test for Each Commit

Async High Level Architecture



Async DAL Service Hierarchy



Async Data Access Maturity

DAL Service Feature

- Client & Server RoR Identification
 - biz-schema aware on Client Side
 - Schema-less on Server Side
 - Traffic Sharding & Routing
 - Active-Active/Active-Standby
 - Auto-Failover
 - Multi-Tenancy
 - ACL
 - Direct/Service-To-Service Replication
-

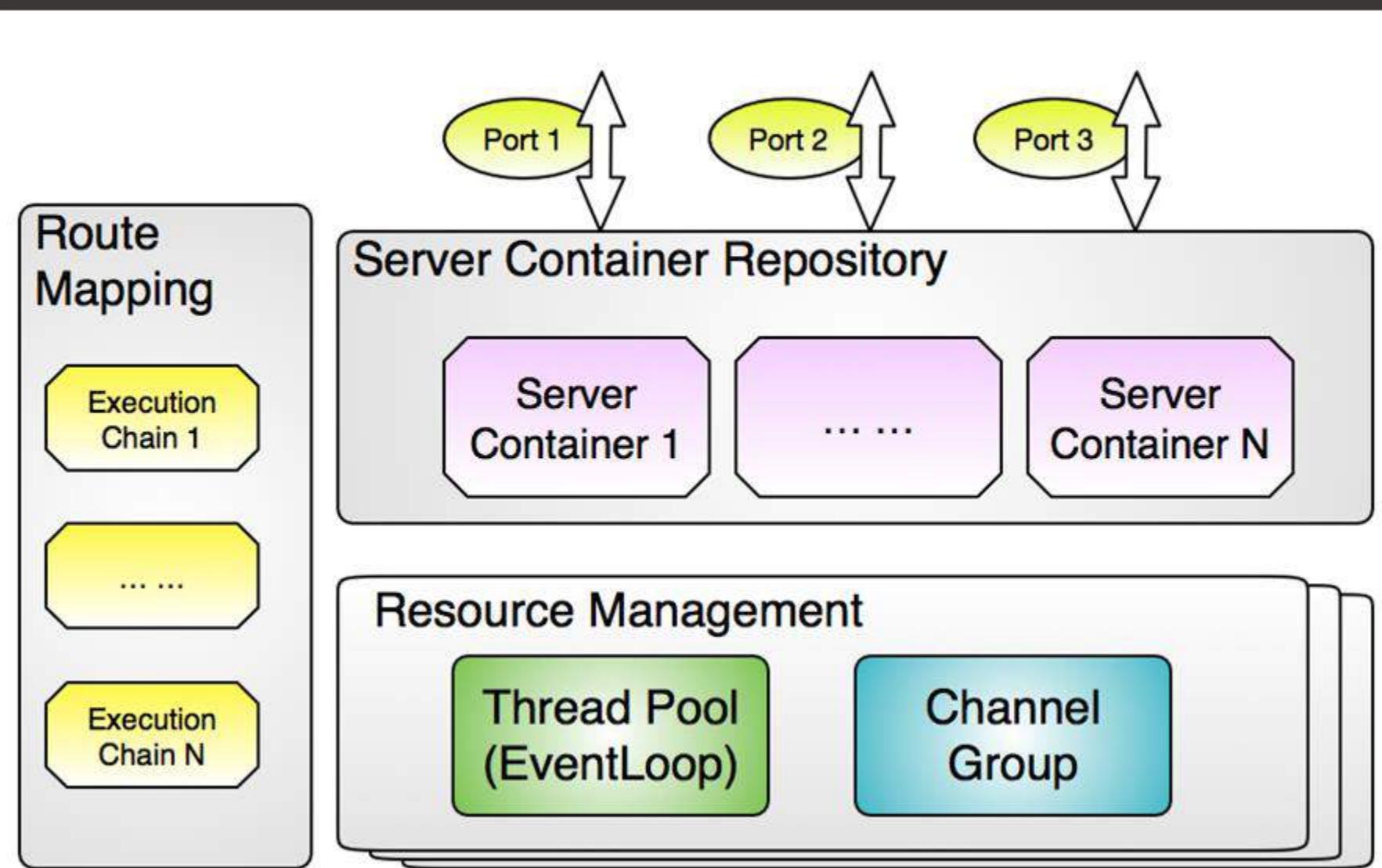
Data Access Mapping

DataSet => KV Mapping
Logical => Physical DataSet Mapping

Metadata Driven

- Source-of-Truth for Online Guideline & Offline Inventory
- Centralized Configuration
- Zero Restart/Auto-Fresh

Async RPC Control Plane Abstraction



Async RPC Maturity

High Flexibility Configuration

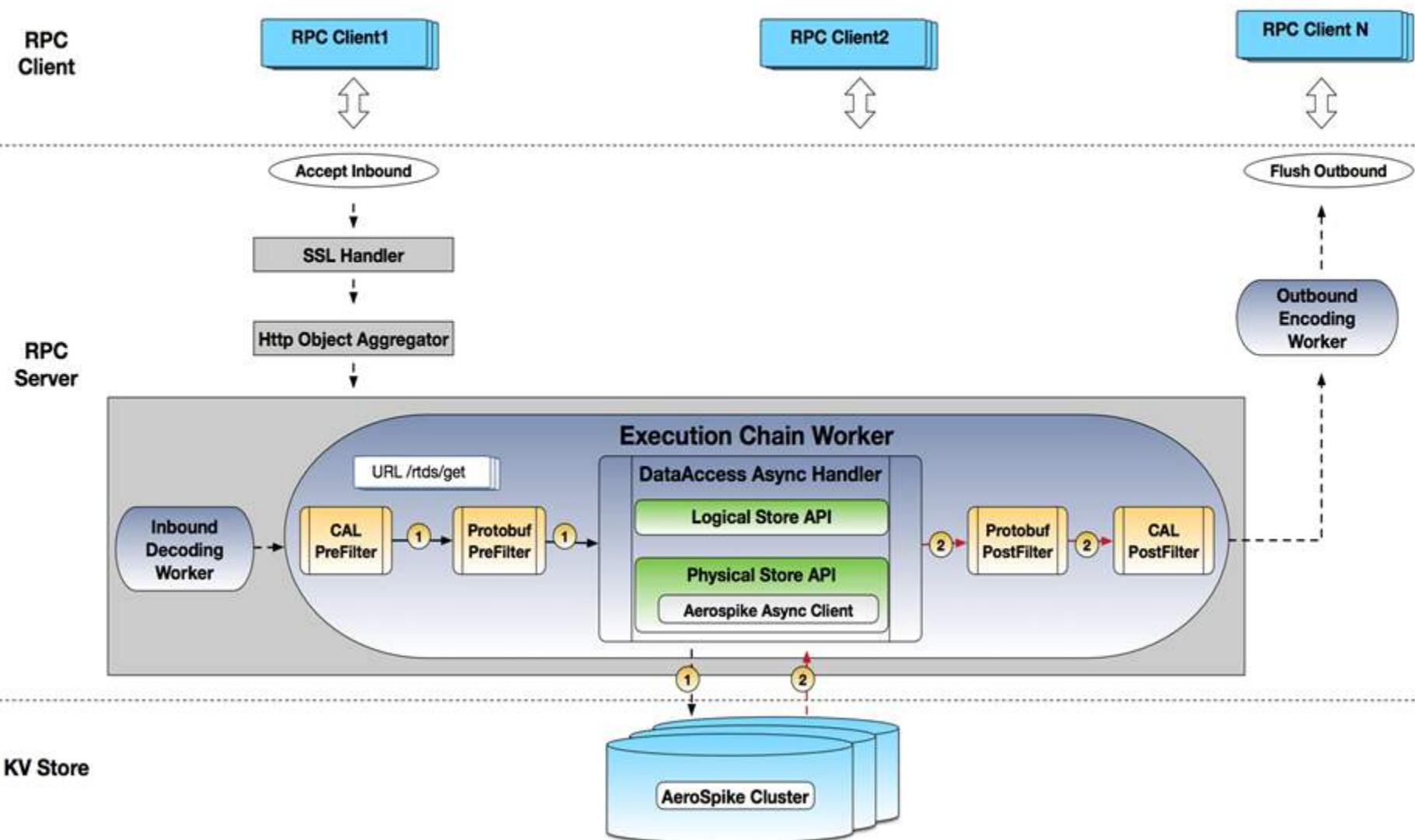
- Configurable Execution Chain per URL
 - Customize protobuf / json encoder
 - Inject Monitoring Module
 - Execution Resource Configuration
 - Threadpool size / netty option (tcp_nodelay)
 - Sharable or not
 - Service Listener Registry
-

RPC Resource Management

- Server Container Life Cycle Management
 - Graceful Shutdown
 - Partial Shutdown Given Container
- Auto Rebuild RPC Client Channel



Async RPC Embrace Async DataAccess



Async Core Value

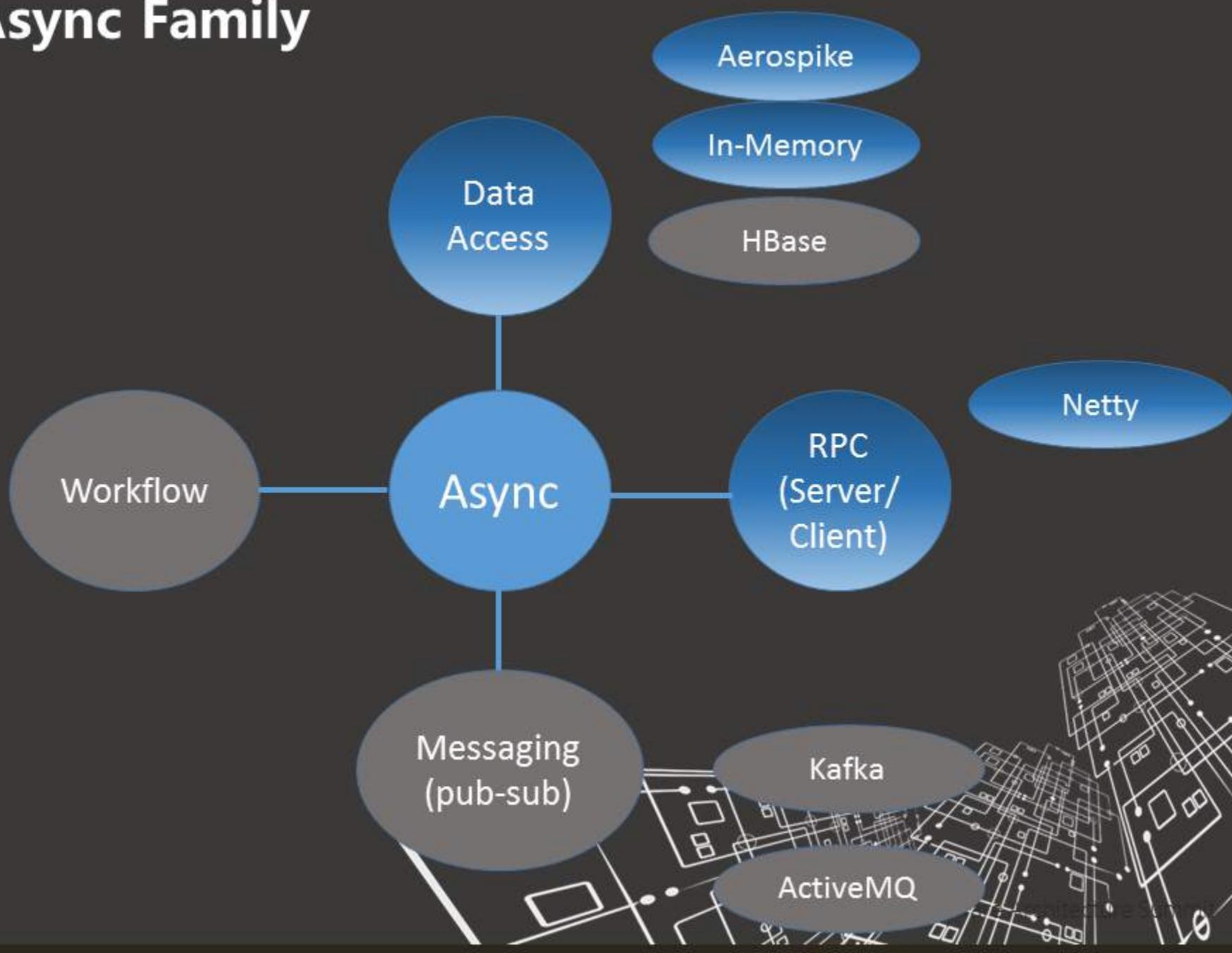
- High Performance**
- Low Latency + High Throughput
 - Low System Load
 - SLA Isolation
 - Understand Performance Contribution More
-

- Cost Saving**
- Less Hardware Investment
 - Loose Constraint for Hardware/VM SKU
-

- Easy Adoption**
- Zero Code Change + Zero Release (new case on-board)
 - Minimize new DB Storage Integration Effort
 - Lego-Style Customization
 - Highly Reusable Functionality
-

- High Flexibility Configuration**
- Execution Chain per URL (RPC)
 - DataAccess Storage & Option [consistency & ttl]
 - Traffic Routing Strategy
 - Replication Strategy

Async Family



AGENDA

PayPal & PayPal Risk (Platform)

Risk DAL Service Challenge

Async Solution

Async Future Plan

Future Plan

Open Source in Year 2019

Async+Sync Hybrid Workflow Execution

Async DataAccess

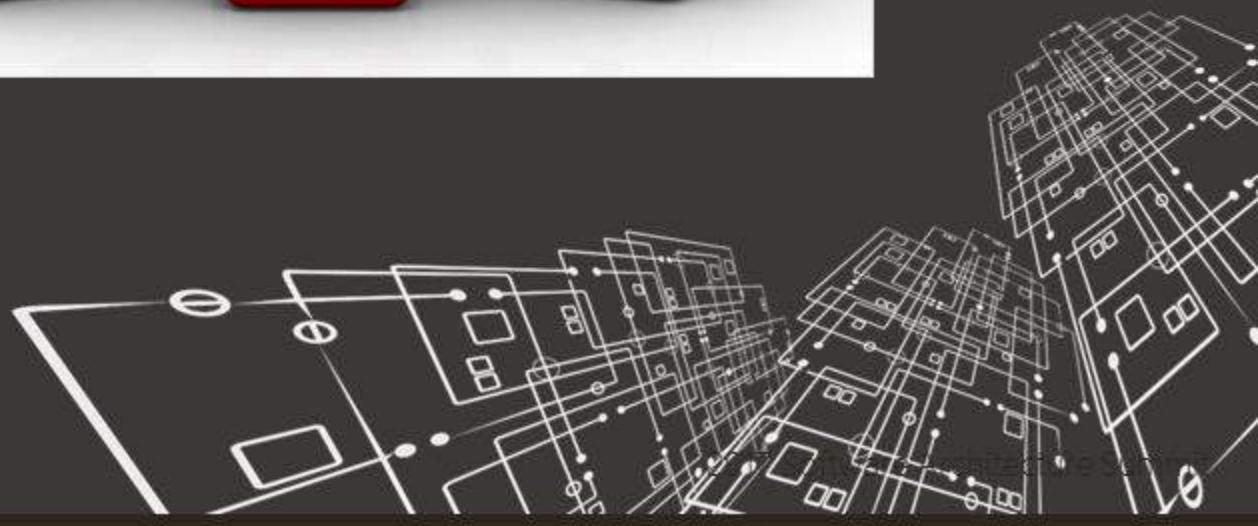
- Compute Operation Support
 - DB Server-side UDF Adoption
 - Smart Client for Direct & Service Access
 - Async HBase Integration
-

Async RPC

- Finer Granularity Monitoring & Throttling
 - Error Handling Injection
 - Client Side Multiplexing
 - Server Push Partial Response + RPC Client Consolidate Response
-

Continuous Performance Tuning Deep Dive

- Shared Eventloop
- Netty Option (IO Ratio)
- NIO vs Epoll SocketChannel
- JDK SSL vs OpenSSL
- Protobuf vs Msgpack
- Sync Client vs Async Client
- W/- Monitoring/Replication features



2017 Software Architecture Summit

打造58速运 高性能、高可用、实时消息平台

任桃术

@58速运



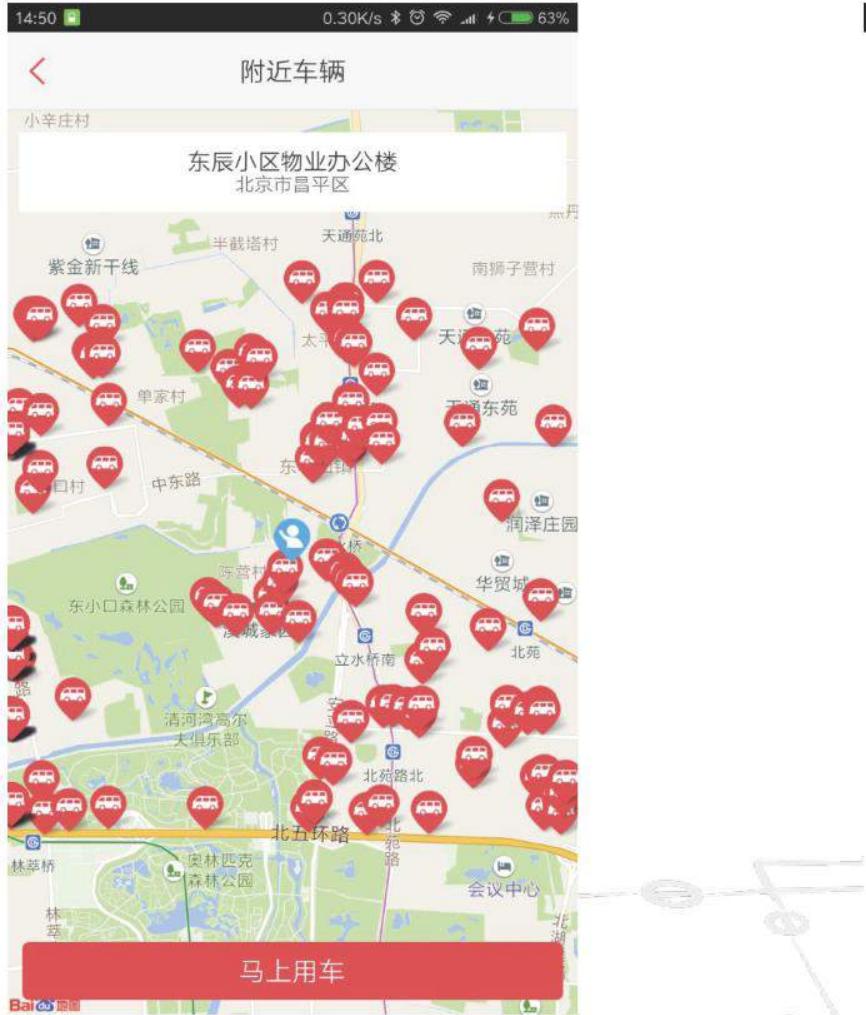
| 目录

- 业务背景
- 早期架构
- 58速运消息平台实践
 - 高性能、高可用、实时、高到达率、高扩展性
- 核心业务流程
- 总结



业务背景

- 58速运司机GPS位置实时上报



- 用户订单实时推送给司机



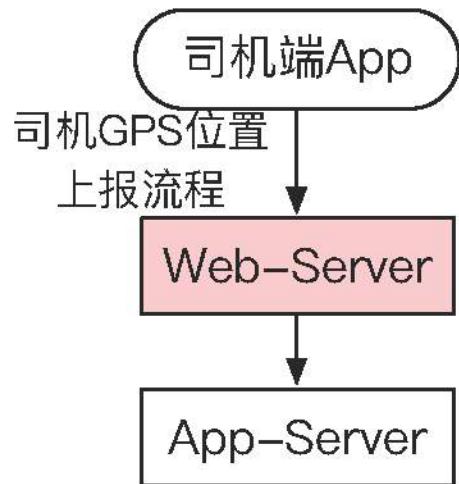
安全快捷

15秒快速响应 安全省心服务

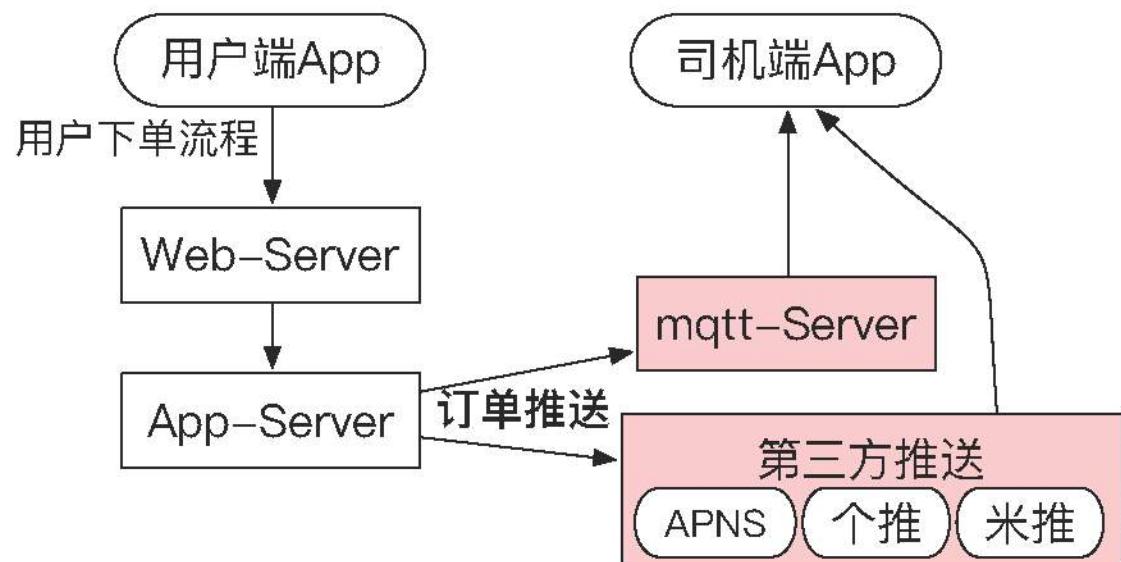


早期架构

- Web-Server性能问题



- 单点问题
- 三方推送及时性、限速、可达性

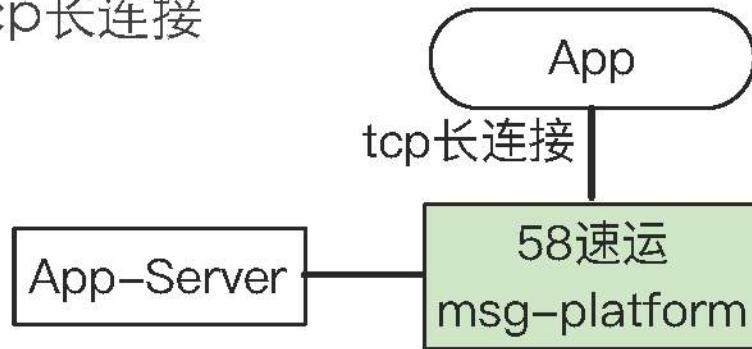


业务方诉求

高性能 高可用 实时 高到达率
消息平台

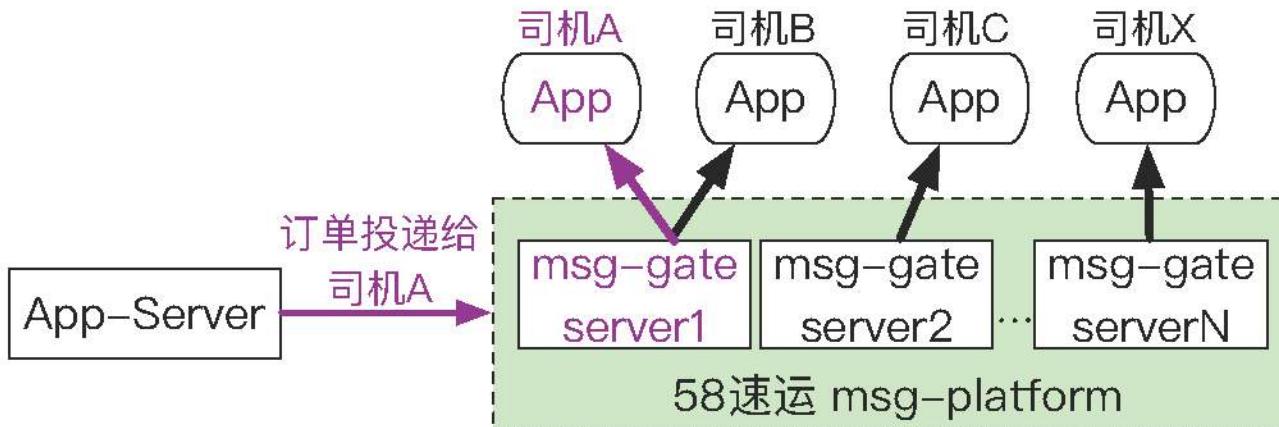
| 高性能

- 改用tcp长连接



- 潜在问题

实时订单推送 长连接状态维护

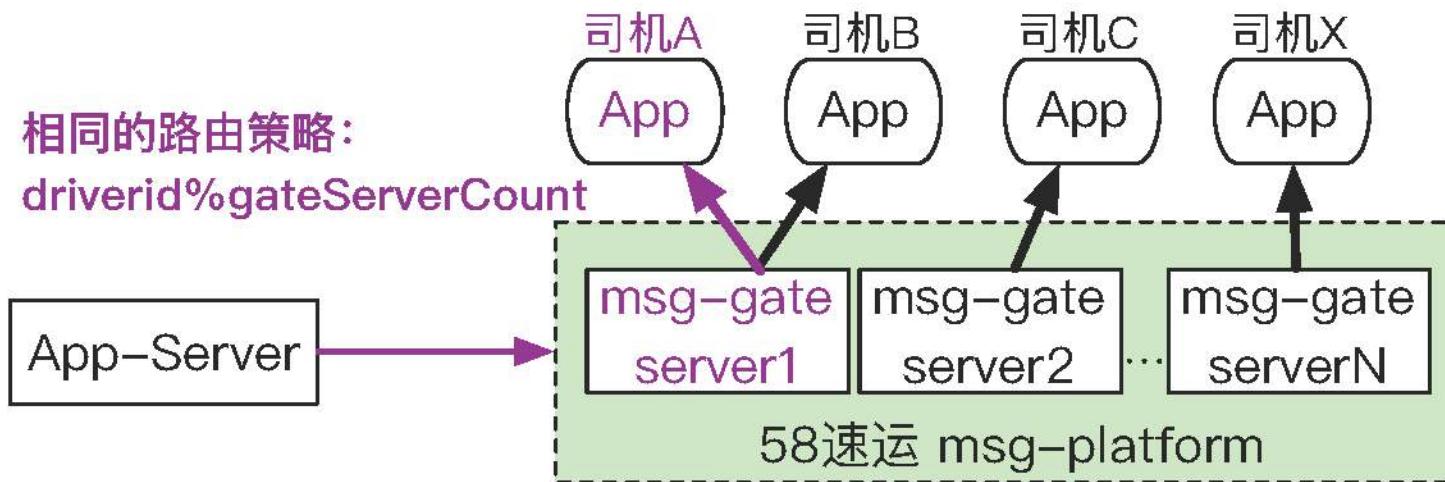


| 高性能 - 长连接状态维护

- 实时订单推送 长连接状态维护

方案：

- 1、App 和 App-Server 使用相同的路由策略；

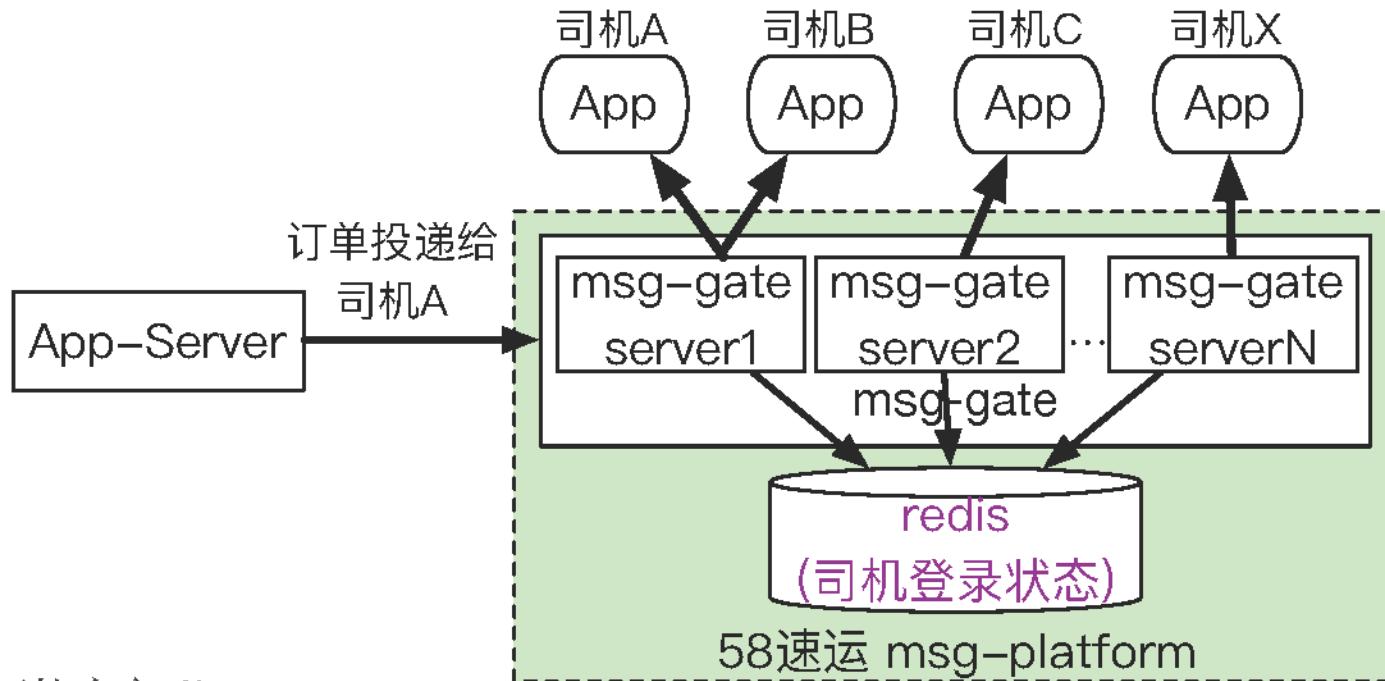


- 潜在问题

msg-gate-server 扩、缩容时，需要通知App端；
路由策略更新时，需要通知App端；

| 高性能 - 长连接状态存储(58速运实践)

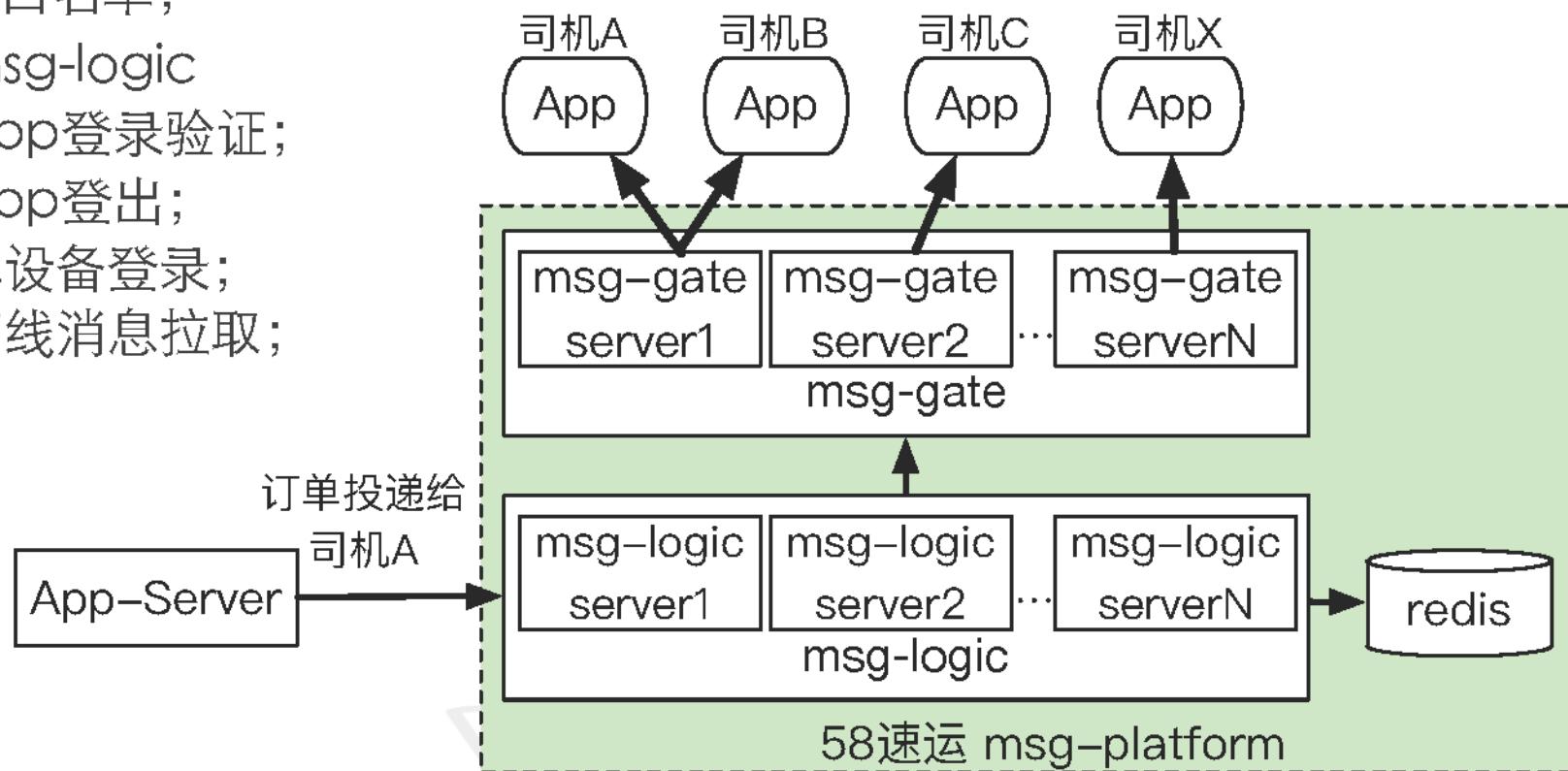
- 司机登录状态存储 <driver_id, {msg-gate ip:port}>
优点: msg-gate服务无状态



- 潜在问题:
msg-gate多层职责
维持长连接、路由、相关业务(司机登录、登出等)
业务更新相对较频繁，需要msg-gate重启，会导致长连接全部重连

| 分层架构

- msg-gate
百万连接管理；
流控；
黑白名单；
- msg-logic
App登录验证；
App登出；
单设备登录；
离线消息拉取；

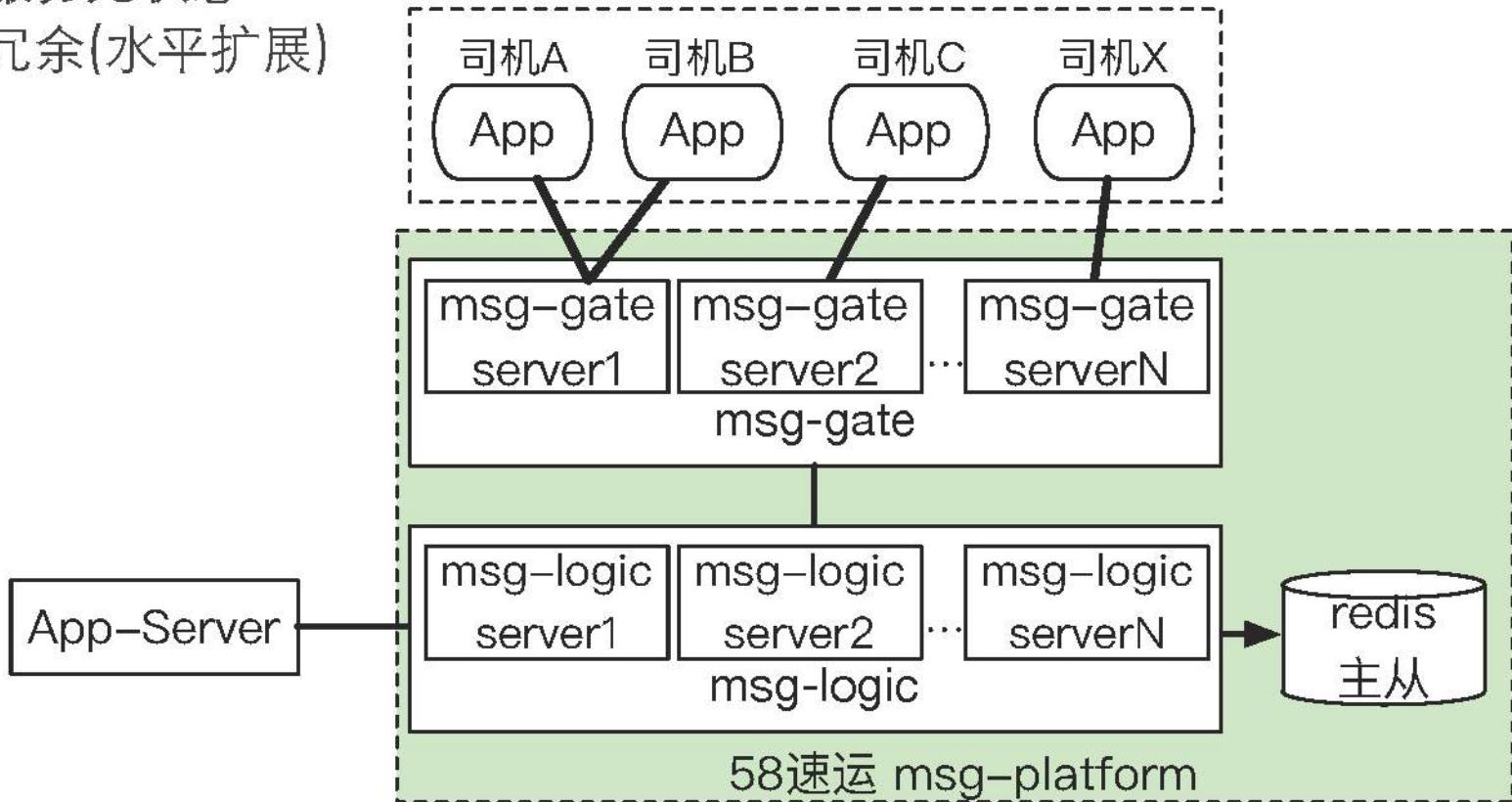


高可用 负载均衡



| 高可用

- 服务无状态
- 冗余(水平扩展)



msg-logic、msg-gate 负载均衡怎么做？

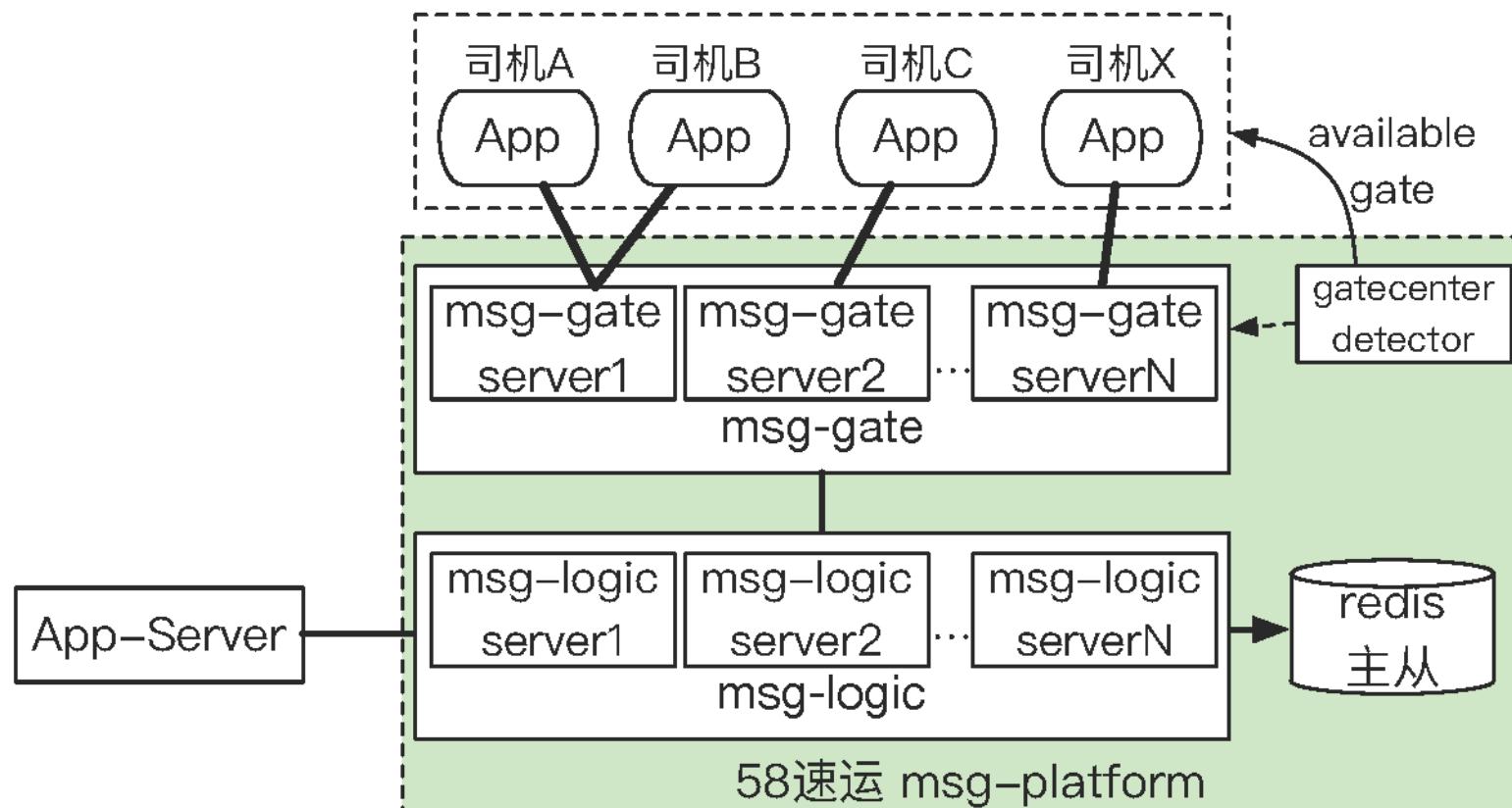
| 负载均衡

- 实时推送订单给司机

App-Server[客户端] -> msg-logic; msg-logic[路由] -> msg-gate;

- App端上报GPS到App-Server

App->msg-gate[服务端]; msg-gate[客户端]->msg-logic;



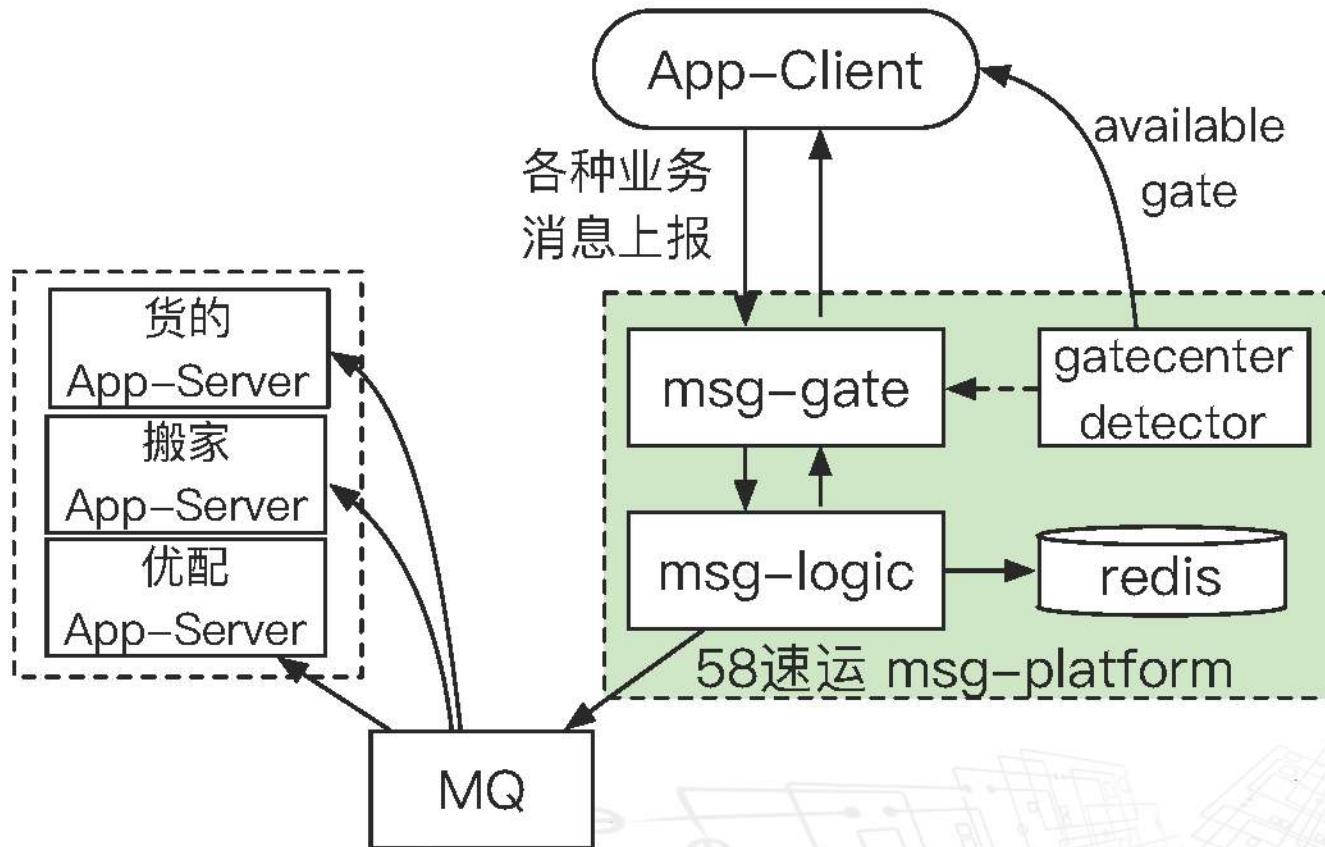
高可扩展性



业务扩展

- 通用消息平台

消息平台不关注具体业务



| 协议扩展

- 定长包头 + cmd + 变长包体

消息平台不关注具体业务

协议 版本	指令 扩展	多APP 支持	多业务 支持	变长包体	包体内容 pb序列化		
magic_num	version	cmd	appcode	msgtype	...	bodylength	data
	login	58速运	用户端	58速运	货的		
	logout	58速运	司机端	58速运	搬家		
	c2s			58速运	优配		
	s2c						
	c2c						
	keepalive						
	kickout						

| 58速运消息平台整体架构

- 分层设计

msg-Gate:连接整流、session管理、攻防

msg-Logic:登录、登出、离线消息拉取(IM)

- 高性能

tcp长连接、redis实时状态存储、异步化

- 高可用

服务无状态+冗余，可快速水平扩展

每个模块都有高可用考虑

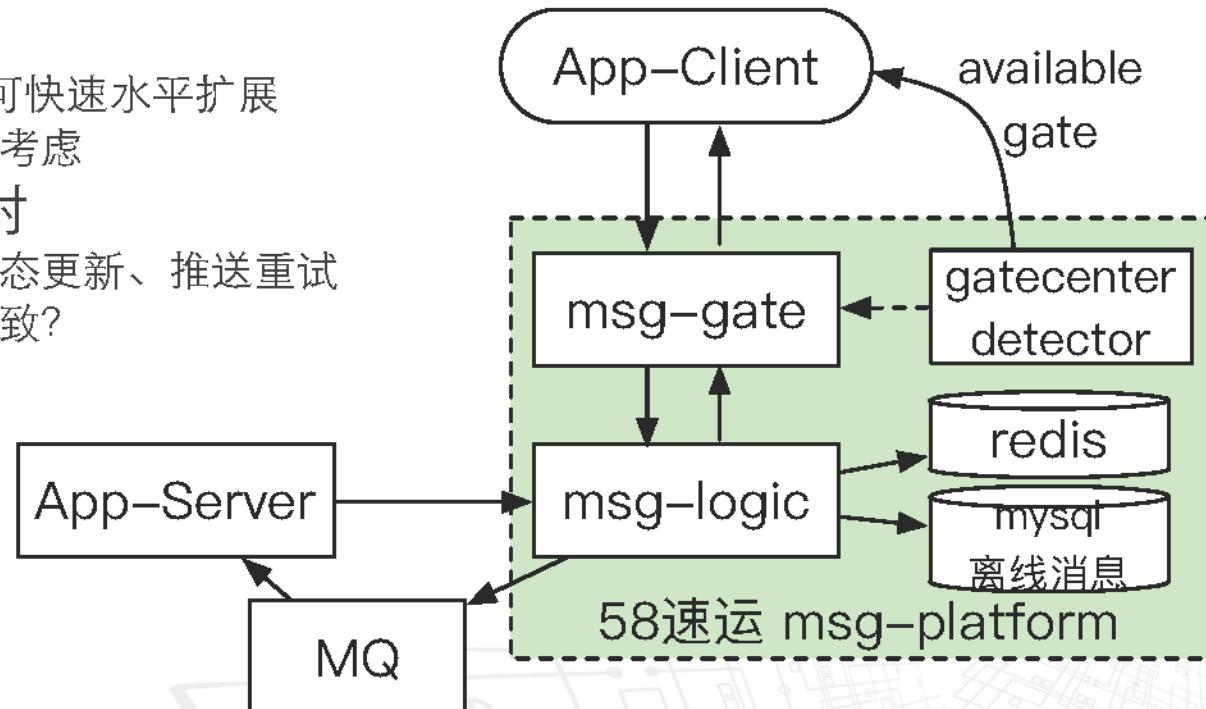
- 高到达率、实时

长连接保持、实时状态更新、推送重试

什么时候状态会不一致？

- 高可扩展

通用消息平台



核心业务流程

login(登录)

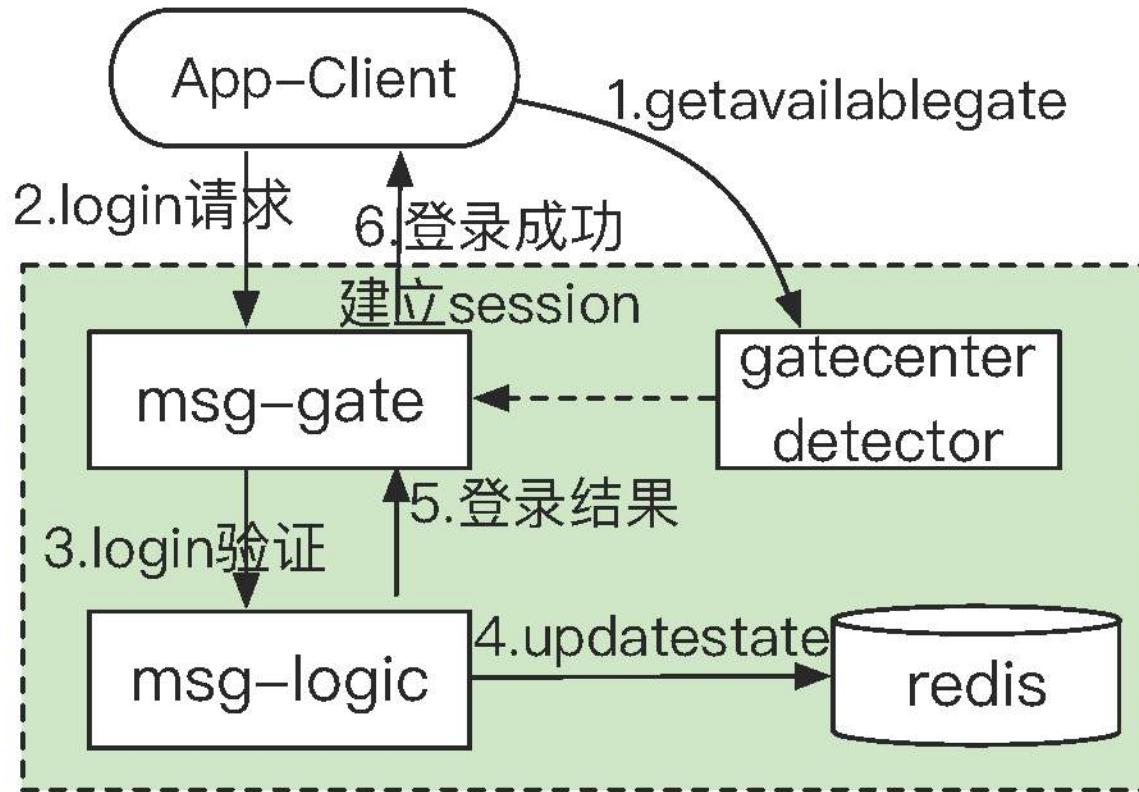
c2s(appClient To appServer)

s2c(appServer To appClient)

c2c(appClient To appClient)

| 核心业务流程 - login

- AppClient登录消息平台，登录成功后，建立连接session
- gate-center-detector负载均衡

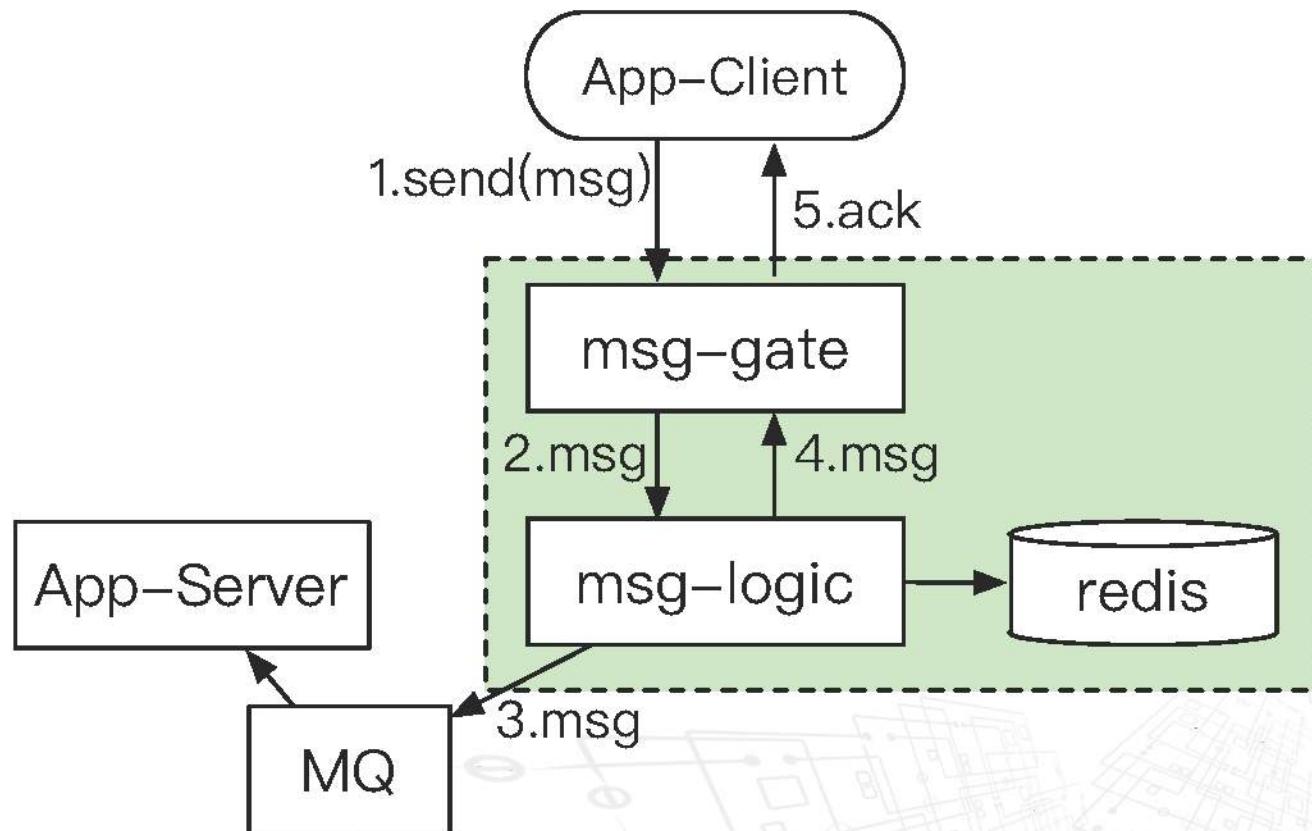


| 核心业务流程 - C2S

- appClient To appServer

消息平台与AppServer业务消息解耦

消息平台不关注具体消息内容

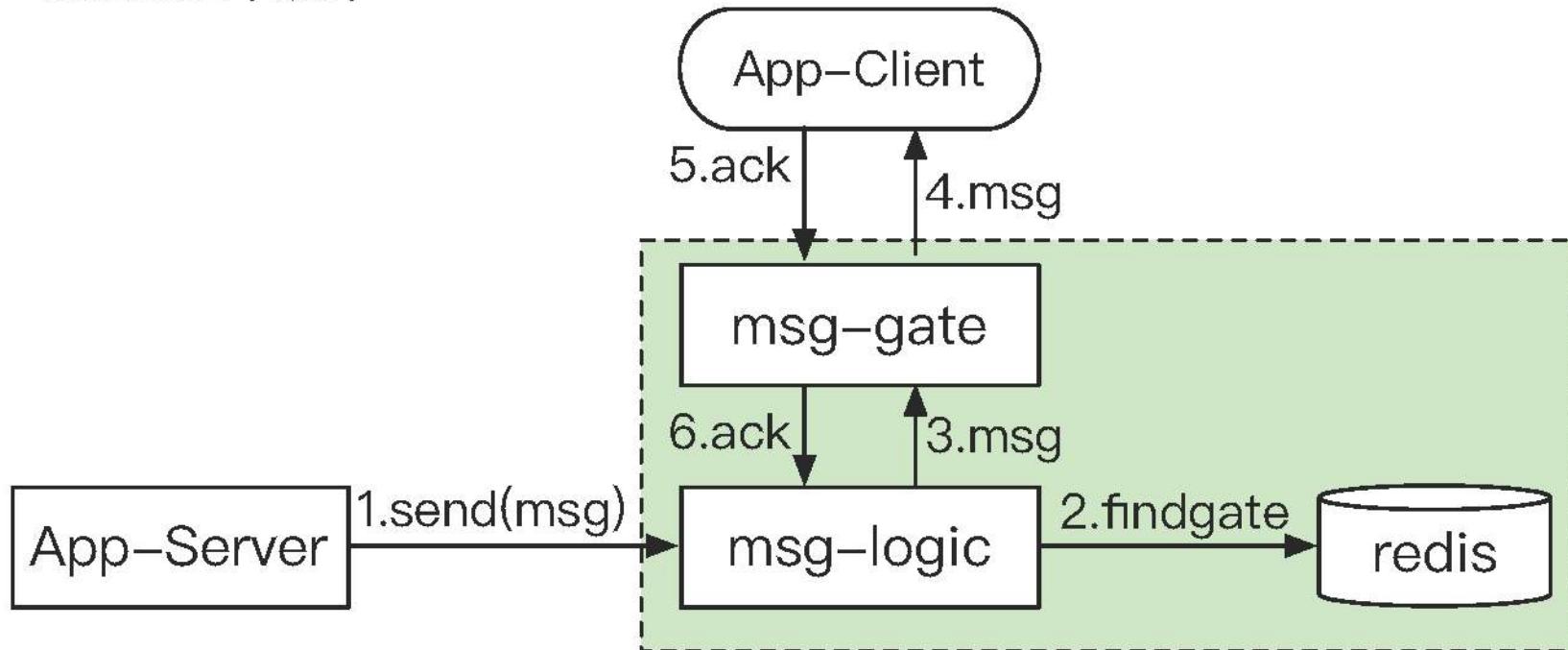


| 核心业务流程 - S2C消息

- appServer To appClient

找到msg-Gate进行实时推送

消息到达率统计



| 核心业务流程 - C2C消息

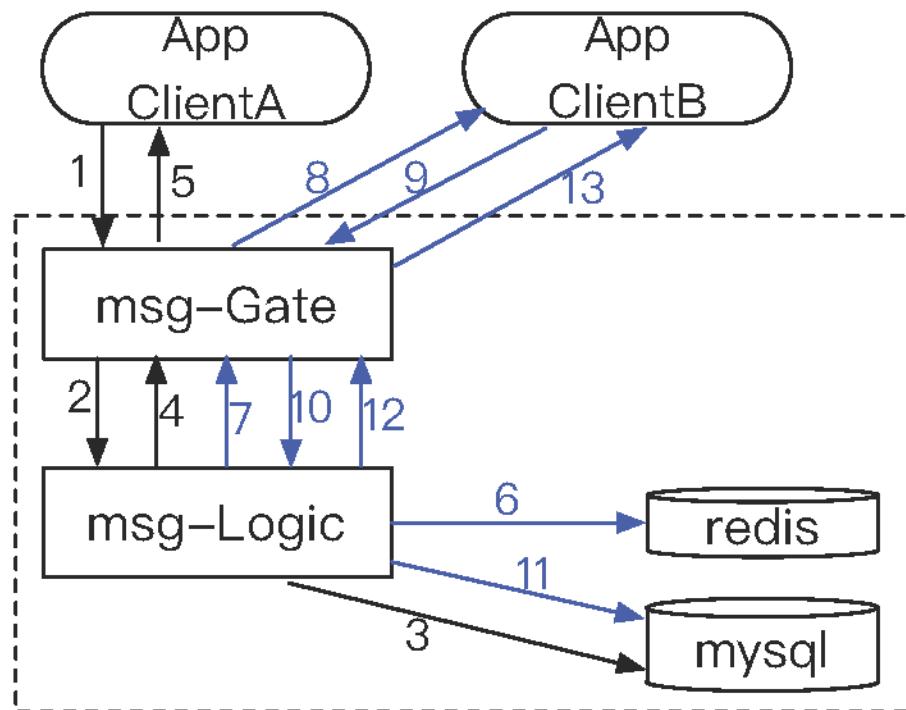
- appClient To appClient

移动端网络环境不稳定下，优化消息发送方体验

服务端代发消息接收方的ACK消息

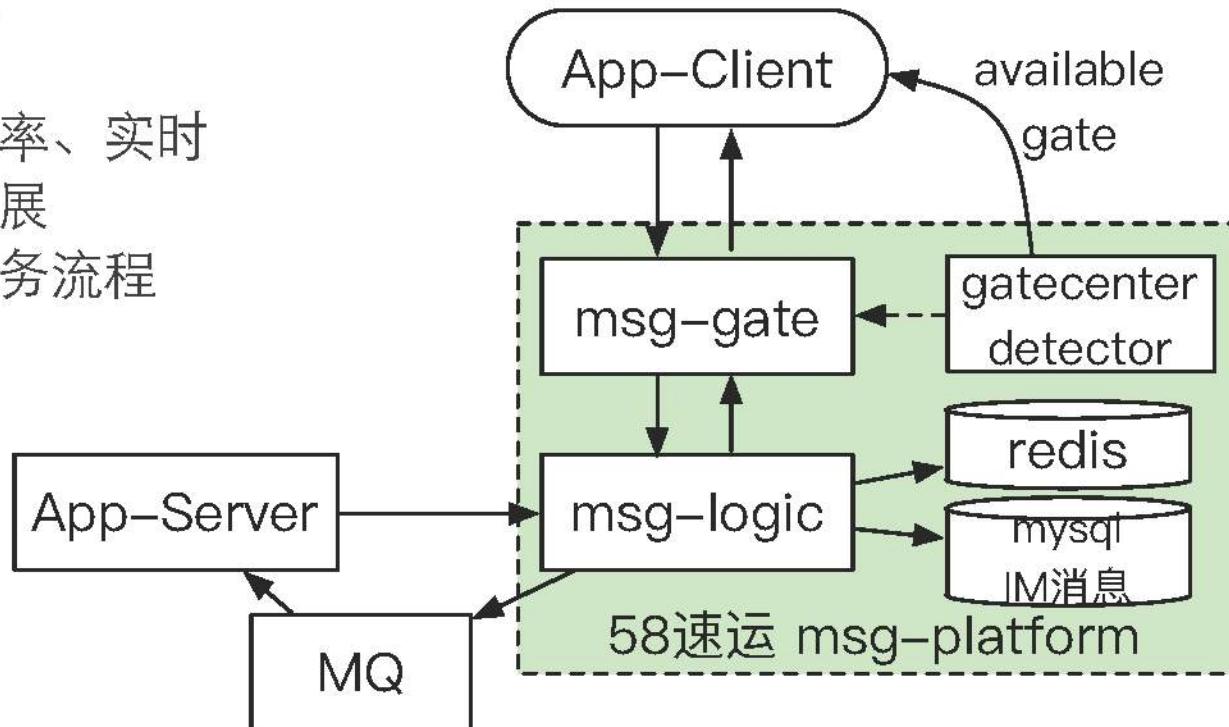
- 流程

- 1: 发送方发送IM消息；
- 2、3: 消息先落地存储；
- 4、5: 服务端代发ACK消息；
- 6: 查询消息接收方是否在线；
- 7、8: 如果在线，则实时推送；
- 9: 接收方ack确认收到消息；
- 10、11: 服务端从数据库中删除消息；
- 12、13: 服务端回复接收方ack；
(避免step 9重复ack)



| 总结

- 分层设计
- 高性能
- 高可用
- 高到达率、实时
- 高可扩展
- 核心业务流程



Q / A

架构师之路

有评论必回哟。



2017 Software Architecture Summit

基于 **Kubernetes** 的高可用 MySQL 微服务实践

张翼飞

七牛开发工程师



| 主题

- 挑战
- MySQL 集群常见故障
- MySQL 集群数据备份/恢复
- Operator 简介
- 关键问题
- 服务保障
- 架构设计
- 交付
- 总结

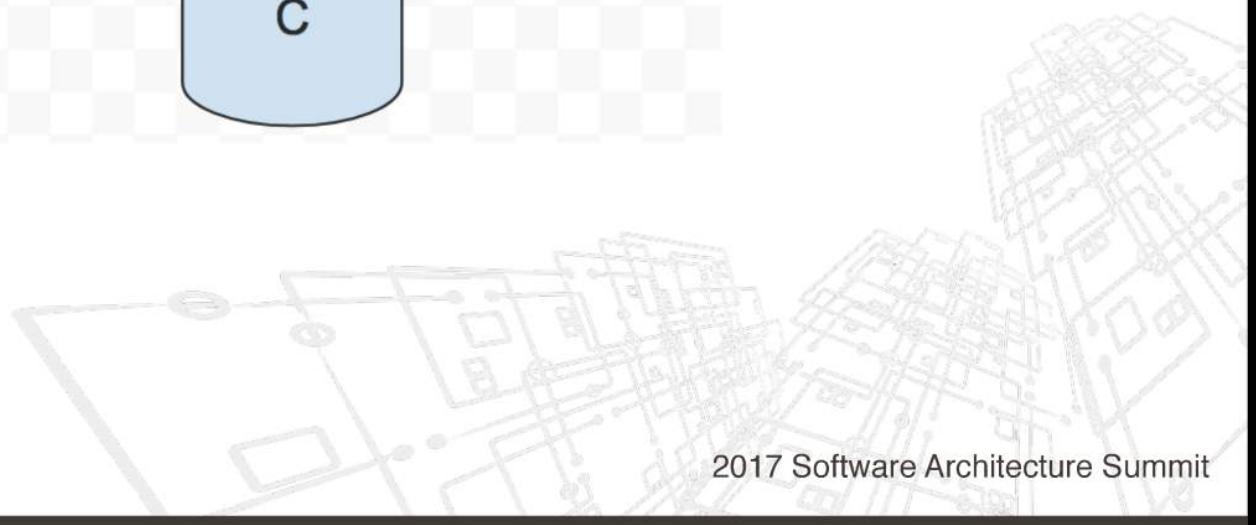
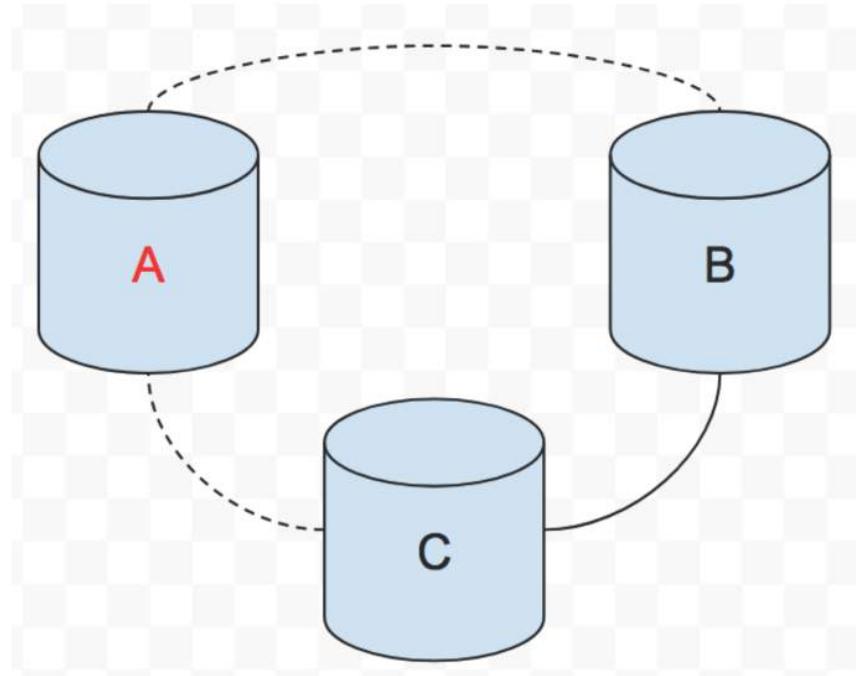
| 挑战

- MySQL 服务高可靠
- MySQL 服务高可用
- 基于 Kubernetes
- MySQL 服务自运维
- 支持多租户
- 易于部署和升级



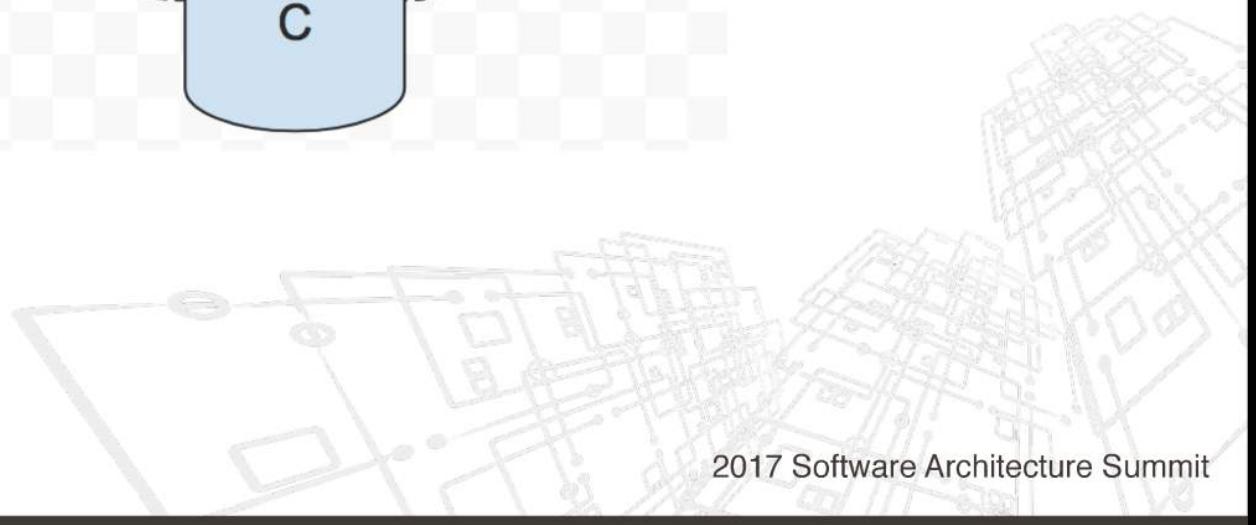
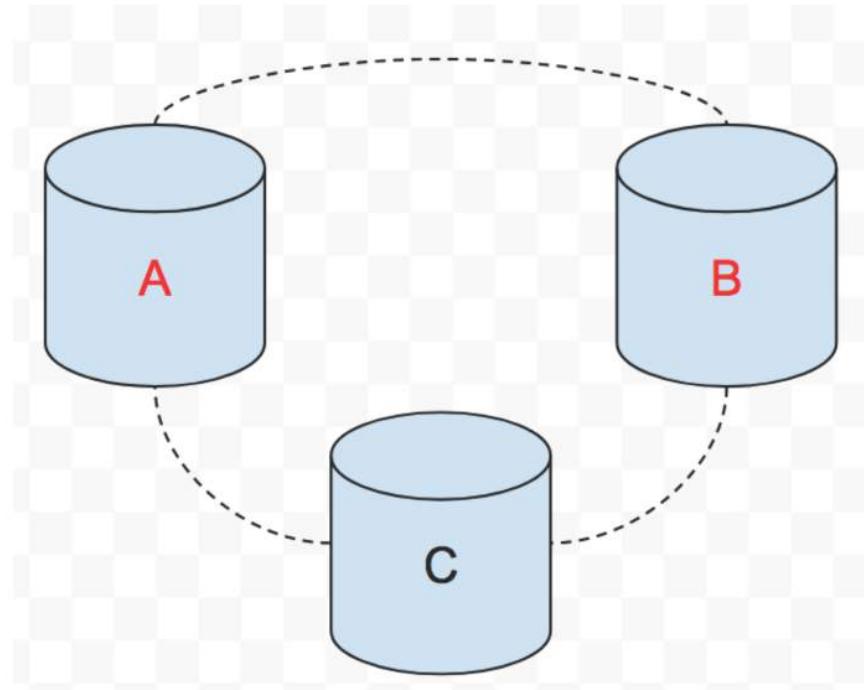


| MySQL 集群常见故障_一个节点停止服务



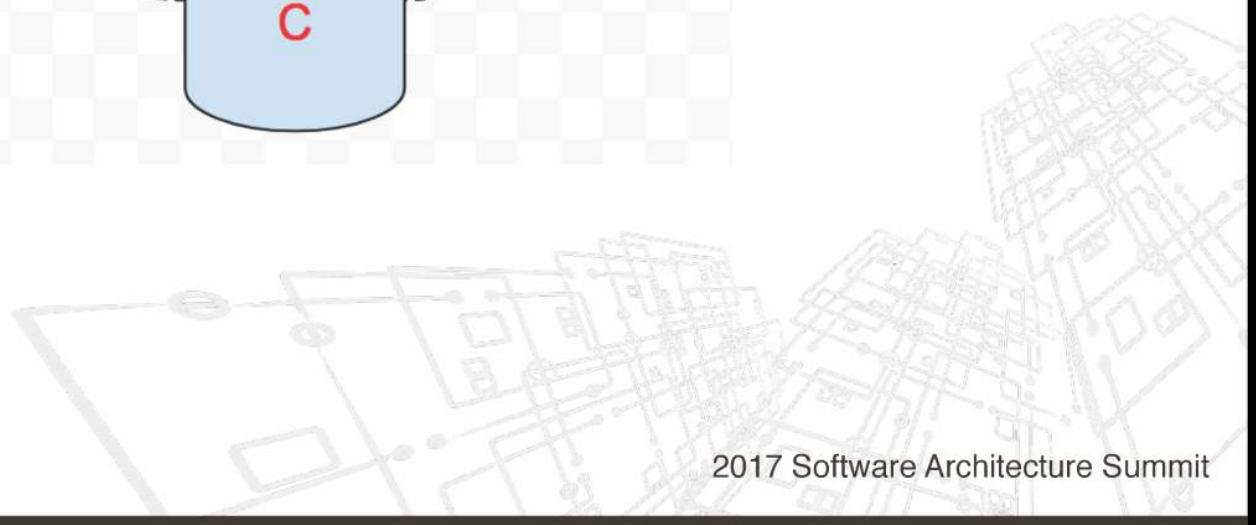
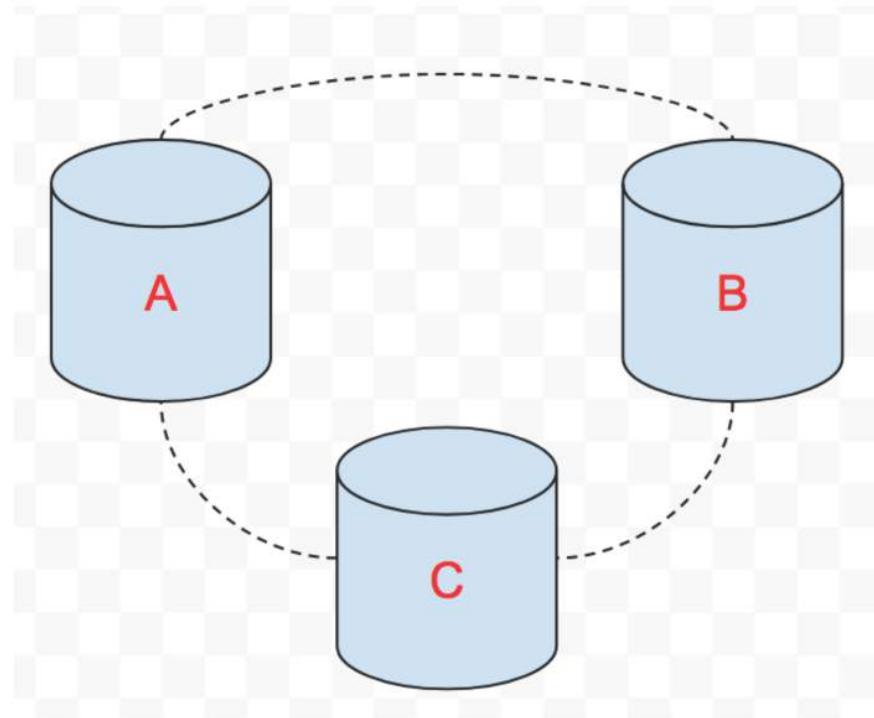


| MySQL 集群常见故障_两个节点停止服务



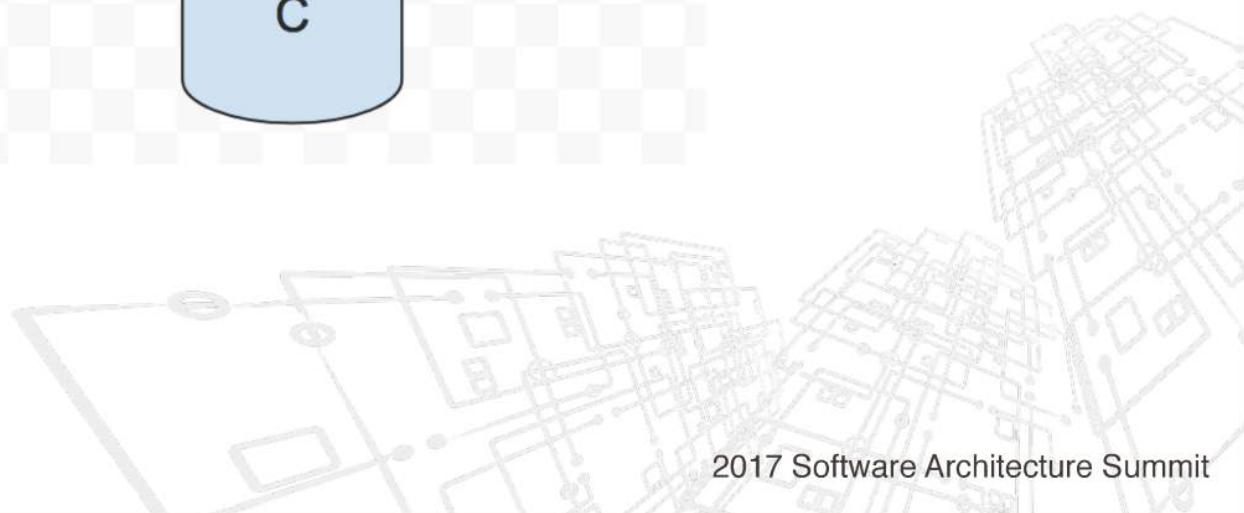
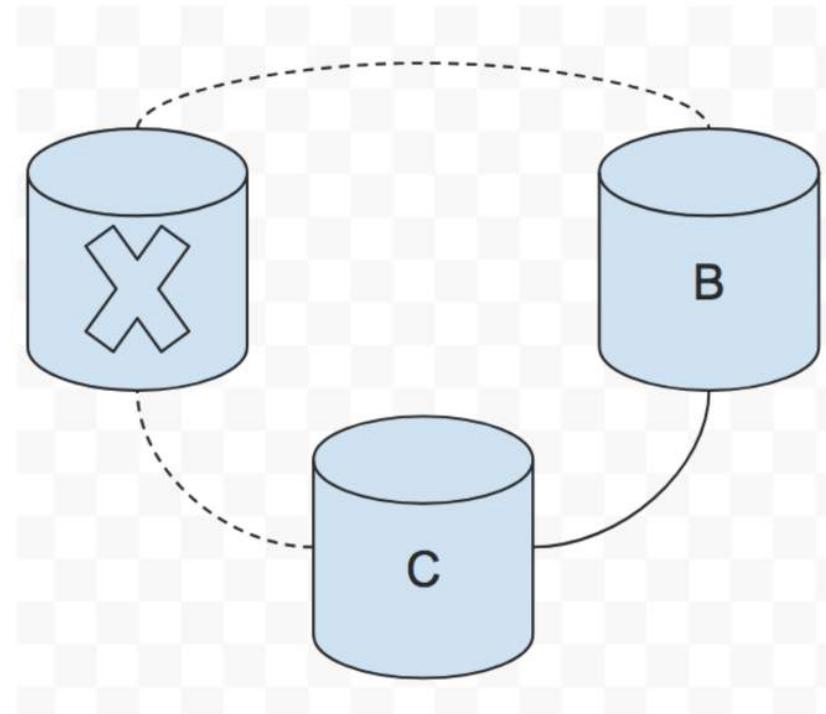


| MySQL 集群常见故障_全部节点停止服务



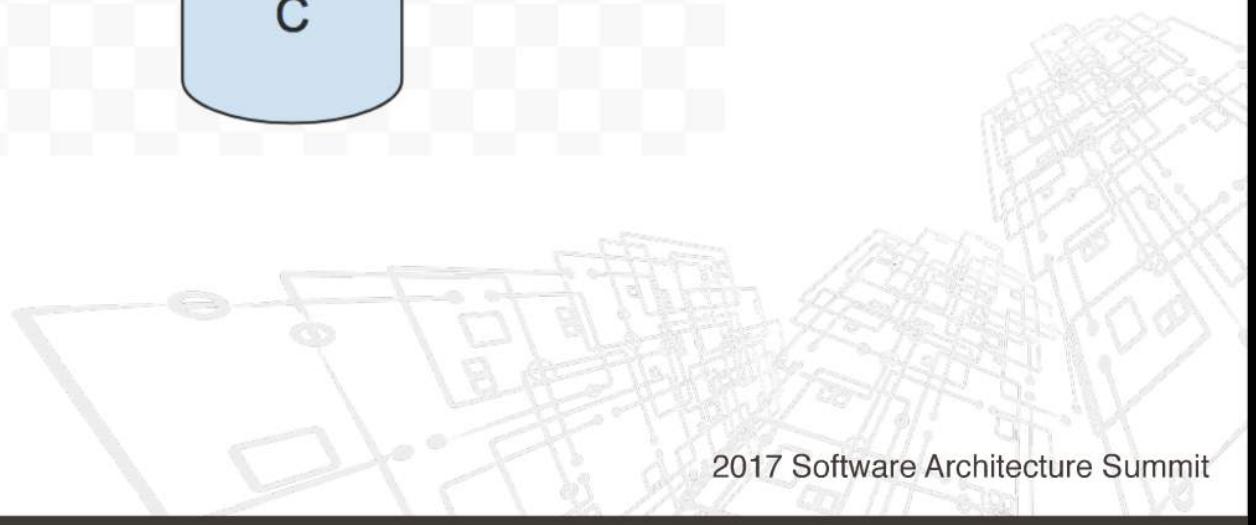
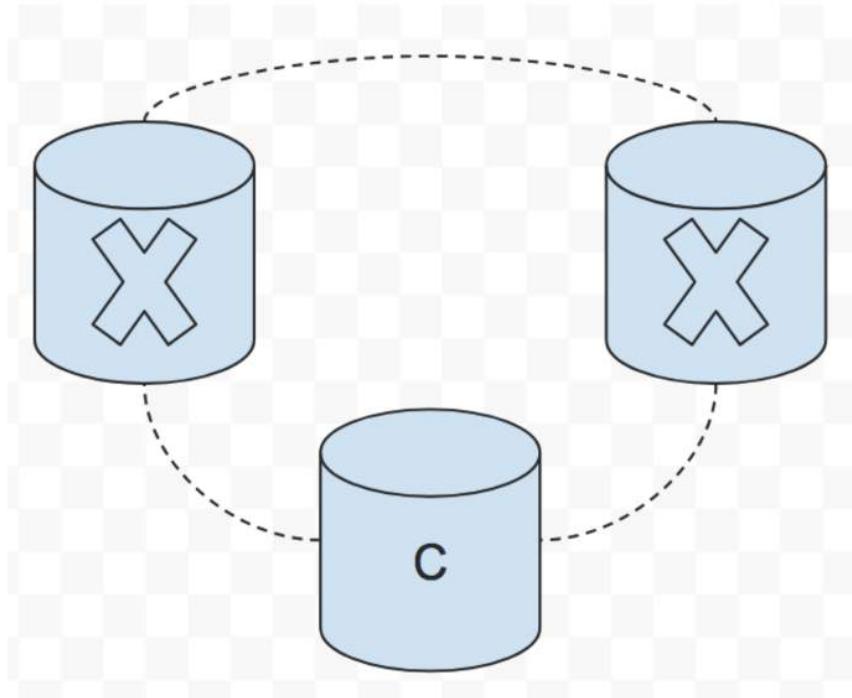


| MySQL 集群常见故障_一个节点挂掉



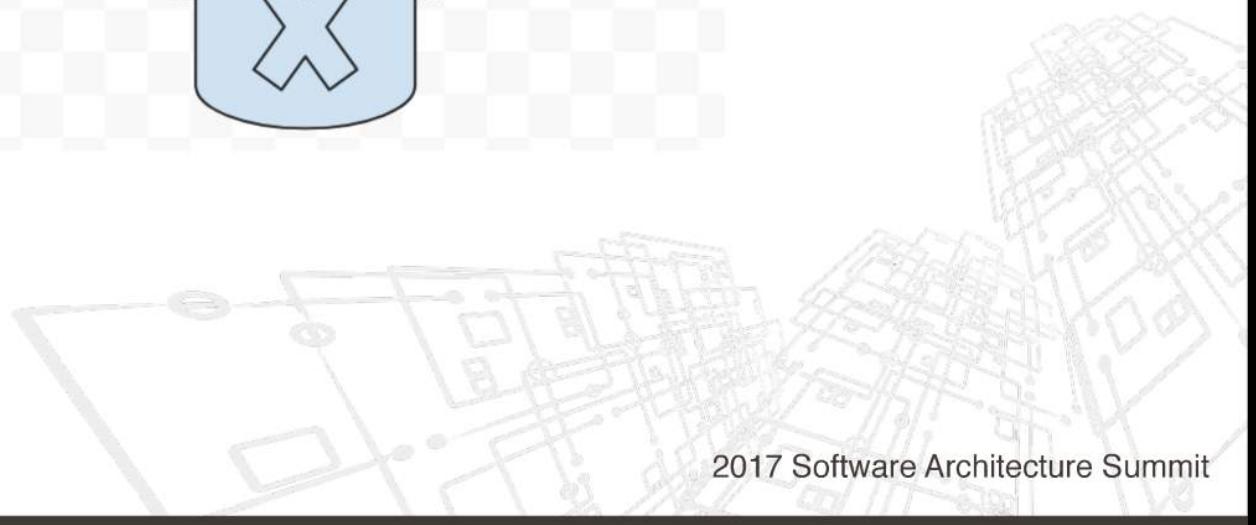
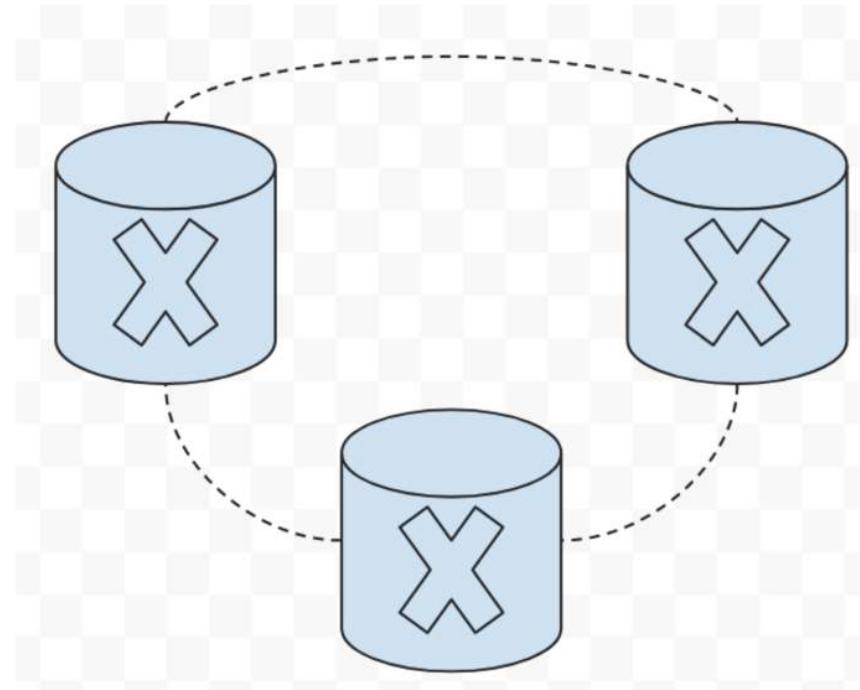


| MySQL 集群常见故障_两个节点挂掉



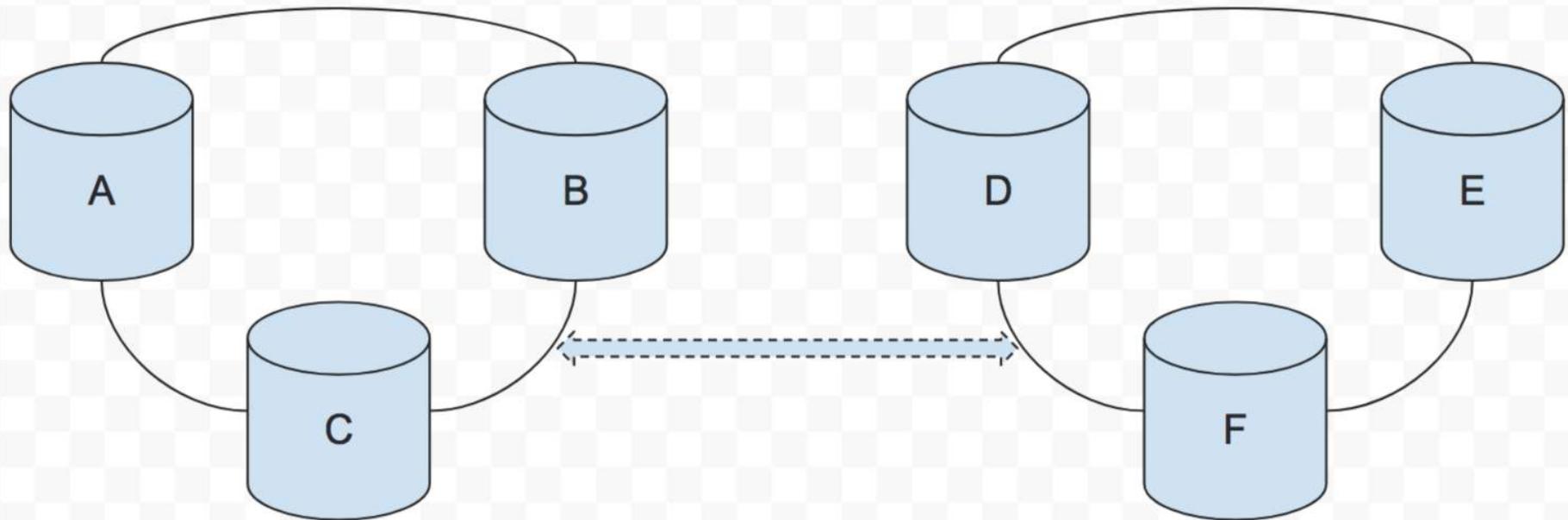


| MySQL 集群常见故障_全部节点挂掉





| MySQL 集群常见故障_脑裂



| MySQL 集群数据备份/恢复_如何备份

- 通过 XtraBackup 工具实现「热备份」

| MySQL 集群数据备份/恢复_如何恢复

- 备份当前状态数据
- 暂停全部节点服务
- 选取某个节点作为 bootstrap 节点
- 在该 bootstrap 节点上恢复数据
- 启动其他的节点



| Operator 简介



Operator 是什么？



针对特定应用的 Controller，封装应用的领域内知识。





| Operator 简介



Operator 解决的问题是什么？



如何为用户提供易用的基于 Kubernetes 的复杂应用。





| Operator 简介



Operator 的设计原则有哪些？

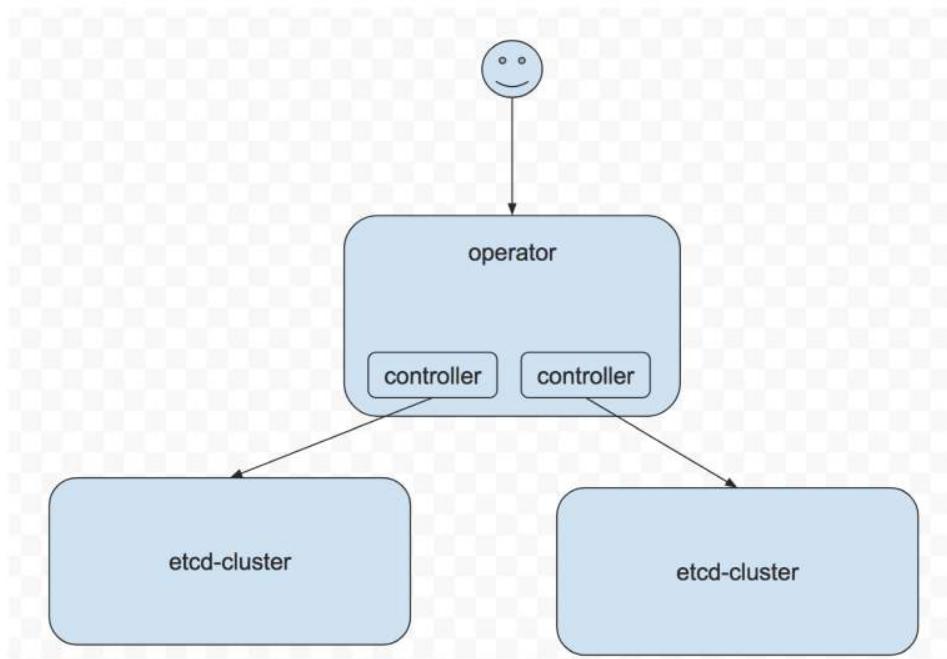


- Operator 仅通过一个简单的 Deployment 部署，且无需其他操作
- Operator 需要在 Kubernetes 中创建一个 Resource 表征应用
- Operator 需要使用 Kubernetes 中现有的服务来管理应用
- Operator 需要设计为它的停止和删除不影响已部署的应用
- Operator 需要支持应用的升级





Operator 简介_etcd-operator 架构



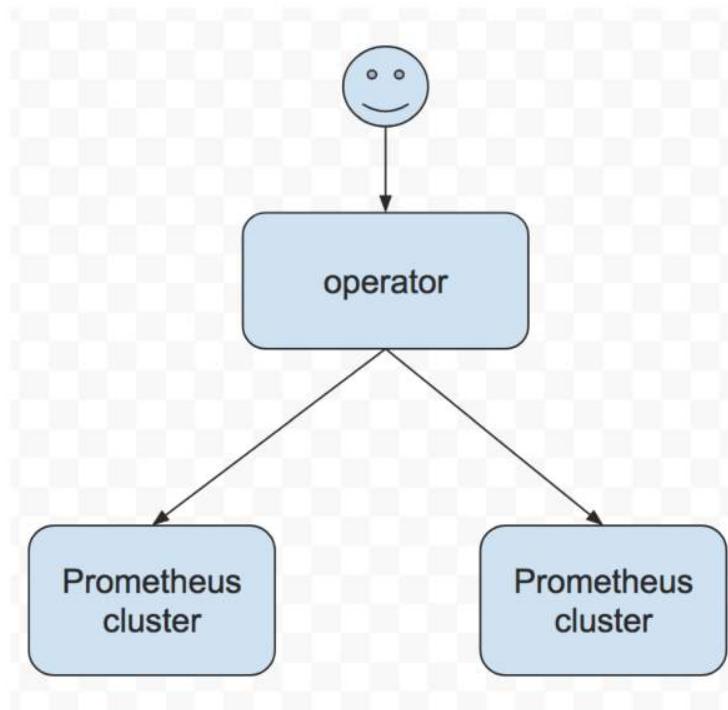
Operator 负责：

- 创建/删除 etcd 集群
- 扩缩容 etcd 集群
- 恢复 etcd 集群
- 备份/恢复 etcd 集群
- 升级 etcd 集群

- Operator 为每个 cluster 创建一个 Controller 管理该 cluster 的生命周期
- Controller 在该 cluster 生命周期内以 goroutine 方式持续运行
- Controller 定期 poll Pod 状态来做故障检测和自动处理



Operator 简介_prometheus-operator 架构



Operator 负责：

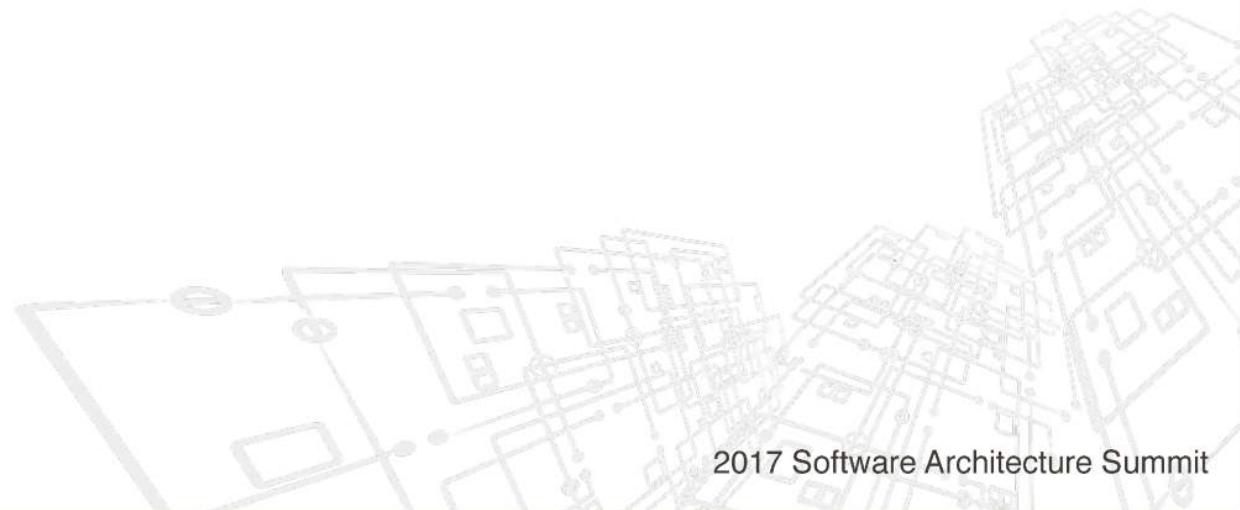
- 创建/删除 Prometheus 服务
- 创建/删除 ServiceMonitor 服务
- 创建/删除 AlertManager 服务

- Operator 使用 StatefulSet 创建集群
- 定时 re-list objects
- 所有监听到的事件入 Queue 串行处理



| 关键问题_如何控制复杂度

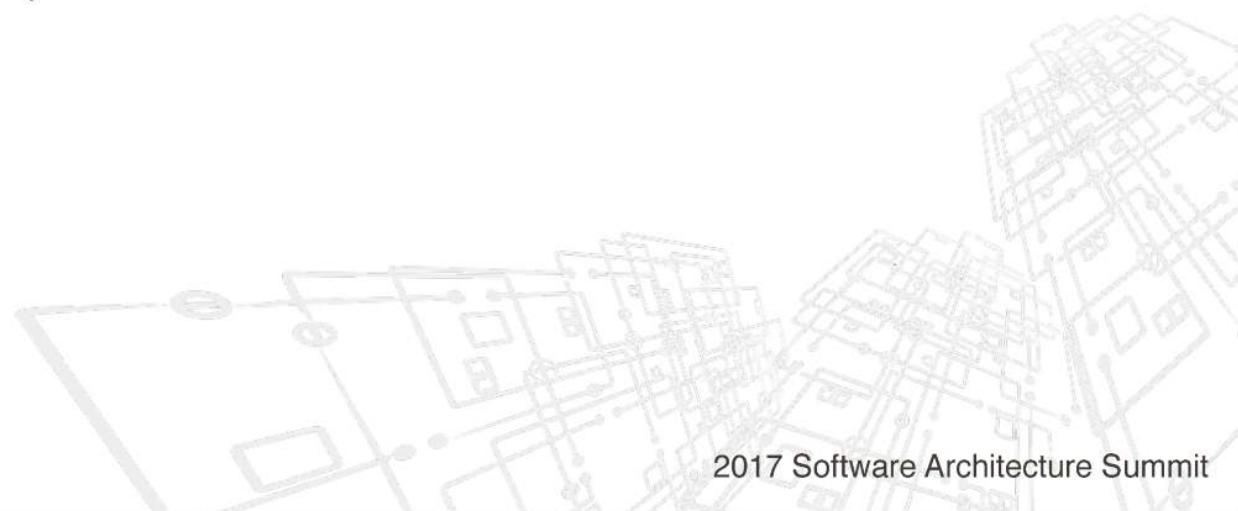
- 分层设计
- 状态驱动
- 机制与策略分离
- 服务解耦





| 关键问题_如何确保服务状态符合期望

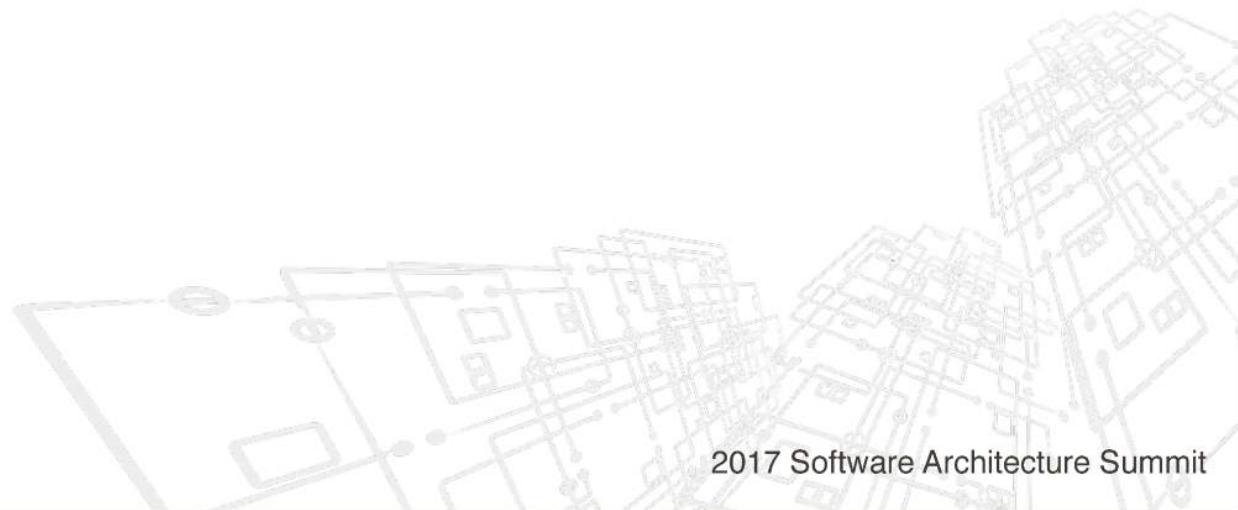
- Operator 不断驱动服务「实际状态」转变为「期望状态」
- 服务的状态信息存储在 Kubernetes 的 objects 中
- Operator 定时 re-list objects
- Operator 根据 object 的状态，触发相应的任务
- Controller 收集状态信息并反馈给 Operator
- Operator 修改 object 状态





| 关键问题_如何控制不同逻辑之间的并发执行问题

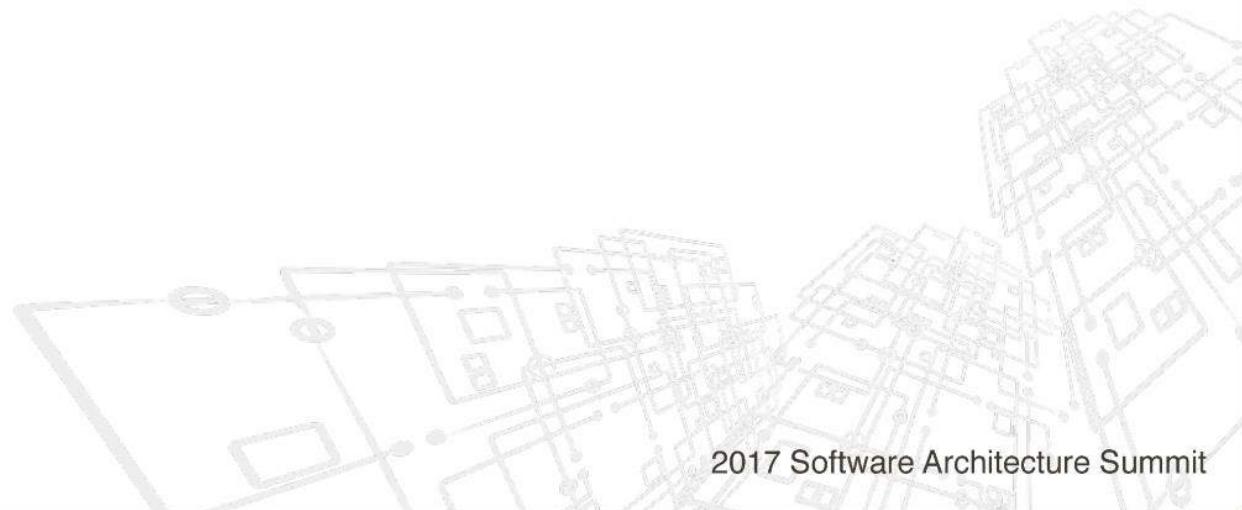
- 由 Controller 管理任务调度
- 同步执行任务





| 关键问题_如何升级服务

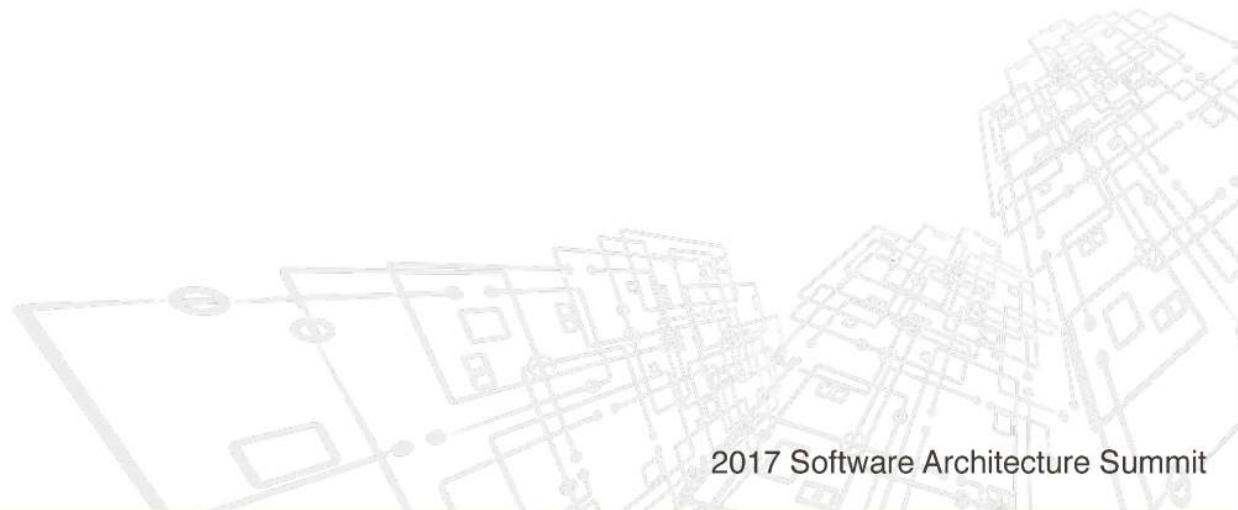
- 使用 Kubernetes 特性，将「升级服务」转变为「升级 SPEC」





| 关键问题_Kubernetes 的环境限制

- 「Pod 异常」是正常情况，且是常态
- 「网络异常」是正常情况，且非常态
- 「存储异常」是正常情况，且非常态





| 服务保障_高可靠

- 数据冗余
- 数据定时/手动备份
- 数据持久化在 PV 上
- 并发逻辑串行化调度





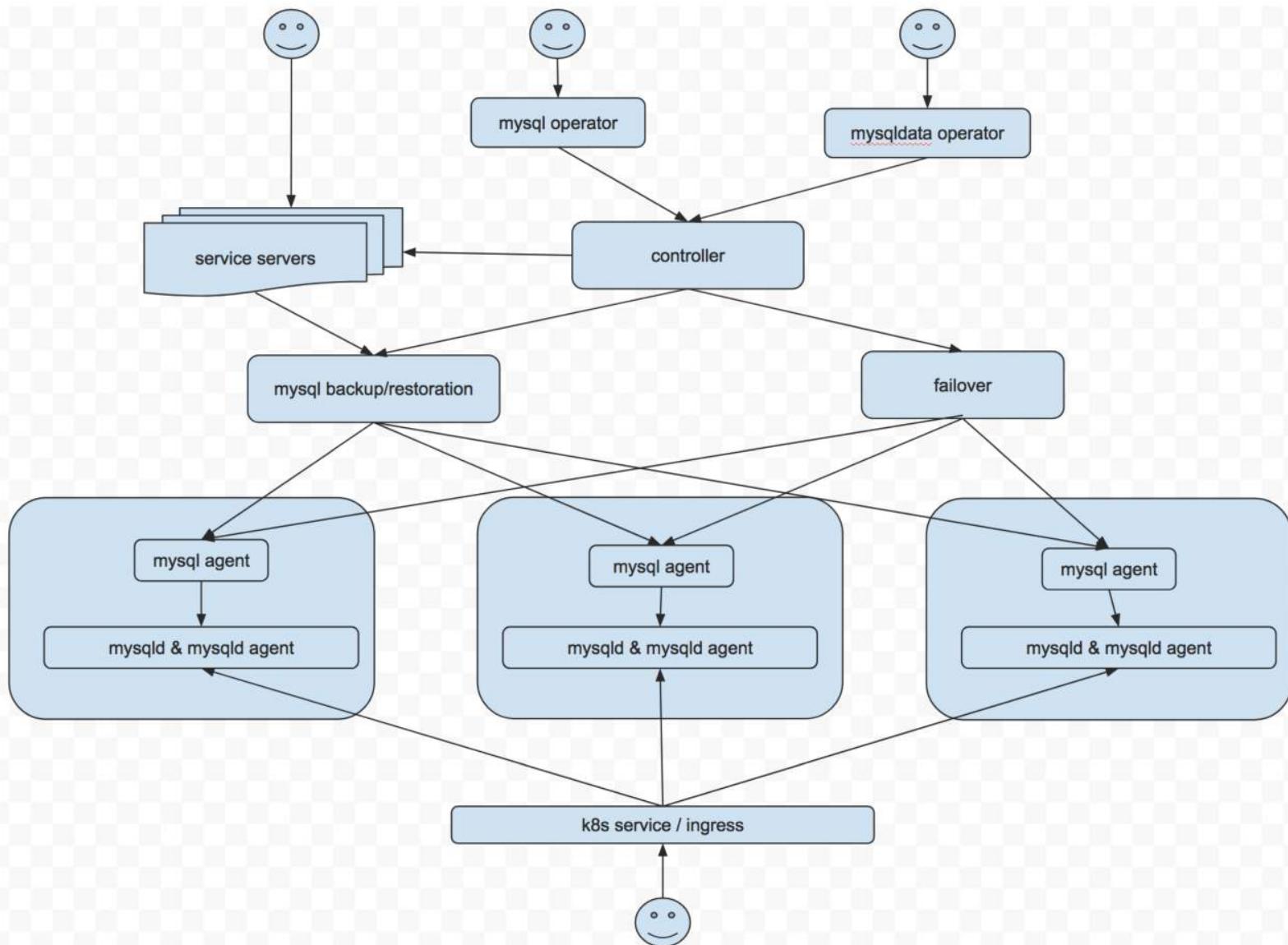
| 服务保障_高可用

- 服务冗余
- 状态驱动
- MySQL Pod 通过 StatefulSet 方式部署
- 每个 MySQL 集群自治





架构设计





| 架构设计 _OperatorInterface

```
type CRD struct {
    Name    string // format: Plural.Group
    Kind    string
    Plural  string
    Group   string
    Version string
    Scope   apiextensionsv1beta1.ResourceScope

    Obj          runtime.Object
    ObjList      runtime.Object
    SchemeBuilder func(*runtime.Scheme) error
}
```

```
type OperatorInterface interface {
    // CreateCRD creates the CRD resource.
    CreateCRD(*CRD) error
    // DeleteCRD deletes the CRD resource.
    DeleteCRD(name string, options *metav1.DeleteOptions) error
    // GetCRD gets the CRD resource.
    GetCRD(string) (*apiextensionsv1beta1.CustomResourceDefinition, error)

    // WatchEvents starts the controller to handle with the CRD's
    // ADD/DELETE/UPDATE events.
    WatchEvents(context.Context, *CRD) error
}
```



| 架构设计_ControllerInterface

```
type ControllerInterface interface {
    // Init initialize the controller.
    Init() error

    // Run starts the loop.
    Run() error

    // AddTask adds a task for the controller.
    AddTask(Task) error
    // PopTask pops a task for the controller to handle with.
    PopTask() (Task, error)

    // Feedback reports the current status of the MySQL Cluster to
    // the MySQL operator.
    Feedback(*Status) error

    // RestoreCluster restores the MySQL Cluster.
    RestoreCluster(*FaultInfo) error

    // RestoreData restores the data of the MySQL Cluster.
    RestoreData(*Dataset) error

    // ScheduleBackup implements the scheduled backup task.
    ScheduleBackup(*ScheduleInfo) error

    // BackupOnce does a backup operation.
    BackupOnce() error
}
```



| 架构设计_FailoverInterface

```
type FailoverInterface interface {
    // Init initialize the worker and create MySQL Cluster.
    Init(*FailoverConfig) error

    // Restore tries to check the fault type and restore
    // the MySQL Cluster.
    Restore() error

    // CheckFaultType checks the fault type.
    CheckFaultType(*FaultInfo) (FaultType, error)

    // ProcessFault tries to process the fault.
    ProcessFault(FaultType, *FaultInfo) error

    // CheckStatus checks the status of the MySQL Cluster
    // periodically.
    CheckStatus() (*Status, error)

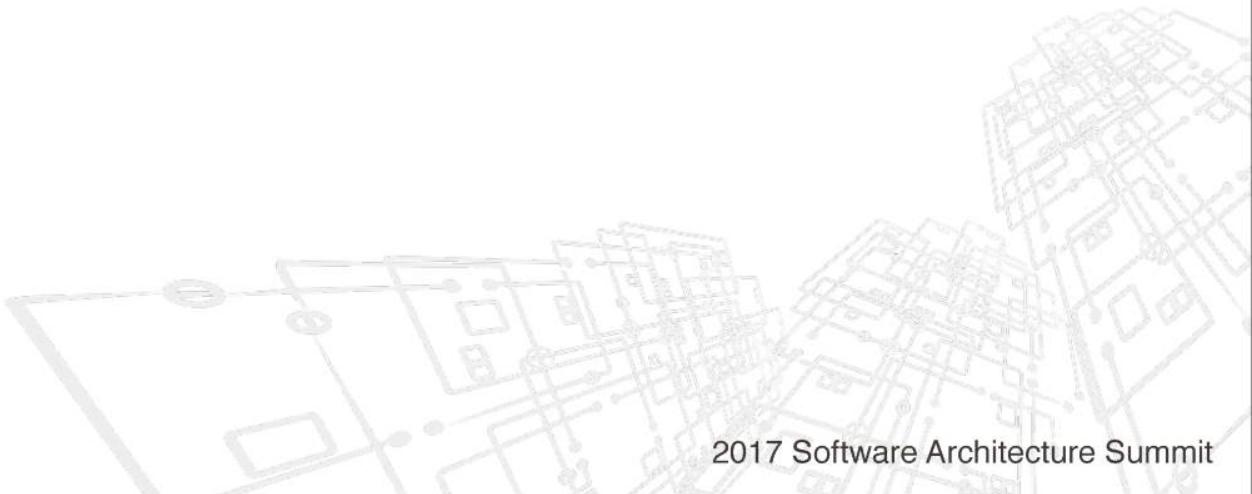
    // Feedback reports to the controller the status of the
    // MySQL Cluster.
    FeedBack(*Status) error
}
```





| 交付_Chart

- Chart 是 Kubernetes 生态圈中开发、分享、使用应用的标准
- 交付 Core Chart 和 Vendor Chart
- 通过 helm 来一键使用 Chart





| 交付_Chart示例

→ ~ helm create example

Creating example

→ ~ tree example

example

|-- charts

|-- Chart.yaml

|-- templates

| |-- deployment.yaml

| |-- _helpers.tpl

| |-- ingress.yaml

| |-- NOTES.txt

| '-- service.yaml

`-- values.yaml



| 交付_部署 MySQL Operator

→ helm inspect ./mysql-operator

```
apiVersion: v1
```

```
---
```

```
---
```

```
replicaCount: 1
```

```
name: mysql-operator
```

```
registry: index-dev.qiniu.io/kelibrary
```

```
imageOperator: mysql-operator:201709281752
```

```
nameMySQLOperatorServer: mysql-operator-server
```

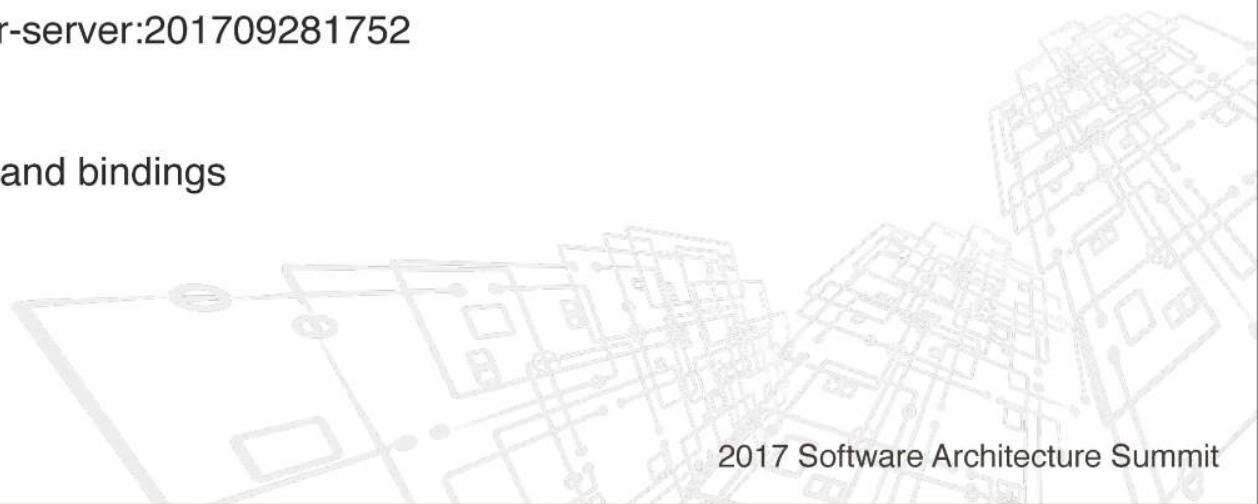
```
imageServer: mysql-operator-server:201709281752
```

```
---
```

```
# Install Default RBAC roles and bindings
```

```
rbac:
```

```
install: false
```





| 交付_部署 MySQL Operator

→ helm install --namespace rds --set name=example-mysql ./mysql-operator/





| 交付_部署 MySQL 服务

→ helm inspect ./mysql-service

...

name: example

backup:

enable: true

backupTime: 01:01,13:01

reservedDays: 7

resources:

limits:

cpu: 100m

memory: 500Mi

storage: 1Gi

uid: default

mysqlRootPassword: PASSWORD





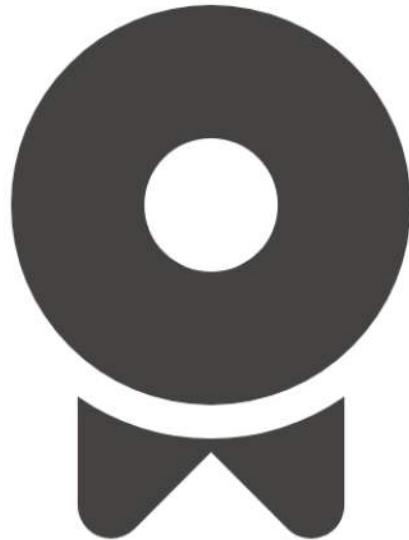
| 交付_部署 MySQL 服务

→ helm install --namespace rds --set name=example-mysql ./mysql-service/



| 总结

- 通过 Operator 和 Controller 封装 MySQL 领域内知识
- 简化 Operator 逻辑，通过 Controller 实现 MySQL 集群自治
- Controller 统一管理所在 MySQL 集群的任务分发
- 以 Chart 方式交付应用



Thanks



2017 Software Architecture Summit

基于Elasticsearch的搜索平台

曹林华

沪江架构师



| 个人介绍

- 前百度 高级开发工程师
- 2016年加入沪江，多年技术研发和系统架构经验
- 目前负责沪江搜索平台、日志平台、实验平台、分布式跟踪系统

| 目录

- 产品介绍
- 搜索实践
- 监控保障
- 总结



产品介绍

主站搜索



搜索 查词

搜索结果

搜索相关课程

雅思口语怎么练习？

1、词汇基础 其实在雅思口试里用词其实也没必要绝对准确，而且对一般的同学来说，7分就是相当好的分数了，即使7分也没必要全是“高级词汇”但是考试当中的基础核心词汇，我们应该保证100%准确，不基础的词汇实在用不准就...

2016-06-11 口语练习 雅思口语 雅思

雅思写作达人养成建议

雅思写作考试考的是语言应用能力，雅思的重点在于考生如何使用语言。雅思作文中需要华丽的词语吗？怎么样才能获得雅思写作的高分~小编为大家整理了一些雅思写作建议，以及正确的技巧，供考生备考之用。雅思写作：不需要华丽的...

2014-08-26 雅思考试 雅思写作 考试热门 雅思

【雅思必读】雅思口语话题大全

近年来雅思口语考试的话题丰富多彩，层出不穷。为了方便烤鸭们能够有针对性地进行口语练习和掌握大致性的口语考试方向，小编最近整理了做常见的雅思口语考试的话题，烤鸭们可以根据实际性需要去模拟训练，请看—— part 1 : ...

2014-11-07 雅思 口语 必读 大全 雅思口语 考试热门 考试口语

阅读排行榜

- ① 新概念英语第一册Lesson 1 Exc...
- ② 史上最全！英语面试问题回答大...
- ③ 新概念英语第二册Lesson 1 A pr...
- ④ 5国语言达人 这10部美剧最适合...
- ⑤ 外媒看中国：雄安新区一夜走红...
- ⑥ 面试中如何回答“做个英语自我介..

热搜榜

- ① 英语在线翻译 30965
- ② 英语翻译 10731
- ③ 英语四六级成绩查询 6906
- ④ 大学英语四级成绩查询 5005

IM 搜索

搜索“英语”，共找到 74438 个结果

节目



韦林文化

WeLearn第一次万人模考解析
（英语一）

2.07万次播放



英语标准发音学习策略

1.27万次播放



口语 | 跟美国大学外教一起学英语

2858次播放



英语词汇学习策略

6153次播放



手把手教你拆解长难词2

3190次播放

全部18915个节目 >

用户

全部34052个用户 >



英语大仙

用户名: 英语大仙

0 关注 0 粉丝 0 节目



英语小虾

用户名: 英语小虾

1 关注 2 粉丝 0 节目



英语逆袭屌丝

用户名: 英语垫底小王子

0 关注 0 粉丝 0 节目



英语渣渣

用户名: 英语渣渣me

0 关注 0 粉丝 0 节目



英语大白

用户名: 英语小白大

0 关注 8 粉丝 0 节目

电商搜索



英语

搜索

我收藏的课程

全部课程 > "英语"

共 507 个结果

全部 英语 小语种 出国留学 考研 财会金融 职场兴趣 中小幼教育

当前水平: 零基础 低水平 有基础 初中水平 高中水平 大学及以上水平 其它

班型: 1对1 VIP定制班 小班 大班 协议班 体验班 其它班型 押题班 刷题班 名校班 基础培优班 考前专项班

授课模式: 课件课 直播课 视频课 混合课 直播+课件

开班时间: 报班即学 十月 十一月 十二月

综合 价格 人气 ↓

1 / 26 页 >



新概念英语1、2、3、4册连读【学霸班】试听

【今日特价课 限时秒杀】超值体验，新概念4册连读直达大学英语六级水平

超级学习团

¥1249

¥2538

随到随学 有效期490天 | 总366课时 老师: Amy老师, Angela老师, Sue老师, Ada老师, Lilian老师

查看详情



英语零基础入门【美式音标随到随学班】试听

零起点，字母音标扎实学，语音障碍全解决。

¥69

随到随学 有效期30天 | 总14课时 老师: 瑶瑶老师

查看详情

搜索遇到的问题



搜索效果不好

中英日文搜索效果不满意



跨库跨表无法查询

数据量变多，需要采取分库分表查询



业务需求变多

越来越多的部门有接入搜索的需求



搜索实践



Elasticsearch 介绍

What





全文搜索



分布式高可用



RESTful API



近实时搜索与更新



面向文档



Schema free



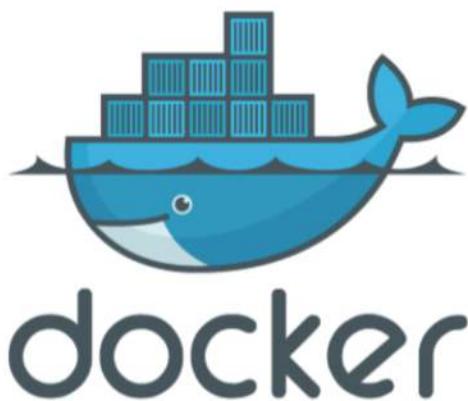
Who





U B E R

ebay



整体架构



业务线

主站

网校

IM

CRM

搜索平台

Index

Search

AB Test

Elasticsearch

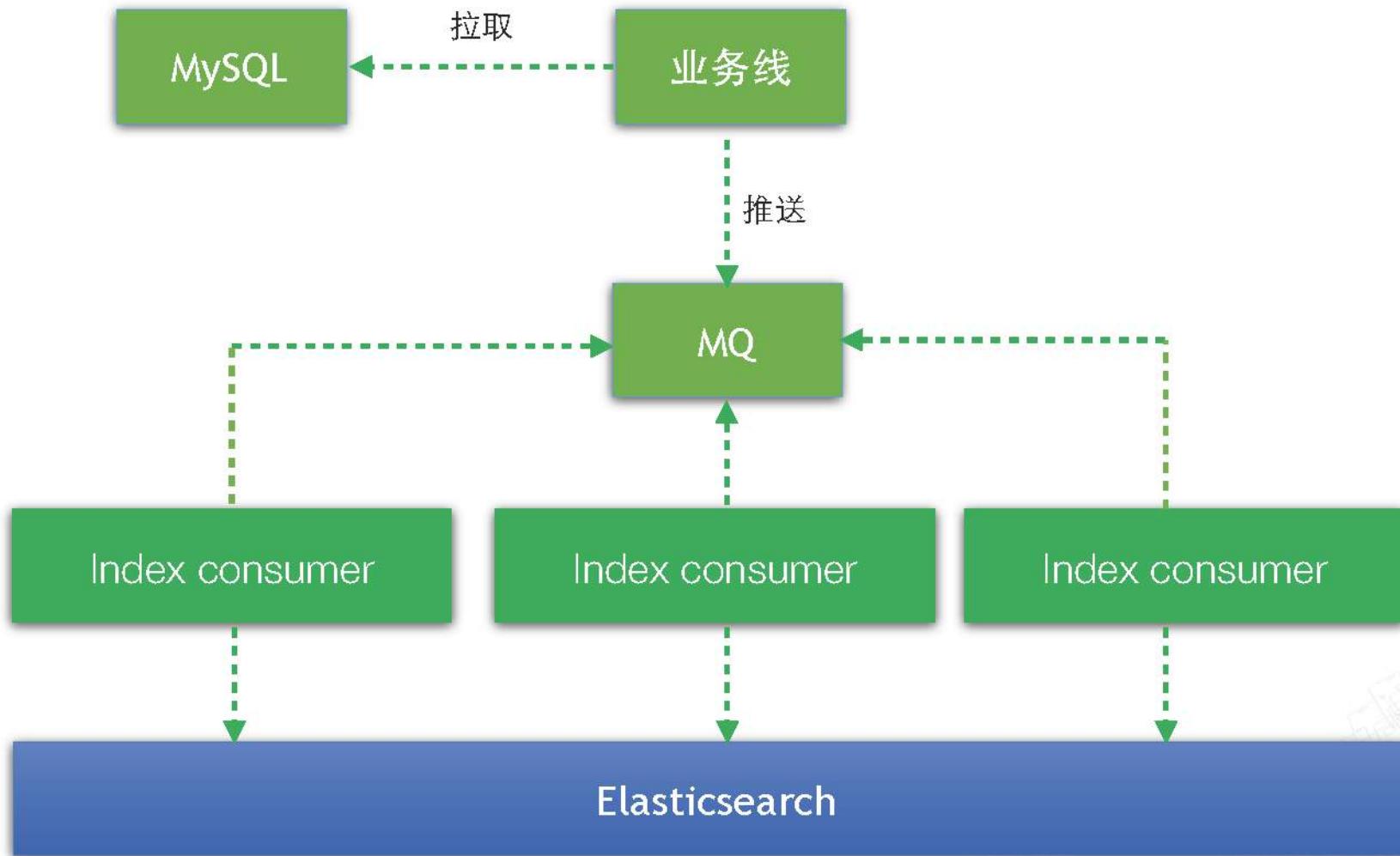
数据层

BI

监控

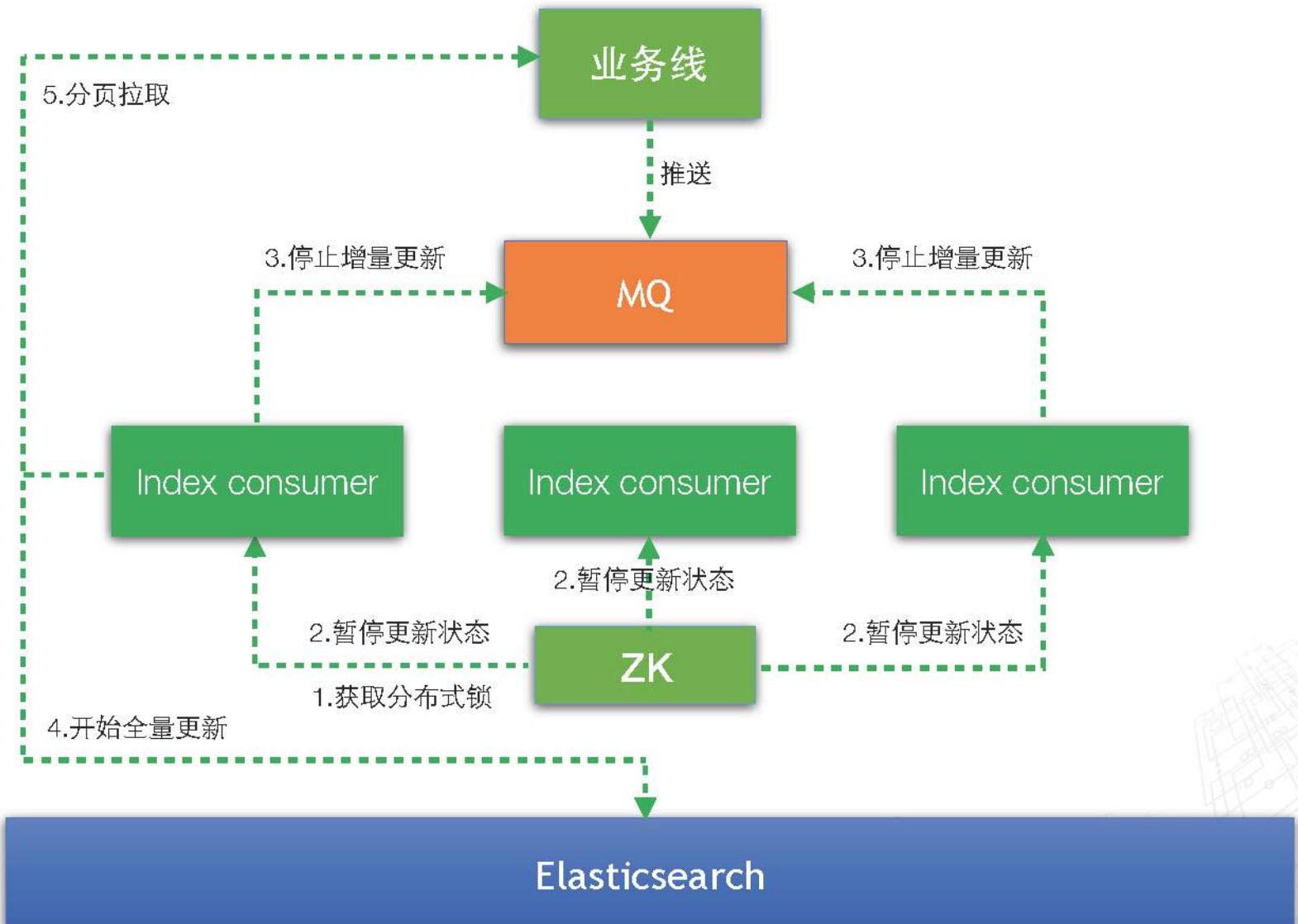
增量更新





全量更新



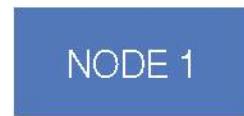


集群扩容

扩容前



扩容中



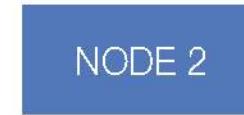
Add Data Node



Restart Master Node



扩容后



监控保障



ZABBIX



[Dashboard](#) [Overview](#) [Web](#) [Latest data](#) [Triggers](#) [Events](#) [Graphs](#) [Screens](#) [Maps](#) [Discovery](#) [IT services](#) [Custom http monitoring](#) [Custom http log](#) [Zabbix](#)

zabbix_dev

导出当前页面数据

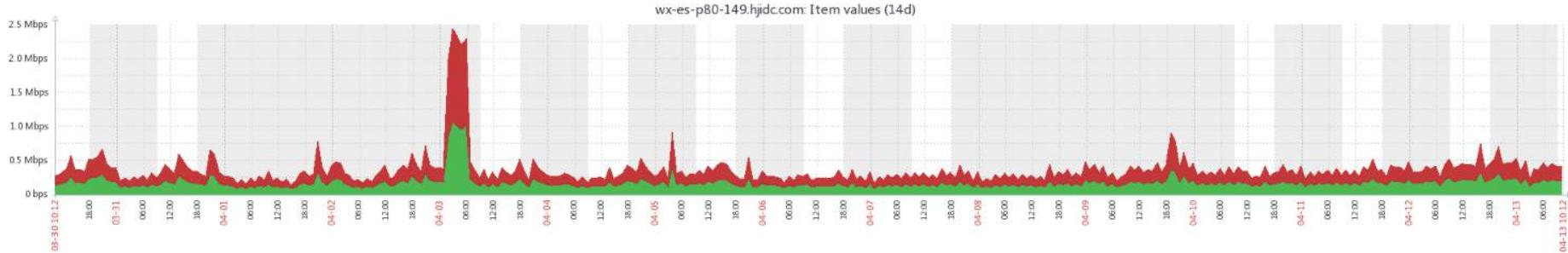
wx-es-p80-149.hjidc.com: 2 items

Filter ▾

Graph type [Normal](#) [Stacked](#)

2017-03-30 10:12 - 2017-04-13 10:12 (now)

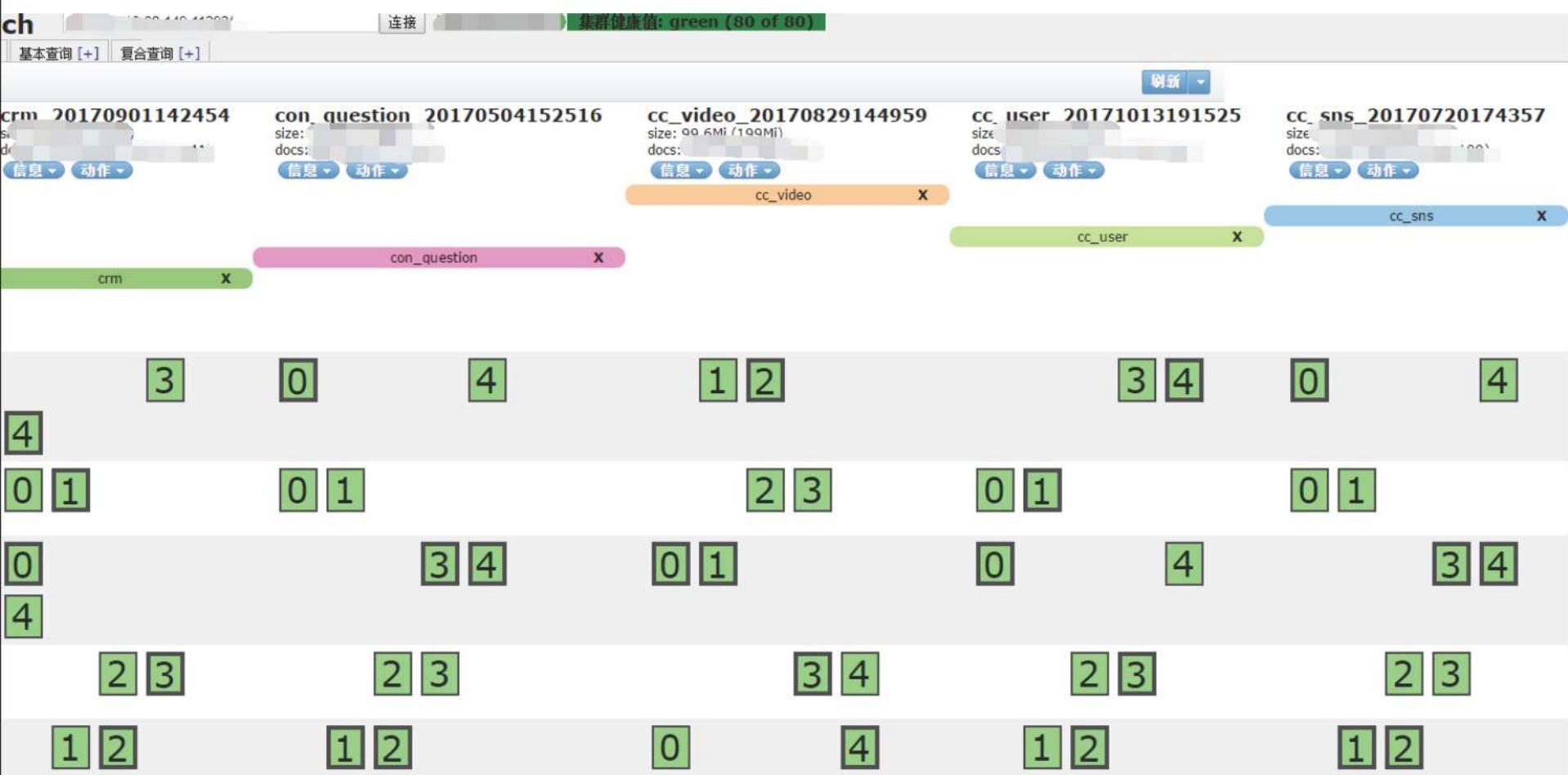
Zoom: 5m 15m 30m 1h 2h 3h 6h 12h 1d 3d 7d 14d 1m All

[«](#) [1m](#) [7d](#) [1d](#) [12h](#) [1h](#) [5m](#) [th](#) [12h](#) [1d](#) [7d](#) [1m](#) [»](#)[«](#) [1d](#) [7d](#) [1m](#) [14d](#) [fixed](#) [»](#)

Data from threads: Generated in 0.21 ms

Head

<https://github.com/mobz/elasticsearch-head>



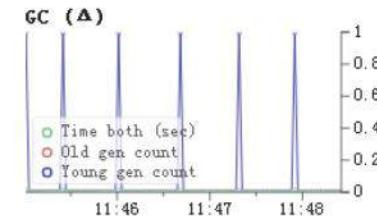
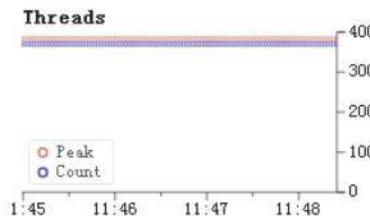
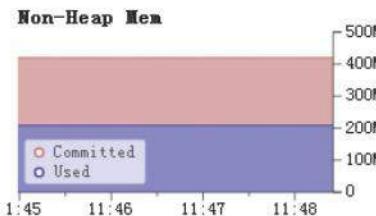
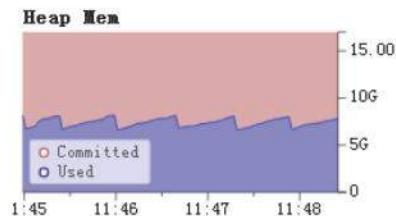
Bigdesk

<https://github.com/lukas-vlcek/bigdesk>

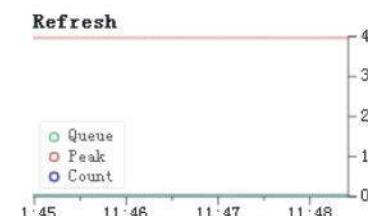
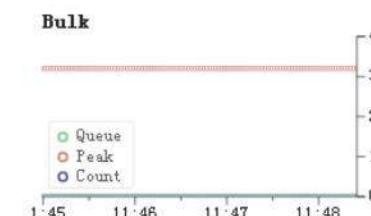
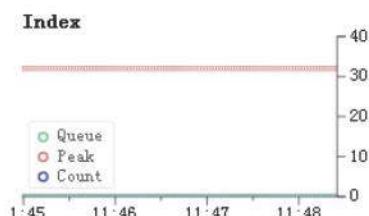
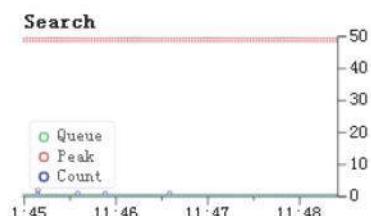
JVI

VM name: Java HotSpot(TM) 64-Bit Server VM
VM vendor: Oracle Corporation
VM version: 25.66-b17

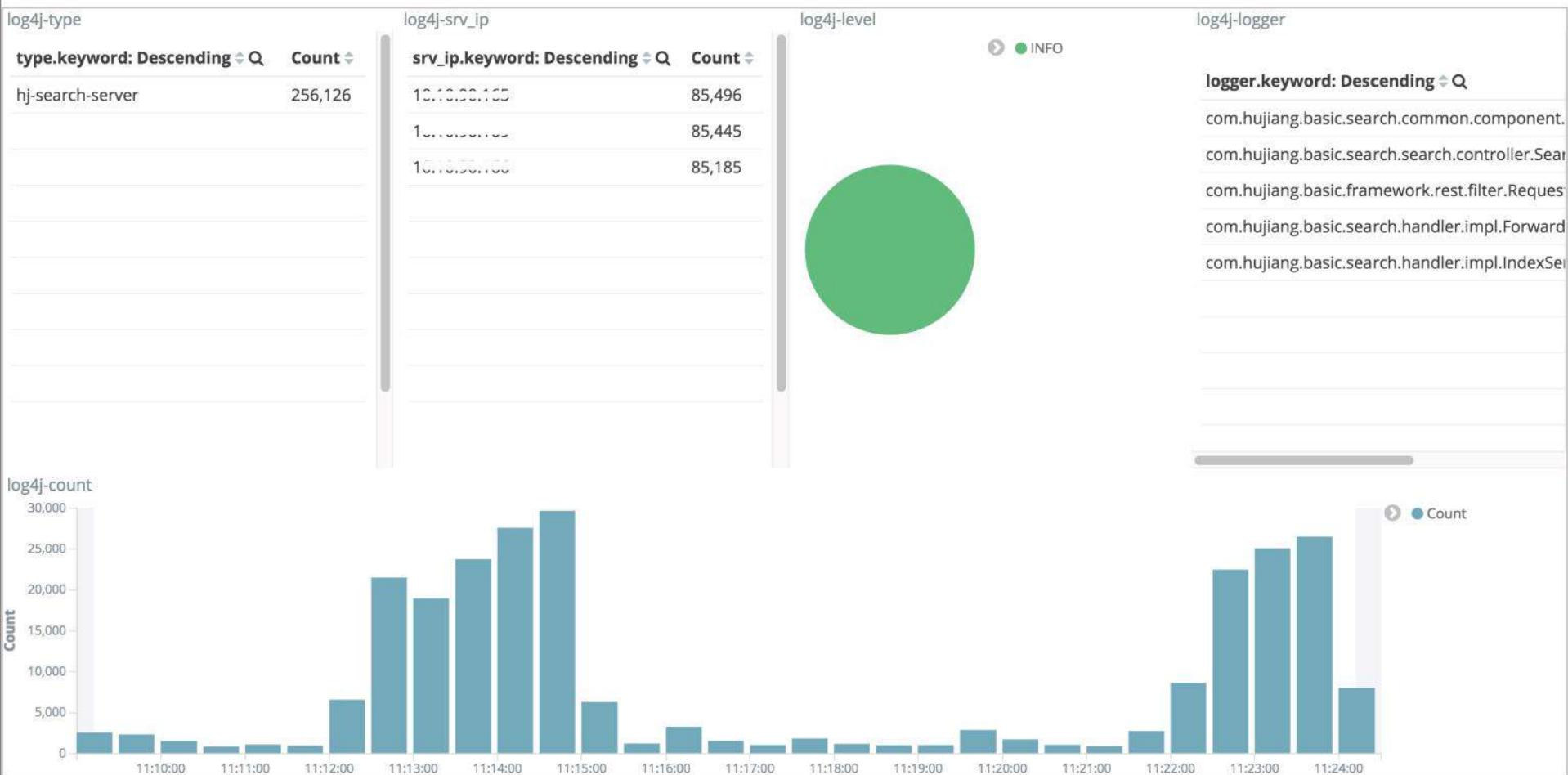
Uptime: 36.9d
Java version: 1.8.0_66
PID: 59417

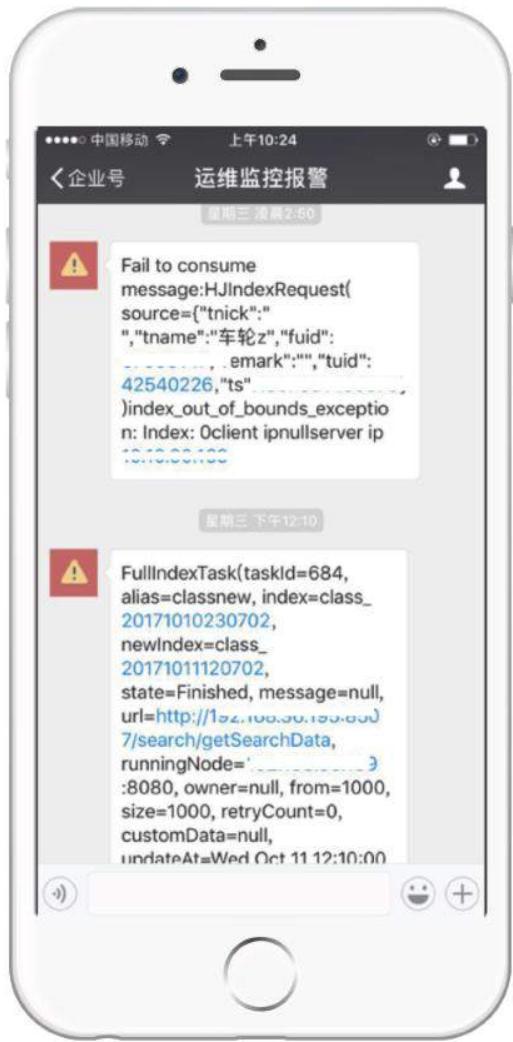


Thread Pools









增量更新错误



全量更新通知

心得体会

- 批量写索引
- 分片数不是越多越好
- 避免使用wildcard的搜索“*ABC*”
- 尽量少用高亮，会导致CPU使用过高
- Index Node 与 Search Node 隔离
- 根据数据量来调整刷新时间

Q & A

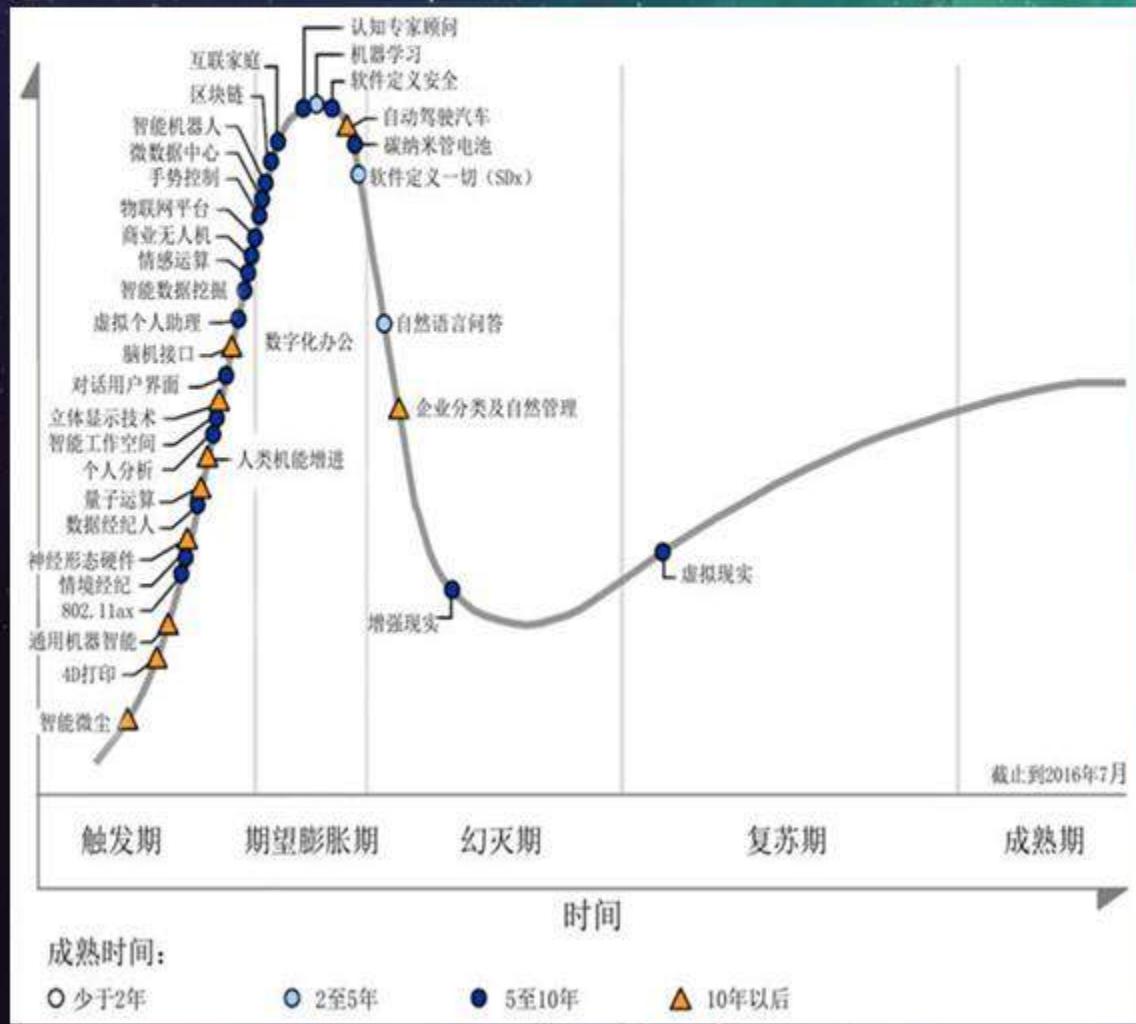


金融科技大数据及 AI技术架构和应用

沈百军
平安银行零售大数据技术总监

新技术趋势：人工智能全面繁荣的时代来临，区块链、物联网将形成平台生态，AR/VR将不断优化“身临其境”的体验。

Gartner 2017年度新兴技术成熟度曲线



解读

- 人工智能正处于成熟度的全盛时期，与各种场景相结合来解决前所未见的问题，将成为未来10年最具影响力的技术

- 区块链、物联网正在接近成熟度的最高峰，将改变我们定义和使用平台的方式，以平台生态转变现有商业模式

- AR/VR在逐步复苏之中，将更加贴合工作与生活的场景，提升人、组织、事物之间的交织度与互动透明度

基于技术趋势、成熟度以及对平安银行战略价值的综合考量，
我们聚焦如下3大技术领域



AI Schema



AI 方向



AI 技术架构、逻辑架构、建模实现流程



AI 相关应用

智能客服
OCR、人脸识别
声纹识别

A.人工智能的发展将分为以下三个阶段：

弱人工智能

专注于特定的领域，只在单一领域比人类强，如人脸识别、自动驾驶等



强人工智能

综合能力达到人类水准，就是人类能做的事情基本都能做到



超人工智能

全方位大大超过人类的智能



A. AI能力集成于产品体

· 验

- 潜客识别
- 流失预测
- 汽融进件预测



A.平安银行AI引擎架构

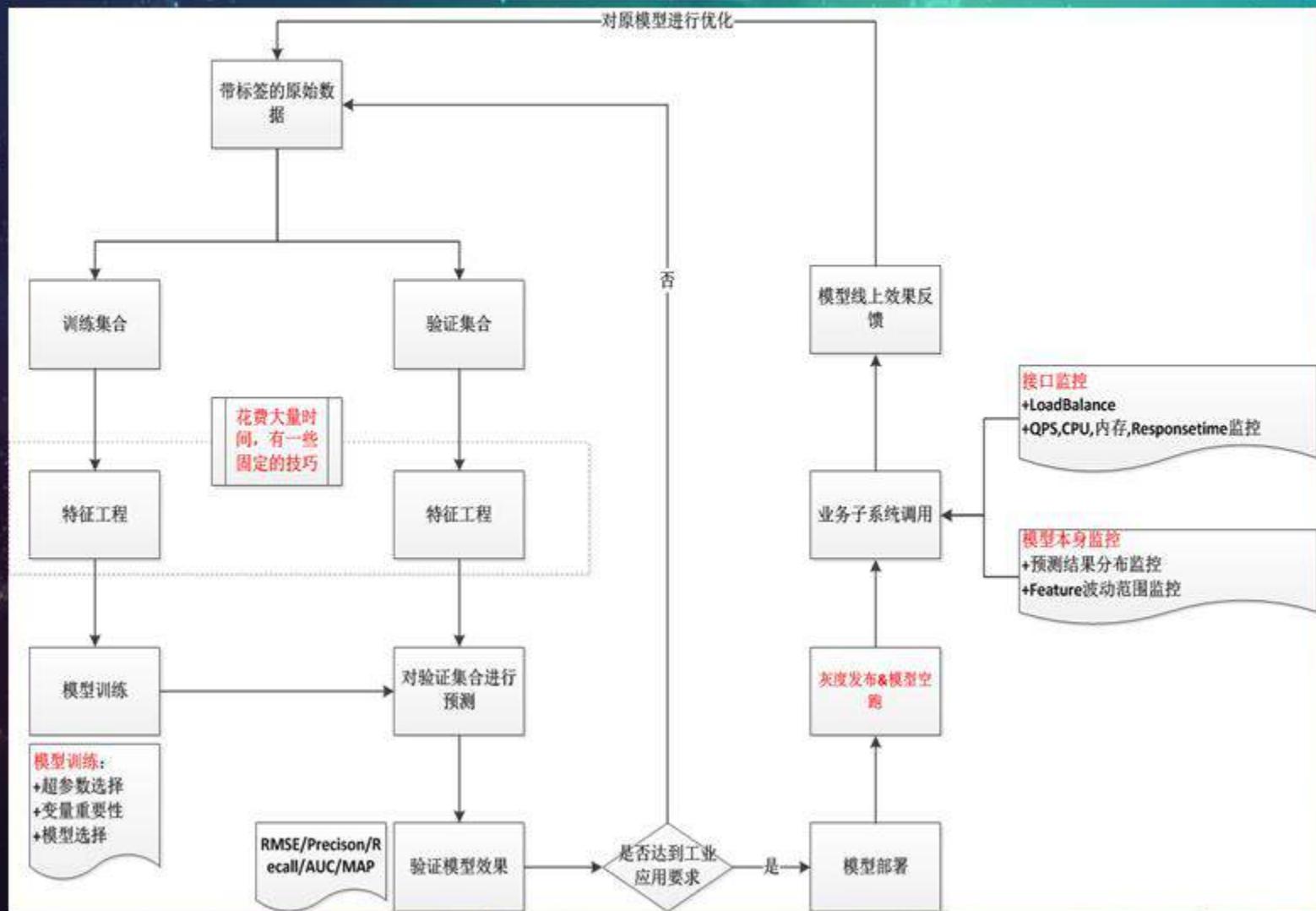
构



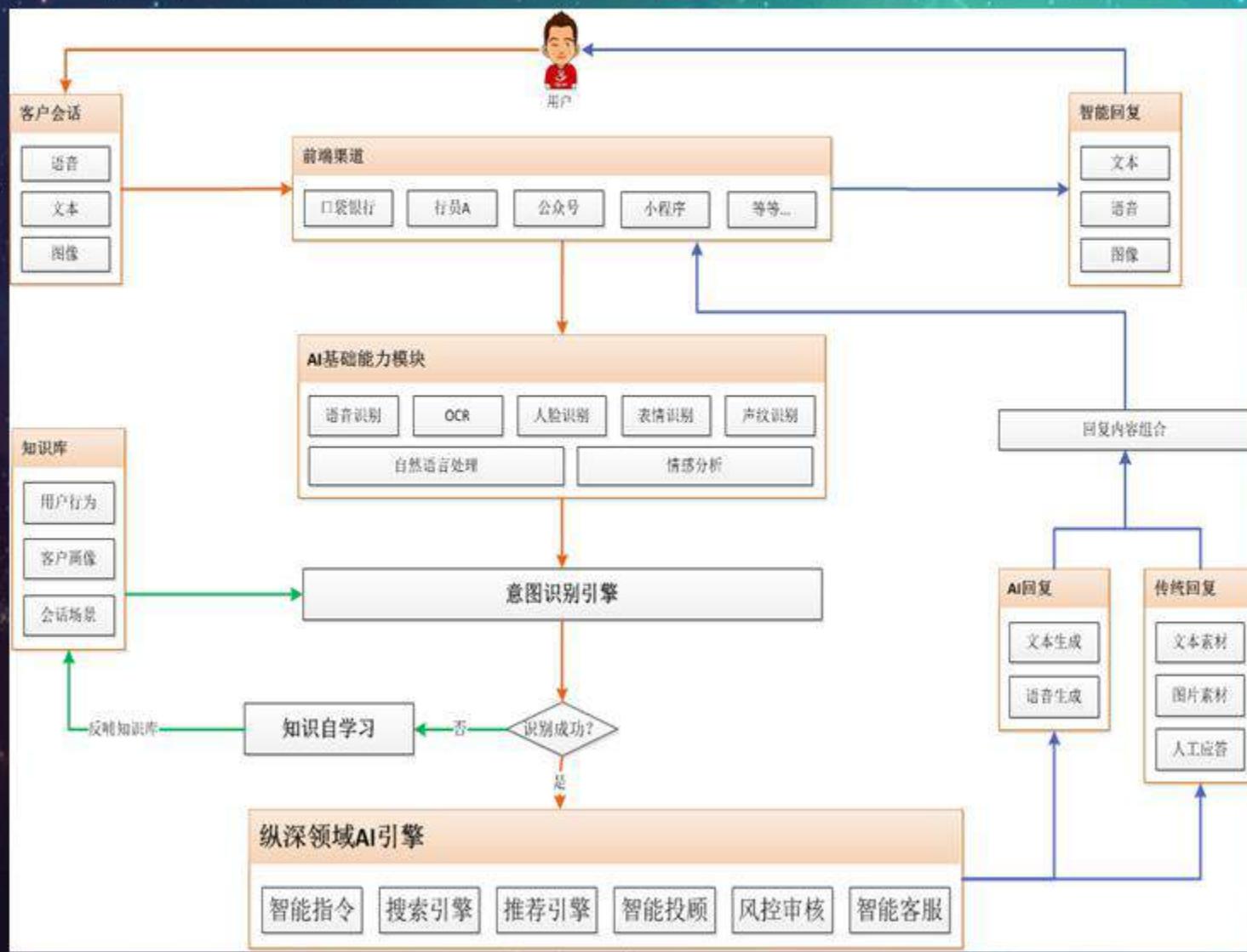
A.I应用的总体架构-平台化+服务化 (API+SDK)



A.机器学习建模流程



A. 平安银行智能客服-（语音识别+NLP）应用



B.数字化：大数据与AI的区别在于，大数据是深度学习的基础



B. 大数据平台目标



数据获取

建立数据中心，如何多元化的获取数据，内部、外部、爬虫、采购等在10PB的数据中，快速获取有价值的数据，是一种能力。



数据打通

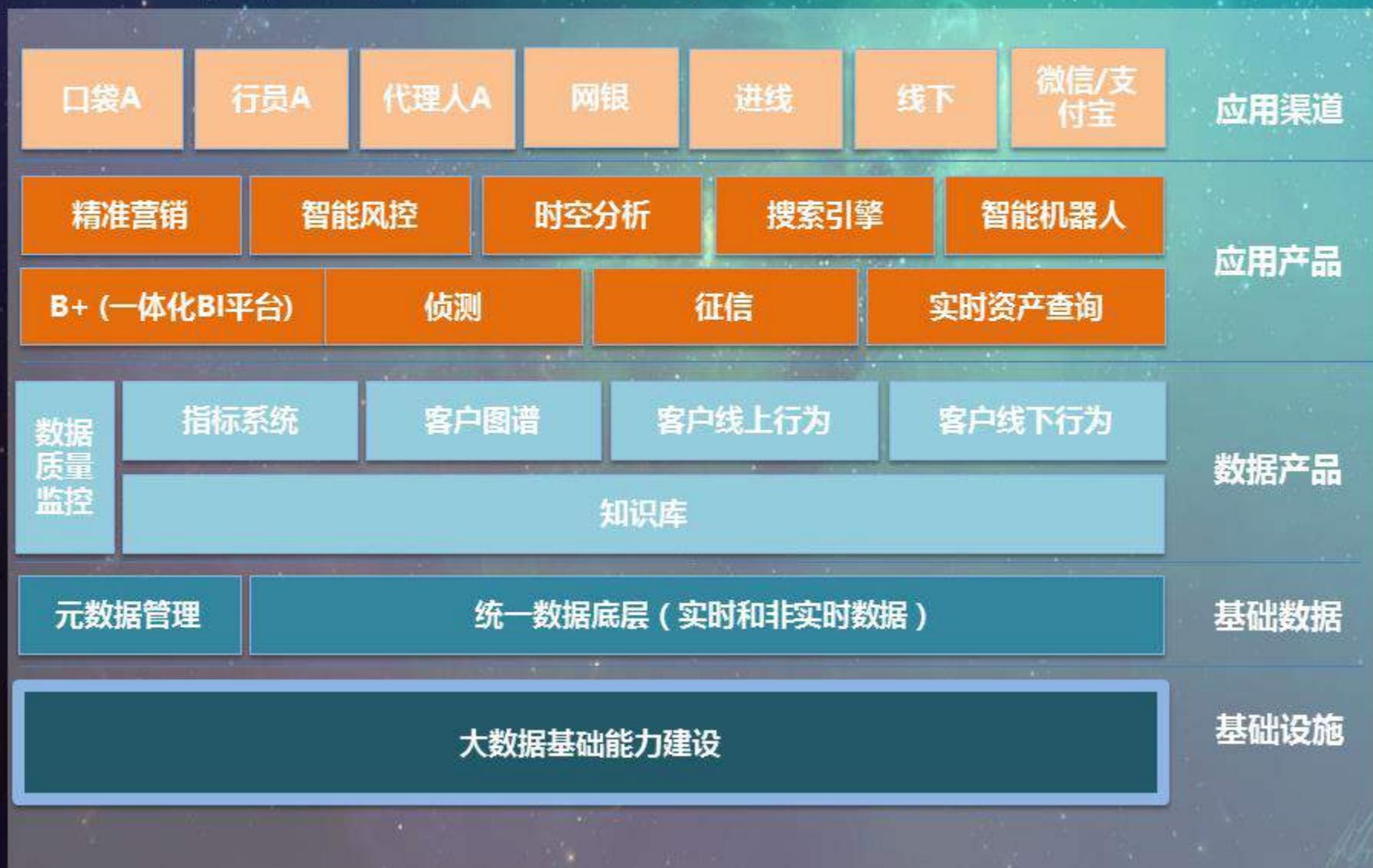
数据融合，需要把数据整合起来，相互利用，发挥其价值。



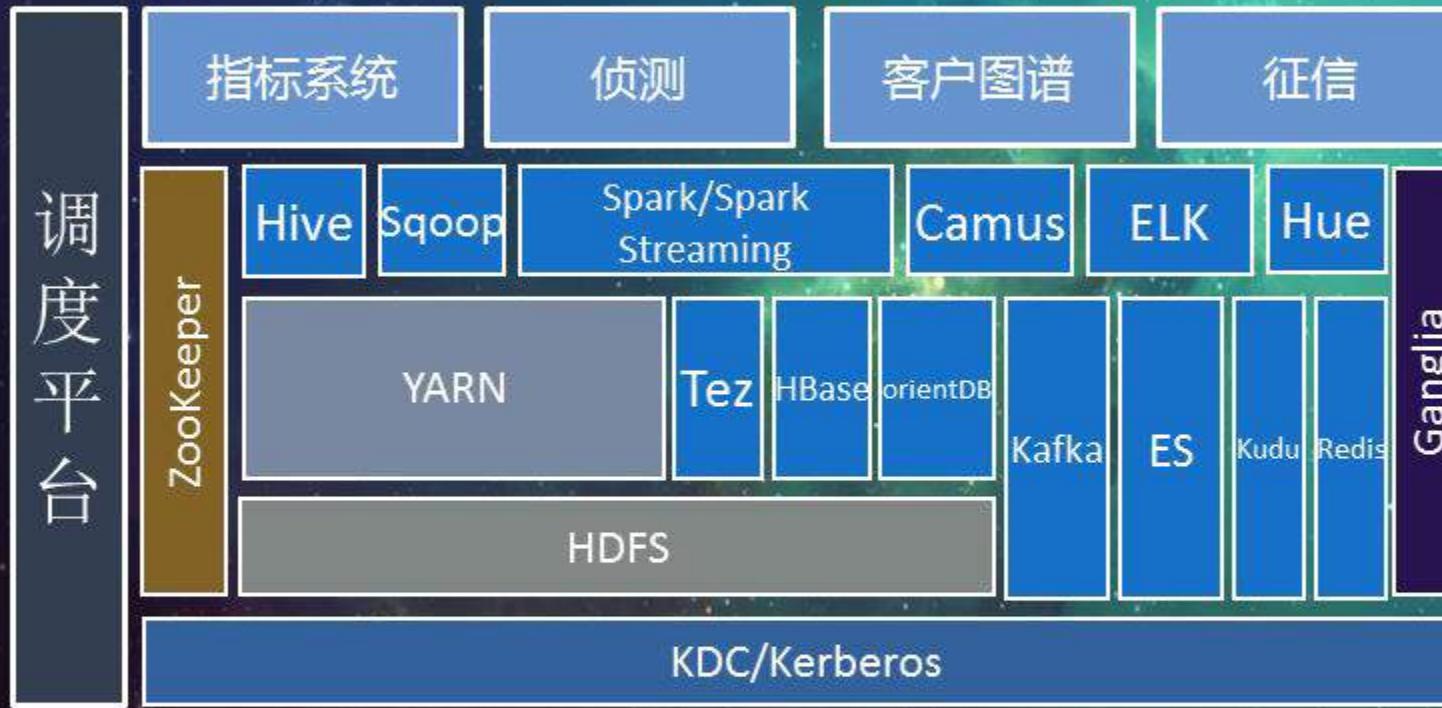
数据应用

数据应用是最终目标，寻找业务、需找场景、和业务共同实现。

B. 大数据总体架构



B. 大数据技术栈



- 半年内从华为Hadoop转向开源Hadoop集群，废弃GP；从底层进行平台和数据重构
- 快速引入全栈大数据技术，离线、流式计算、图谱、NoSql、多维分析
- 日新增数据100T，包括金融数据、非金融数据、用户行为、网站爬虫数据
- 日计算Job 7万+，300+用户，提供一个全新的大数据平台
- 建立1000+个标签，为App千人千面提供数据基础，毫秒内检索用户画像和行为数据
- 大规模使用Elastic Search，写入量20W/S，为健测提供平台支持

B. 后期方向

- 集群更稳定
- 集群更安全
- 集群计算更快



B. 指标系统



标签市场

营销
push平
台

潜力投资送流量

- 是否理财偏好客户 = 'Y'
- 是否口袋客户 = 'Y'
- 开户日期 > 2016/10/01
- 开户日期 < 2016/12/31
- 客户本日资产余额 < 10000
- 最近30天登陆口袋银行次数 <= 5

个性化外部标签



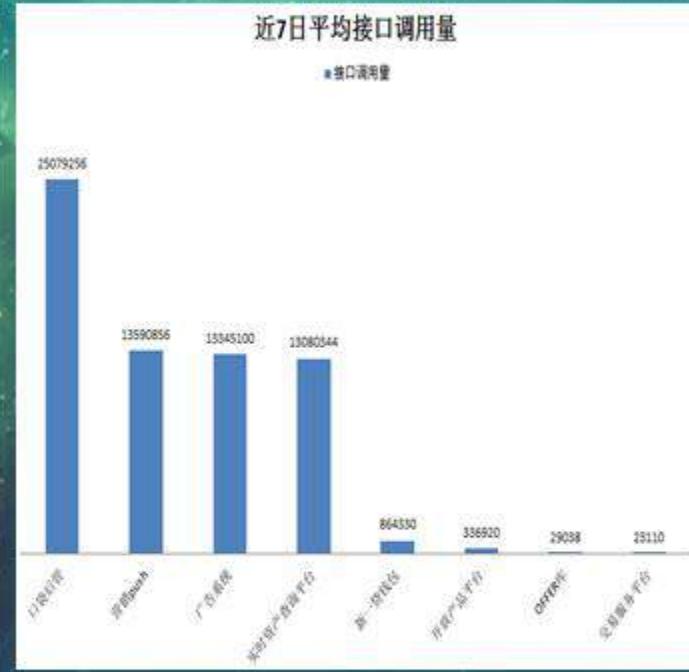
B.客户标签系统

用户基本信息

客户姓名：	黎*
年龄：	40
性别：	男
客户类型：	个贷客户
手机号：	139****9964
开户时间：	20000721
所属分行：	深圳分行
风险评级：	进取型
是否平安员工：	否
集团资产层级：	M5



用户行为信息



算法类标签

客户钱包大小：	50万以上
和盈购买倾向：	高
新一贷贷款倾向：	低
客户流失风险：	中风险
产品购买倾向：	现金宝, 和盈系列, 贵金属份额

标签使用情况

客户AUM余额
客户是否已经领取过抽奖券-是否(是:是;否:否)
首次成为千元户时间
客户开户时间
客户是否口袋客户
首次成为万元户时间
客户持卡类型
客户是否已经领取过优酷优惠券(Y:是;N:否)
是否理财偏好客户
客户是否已经领取过抽奖券-是否(是:是;否:否)
首次成为万元户时间
私期t1, t2标志
客户资产本日余额
5月度客户总资产小于1万
业务员工号
开户日期
客户是否已经领取过抽奖券-是否(是:是;否:否)
资产总额

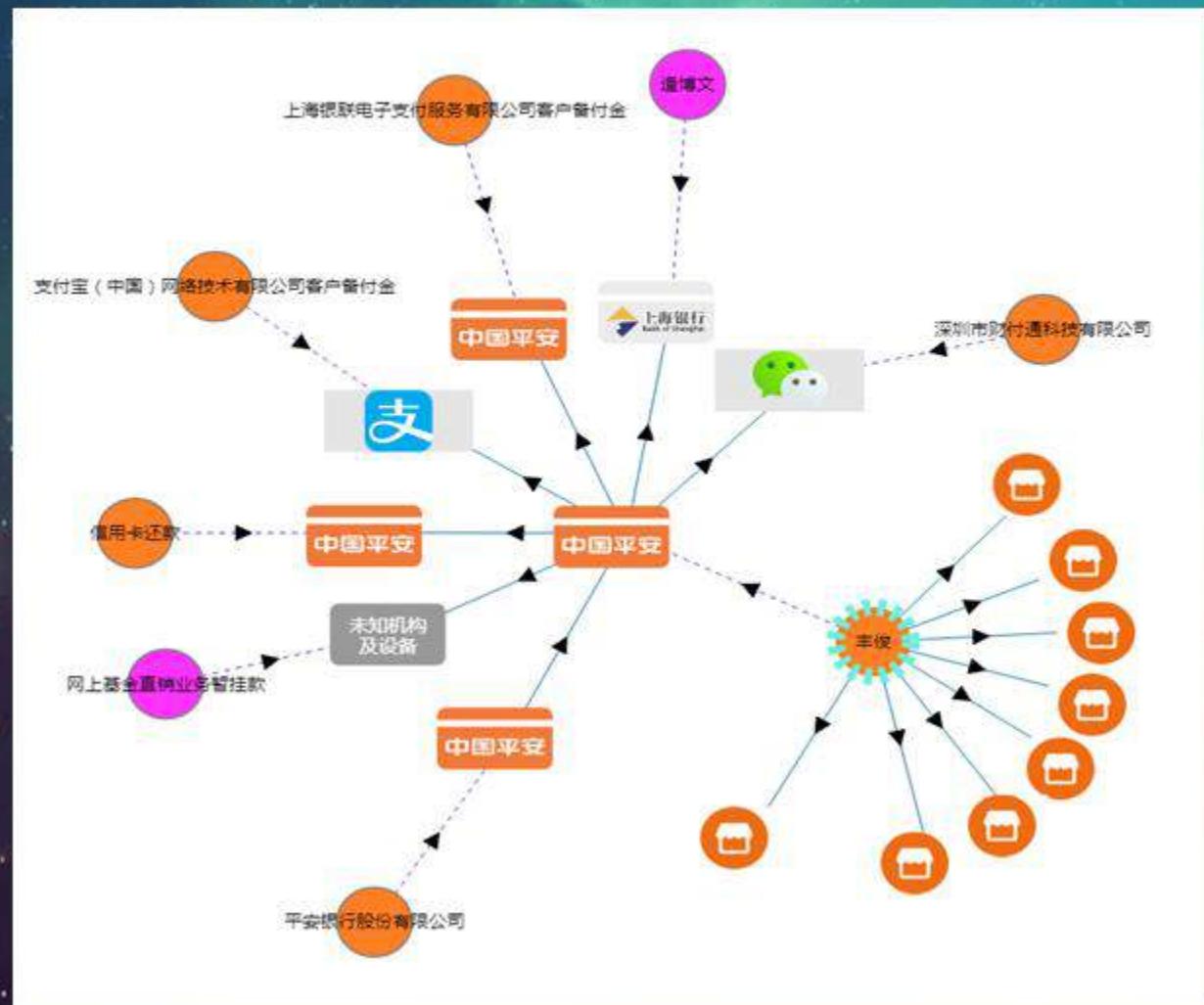
B.客户关系图谱

已支持关系类型：

- ✓ 借记卡类交易关系
- ✓ 信用卡类交易关系
- ✓ 管户人关系

开发中：

- ✓ 同事关系
- ✓ 空间相似关系



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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

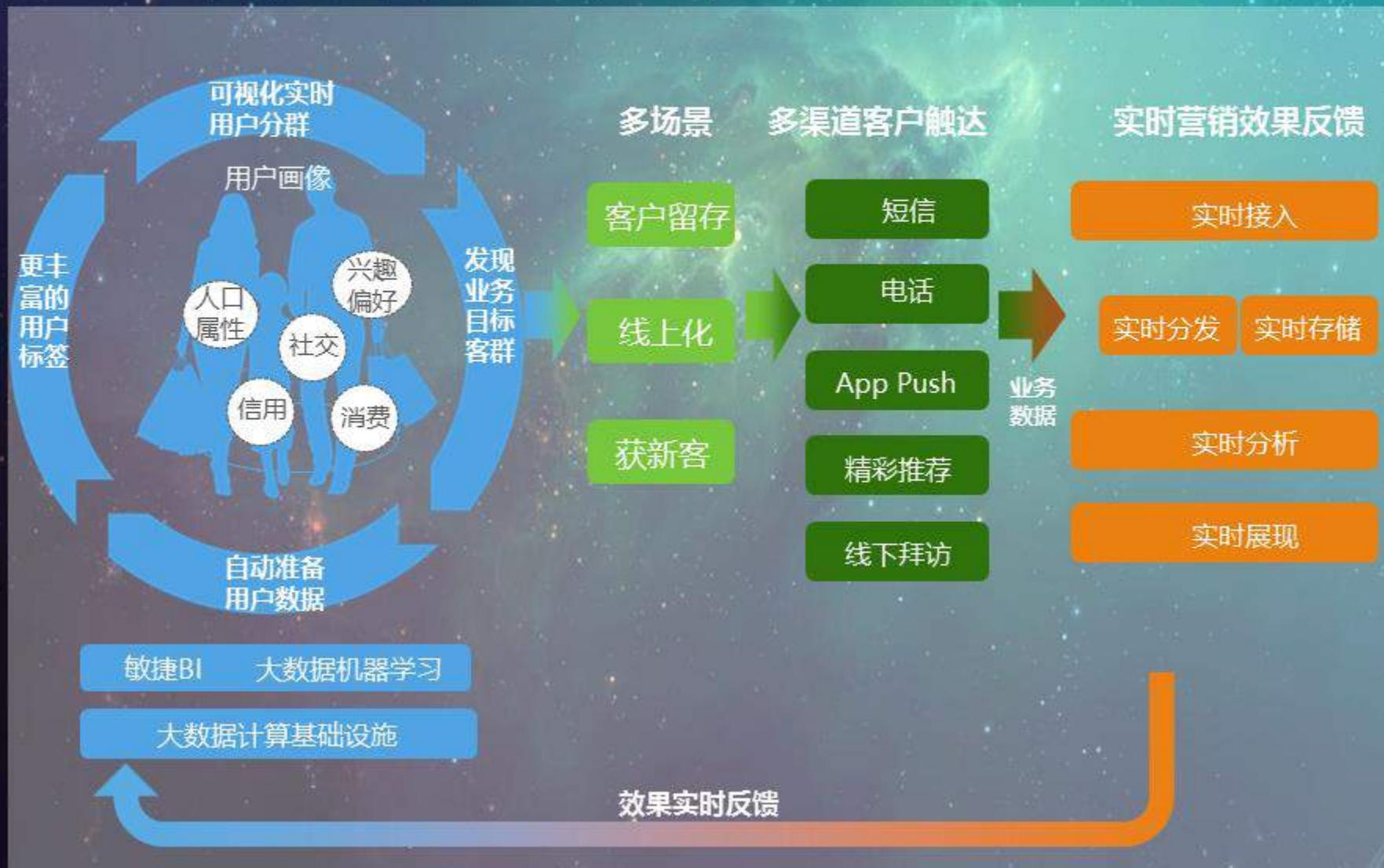
10 敏捷BI

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。

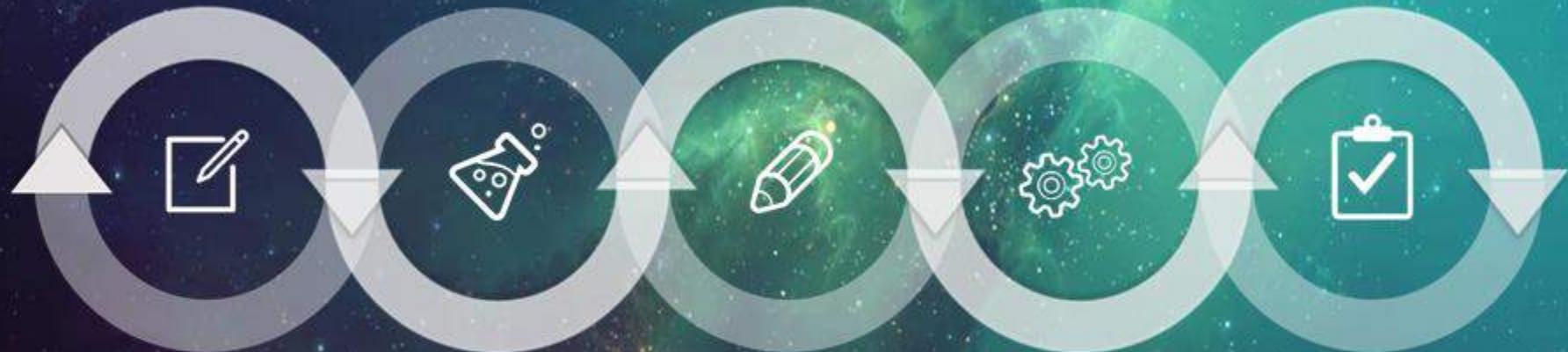
12 智能客服

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
识别假冒。



事后

聚类排查、链式分析
提前挖掘关联欺诈。



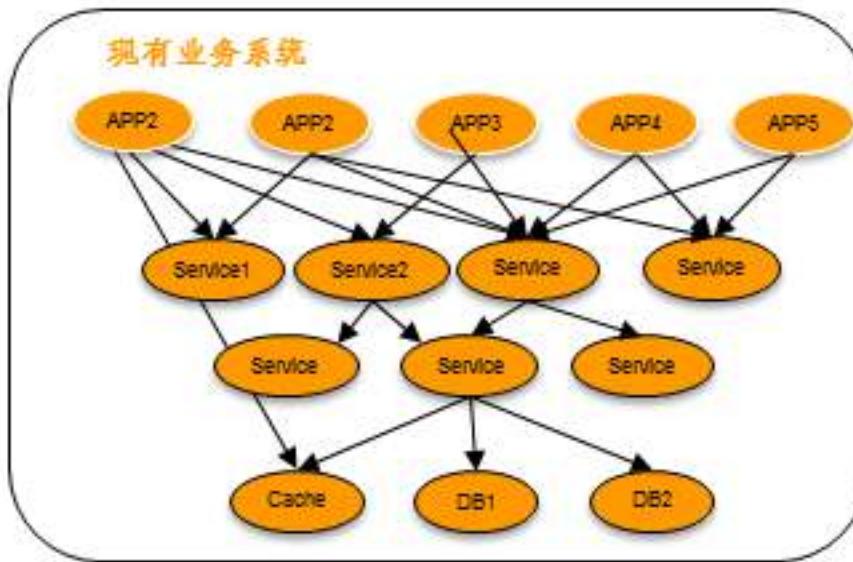
谢谢

沈百军
平安银行零售大数据技术总监

容量规划和流量管控

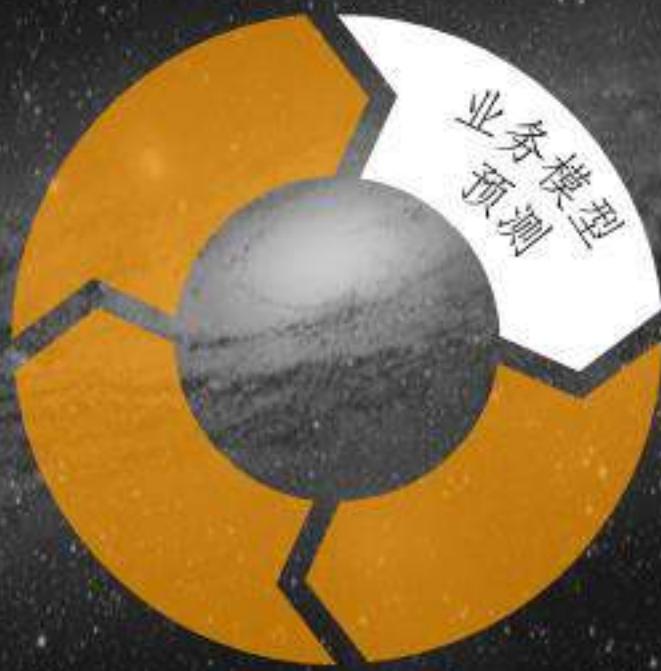
阿里巴巴高可用架构团队
张军,林佳梁

容量规划



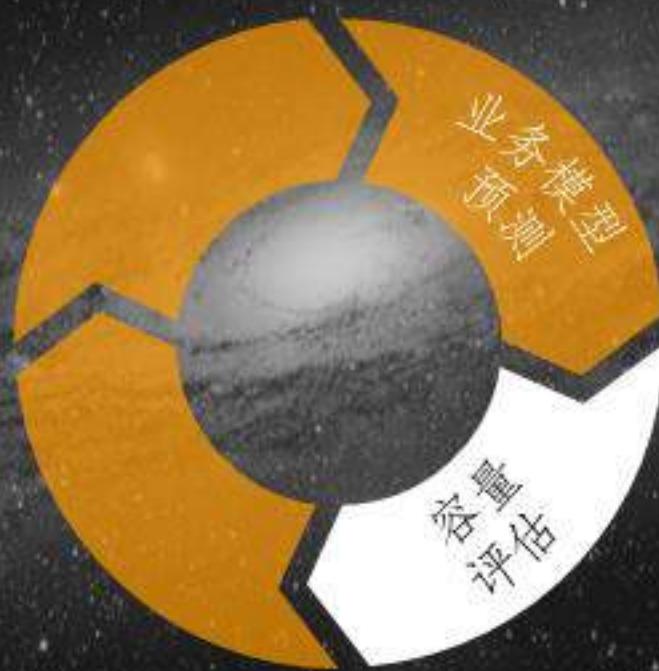
Capacity planning is the process of determining the capacity needed by a complex distributed system to meet the workload with guarantee on certain level of performance

容量规划



- 流量模型
- 历史数据
- 预测算法

容量规划



- 流量模型
- 历史数据
- 预测算法
- 单机容量
- 应用模型

单机容量评估的四种方式



模拟



复制



重定向



Load
Balance

模拟



生产环境



测试环境

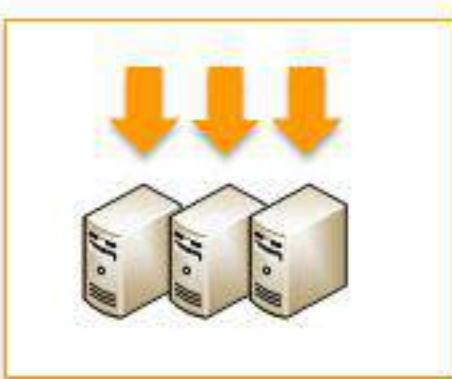
优点

- 容易实现
- 适合新应用

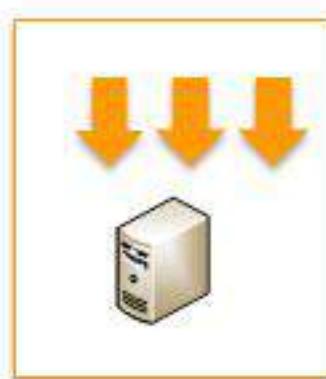
缺点

- 请求不够逼真
- 额外的脏数据

复制



生产环境



测试环境

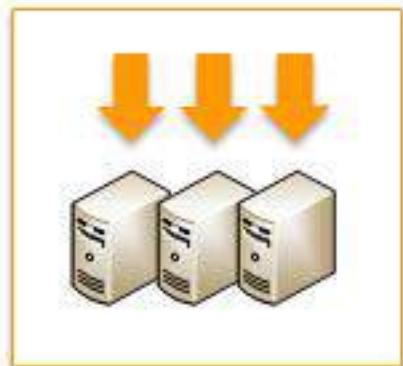
优点

- 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.

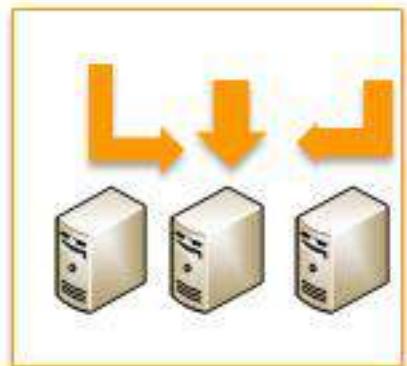
缺点

- 需要额外的机器
- 产生脏数据

重定向



生产环境



生产环境

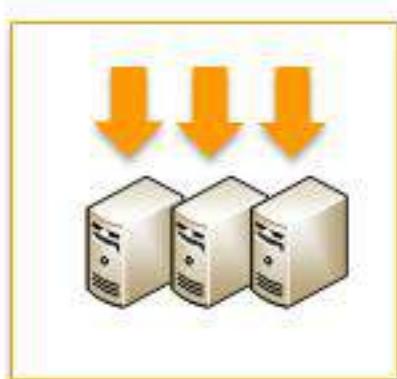
优点

- 所有的数据都是来源真实
- 无脏数据

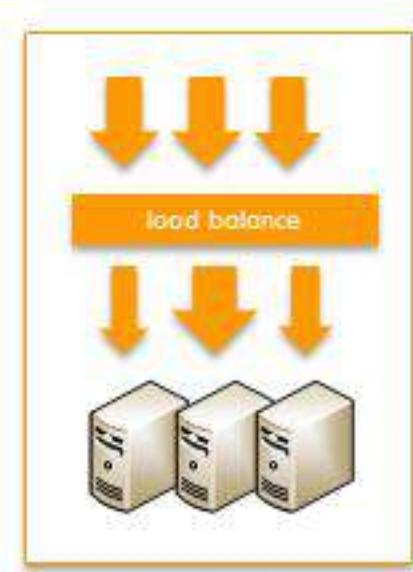
缺点

- 无法实施于新应用或者流量较少的应用

Load Balance



生产环境



生产环境

优点

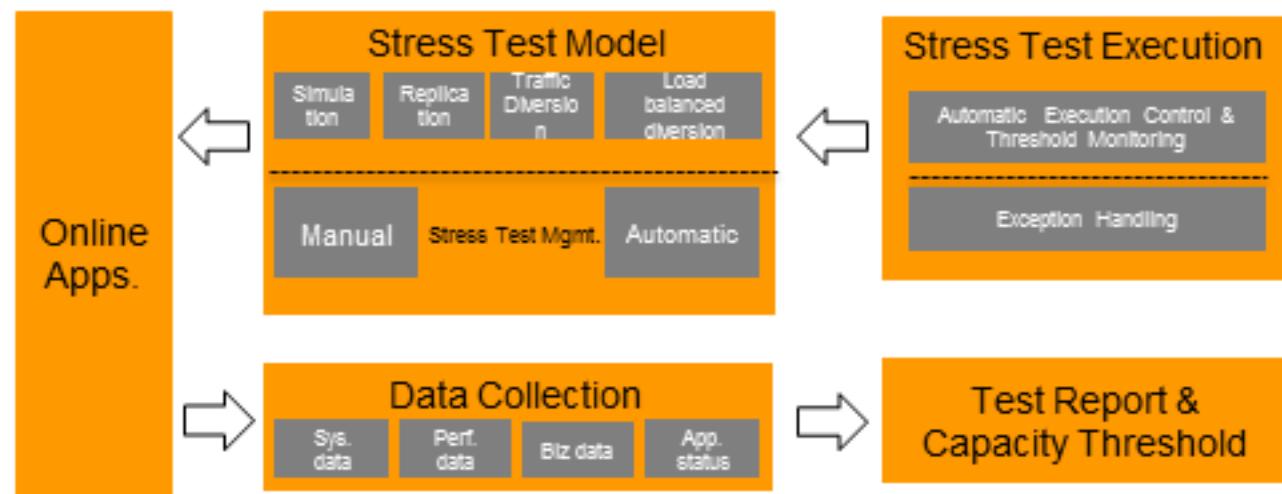
- 所有的数据都是来源真实
- 无脏数据

缺点

- 无法实施于新应用或者流量较少的应用

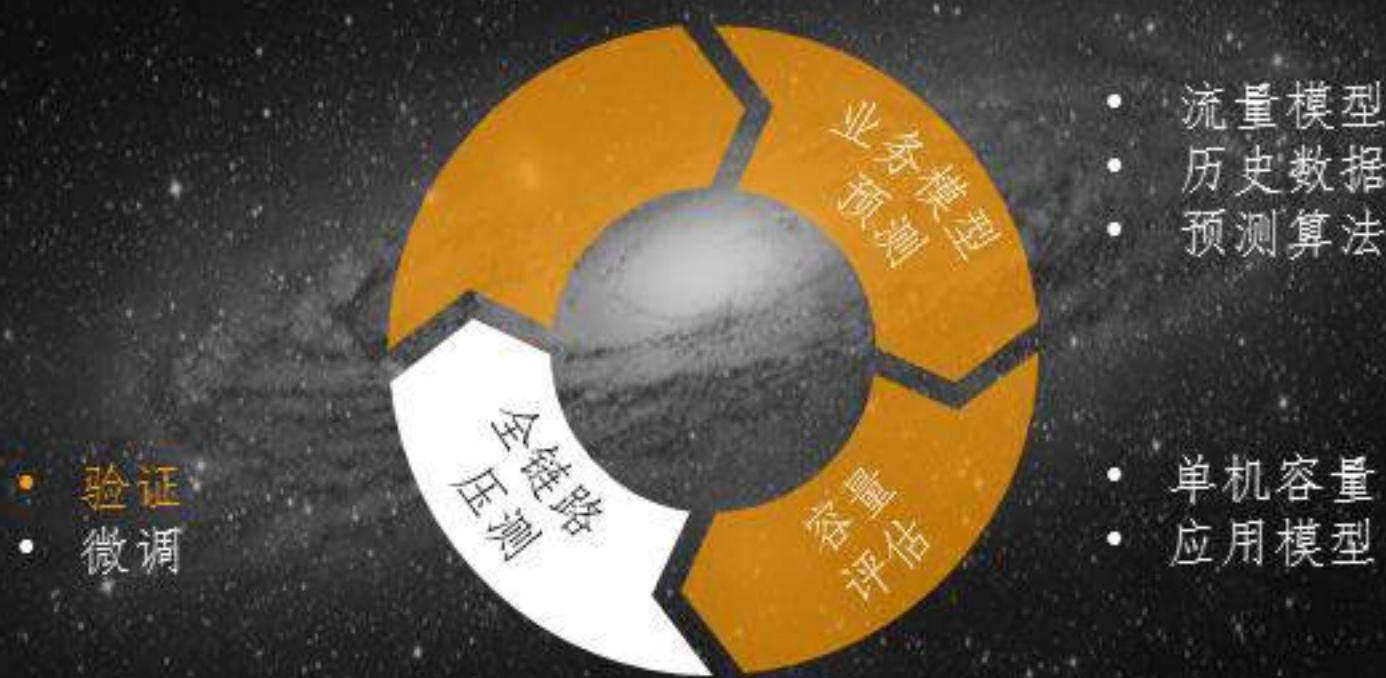
单机压测平台的架构

- 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

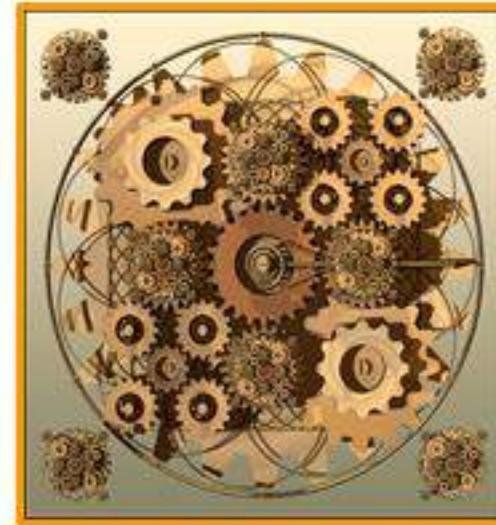


需要的机器数目 =
业务估算 / 单机容量 + 冗余

容量规划

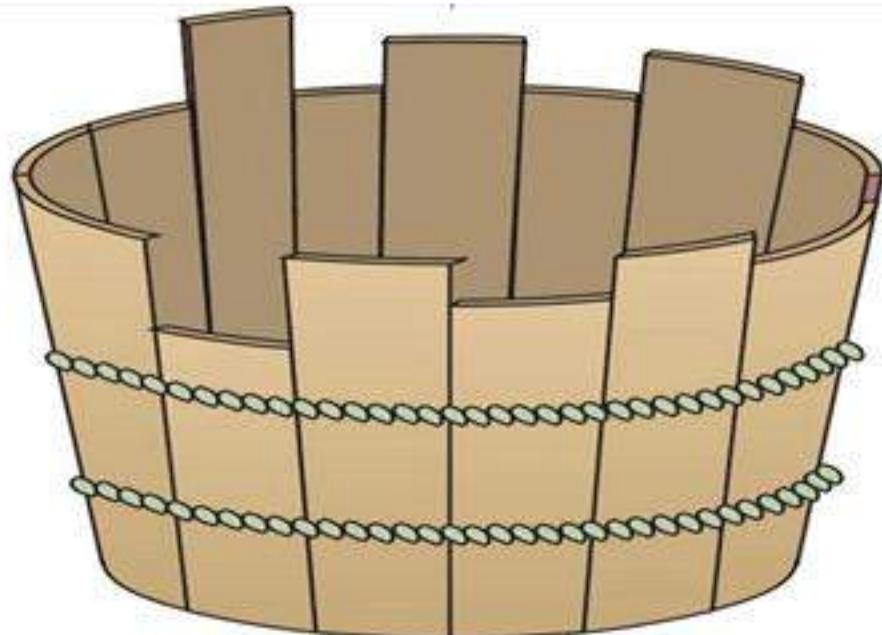


为什么会有全链路压测



“Single point” approach is totally different from “scenarios”

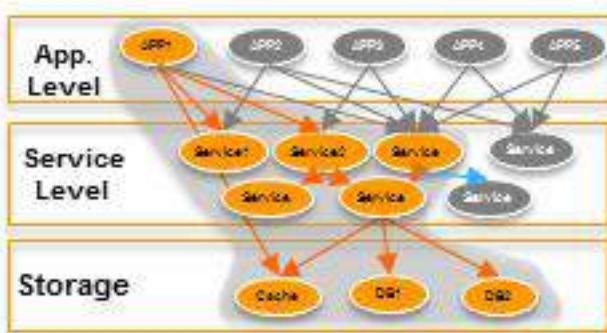
木桶理论



- 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

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

困难点



请求规模大

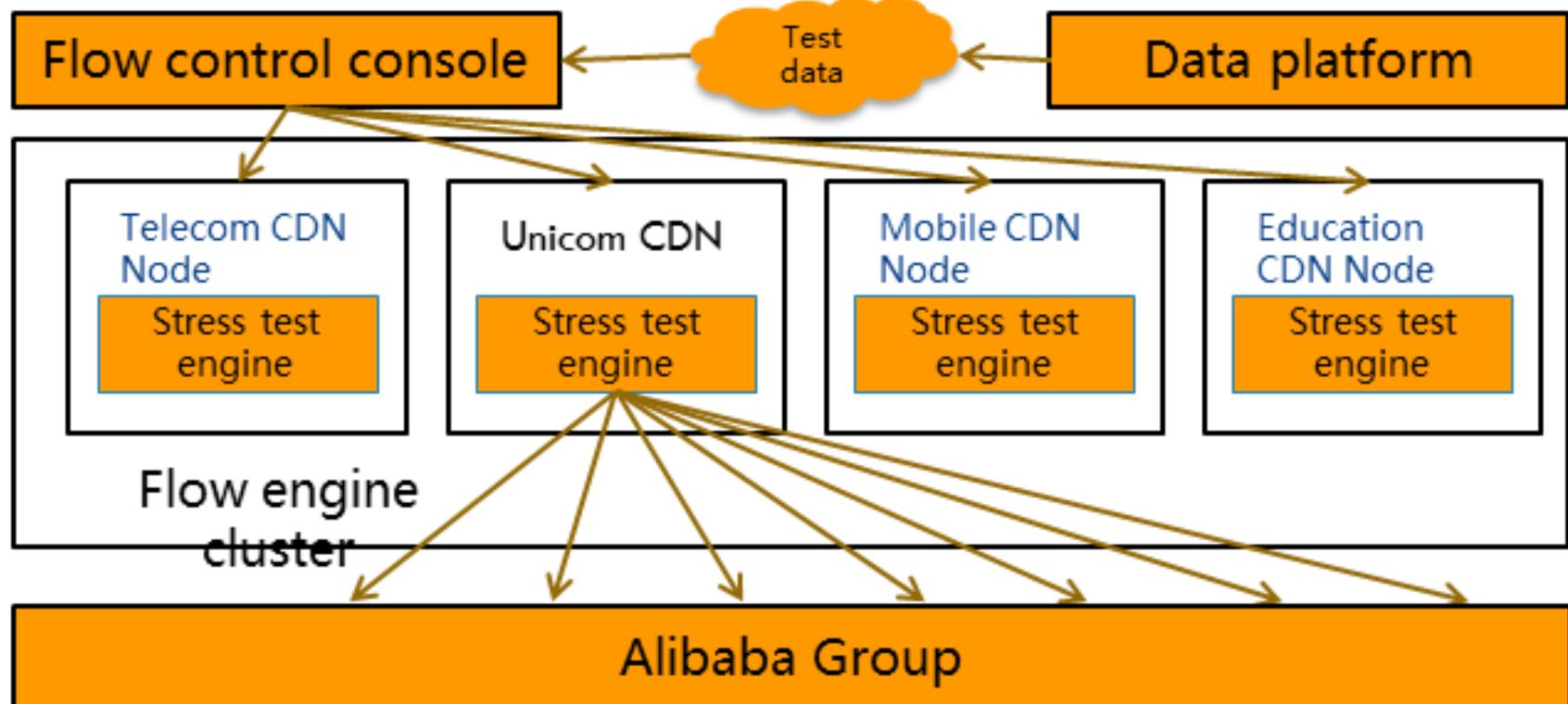
- 10,000,000 requests/sec

用户行为复杂

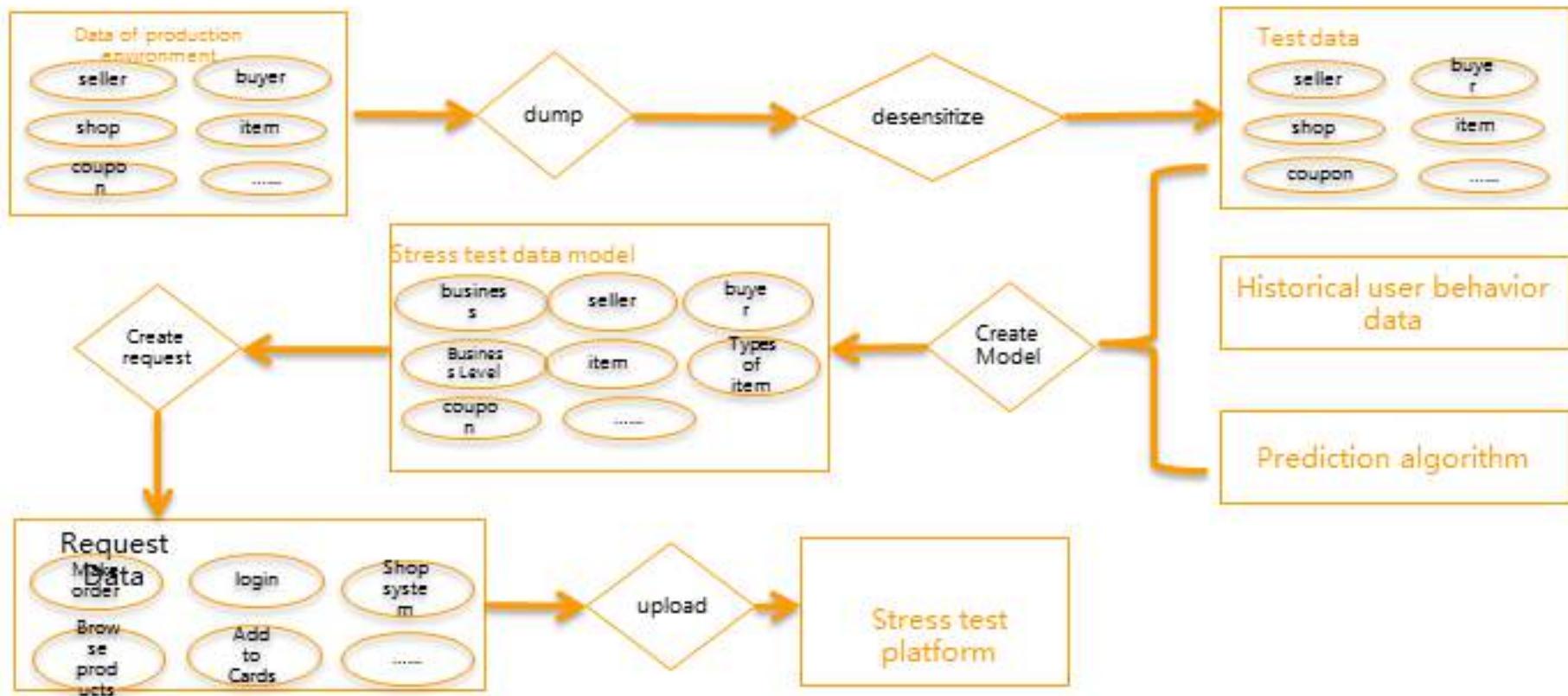
大促场景特有

不能对生产环境有影响

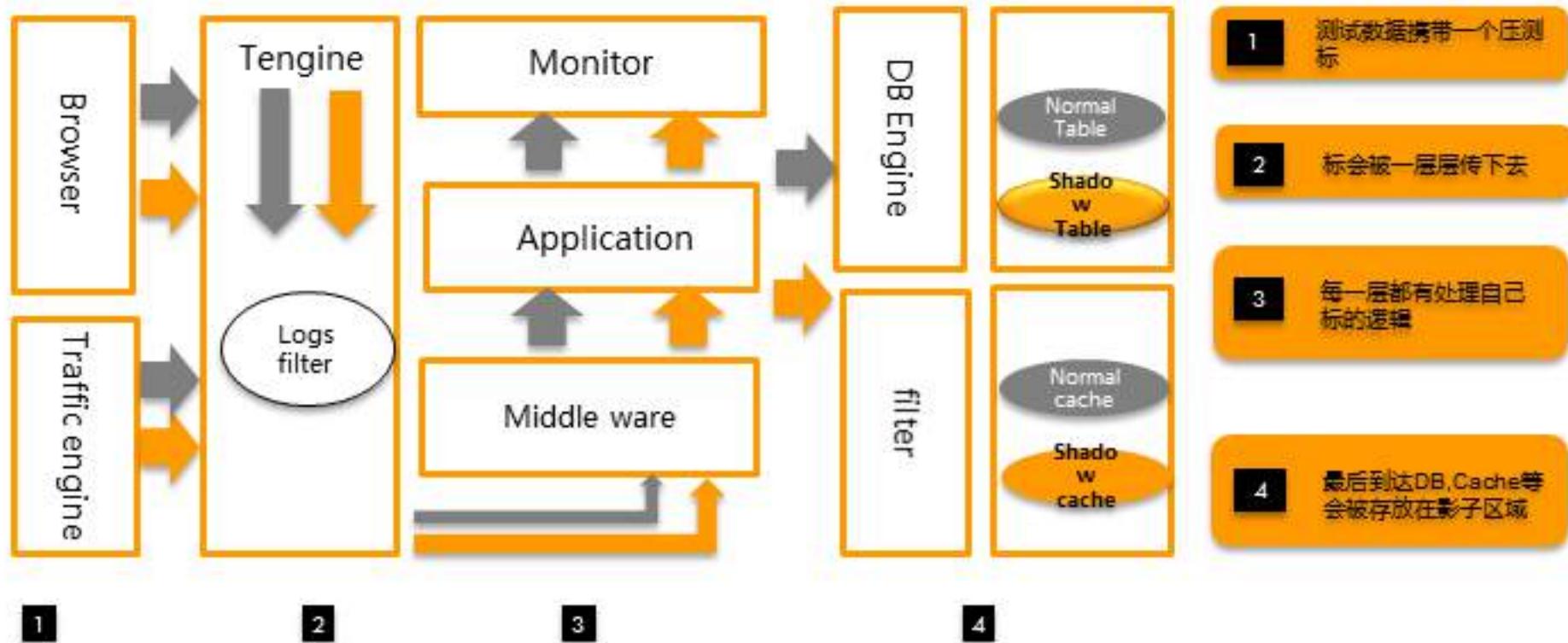
模拟用户请求



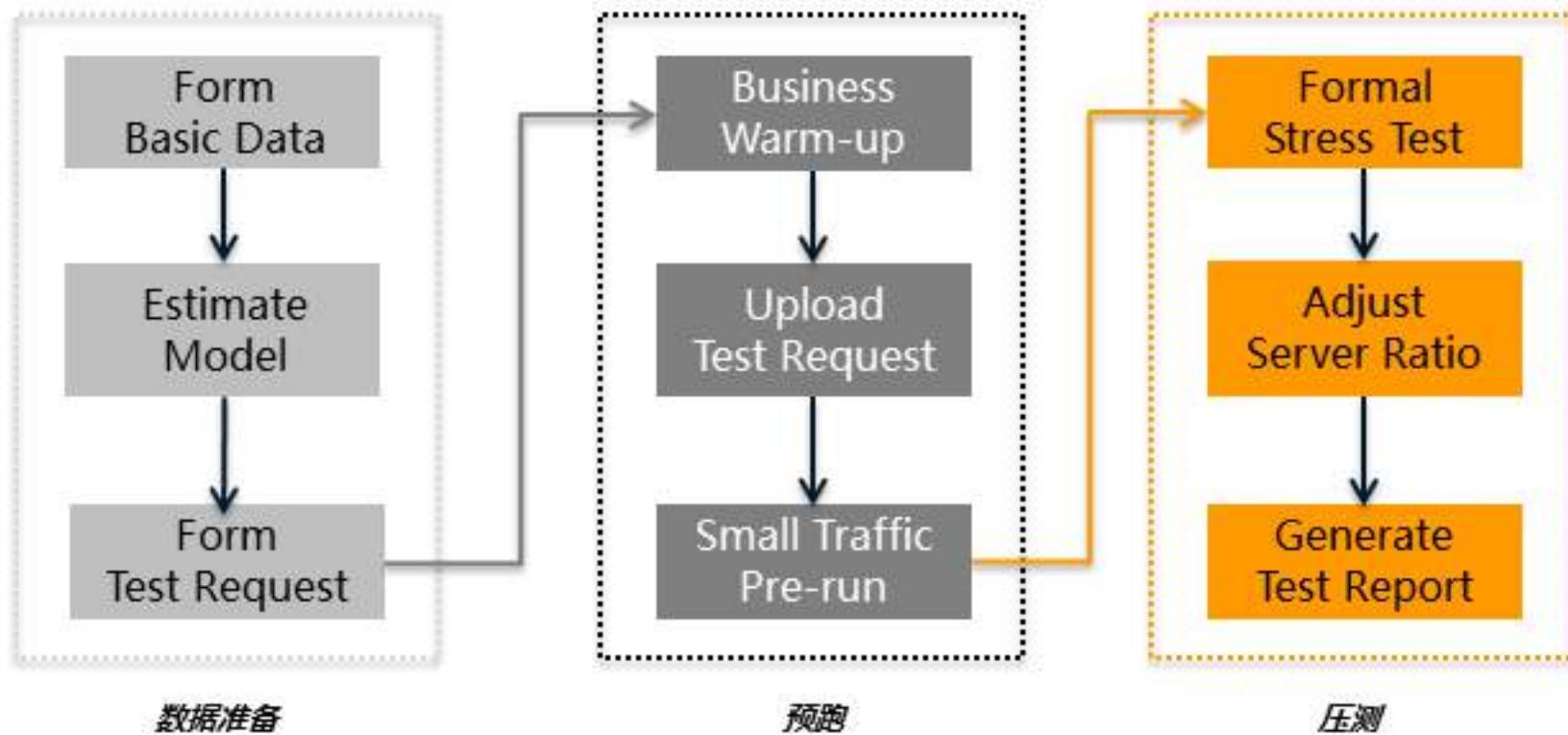
如何构造用户请求模型



隔离测试数据

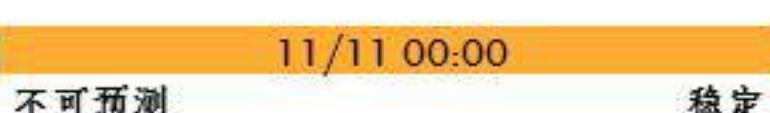
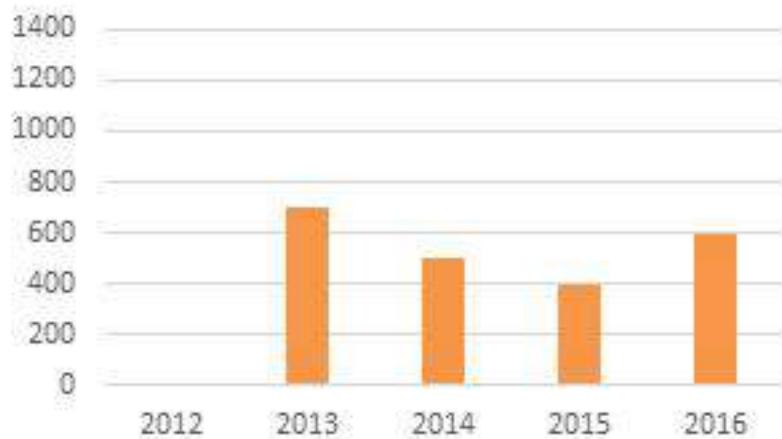


关键步骤



成果

Problems Detected



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)

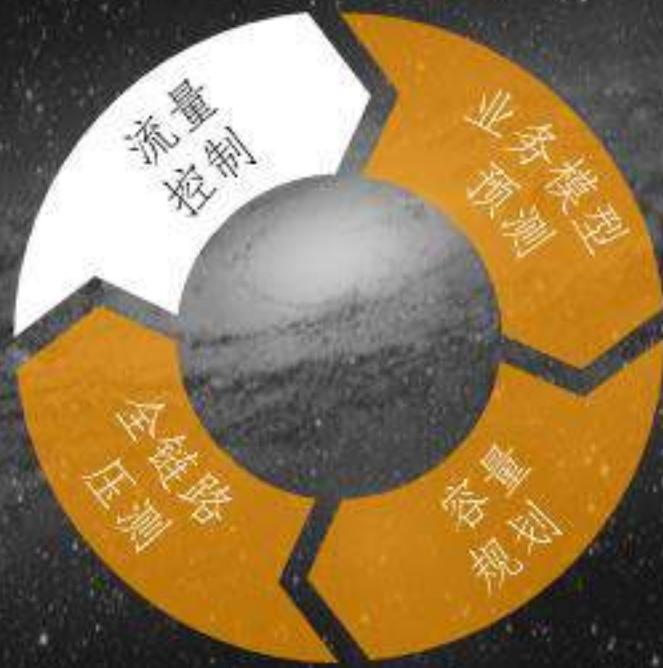
2013 to 2015 平台化

2016 的能力

- 4000+ 链路压测
- 兼容优酷，土豆等子公司模式

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

流量控制的重要性

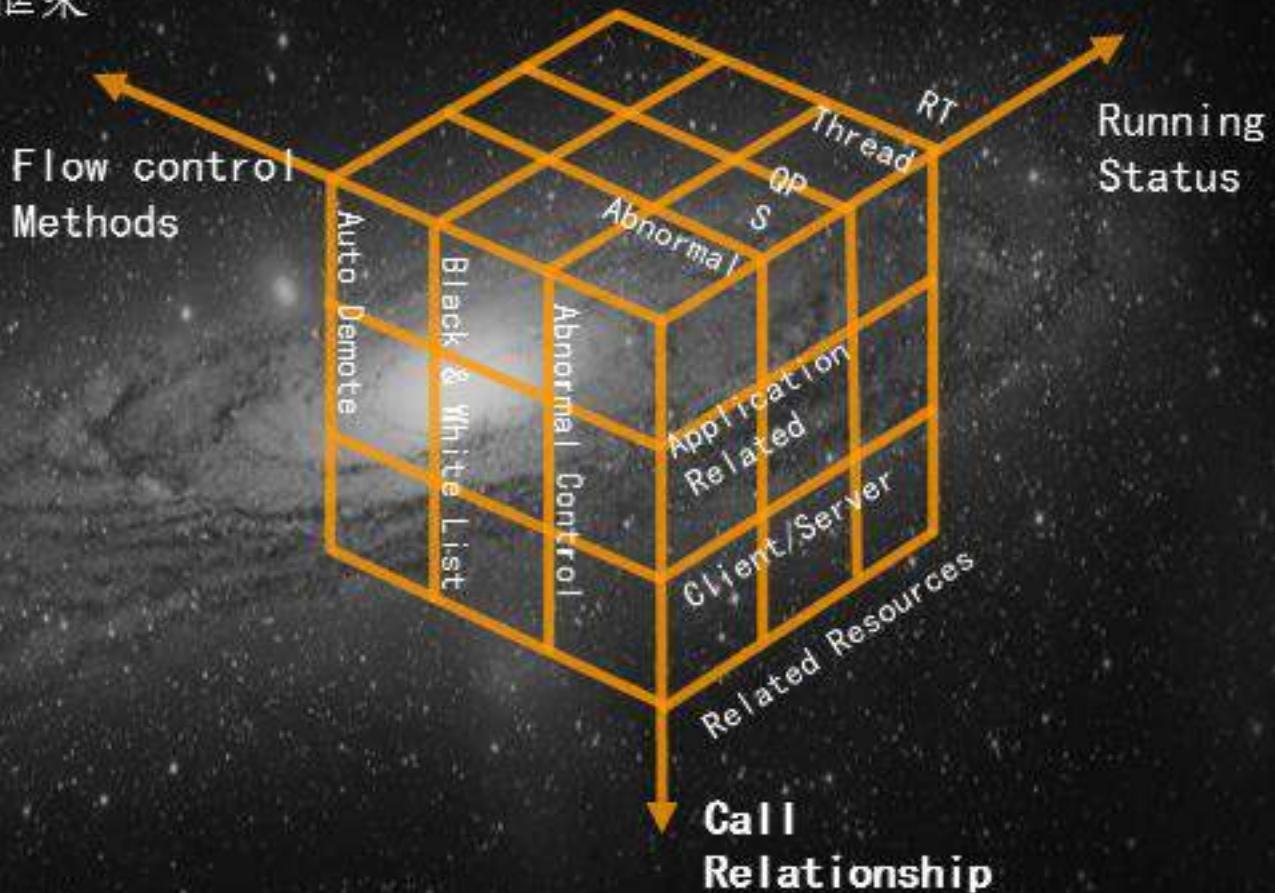


超过容量的流量会造成

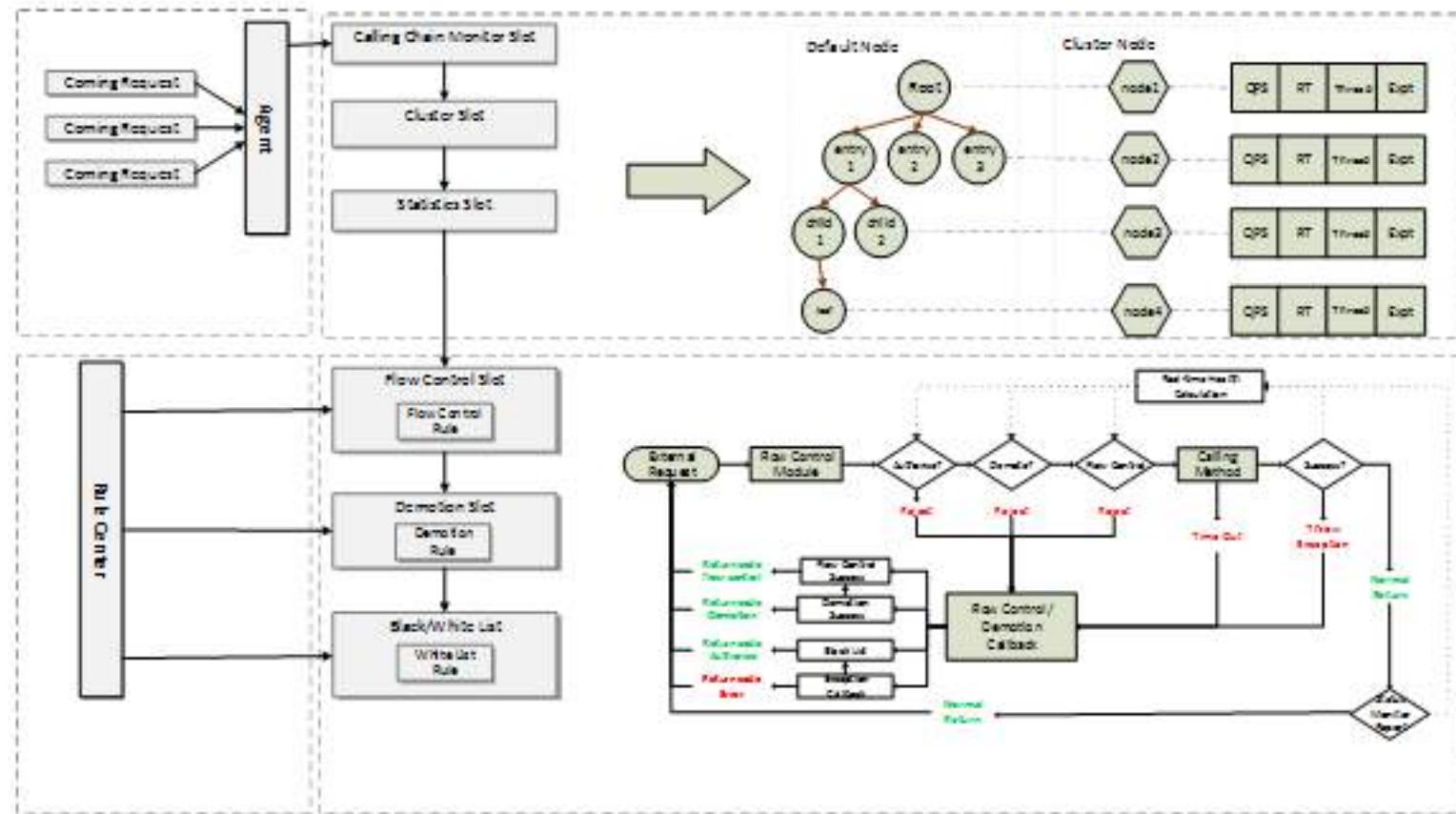
- 影响服务器的性能
- 拉长响应时间
- 影响用户体验.

雪崩效应

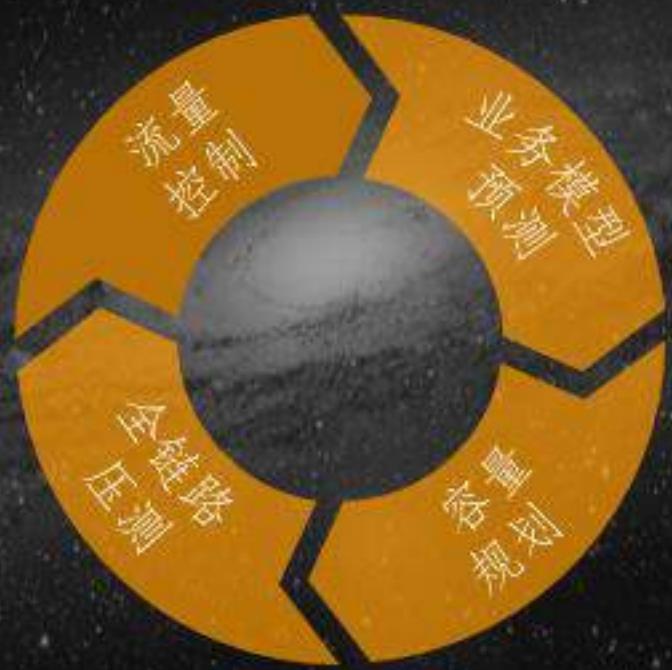
流量控制模型的框架



流量控制的工作流



容量规划



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



阿里云

中国站 搜索台合 购买 登录

全部导航 产品解决方案 数据·智能 安全 云服务 支持 合作伙伴 免费试用

性能测试 PTS

性能测试 (Performance Testing Service) 是全球领先的SaaS性能测试平台，具备强大的分布式测试能力，可模拟海量用户的商业业务场景，让所有行业领跑无所遁形。PTS 还帮助推出了阿里云飞天超大规模弹性伸缩系统。点此 申请试用套餐

立即开通 产品价格 相关文档

无限接近真实的流量
全球分布在亚洲一百多个城市机房，连接来自200+国家和地区的用户

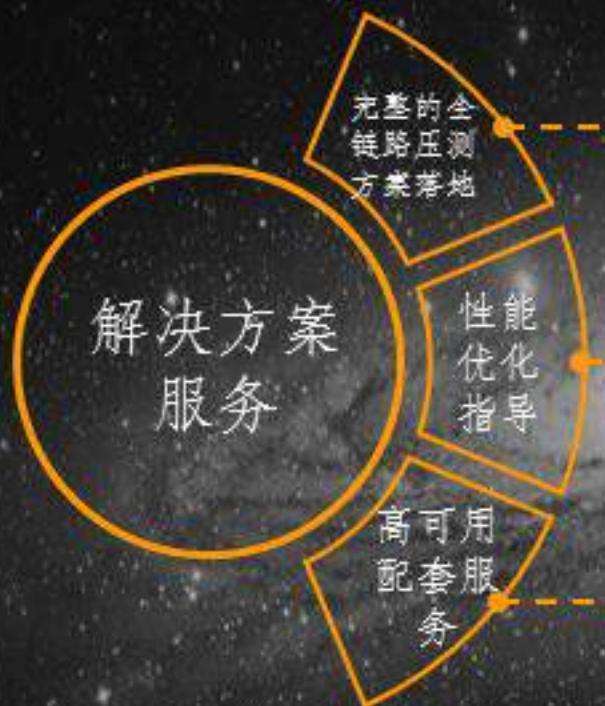
超强并发能力
依托多种技术积累和全球节点，轻松支撑千万级店铺和用户并发

操作零门槛
图形化的交互设计，开发白屏测试，接入门槛极低

复杂场景也能应对
即使双线双链路场景，亦可通过链路负载均衡实现

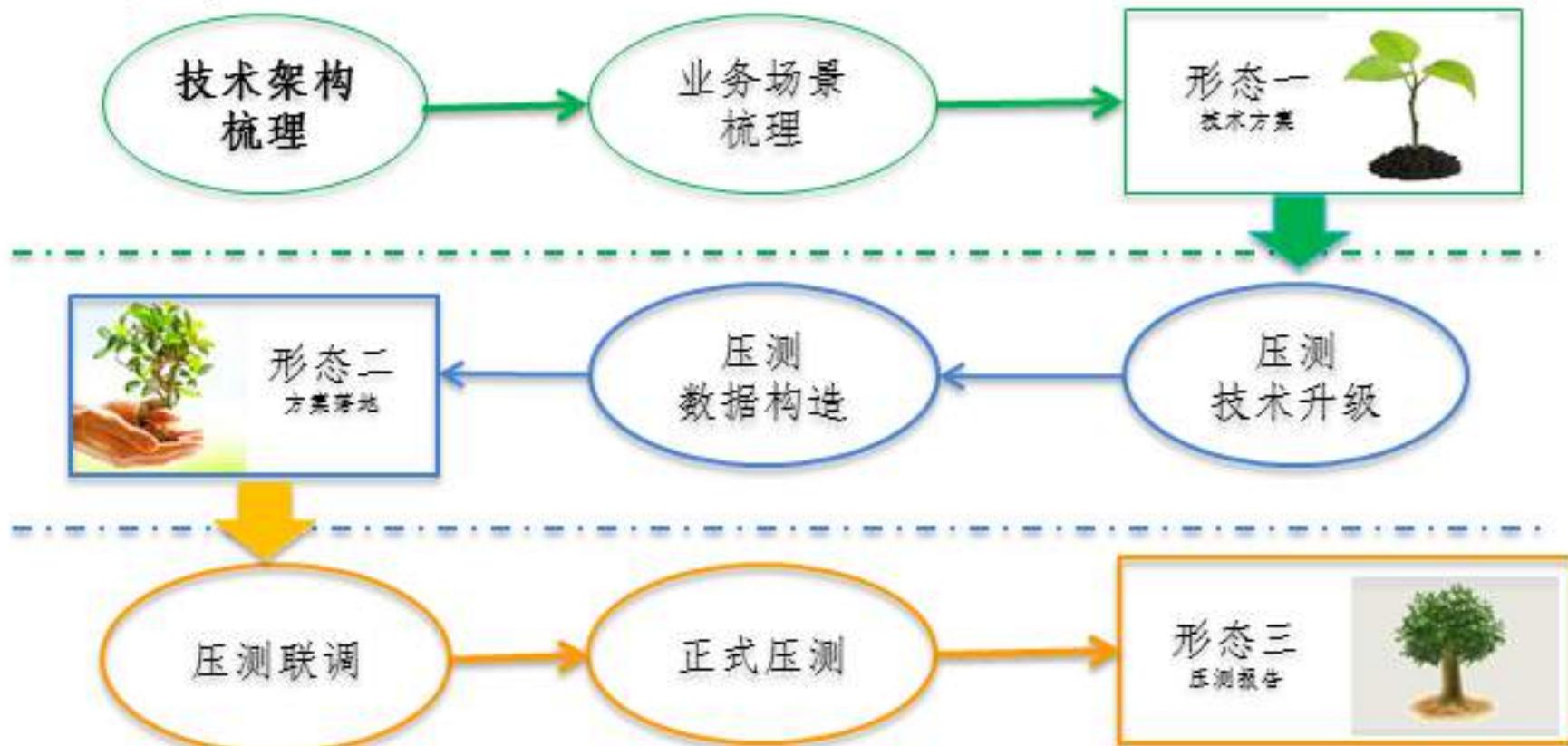
咨询与购买

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

解决方案流程



Thank you!

子矜

子矜



在钉钉上扫一扫加我

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: Machine Learning Pipeline



议程

问题

思路

方案

我们要解决什么问题？



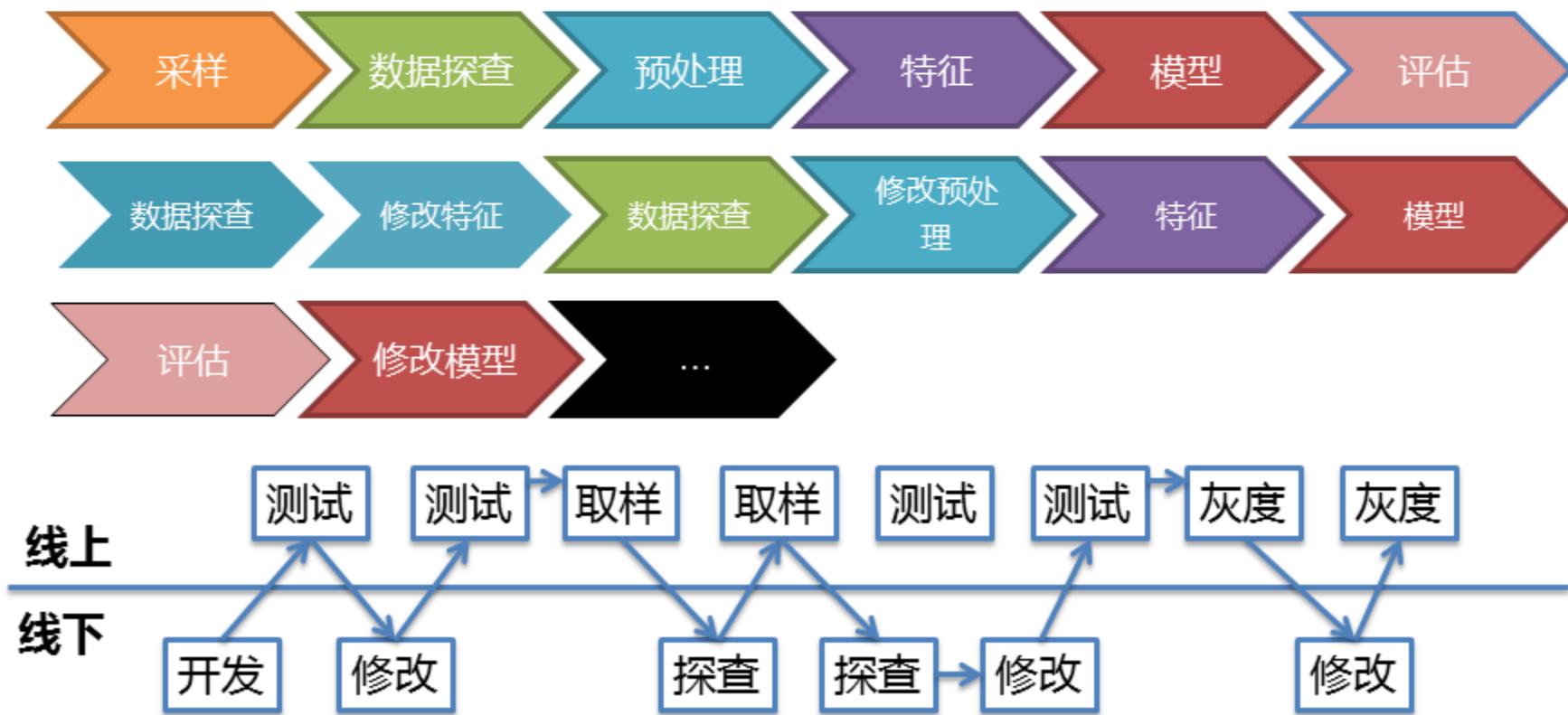
问题1：共享协助的问题

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**？



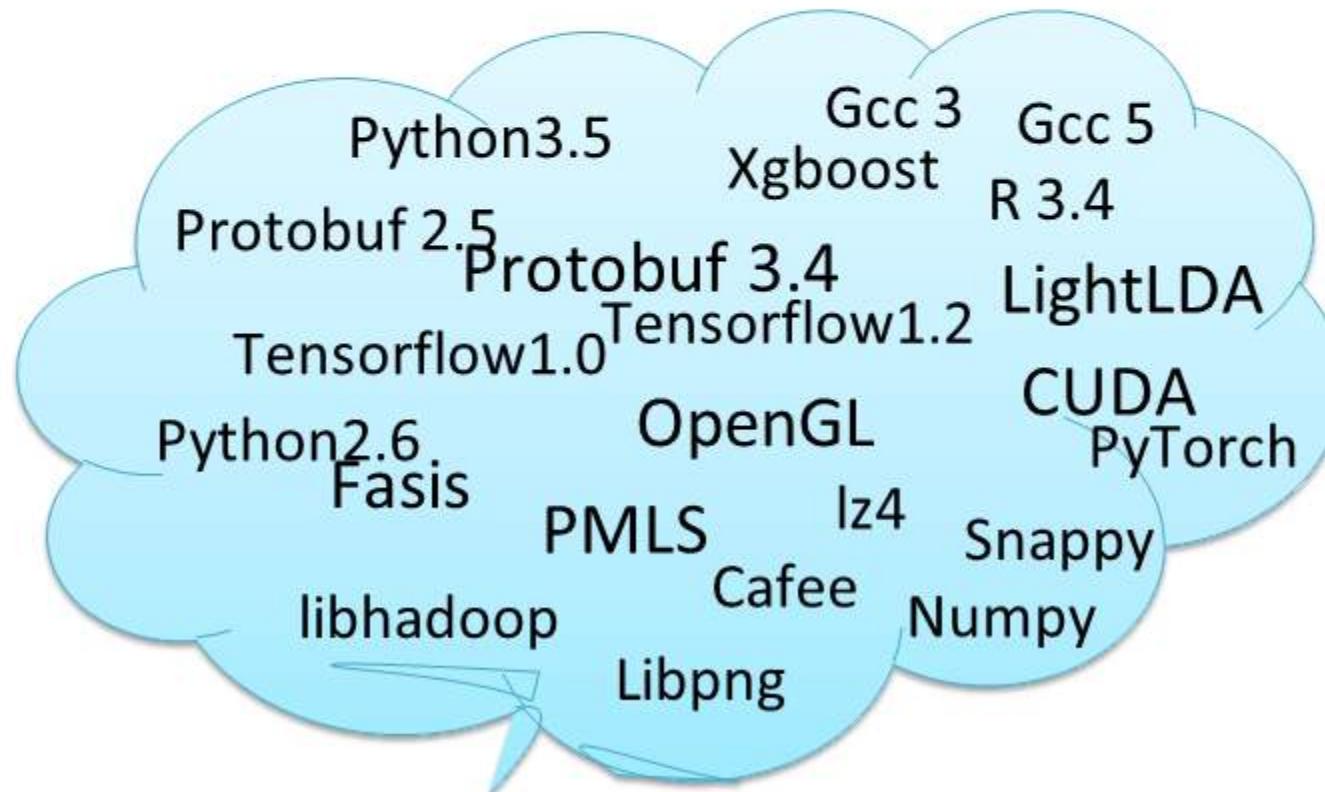
问题2：开发模型周期长，时间成本高的问题

- 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

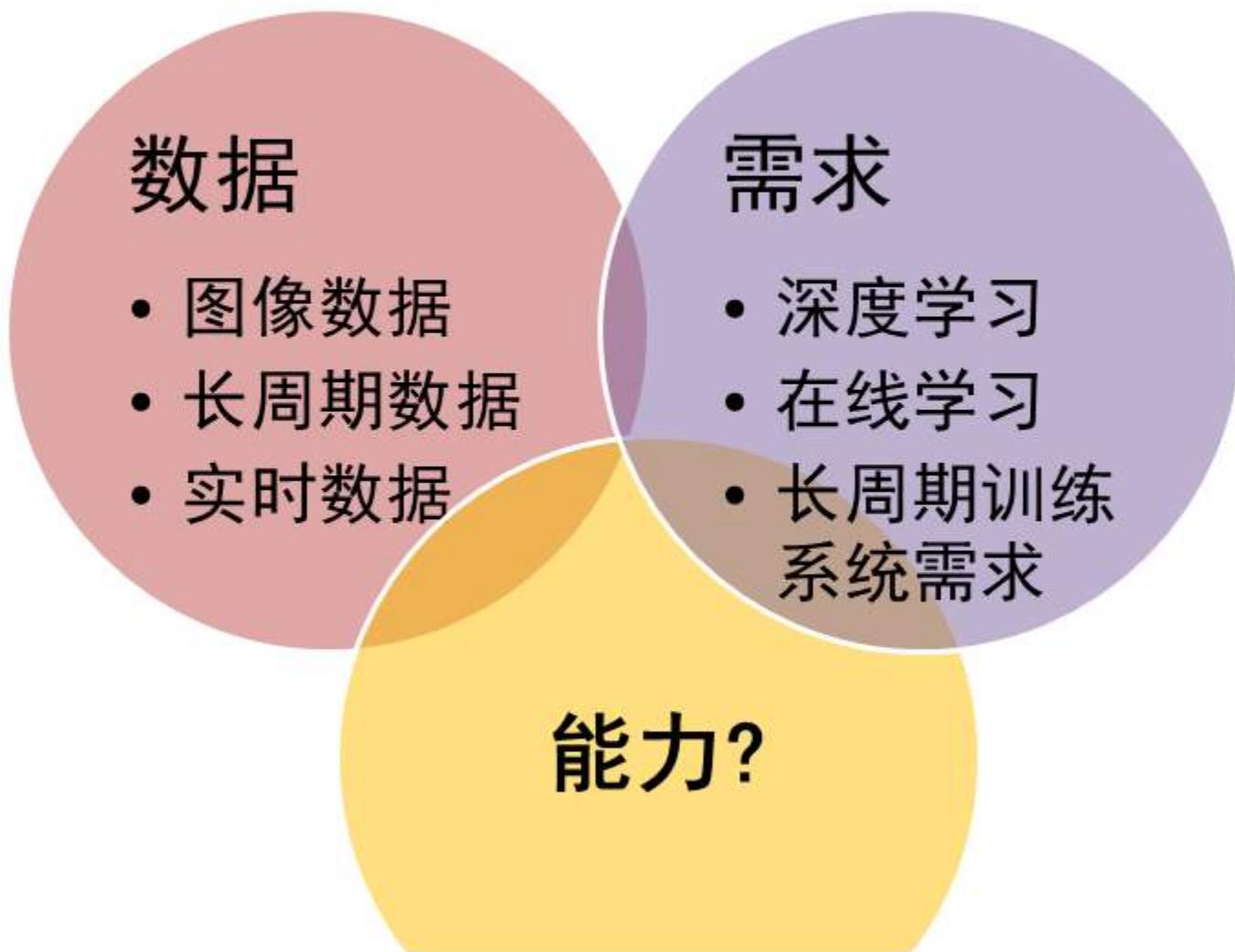


问题3：运行环境多样，系统维护成本高的问题

- 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：

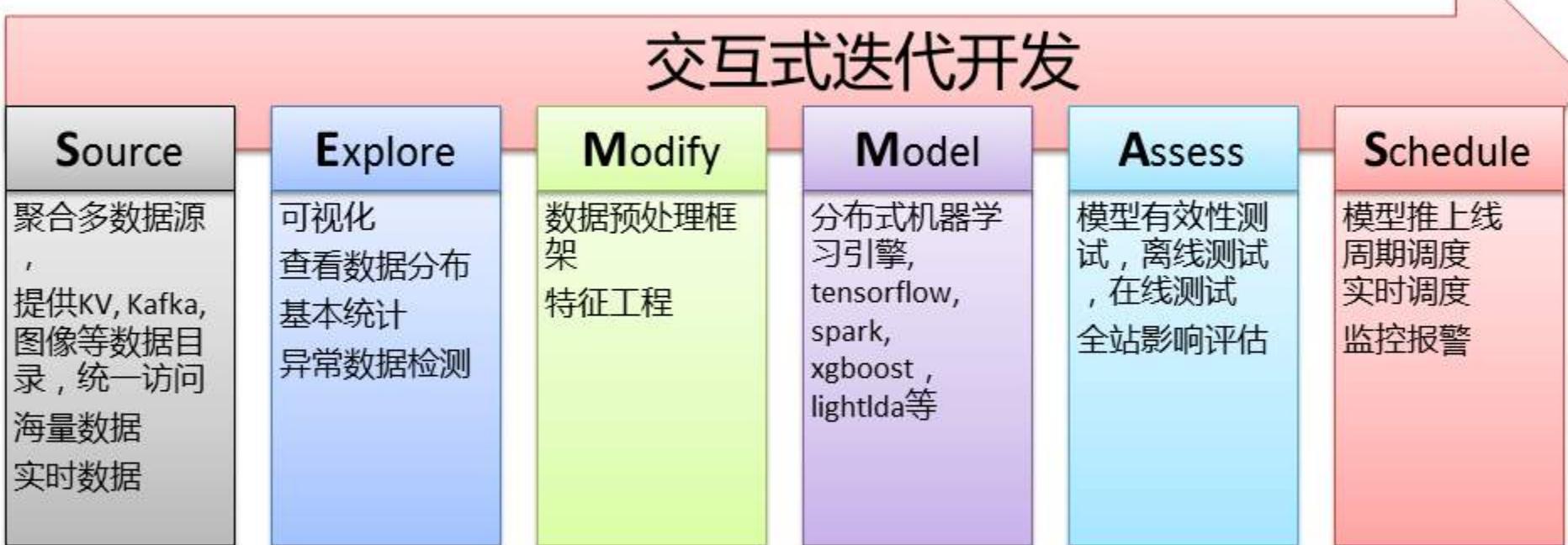
- 设计一个系统，
 - 解放生产力，机器学习现代化。
 - 在实时数据，长周期数据，图像数据上具备处理能力，满足在线学习深度学习需求；
 - 鼓励共享协作。

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。



数据共享，模型共享，算法共享

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

大规模
分布式

稳定

有效

高性能

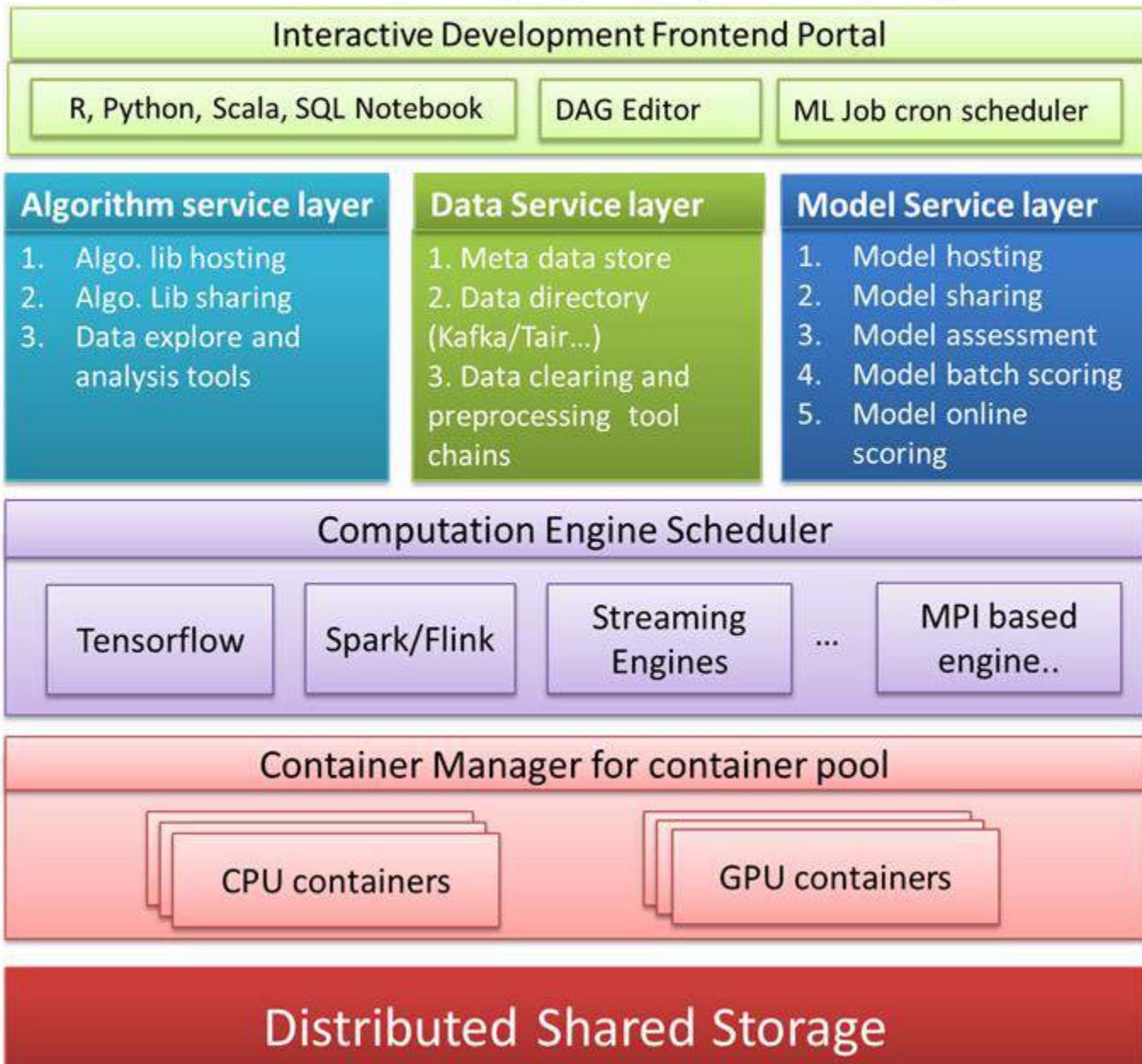
对计算框架的选择技术上不做限定，按性能和稳定性作为ML引擎选择的标准

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



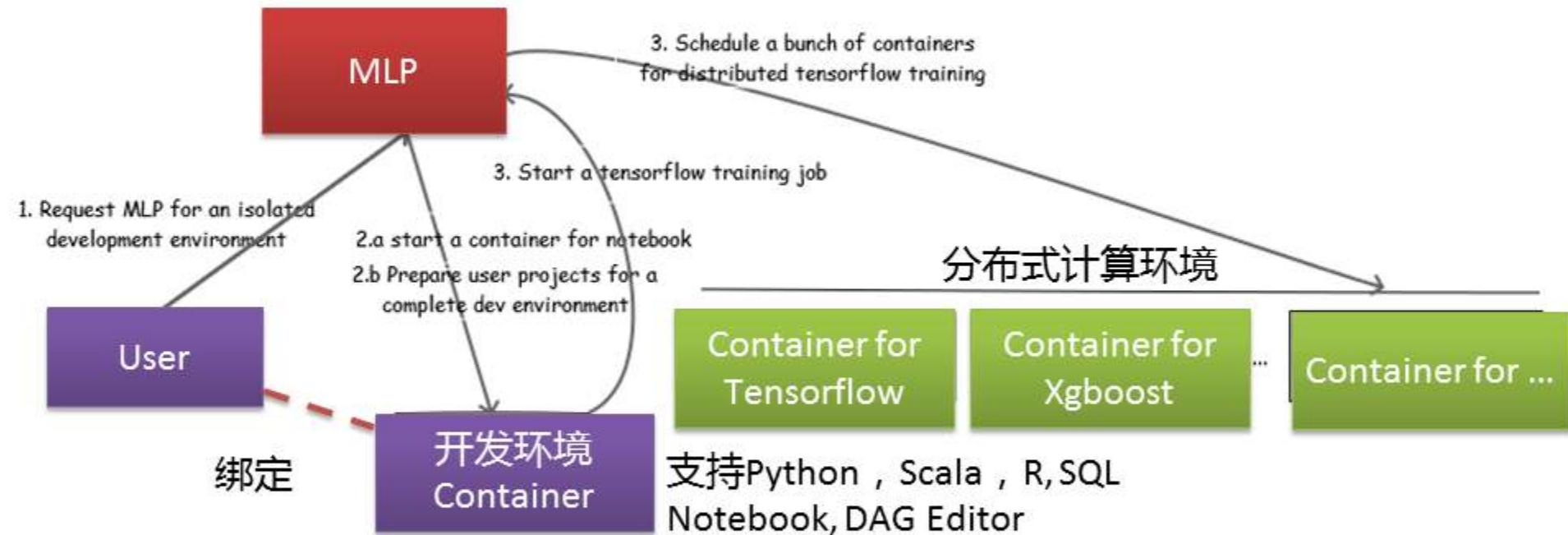
Monitoring and HA

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

User workflow

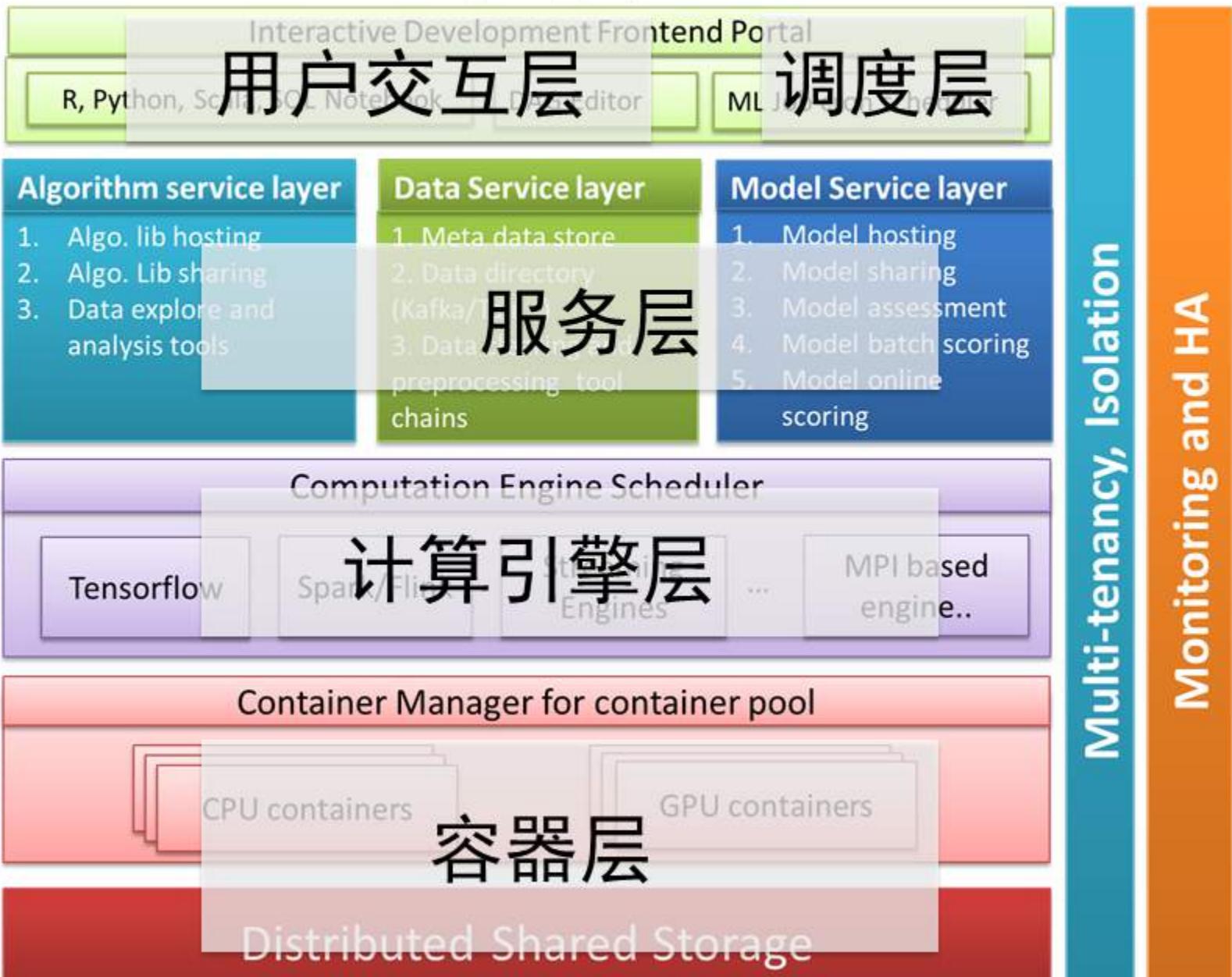


支持多租户，每个用户会拿到一个独有的，隔离的开发环境
成熟后开发中的Notebook/DAG直接转为生产，
添加到工作流调度器中。

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

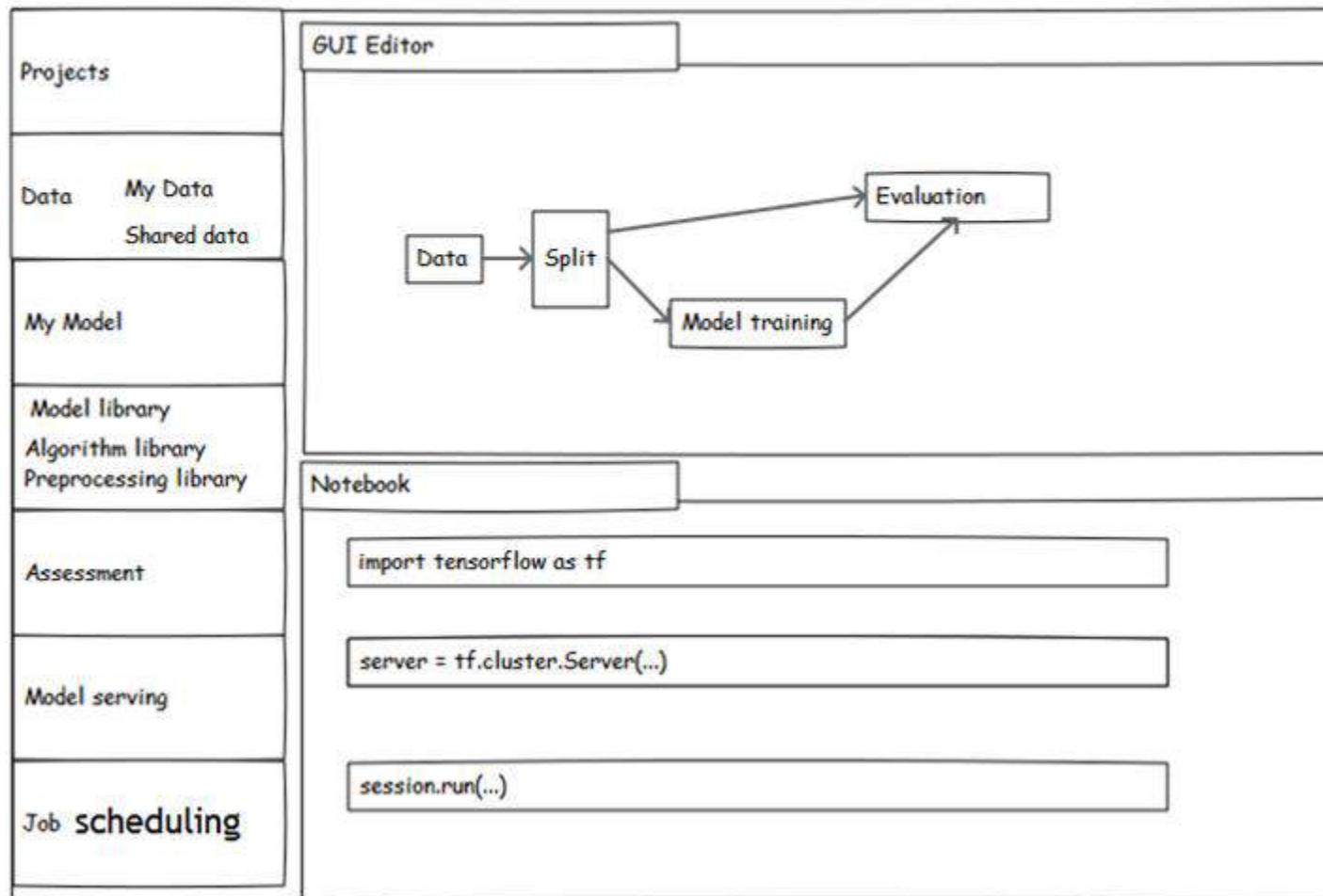


1. 用户交互层

- 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。

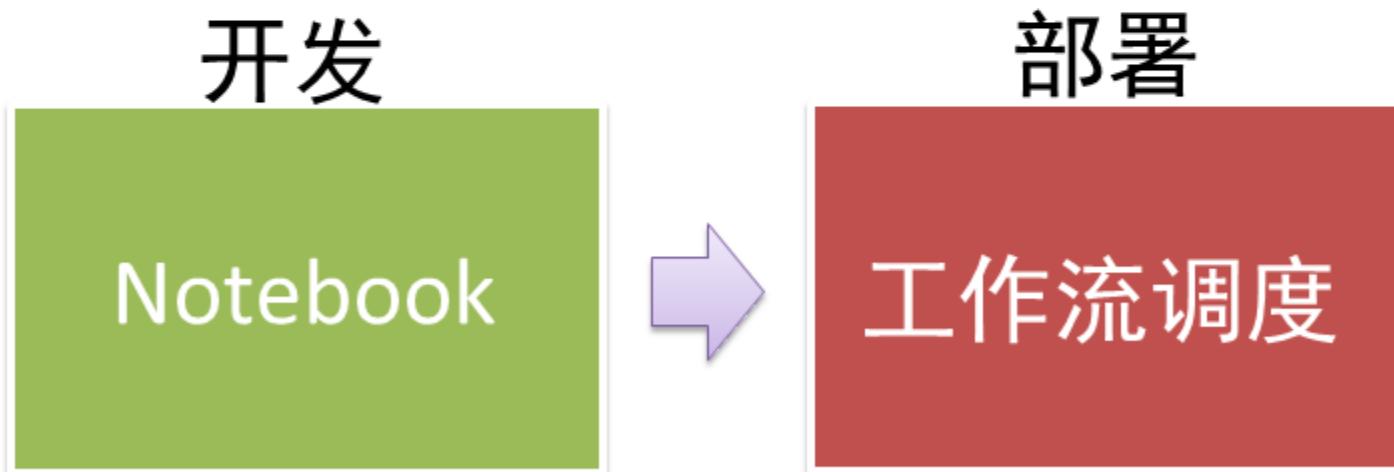
Notebook UI

- 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

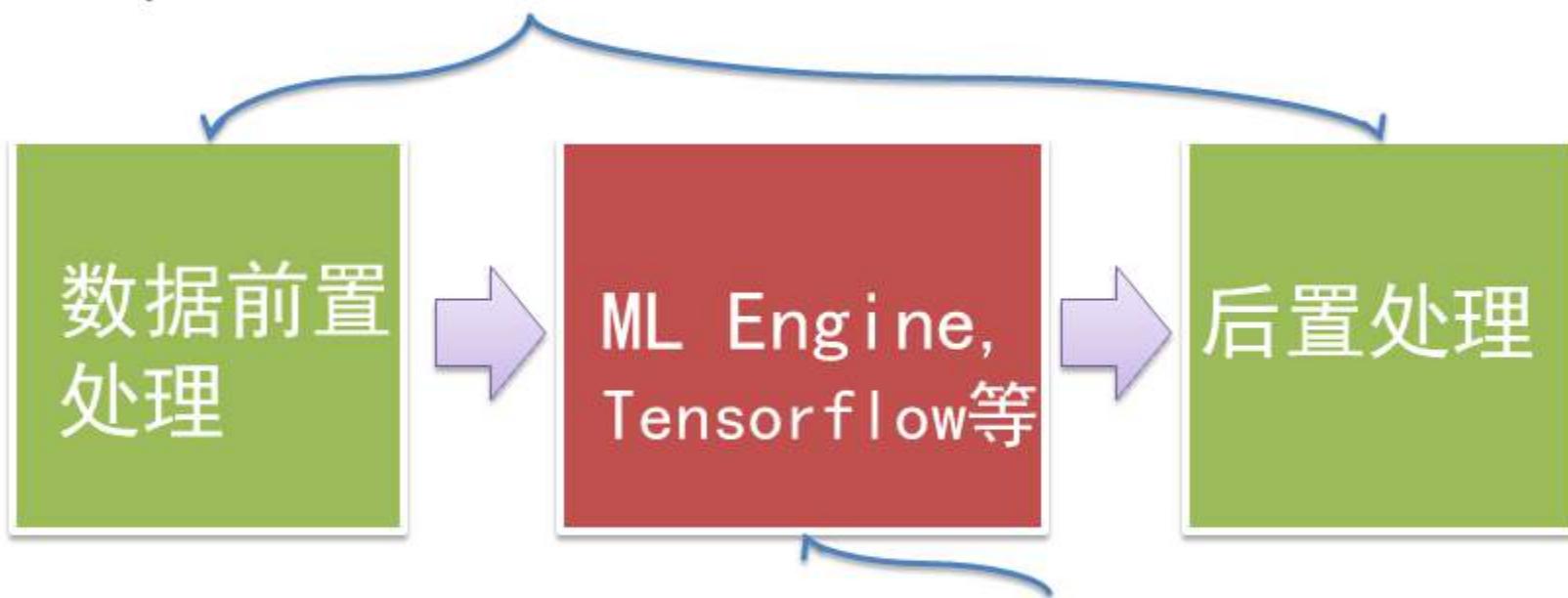
Backed By Gluster

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

4. 服务层

算法服务

通用算法
基础算法
算法共享，协作开发

模型服务

Model directory
Model sharing
Model deployment
Model validation
Model Assessment

数据服务

统一数据访问
基础数据大联通
特征工程数据共享
数据处理工具链

总结

1. 通过在线服务提供机器学习一站式平台。
2. 通过容器实现多租户隔离，版本管理，和资源弹性扩展。
3. Notebook和DAG Editor相辅相成成为用户入口。
4. 强调共享协作。数据共享，模型共享，算法共享。
5. 以提高生产力和支撑前沿探索系统需求为最高目标。

Thank you!



2017 Software Architecture Summit

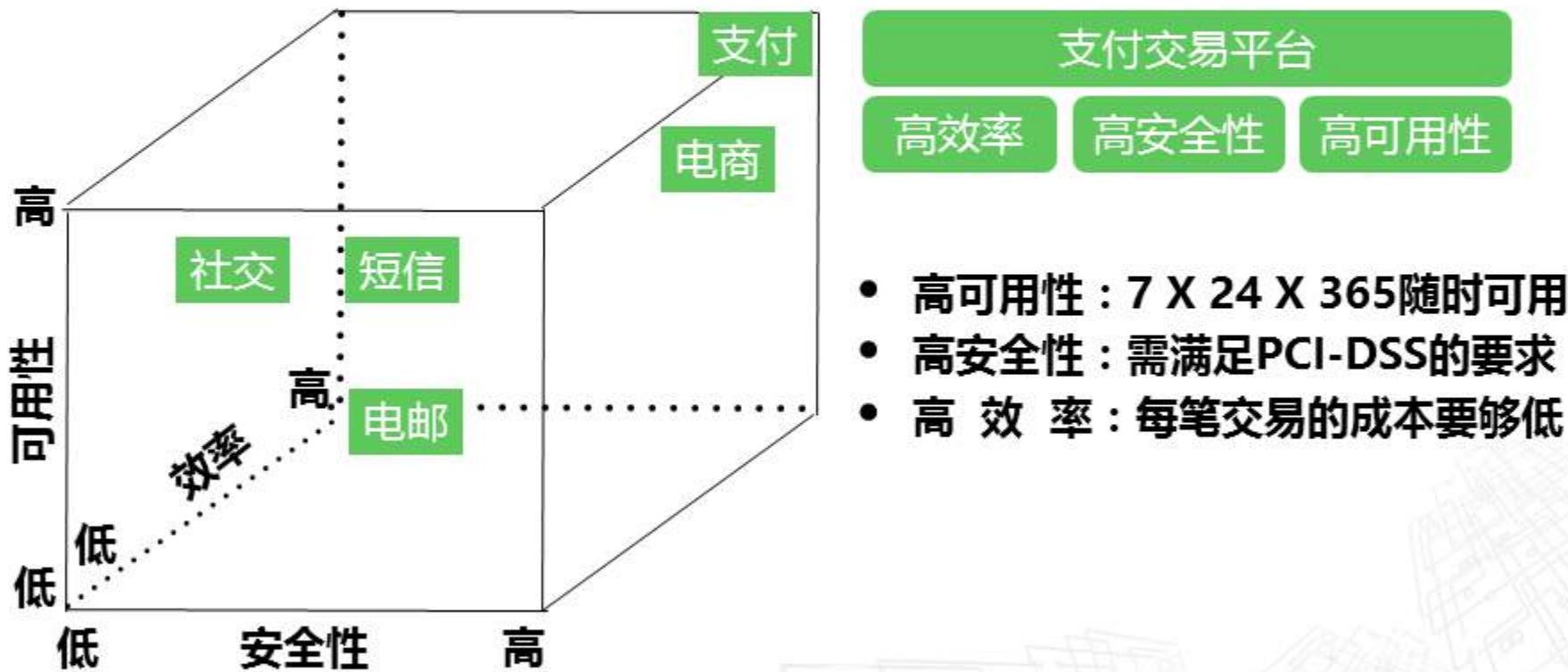
如何构建大型支付交易平台

陈斌

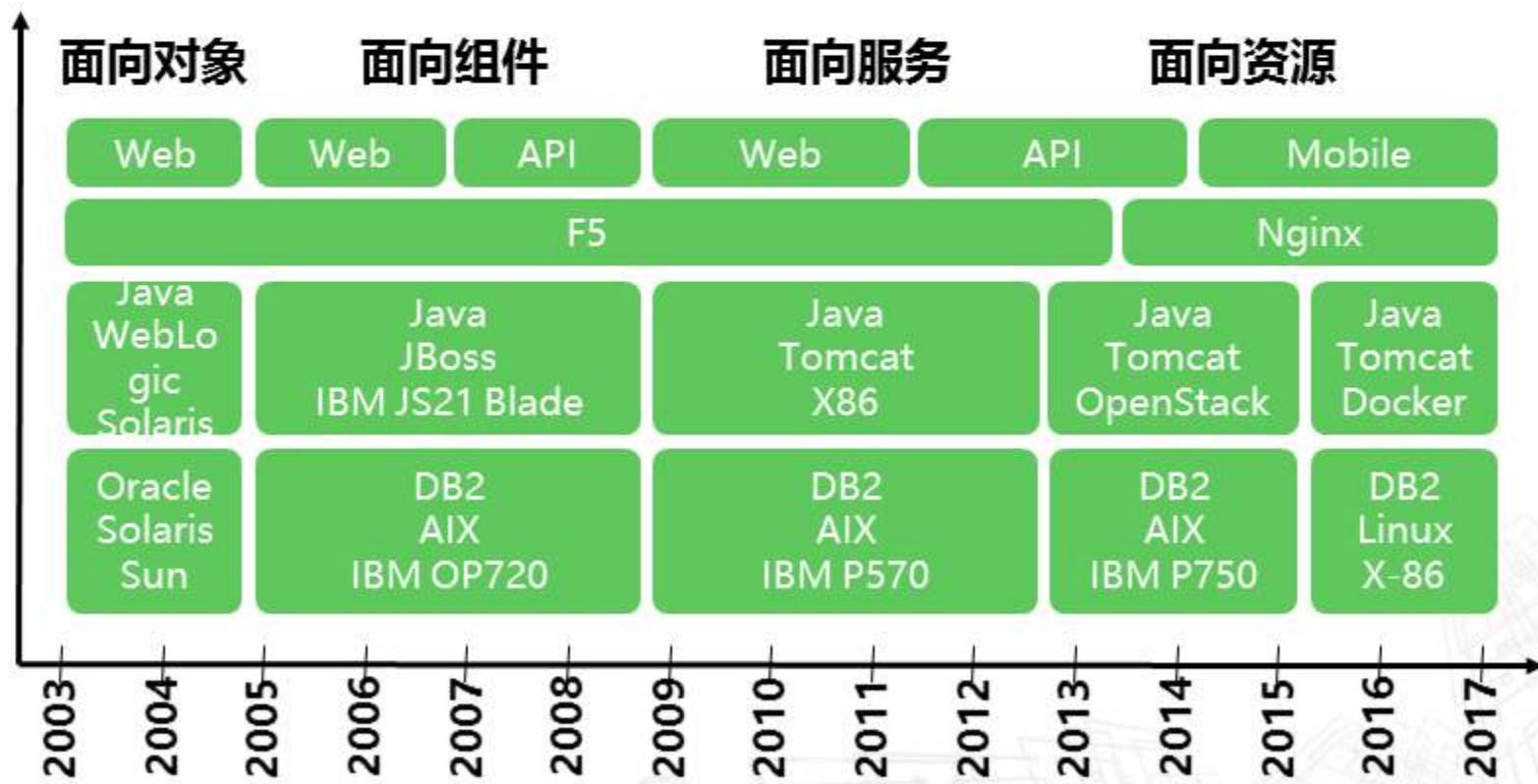
易宝支付CTO



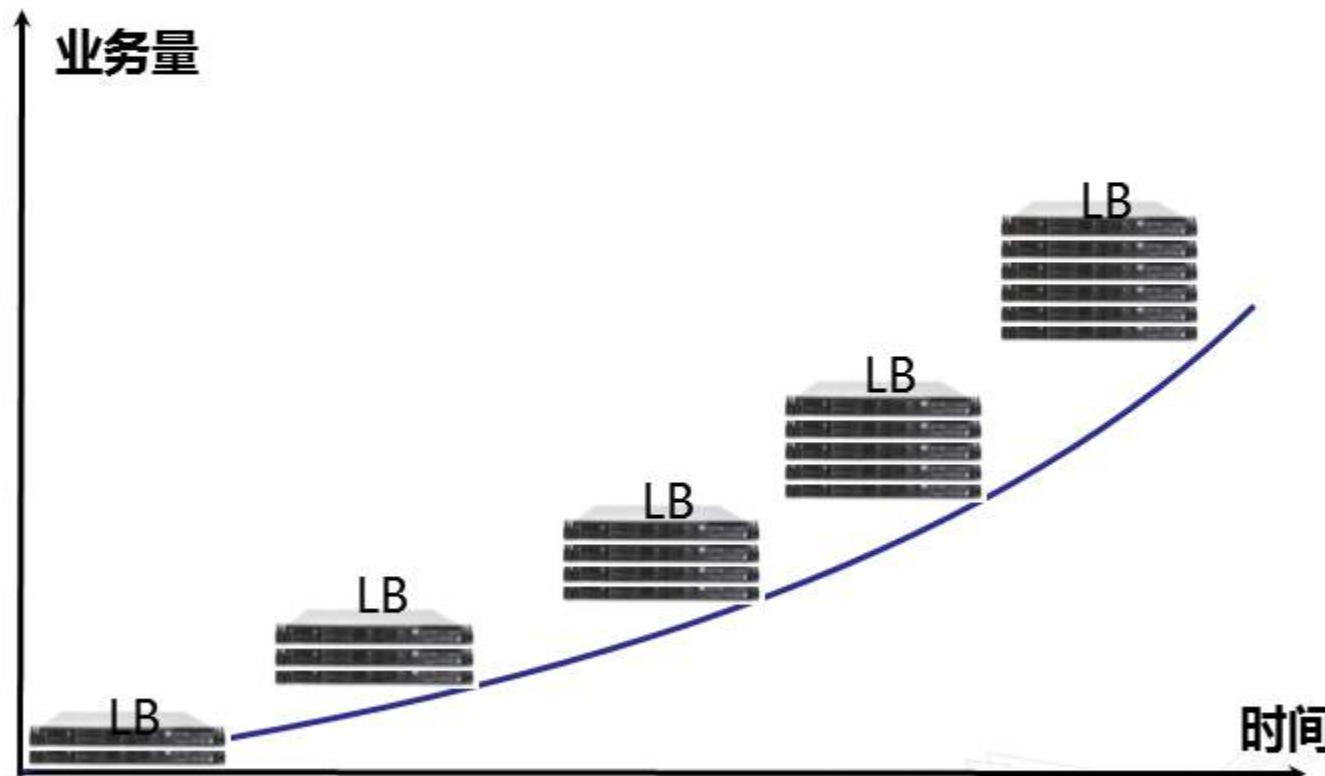
支付交易平台的特点



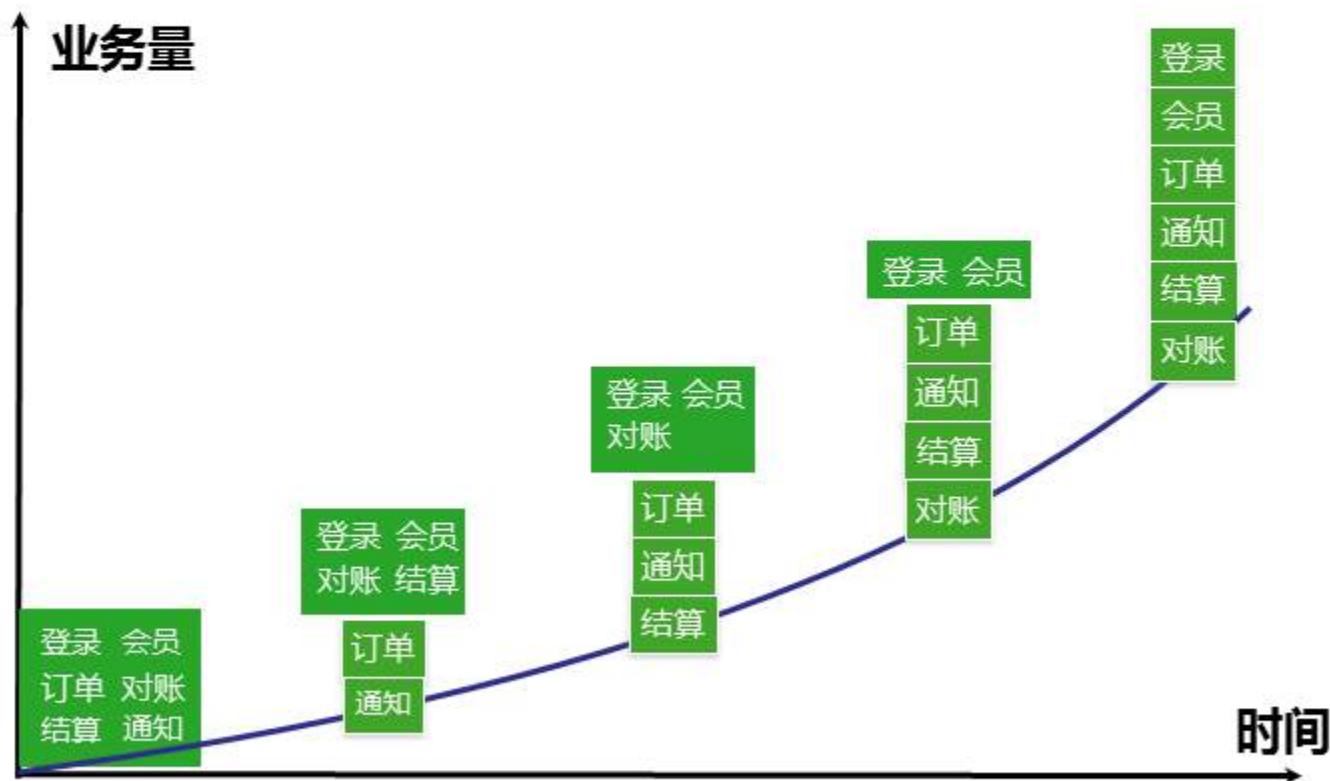
易宝支付交易平台的发展历程



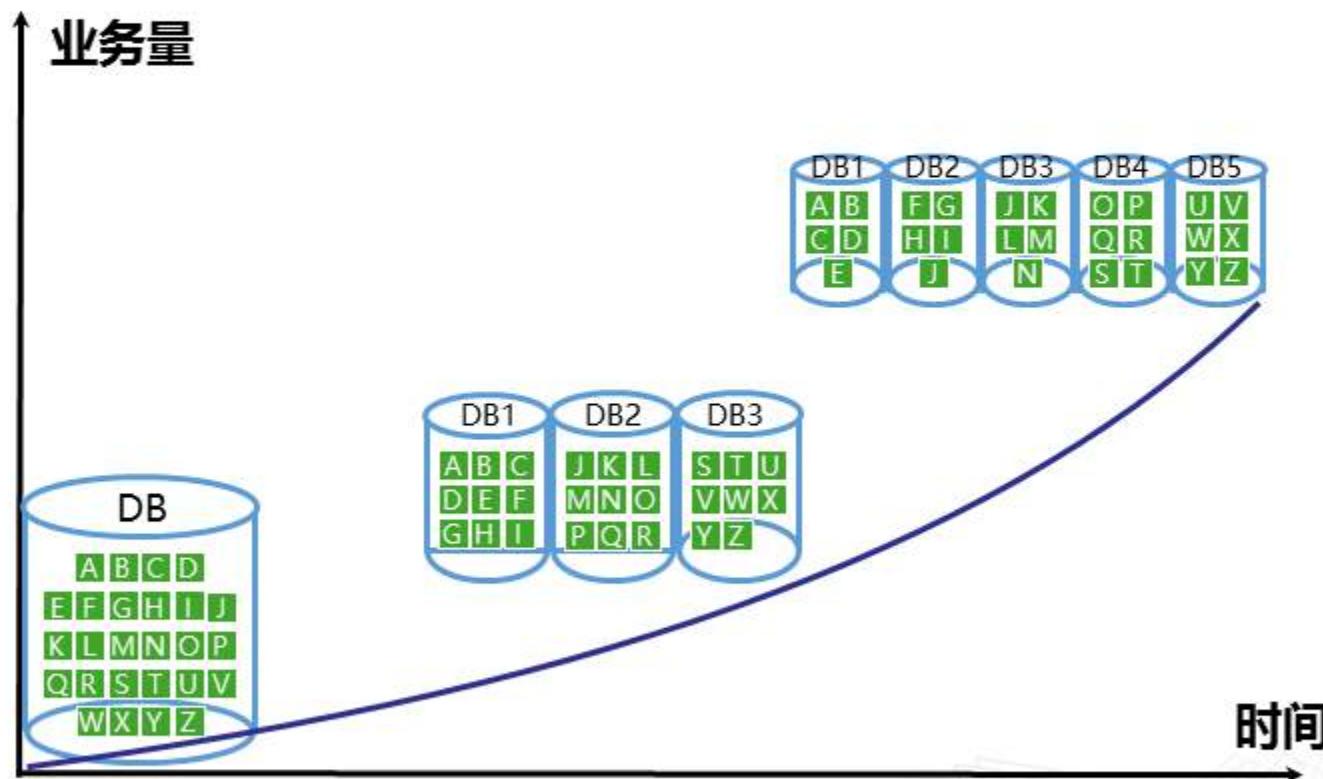
X轴扩展-水平扩展



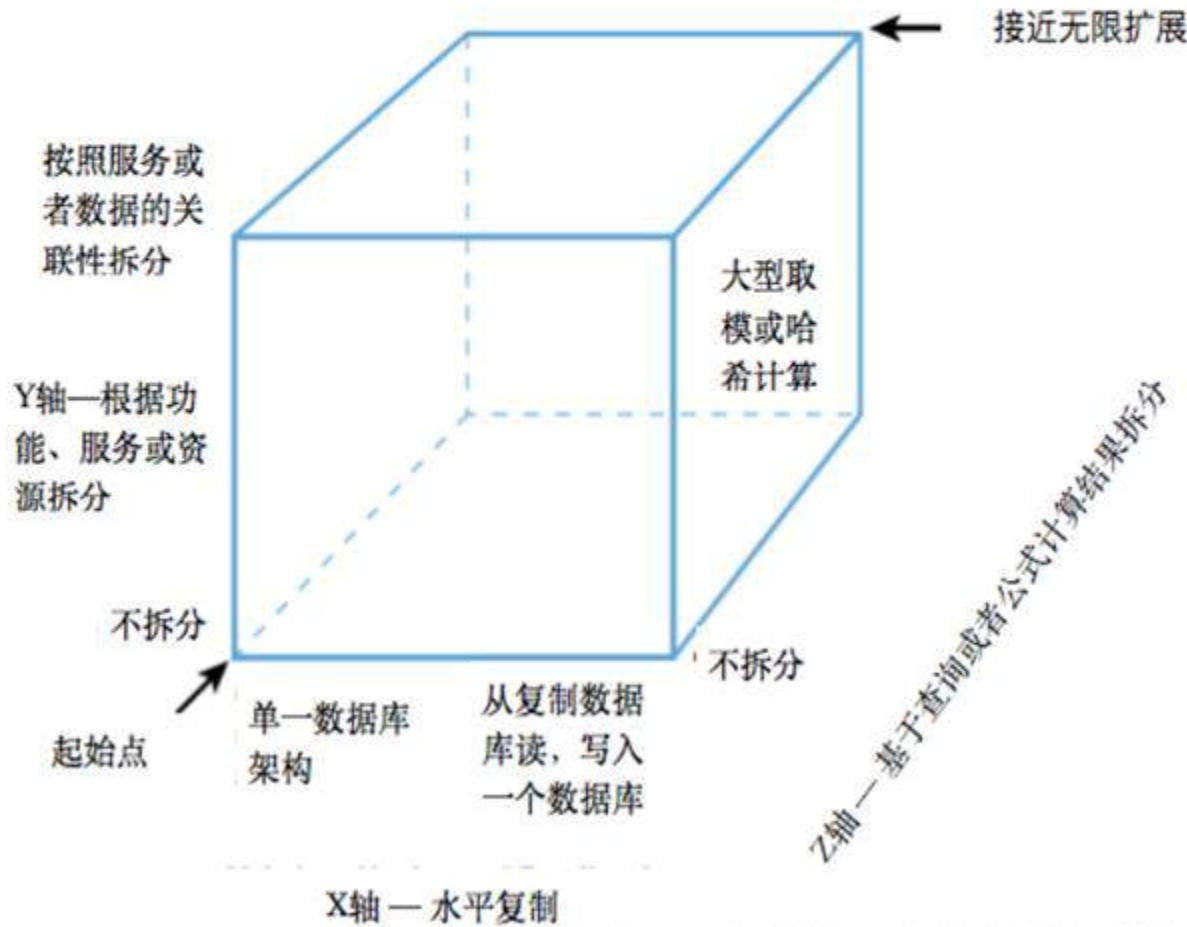
Y轴扩展-应用拆分



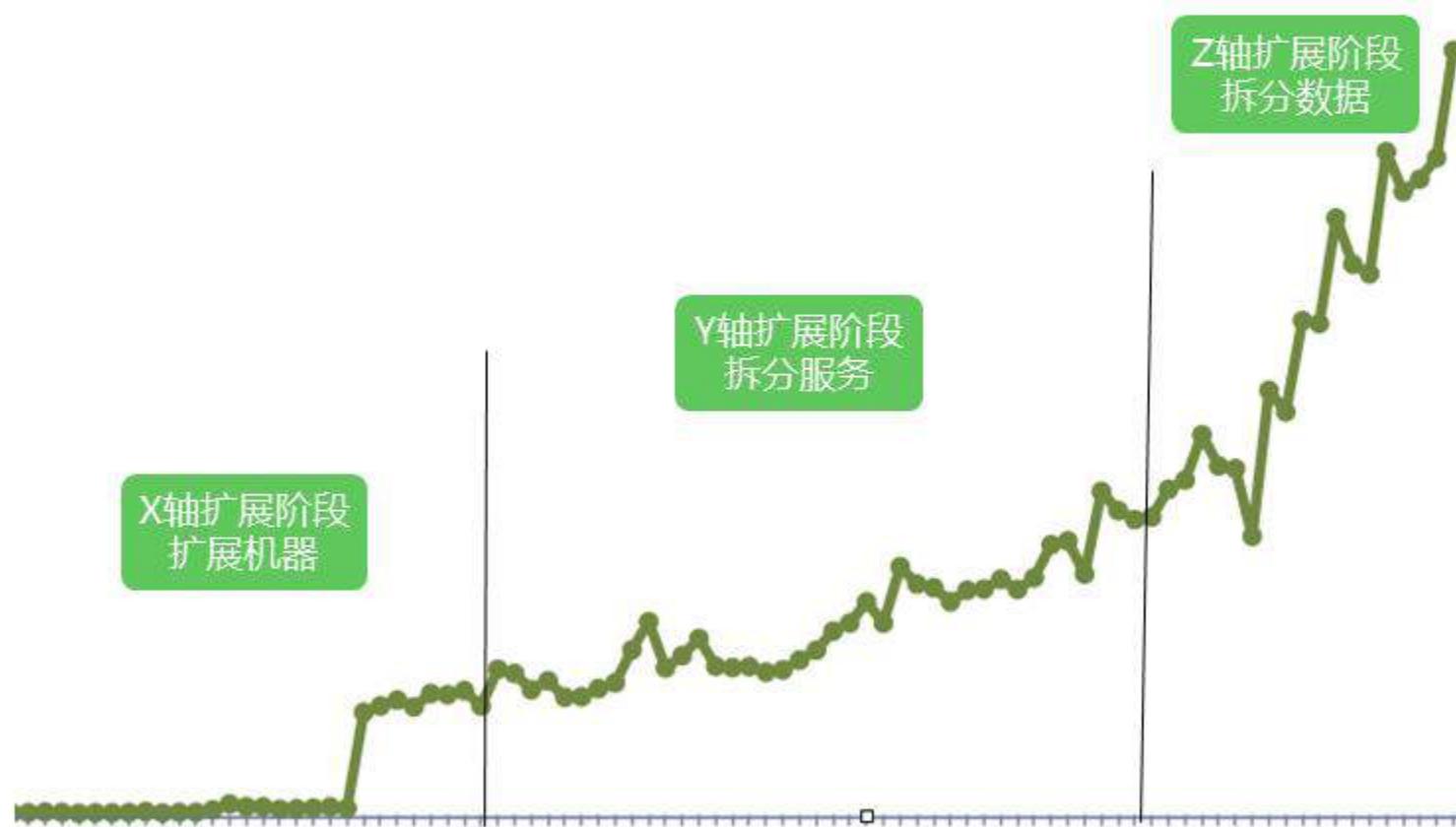
Z轴扩展-数据拆分



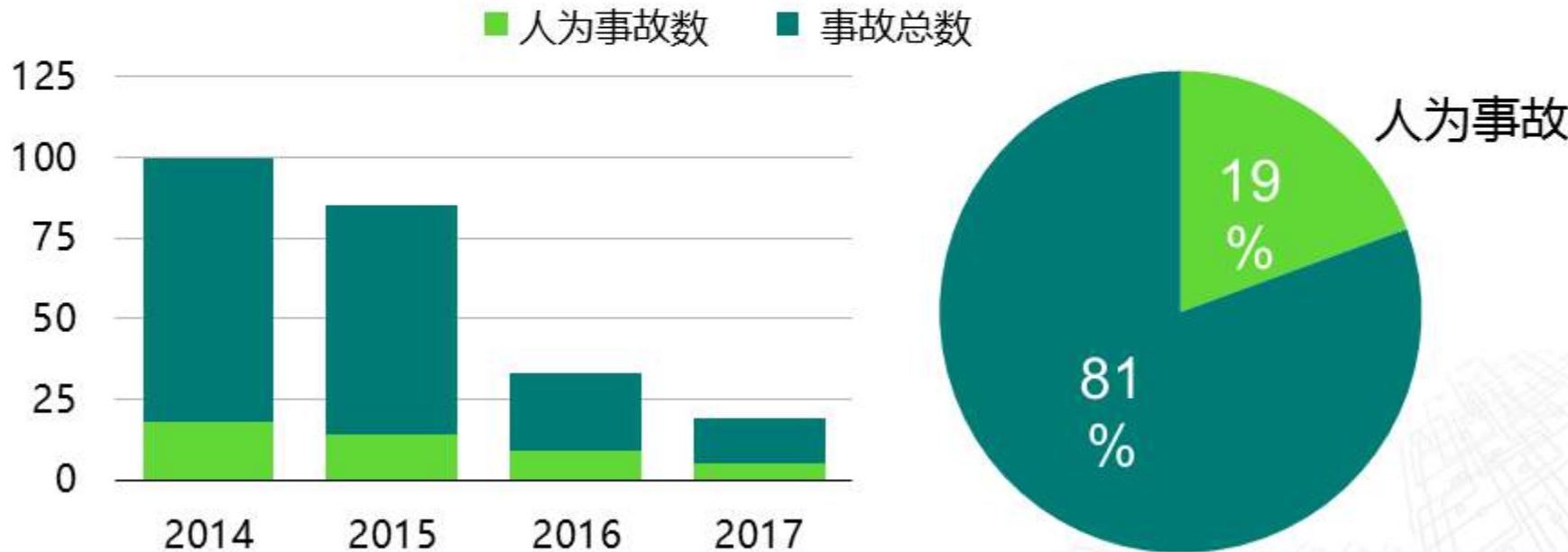
NKF的三轴扩展理论



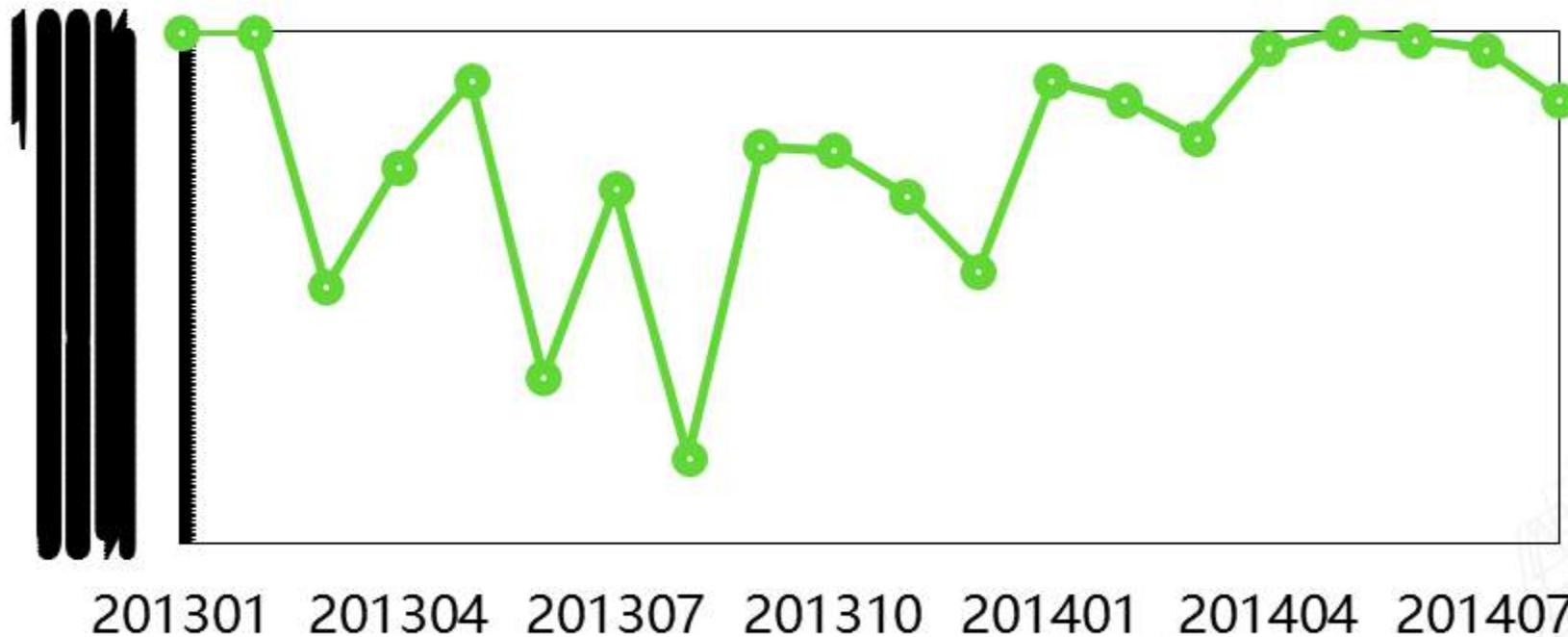
易宝支付平台扩展演进历史



版本多、环境杂，步骤多、指令烦

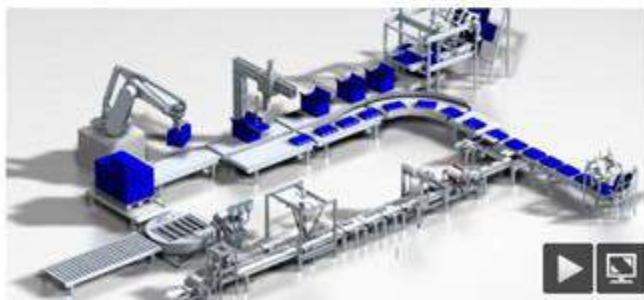


支付平台的可用性不断波动



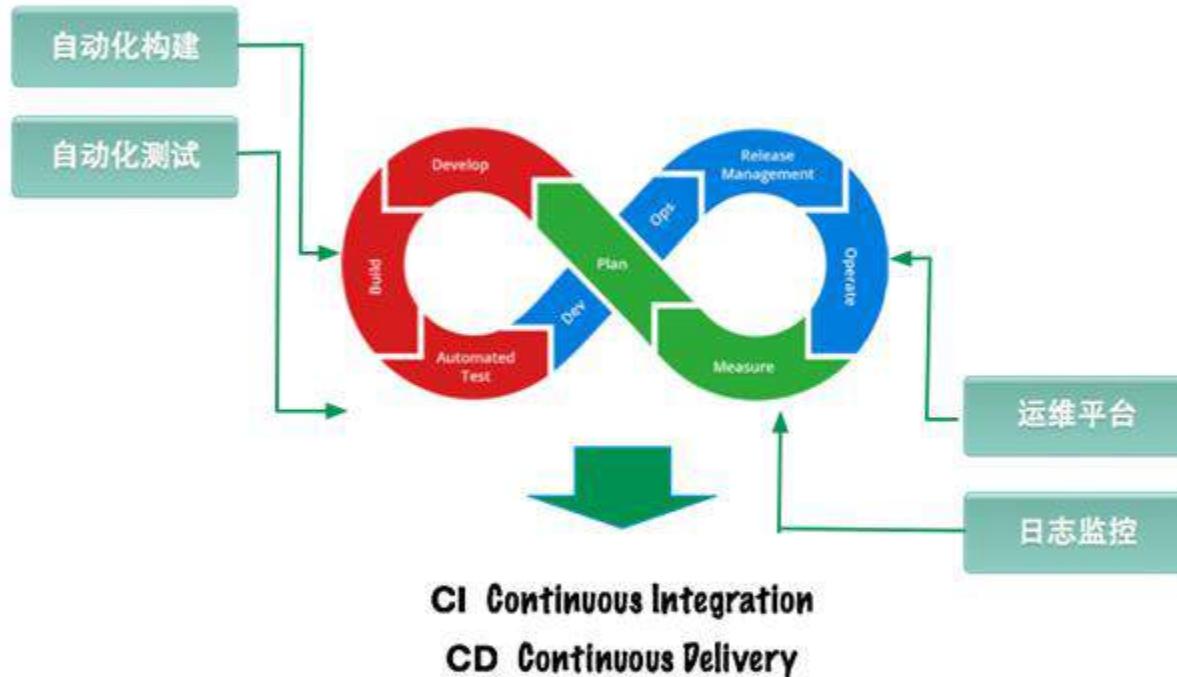
运维面对挑战

人常犯错误，而且往往多次犯同样的错误。人也容易对重复性或琐碎的工作失去兴趣。对企业而言，人员成本都是显著的，而且好的人才价值不菲。



自动化机器每次都会完成简单重复的任务，或取得相同的成功，或犯同样的错误。所有系统都该在设计阶段考虑自动化部署、构建、测试、监控。

DEVOPS



技术选型



功能比较简单
技术栈简单
非稳定版



偏重资源抽象
技术栈复杂
大规模场景



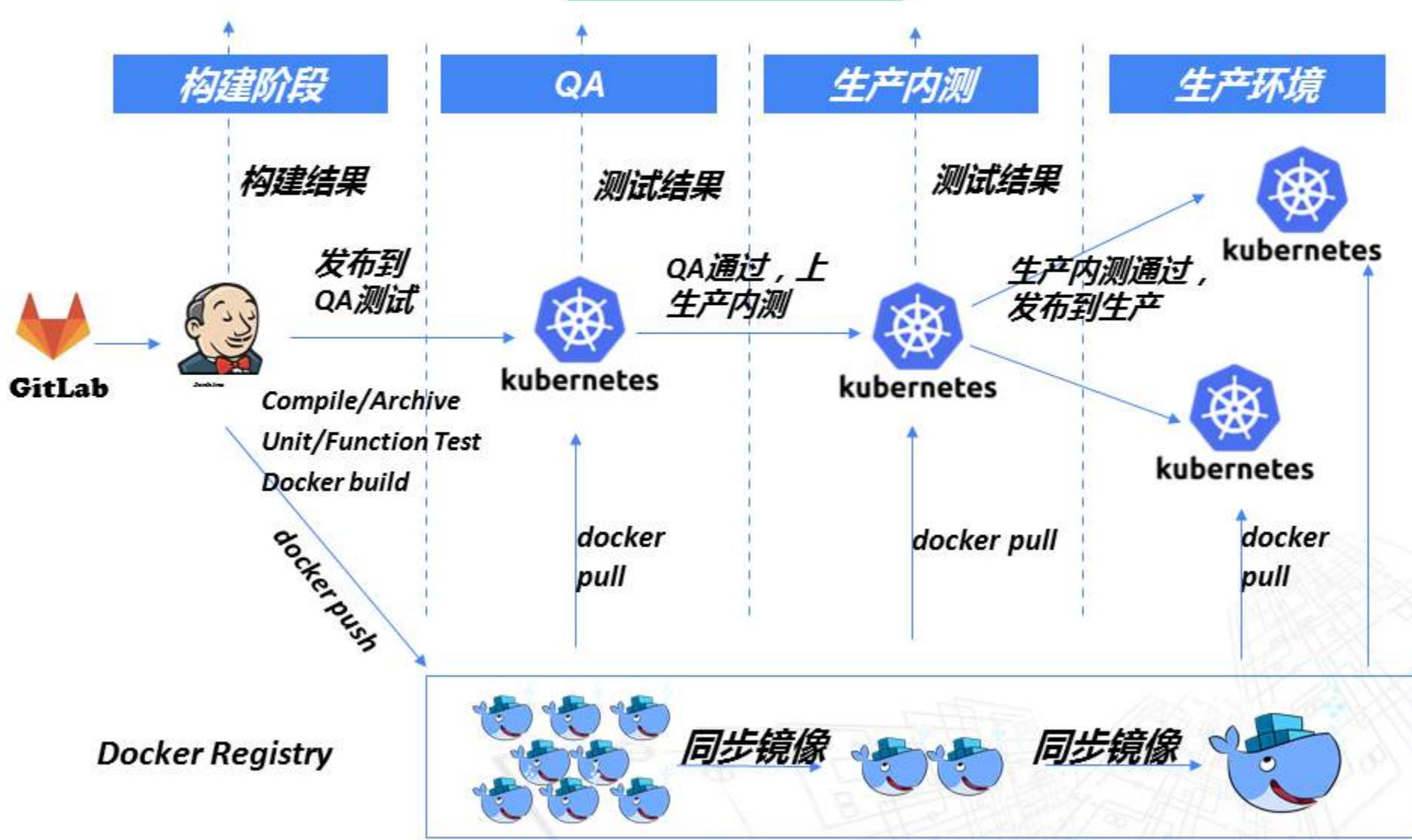
自主研发
投入产出比低
项目风险大
机会成本太高



理念先进
以应用为中心
Web应用为主
功能完善

Pass!

易宝容器云



搭建基于容器云的基础设施



保持弹性

繁忙



正常



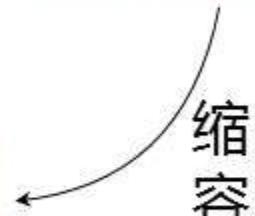
扩容



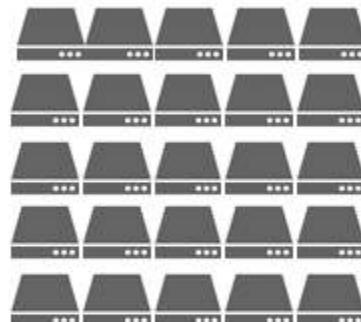
空闲



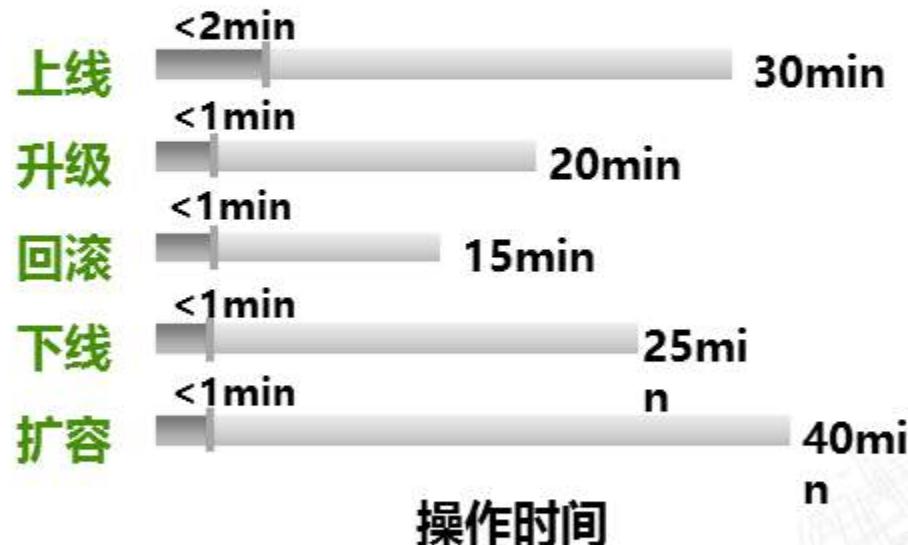
缩容



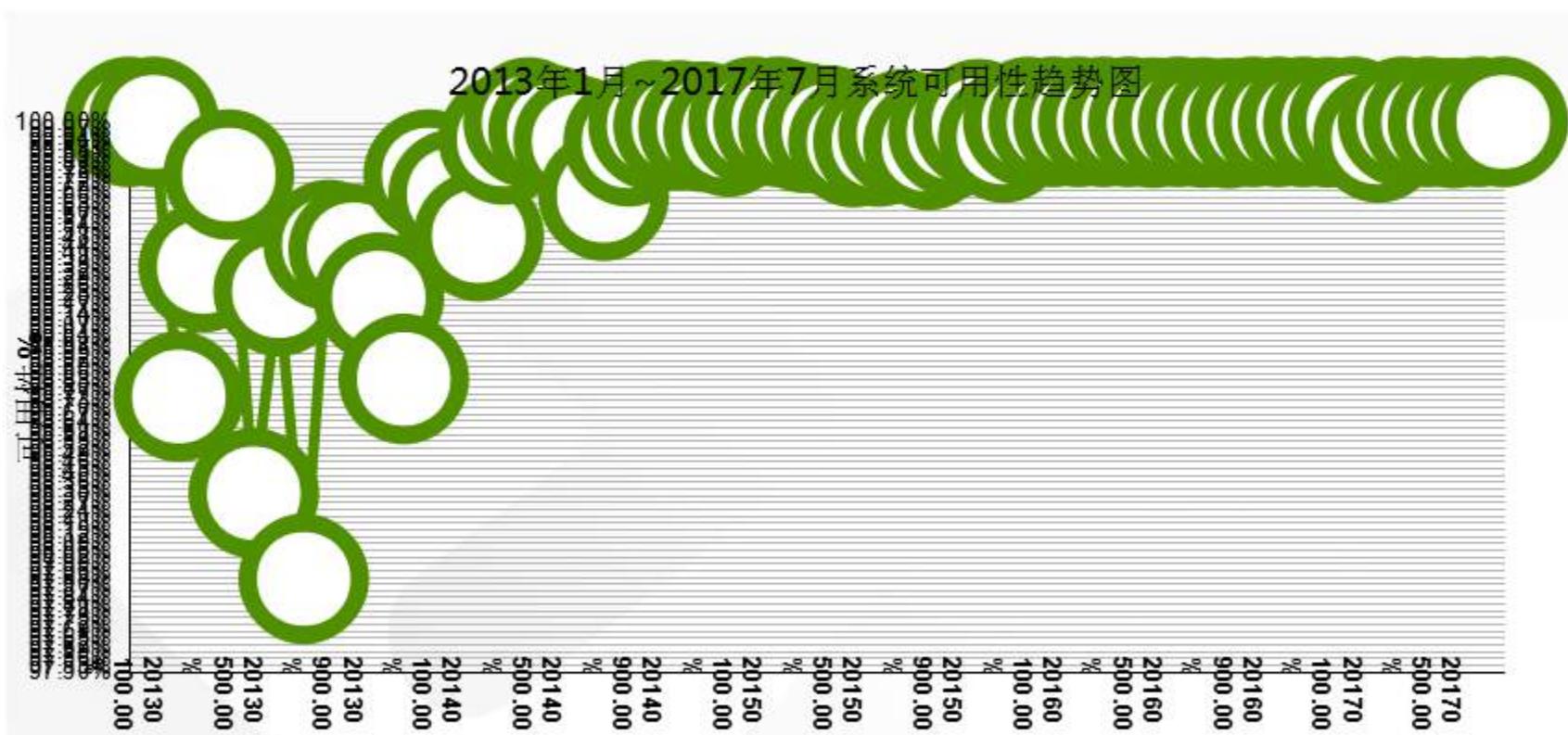
支付平台的运维效率得到了明显改善



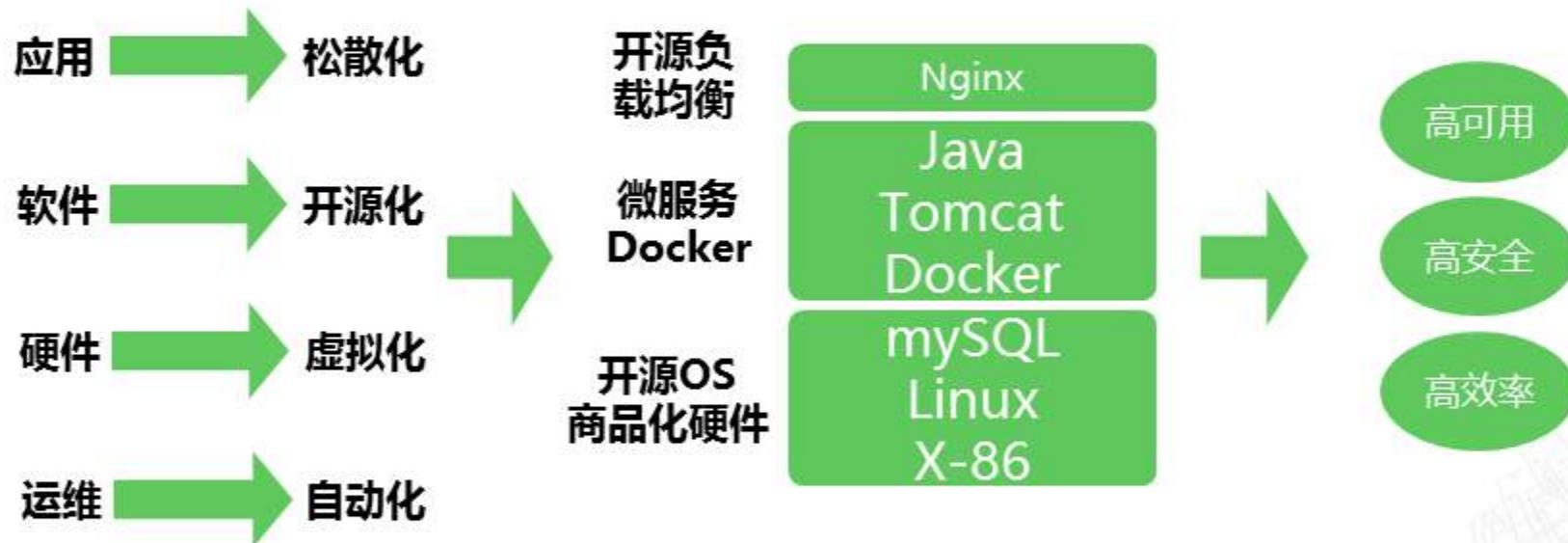
迁移前后



支付平台的可用性得到了大幅度的提高



易宝支付平台的发展趋势





2017 Software Architecture Summit

Thank You !



DevOps在链家上研的实践与探索

郭飞

2017年10月



目录

- 为什么用DevOps ?
- DevOps的探索
- 构建要点

为什么用DevOps ?

存在什么痛点？

- 新项目资源申请与发布过程繁杂
- 项目众多，各项目资源使用不均
- 资源分配、配置调整为人工操作，出错率高
- 项目动态伸缩困难
- 项目下线、资源不能及时回收

为什么用DevOps ?

什么是DevOps ?

- 一组工具、过程及系统的统称
- 是一种理念，提倡开发和运维之间的高度协作，打破部门墙
- 强调对应用全生命周期的管理
- 为什么是现在？
 - 1.微服务与容器技术的应用
 - 2.敏捷开发模式的普及
 - 3.系统架构越来越复杂

DevOps的探索

DevOps的探索

目标是什么？

- 能对资源的使用情况进行全局掌控，合理分配资源
- 对项目全生命周期进行线上化管理
- 尽可能减少人工操作，降低风险
- 减少跨部门的不必要沟通，提升持续交付效率
- 提升持续交付的健壮性与安全性

DevOps的探索

- 核心关键词

1

自动化

2

自助化

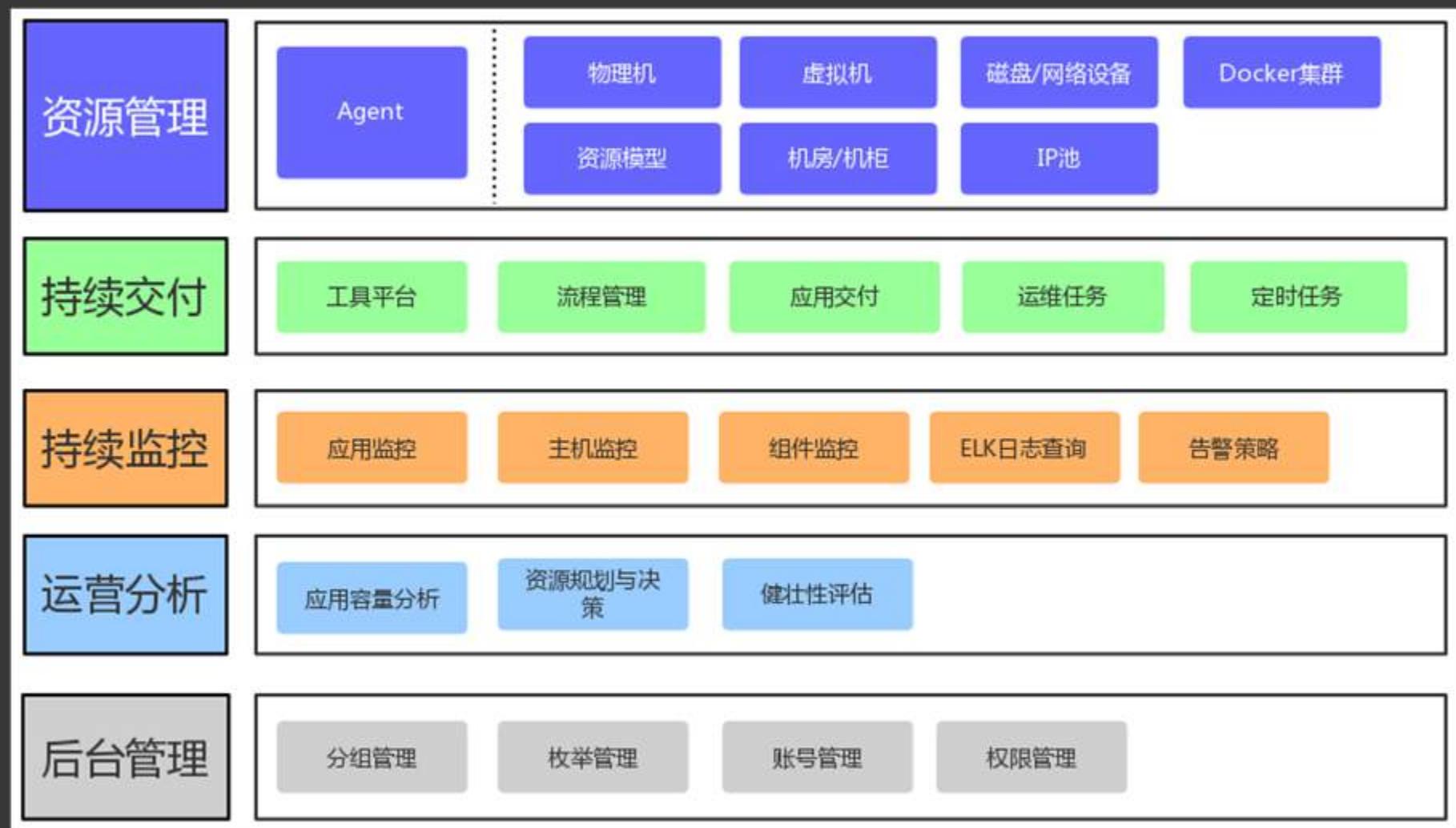
3

协作

4

智能

系统结构



一、 资源管理

主要资源

物理机

K8S集群

IP池

虚拟机

其他资源

机房

机柜

网络设备

存储设备

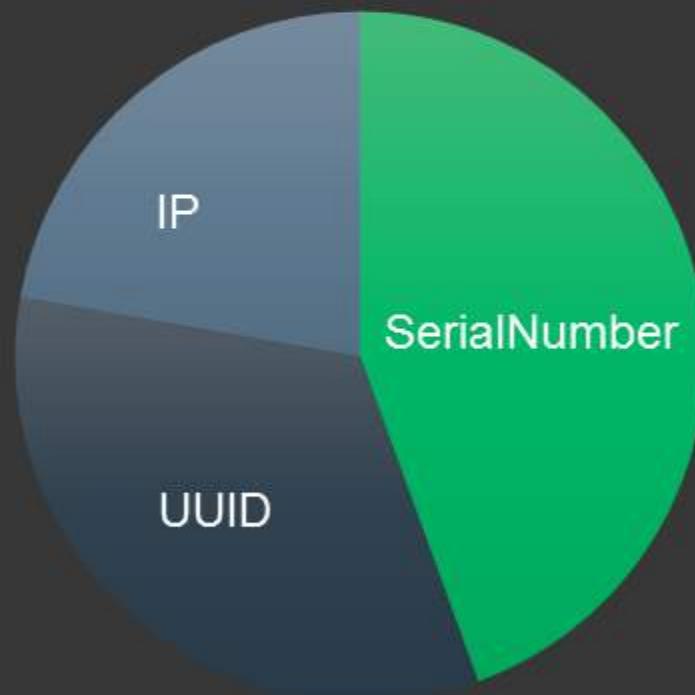
一、 资源管理

核心是自动化 (Agent)

- 上下架自动发现
- 资源配置变更自动同步
- 状态变更自动处理
- 运行进程自动采集
- 使用容量动态更新

一、资源管理

资源唯一标识设计

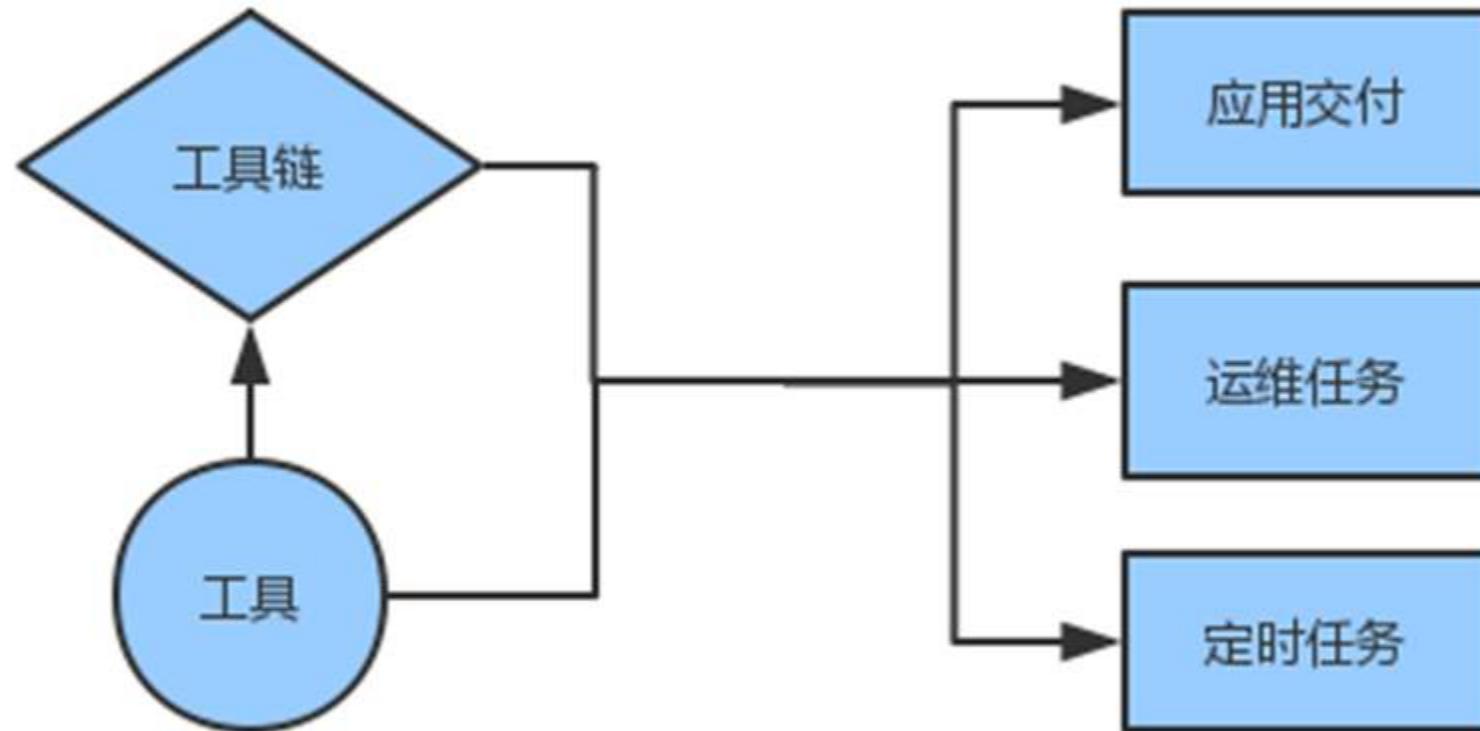


一、 资源管理

资源调度决策

- K8S调用API
- Xen调用API
- 资源申请的自动决策
- IP申请的自动规划

二、持续交付



二、持续交付

工具管理平台（在线脚本管理）

- 丰富的脚本类型
 - Shell、Python、PowerShell、Java
- 标准输入、输出
- 开放的工具平台
 - 系统内置
 - 面向全体工程师群体
 - 也对接外部api

二、持续交付

工具的安全控制

- 安全级别定义
- 加入审批机制
- 脚本安全检测

二、持续交付

流程管理

- 多个工具的组合
- 核心是流程执行器的设计

二、持续交付

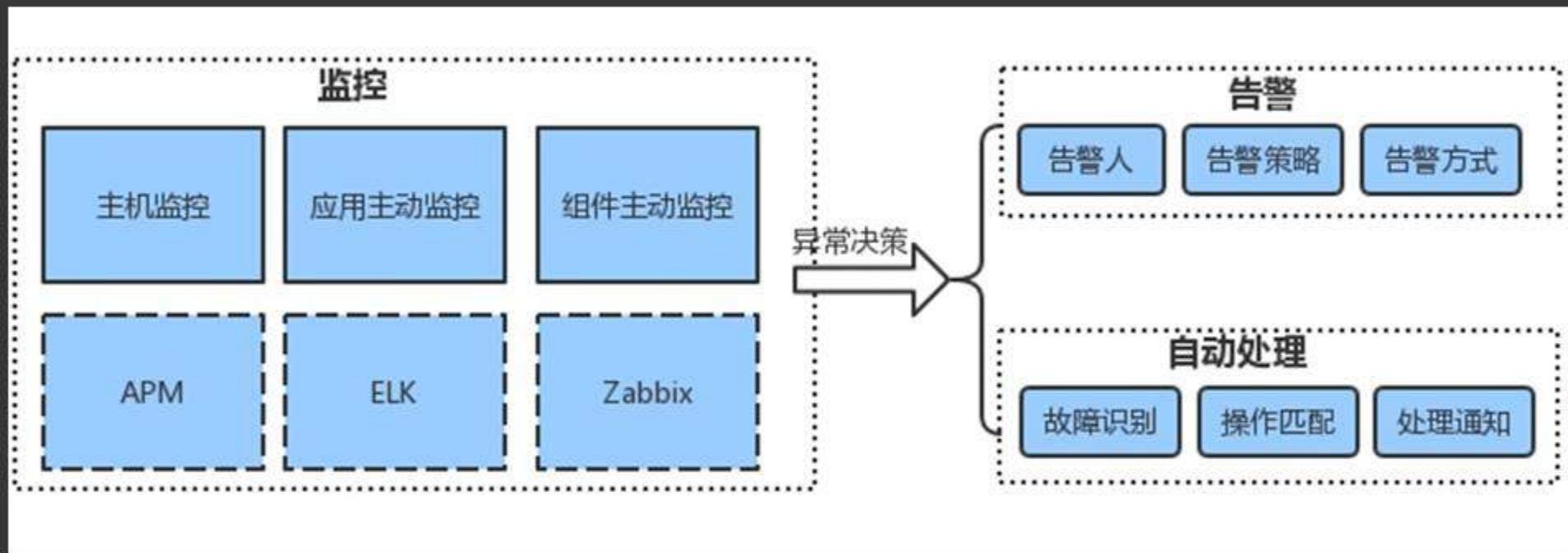
应用交付

- 场景定义
 - 初始化、部署、回滚、重启、下线、配置变更、节点伸缩。。。
 - 通过场景实现对应用全生命周期的管理
- 多环境下的不同流程
 - 非正式环境下自助化操作
 - 正式环境下的审批流设计
- 资源使用自动规划（如新系统资源分配、添加节点）
- 权限的开放与公司管理制度相匹配

二、持续交付

- 运维任务
 - 运维的常用操作线上化
- 定时任务

三、持续监控



三、持续监控

与原有监控系统是什么关系？

- 对原有监控报警的整合与补充
 - 对接zabbix、APM，让其纳入应用生命周期管理范畴
 - 提供更好、更准确的告警服务（告警人）
 - 整合各种组件监控与告警
- 侧重点不同
 - 关注点不同：更关注系统可用性，淡化故障排查过程
 - 结合APM系统达到自动故障定位、自动处理的目的

三、持续监控

- 应用监控
 - 规范化应用可用性检测接口
 - 主动心跳检测
- 主机监控
 - 通过Agent完成对主机的运行状态监测
- 组件监控
 - Redis、nginx、mysql、mongoDB、ES的心跳检查
- 告警策略
 - 多种告警方式（结合工具的使用）
 - 告警频率与阈值
 - 告警人与分组告警

三、持续监控

自动处理

- 异常类型定义
 - 故障：失去响应
 - 负载过高：CPU、内存、线程
- 异常处理流程匹配

四、运营分析

解决什么问题？

- 帮助解决资源的合理使用问题
- 帮助进行资源规划
- 帮助对应用进行健壮性评估

四、运营分析

1. 应用容量

- 帮助发现应用容量的瓶颈风险
- 帮助发现应用资源使用上的浪费
- 自动对应用资源进行伸缩
 - 如何做到？

四、运营分析

2. 资源规划与采购决策

- 资源剩余量预警
- 可用时长评估
- 业务组资源占用统计与费用测算
 - 资源使用是否合理？
 - 与业务价值是否符合？
- . . .

四、运营分析

3. 应用健壮性评估

- 部署成功率
- 故障率
- 故障时长
- 非正常发布频率

构建要点

构建要点

- 尽可能范围内自动化、自助化
- 零停机部署（工具与流程设计）
- 工具共享
- 从实践中学习
 - 忌大而全，与公司实际情况相结合
 - DevOps的完善也是持续迭代的过程
- 微服务与传统架构均可使用



连 接 每 个 家 的 故 事

2017 Software Architecture Summit

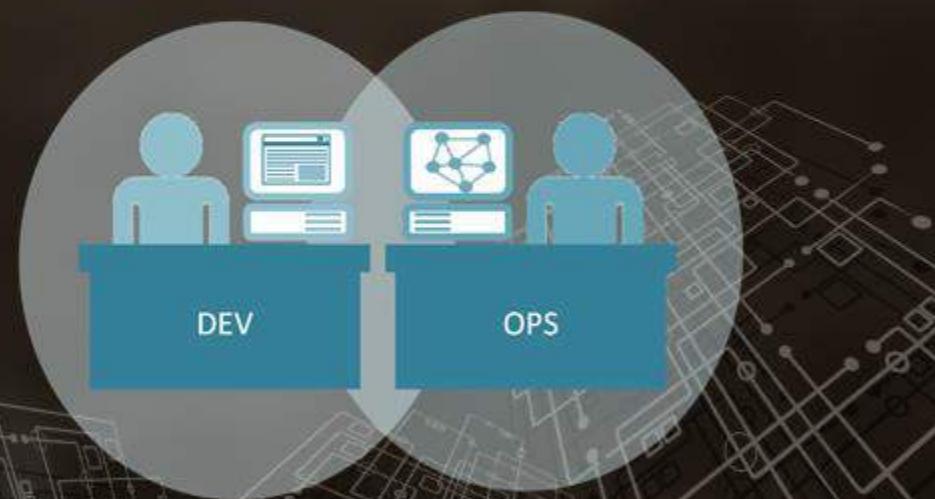
一站式软件交付: 世界五百强企业中的DevOps转型之道

马致杰 *George Malone*

JFrog 中国创始人 / CEO

前Marvell ARM服务器产品负责人

前Vungle亚洲总监



ORACLE®

CISCO™

ING 

| 个人介绍



马致杰 *George Maloney*
JFrog 中国创始人 / CEO

前Marvell ARM服务器
产品负责人

前Vungle亚洲总监

个人微信：



JFrog微信：



软件开发趋势

PROJECT EXECUTION METHODOLOGIES – THE CHANGE

WATERFALL 1985年



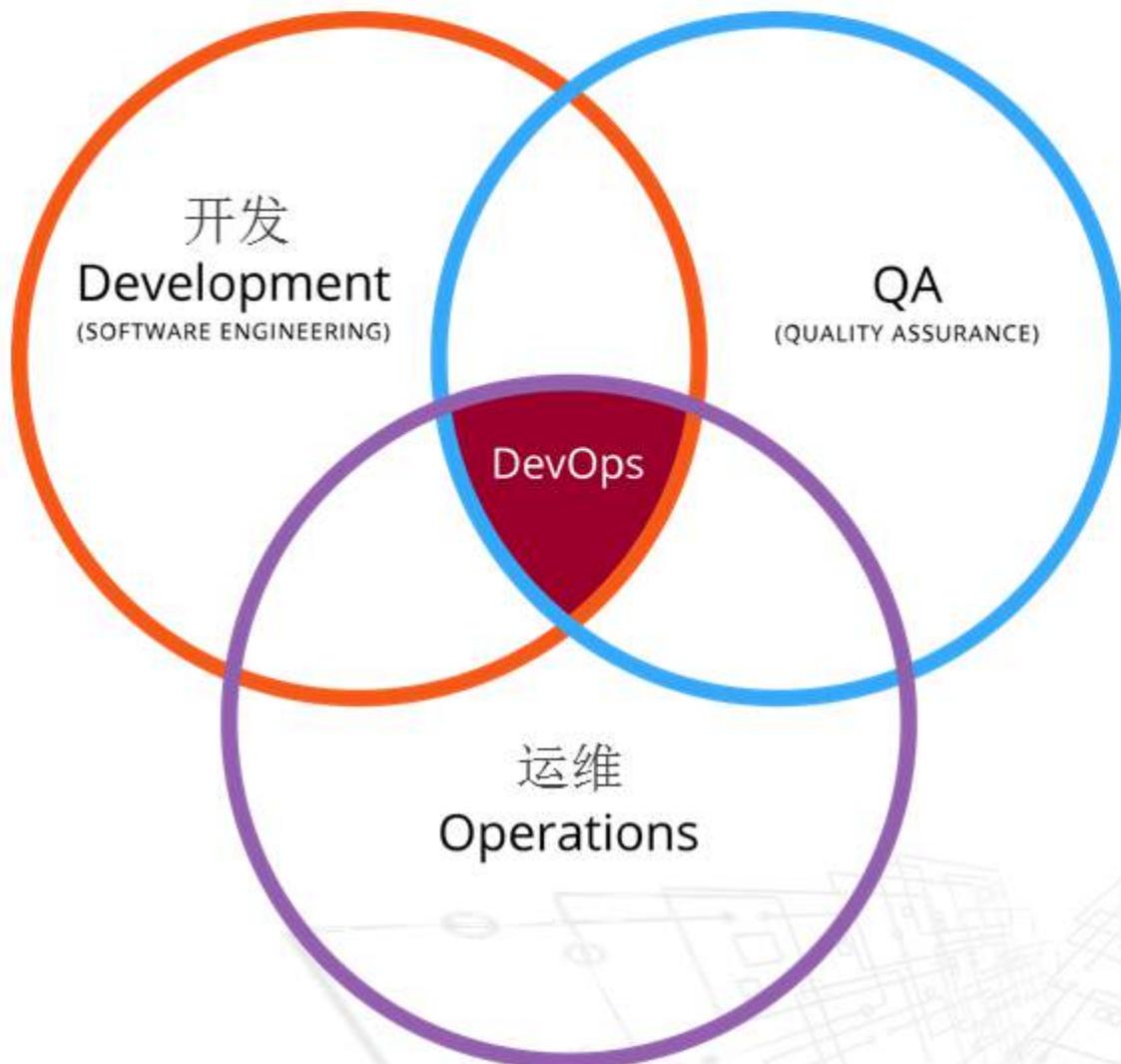
AGILE 2005年



DEVOPS 2009年



DevOps是什么？



DevOps 工具链



DevOps的发展现状

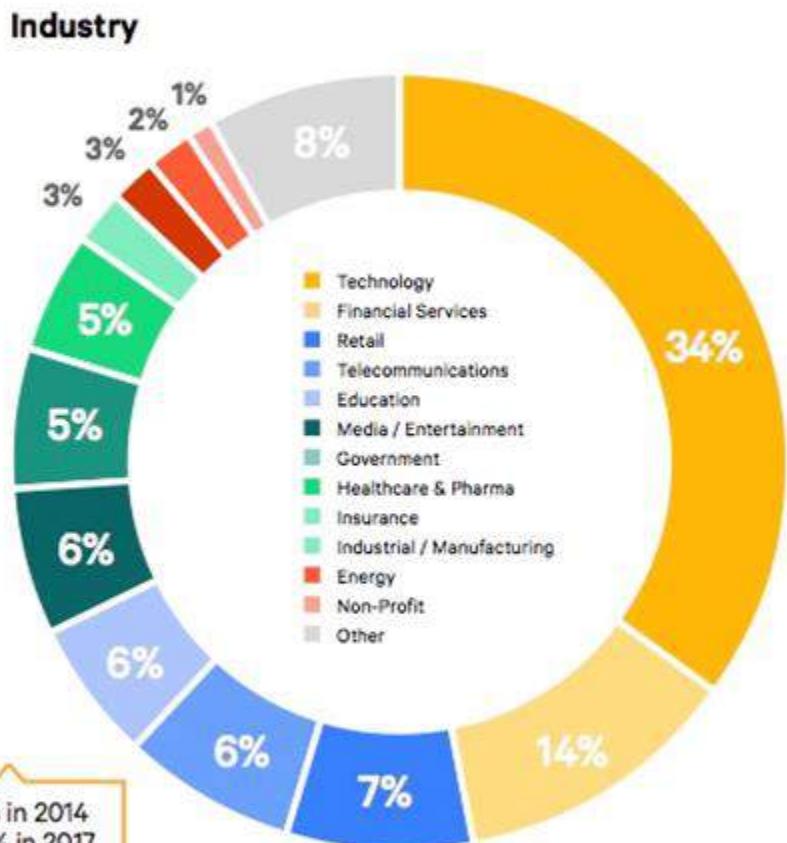
2017 DevOps 现状调查报告

Presented by:

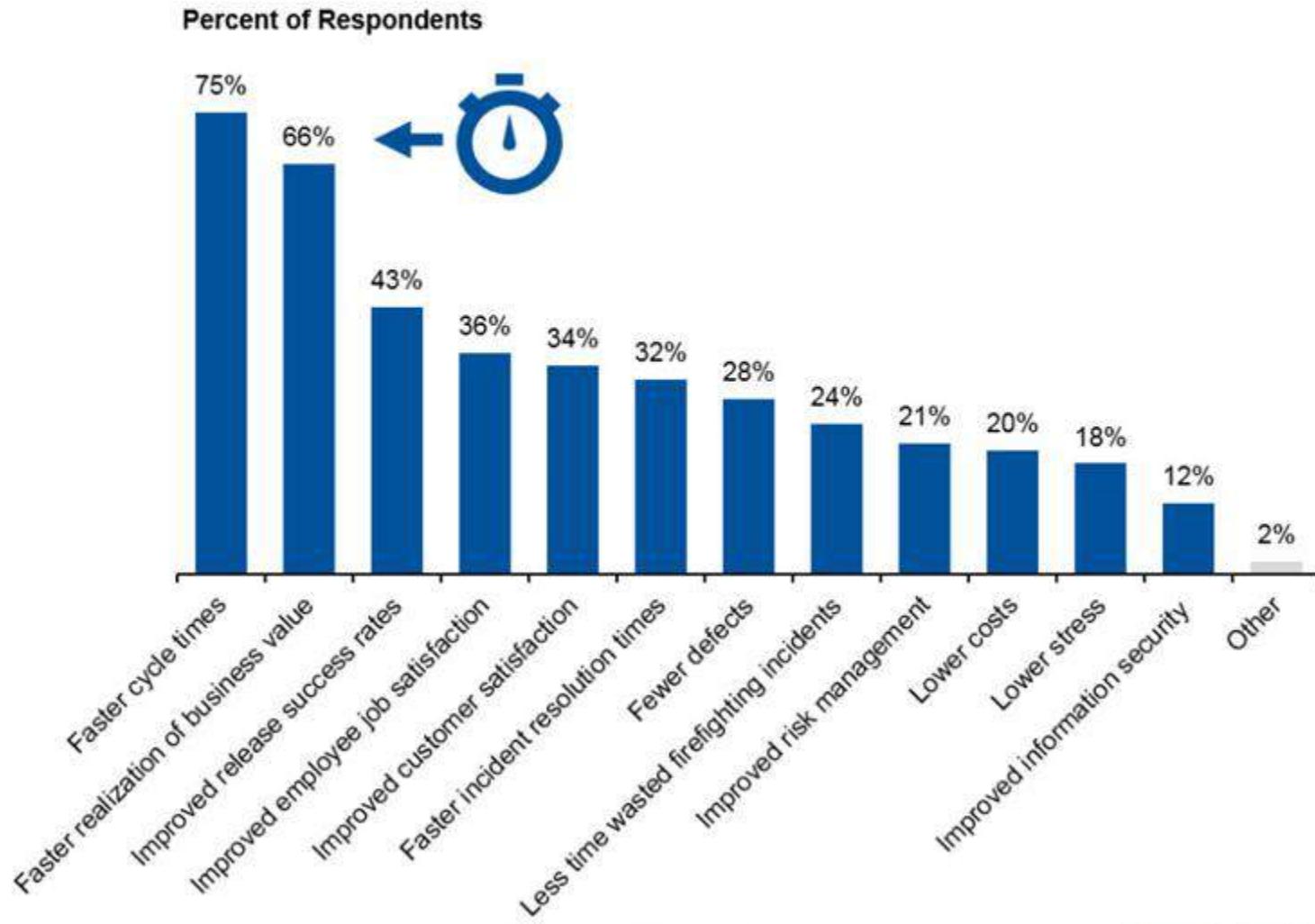


世界500强的DevOps员工
比前三年增长了两倍

DevOps teams increased from 16% in 2014 to 19% in 2015 to 22% in 2016 to 27% in 2017.



DevOps 的收益



DevOps全球开发者分布

Region

North America South America Europe + Russia Asia Africa Australia + New Zealand

54%

3%

27%

1%

10%

5%

全球最超前做DevOps的公司



自助式DevOps

自定义开源持续交付流水线

模版化交付任务
统一接入
跨团队数据分析

可视化报表
可扩展容器化管理

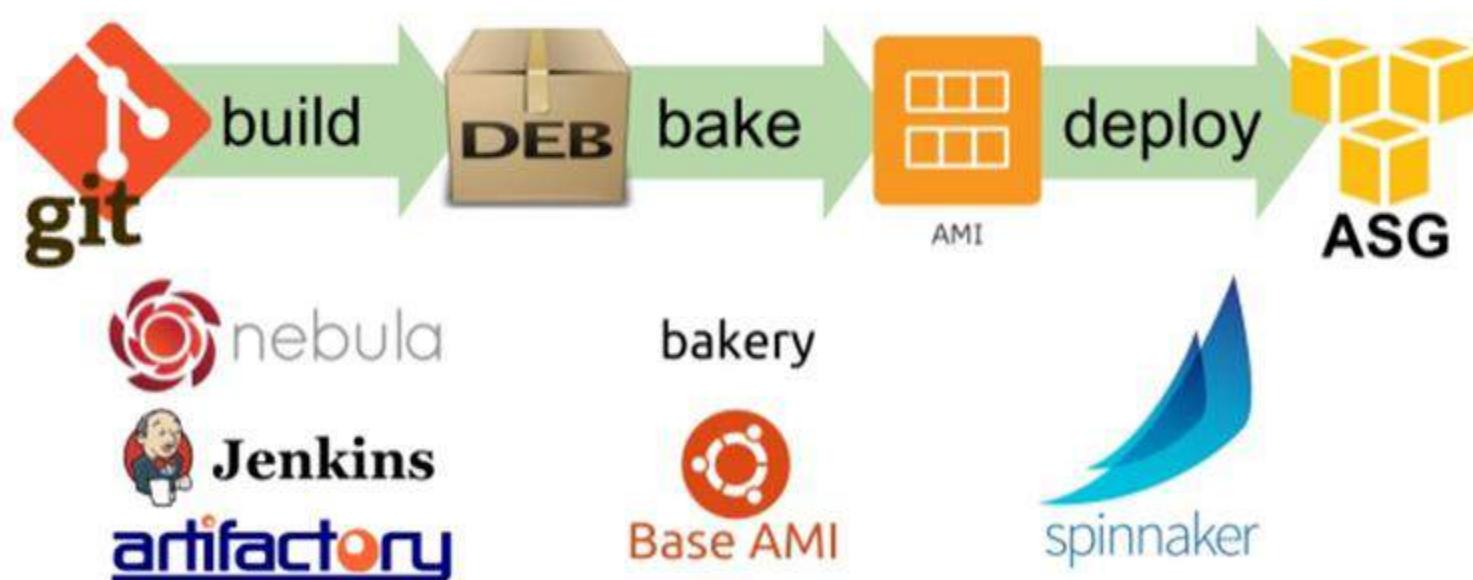




全球领先DevOps实践者

500种微服务

每天发布4000次



ORACLE®



2017 Software Architecture Summit

ORACLE®

40k
developers

59
countries

420k
customers

460
products

52
million artifacts

甲骨文的规模

Usage Stats

Upto 300

Tags per repository

1.5 million
Requests per day

100,000
Count of Folders

80 TB
Total Artifacts

4 TB
Size of Docker Registries

400,000
Count of Files

500,000
Item Count

150
Registries/Artifact Repositories

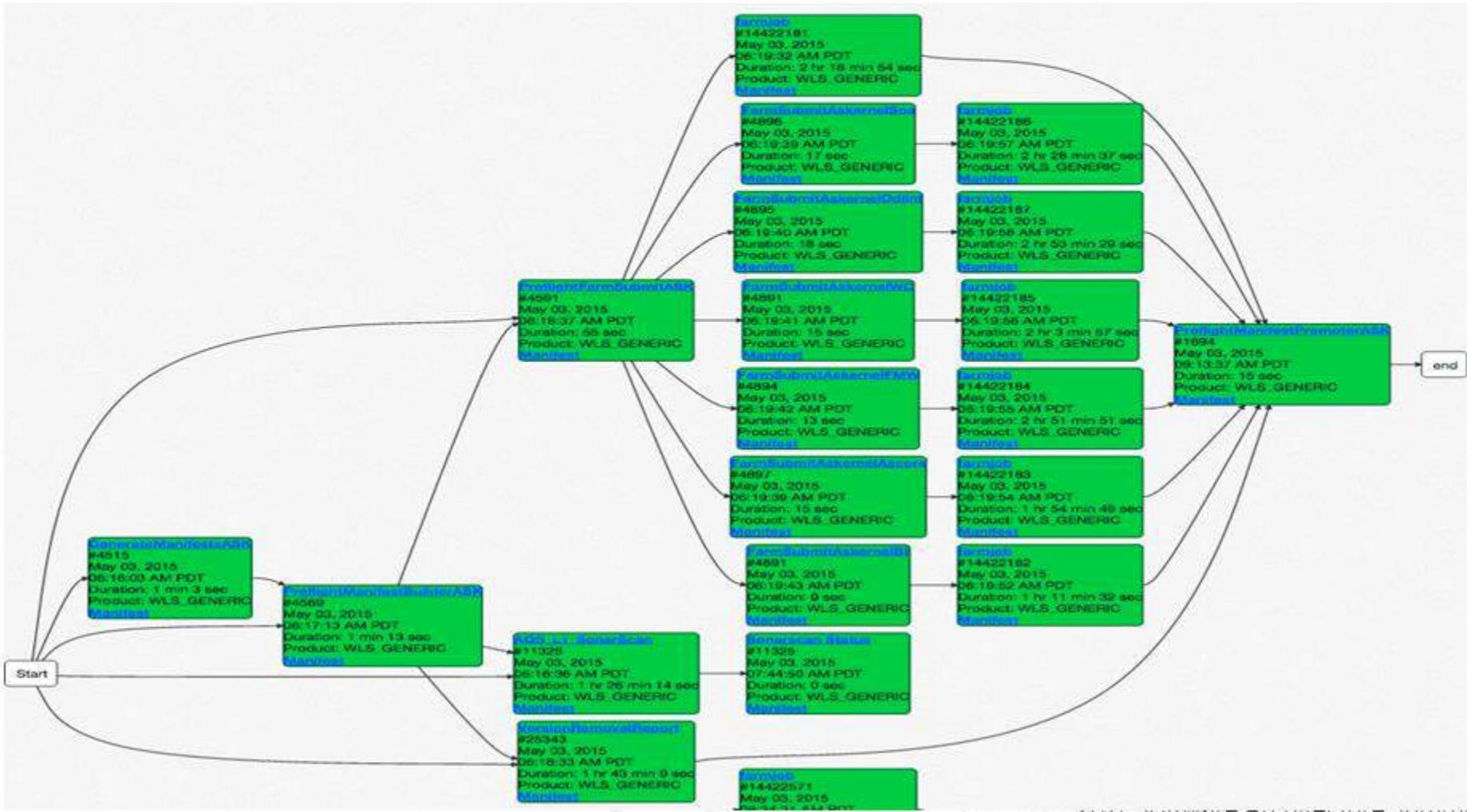
150

Docker Repositories or Images on v2 API

ORACLE®

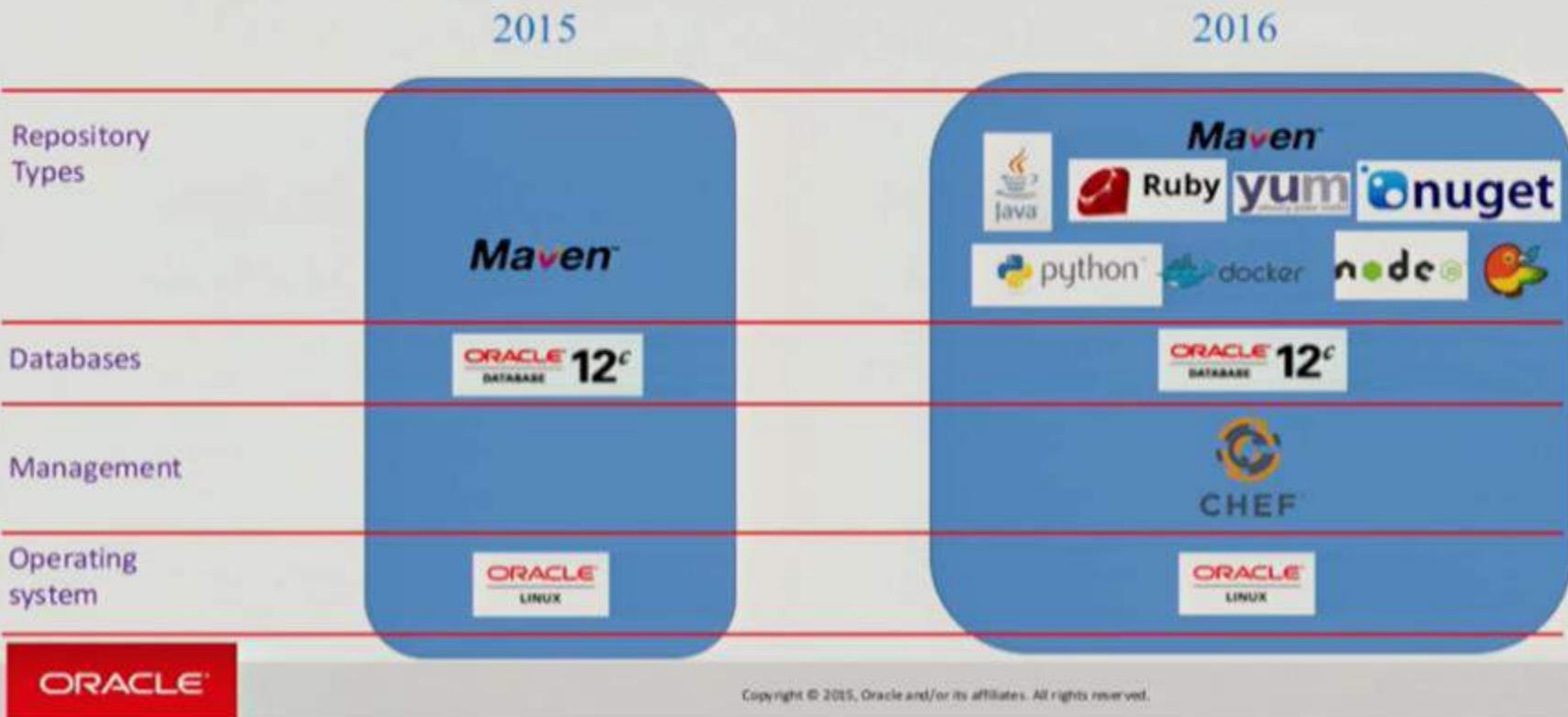
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

甲骨文复杂流水线



集中工件管理

Artifact Diversity



可视化流水线

OrchViewer

OrchID: 883 - InstanceID: 1820965

[View Old Visualization](#)

Instance Information

Duration: n/a

State: ERROR

Start Event:

```
{  
    CIM_TYPE: "SBE",  
    RELEASE_BRANCH:  
    "release-16-4-5",  
    EM_ORACLE_HOME:  
    "/u01/ops/em/core/12.1.0.1",  
    EMAGENT_IMAGE_FILE:  
    "http://bigfiles.us.oracle.com/agent/oracle-emagent-12.1.0.1-1.0.1.jar",  
    POOL: "pool2",  
    MGMT_VM_NAME:  
    "bdpmgmtpp11",  
    SPARK_ASSEMBLY: "spark-assembly-1.2.1"  
}
```

1820965

Graph View [Json View](#)

Success Failure Running Waiting Skipped

[Retrigger From](#) [Retrigger](#)

[Abort](#)

统计与报表

Syseng Report

Health Card Summary View

Last Run: 161213.0636.253

Last Promoted: 161213.0835.253

161213.2324.254

161213.0935.253

161212.2033.252

161212.0627.251

161211.1117.250

161210.2110.249

161210.0545.248

161209.1613.247

161209.0336.248

syseng.oraclecorp.com:8282/dashboard/index.html#/pipeline/main

12.2.1.3.0 | askernel:generic:12.2.1.3.0 | All | Go | BOOKMARK YOUR SELECTION

Standard Cloud Testing Pipeline (CPO) Dashboard Home Hits (242) Consumption Matrix Help

Health Card for 12.2.1.3.0 L2 (Last 7 Days Data)

Number of Runs

Total	Success	Failure
22	11	11

Time Metrics Task Wise (hh:mm:ss)

Pipeline Jobs Time

Task	Time
mbuilder	00:00:22
l2_mbuilder	00:00:43
l2_getting_helloworld	00:00:07
l2_somchange	00:00:03
l2_getting_helloworld	00:00:11
submit_config_jobs	00:00:12
l2_getting_tempfile	00:00:00
l2_non_getting_tempfile	01:06:13
mbpromoter	00:03:43
labelproduid	00:05:45
labelprodid	00:05:28
labelproduid	01:34:52

farmjobs Time

Task	Time
askernel_l2i	03:18:47
askernel_webcenter	03:38:11
askernel_asone	02:05:48
askernel_l2m	07:04:32
askernel_sso	03:28:05
askernel_odint	00:38:13
askernel	04:42:21
askernel_farmtools	01:15:00

Manifest for Early Pick Up

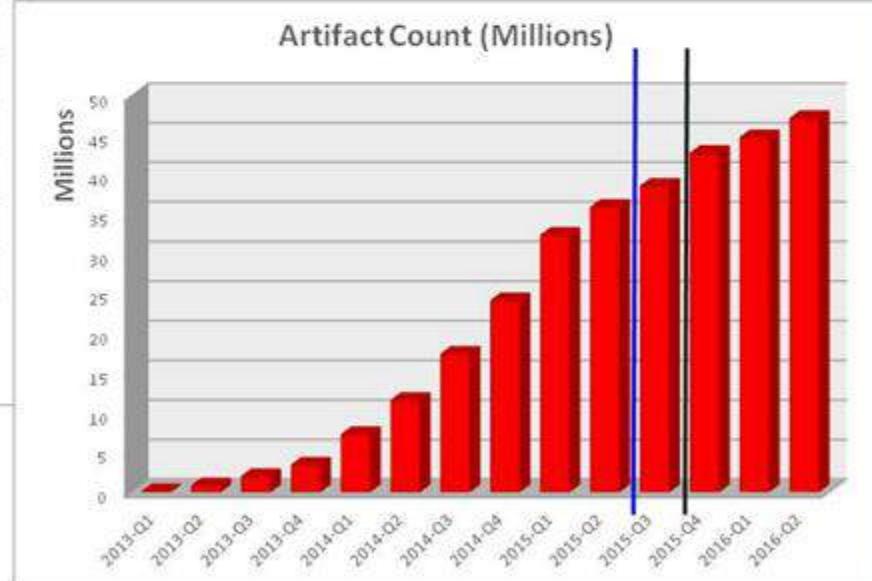
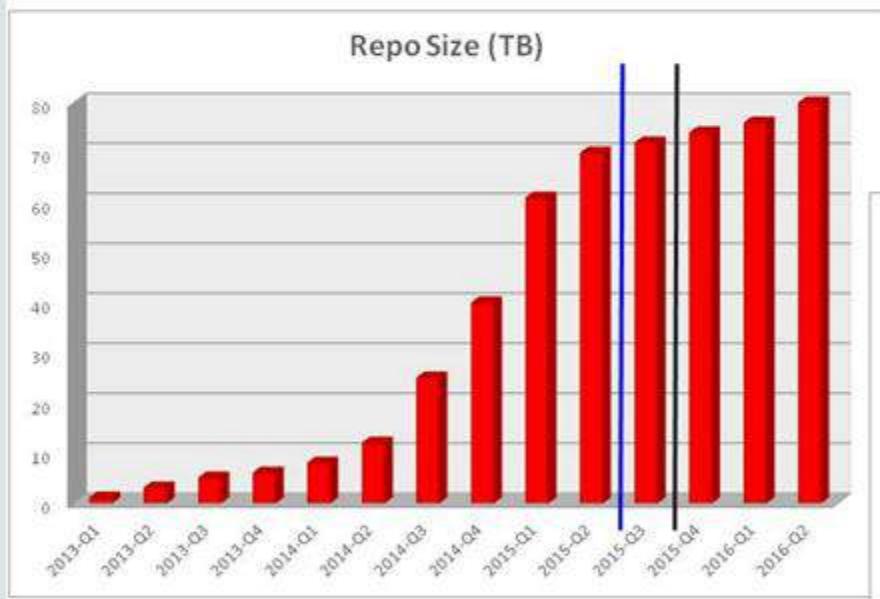
manifest.integration.release.askernel-manifest:12.2.1.3.0-161213.2324.254
ASINST_GENERIC_12.2.1.3.0-161213.0045
ECMLCLIENT_GENERIC_12.2.1.3.0-161213.1200.8254
EMGC_FMWC_GENERIC_12.2.1.3.0-161212.2201
FMWPLATFORM_FMWPROM_GENERIC_12.2.1.3.0-161213.0814.13088

Time Metrics(hh:mm:ss)

Average	Best Case	Worst Case
10:53:55	06:36:15	15:24:57

甲骨文平台扩容

Ramp Up





2017 Software Architecture Summit

金融巨头

2016年：¥100BN 收入

Global Continuous Delivery Journey

A worldwide presence in commercial and retail banking



Disclaimer: ING Bank does not have a commercial banking licence in the U.S. and therefore is not permitted to conduct commercial banking business in the U.S. Through its wholly-owned subsidiary, ING Financial Holdings Corporation, and its affiliates, it offers a full array of wholesale products such as commercial lending, corporate finance and a full range of FM products and services.



ING持续集成

Build - Continuous Integration



Code

Orchestrate

Build

Check

Store

ING持续交付

Deploy - Continuous Deployments



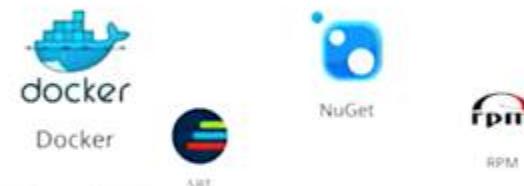
多语言开发

Artifactory

Numbers

#artifacts	3.500.000
#binaries	2.500.000
#local repos	1800
#remote repos	39
#users	4000
#groups	2600
#permissions	900
#storage	11.5 TB
#growth	1TB / month

Artifact types



Maven™

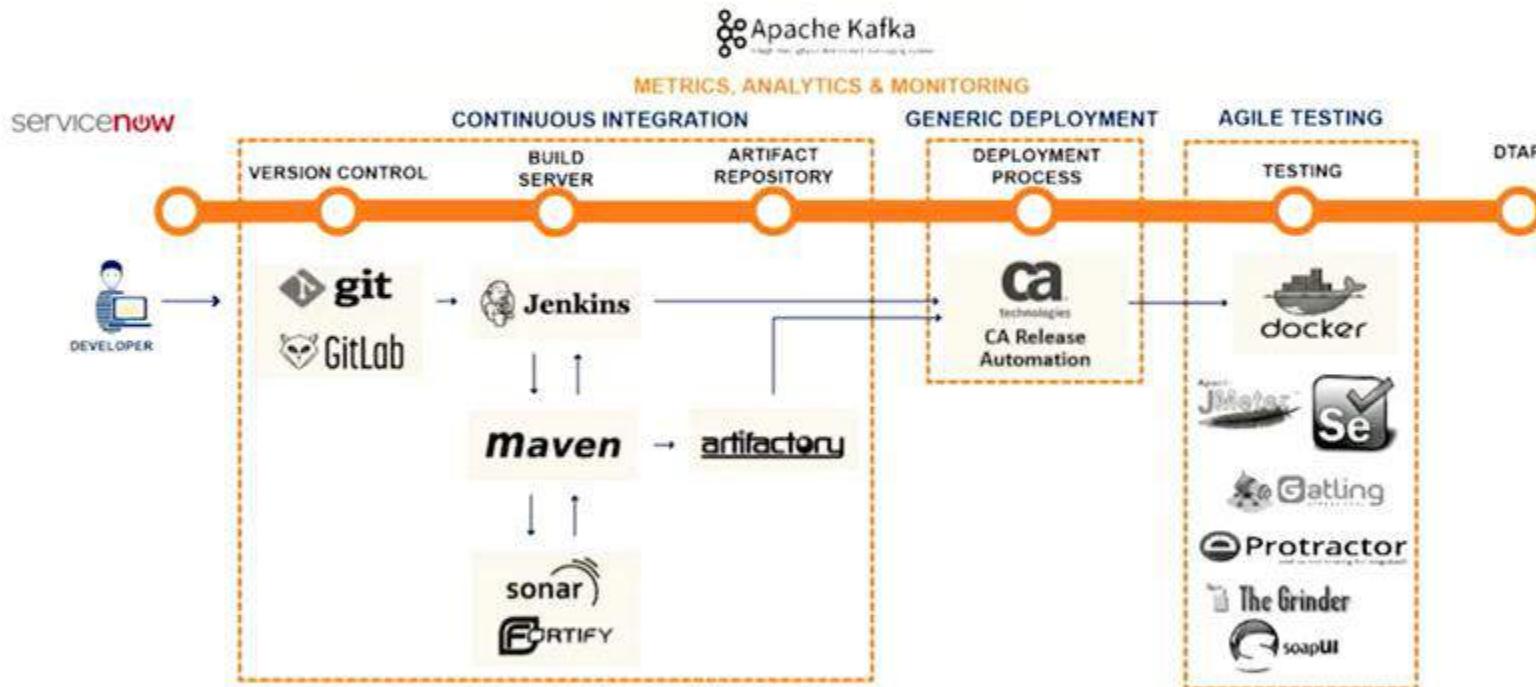


npm



Generic

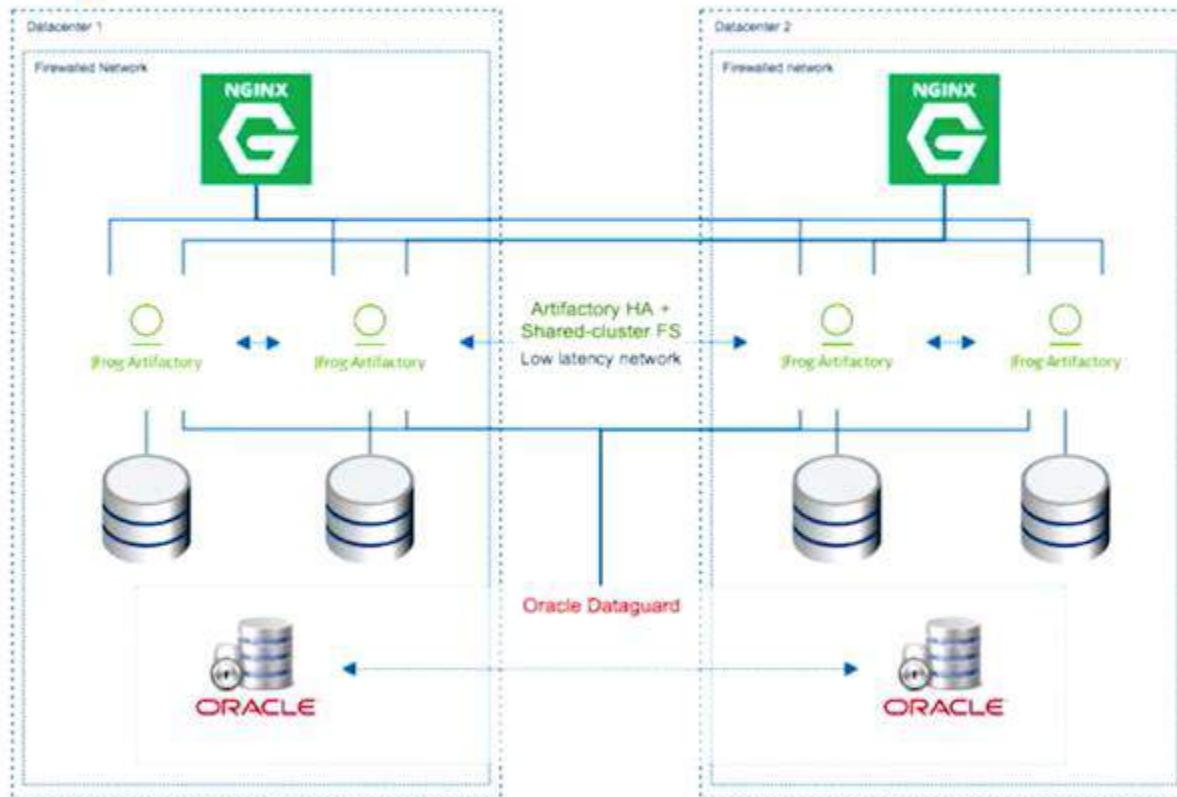
ING自定义流水线



One opinionated flow for all the (≈ 600) teams

高可用 / 容灾备份

Artifactory HA architecture end state





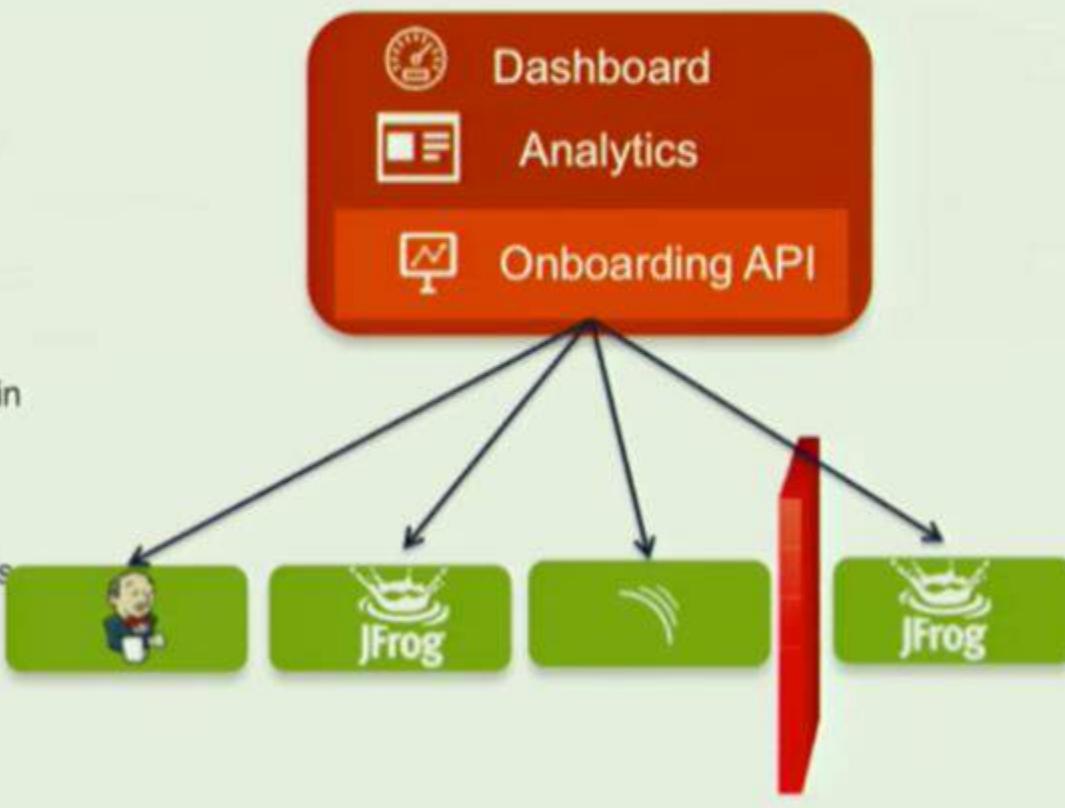
2017 Software Architecture Summit

思科 Account Manager

5个人DevOps团队支持4万+开发者

Account Management

- Account Orchestration
 - Automated onboarding through APIs
- Extensible plugin system
 - Jenkins, Artifactory, and SonarQube
 - Other systems can be added via plugin development (Artifactory-X)
- Unified dashboard
 - Manage customer identity across tools
- CI-Stats analytics
 - Number of requests
 - Number of uploads/downloads



思科 Account Manager

Account Management

pm/account/#/main/service/create/188

New Service

Limited fields can be edited at this time. Additional fields will be enabled in the next beta release of the account management service.

Artifactory Jenkins SonarQube

Team *
BMS Testing [BMS Testing]

Name *
swampup

Artifactory
Artifactory is a global service that distributes your artifacts all over the world. There is a single hostname that you should deploy to and read from, and it will resolve to the server that is physically closest to you.

Example config for your build management tool:

Maven Yum Docker

```
<repositories>
    <repository>
        <id>artifactory</id>
        <url>http://engci-maven.cisco.com/artifactory/-group</url>
    </repository>
</repositories>
<pluginRepositories>
    <pluginRepository>
        <id>artifactory</id>
```

Deployer User *
swampup-deployer

Deployer Password *
scj1c3d300Dedikt9

Users Hosted Remote
Virtual

+ Add Name

思科 Account Manager

Account Management

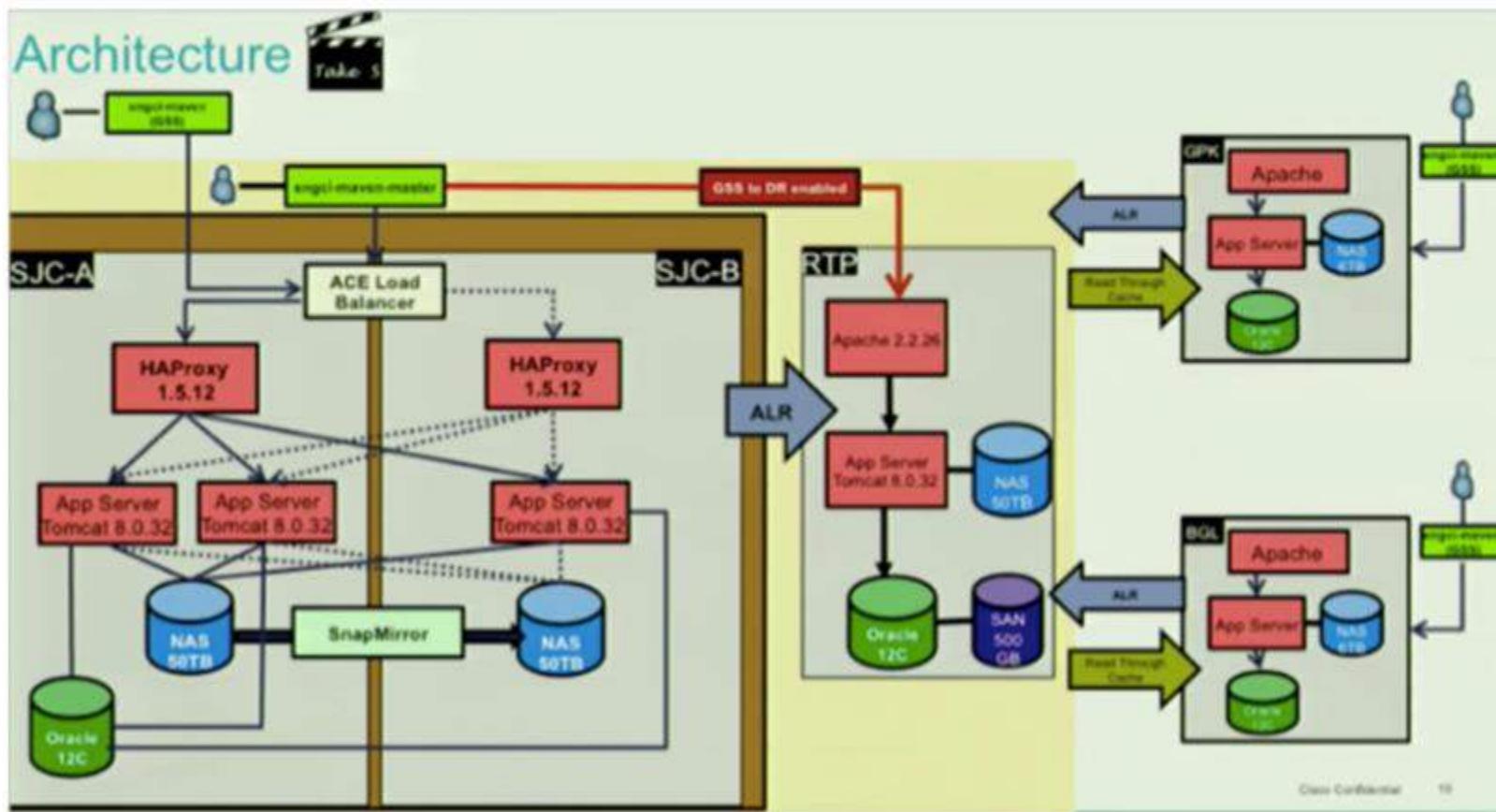
Audit Trail!!

Events for Service 16003

1

Started By	Status	Started	Completed	Log
prayyapp	SUCCESS	Apr 12, 2016 7:38:59 PM	Apr 12, 2016 7:41:28 PM	2016-04-12T14:40:37.451+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.ReplicationHandler: Setting replication settings to 'Satellite' for repo [frogs-demo-adapter] 2016-04-12T14:40:37.451+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.ReplicationHandler: Setting replication settings to 'Satellite' for repo [frogs-demo-deployer] 2016-04-12T14:40:40.126+05:30[INFO] com.claus.ei.accountmanagement.helpers.artifactory.ArtifactoryOperations: Creating [/frogs-demo] as satellite (http://frogs-claus.com/artifactory) 2016-04-12T14:40:41.394+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.SecurityHandler: Creating deployer user [/frogs-demo-deployer] 2016-04-12T14:40:42.541+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.SecurityHandler: Creating group [/frogs-demo-group] 2016-04-12T14:40:42.540+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.SecurityHandler: Adding users [admin, prayyapp] to group [/frogs-demo-group] 2016-04-12T14:40:43.203+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.SecurityHandler: Setting permission target [/frogs-demo-right] 2016-04-12T14:40:43.571+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.strategies.HarmoStrategies: Creating local Nexus repos [/frogs-demo-1a] 2016-04-12T14:40:44.754+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.strategies.HarmoStrategies: Creating local Nexus repos [/frogs-demo-1a] 2016-04-12T14:40:45.52.188+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.strategies.HarmoStrategies: Creating virtual Nexus repos [/frogs-demo-q] 2016-04-12T14:40:45.52.291+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.strategies.HarmoStrategies: Creating virtual remote cache Nexus repos [frogs-demo-group-qm900ta] 2016-04-12T14:41:07.212+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.SecurityHandler: Adding repos [/frogs-demo-thirdparty, [/frogs-demo-group, [/frogs-demo-adapter]]] to permission target [/frogs-demo-right] 2016-04-12T14:41:08.009+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.ReplicationHandler: Setting replication settings to 'Satellite' for repo [frogs-demo-adapter] 2016-04-12T14:41:29.319+05:30[TRACE] com.claus.ei.accountmanagement.helpers.artifactory.ReplicationHandler: Setting replication settings to 'Satellite' for repo [frogs-demo-adapter] 2016-04-12T14:41:29.354+05:30[INFO] com.claus.ei.accountmanagement.provisioning.serviceorchestrator: CREATE of com.claus.ei.model.registry.ArtifactoryService (100)

思科异地DevOps

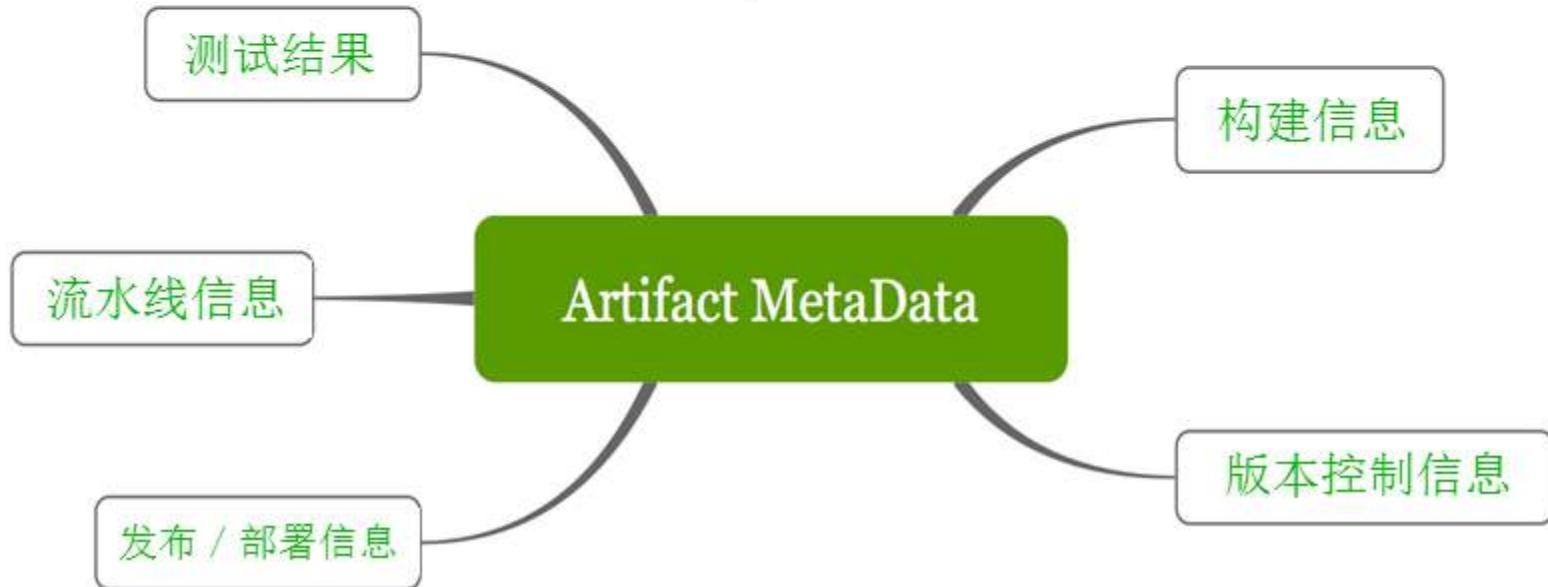


思科元数据

思科通过一个命令行能找到上亿个包里面的目标包

JFrog Artifactory制品库 + AQL查询语言

- 所有的软件包加上多样化DevOps各阶段的信息
- 根据**Metadata**的信息来实现自动化的快速发布
 - 提高发布产品的自信





思科元数据

JFrog Artifactory

Artifact Repository Browser

multi3-6.238.war

Properties

Property	Value(s)
qa	done
passRate	1
platforms	osx, win386, win64, debian
vcs.revision	251ab9ba9bea19fdcaa48622b43c8fe2da155df17
qaType	unitTest,sonar
testType	junit
build.timestamp	1502195316210
build.name	mvn-pro-webinar
build.number	80
chef.env	TestEnv1,ProdEnv1
env	TESTING
buildStatus	RC
sonarUrl	http://47.93.114.82:9000/dashboard/index/jfrog/multi3
sonarIssue	2

Actions

Set Me Up **Deploy**

Tree Simple Q

- npm-virtual
- pip-virtual
- conan-local
- docker-dev-local
- docker-dev-local2
- docker-local
- docker-prod-local
- docker-release-local
- docker-release-local2
- docker-test-local2
- docker-webinar-local
- generic-local
- git-lfs-local
- gradle-dev-local
- libs-release-local
- libs-snapshot-local
- libs-test-local
- maven-local-2
- maven-releases-local
- org
- Trash Can

Home **Artifacts** **Search** **Builds** **Admin**

ARTIFACTORY Enterprise
5.8.0 rev 90049
Licensed to Waterway
© Copyright 2017 JFrog Ltd.

JFrog DevOps

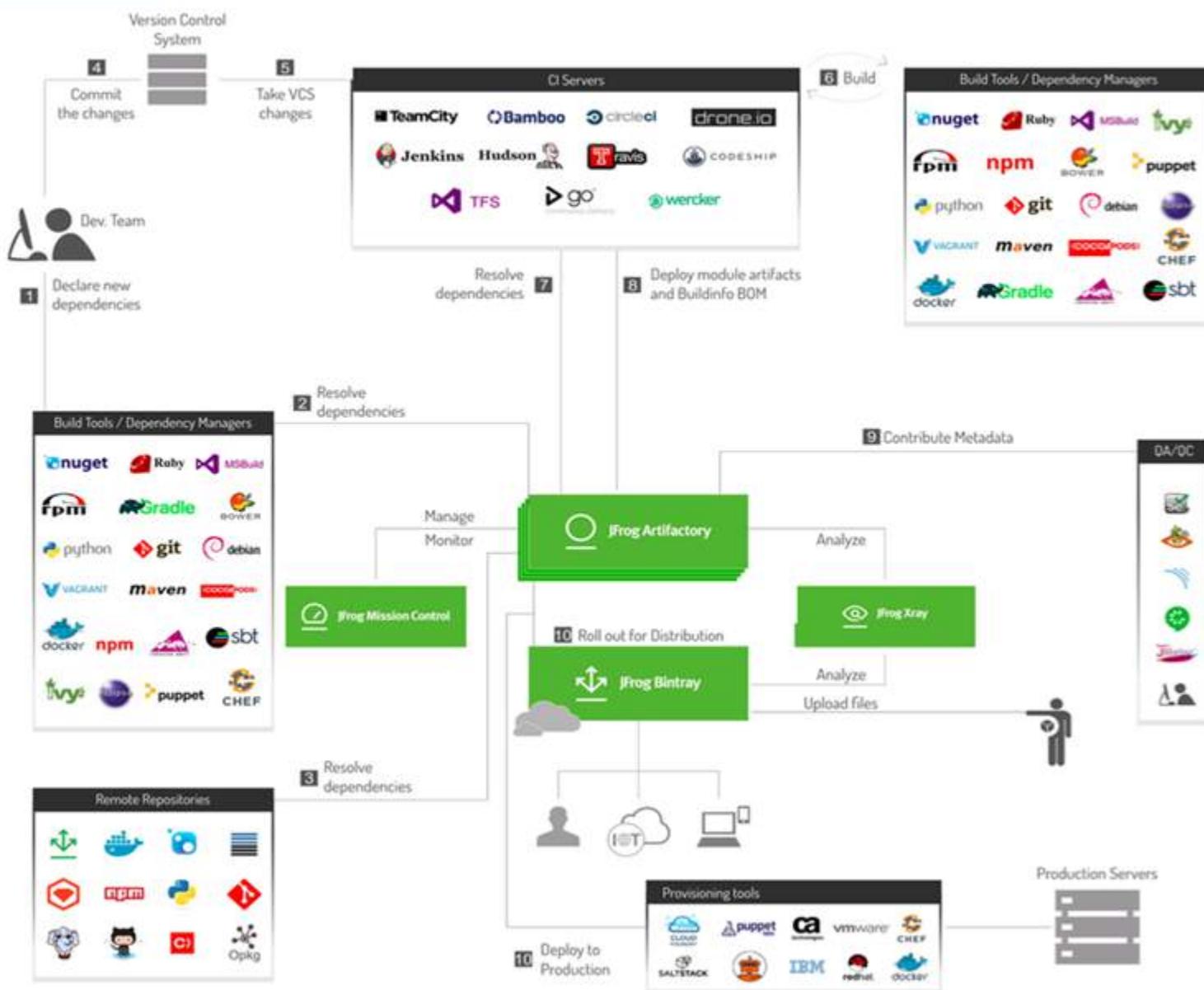


2017 Software Architecture Summit



JFrog Artifactory

- 多语言仓库
- 统一CI/CD平台
- 收集测试结果
- 设置质量关卡
- 包升级，移动
- 统计数据，进行评估
- 高可用，容灾备份



JFrog DevOps



互联网 & 软件

Google **Tencent** 腾讯

vmware



ORACLE

ca
technologies

YAHOO!

JD.com 京东

4,000
客户

科技 & 电信

HUAWEI

LinkedIn



CISCO

CHEF



ERICSSON

银行 & 金融

BARCLAYS

Bloomberg

JPMorganCh

FIS

CREDIT SUISSE

WELLS FARGO

ING DIRECT

+100
New Logos
a month

工程 & 航空

ANSYS

GM

NASA

AIRBUS

BOSCH

VC

GE

HALLIBURTON

Raytheon

零售 & 消费

COSTCO
WHOLESALE

NETFLIX

GAP



macy's

TESLA

RIOT
GAMES

95%
续费率

教育 & 研究

MIT

UNIVERSITY of
WASHINGTON

Yale

OHIO
UNIVERSITY

Stanford
University



DevOpsOne

全开源自定义流水线



Jenkins



JUnit



maven



JMeter™



sonarqube



ANSIBLE



dcld dcloud paas.devopsone.

DevOpsOne 总览 构建 包管理 部署 流水线 报告 资源池 admin

springboot-demo

1 开始 1个参数 + 2 stage 3个任务 + 3 结束 1个Stage

全局参数 源代码 本流水线共有：
添加参数 maven构建 1个Stage
+ 代码扫描 1个参数
+ 添加任务 3个任务

保存 取消



Sonarcube Jenkins Artifactory Jira Bitbucket

MyApp 2017-03-21 Normal

Old Defects

8
open defects over 90 days
QTES: 3 months ago

Old Stories

10
open stories over 90 days
QTES: 3 months ago

Stories In Multiple Sprints

9
open stories over 2 sprints
QTES: 3 months ago

Defects Density

0
open defects over attempted storypoints
QTES: 3 months ago

Velocity

Source: QTES Updated: 4 months ago

Sprint	Planned	Actual
S0	8	8
S1	22	22
S2	10	10
S3	22	22
S4	10	10
S5	10	10
S6	12	12
S7	10	10
S8	8	8
S9	20	50
S10	15	15
S11	5	5

Story points

Current Sprint

sprints

Time

Source: QTES Updated: 4 months ago

Date	Planned	Actual	MVP
2017-03-15	20	20	180
2017-03-16	40	40	180
2017-03-17	60	60	180
2017-03-18	80	80	180
2017-03-19	100	100	180
2017-03-20	110	110	180
2017-03-21	160	160	180

Story points

Planned

Actual

MVP

Projected End

Current Sprint

sprints

Stories Inbound Outbound

Source: QTES Updated: 4 months ago

Date	Inbound	Outbound
2017-03-15	10	10
2017-03-16	10	10
2017-03-17	10	10
2017-03-18	10	10
2017-03-19	10	10
2017-03-20	10	10
2017-03-21	10	10

Num Of Stories

Inbound

Outbound

date

多谢！

个人微信：



马致杰 *George Malone*
JFrog 中国创始人 / CEO

www.jfrogchina.com

george@jfrogchina.com

JFrog微信：



2017 Software Architecture Summit

360展示广告系统技术架构实践

陈东

360商业产品部技术总监，负责展示广告系统技术架构，主导了360 DSP和ADX平台的创建。

于北京大学获得学士和硕士学位。在加入360前，曾在微软中国广告技术中心，聚效广告担任技术研发和管理等职位，积累了10年广告行业服务端技术架构经验。



| 内容摘要

360展示广告系统介绍

系统规模扩大的挑战

高可用性管理

总结





MARKETER 广告主

PUBLISHER 媒体
CONSUMER 消费者

Super Platforms | 综合大型投放平台



DSP & DSPAN (DSP + Ad Network) | 程序化广告采购方



Ad Exchange & SSP | 程序化广告供应方



Data Supplier & Data Management



Trading Desk & Tech

采购交易平台及技术



Verification

广告验证



Measurement & Analytics

监测分析工具



Programmatic TV | 程序化电视广告



Programmatic DOOH | 程序化数字户外广告



| 360展示广告系统介绍

335亿日均流量 | 8大定向模式 | 80万标签总量 | 1000万投放设置



360站内独家资源

PC



360导航



360搜索

站外优质合作媒体



爱奇艺

精选数百家主流媒体

移动



国内数千款TOP应用

PC

285亿

移动

50亿

=

一站投放

335+亿

每秒广告请求>40万

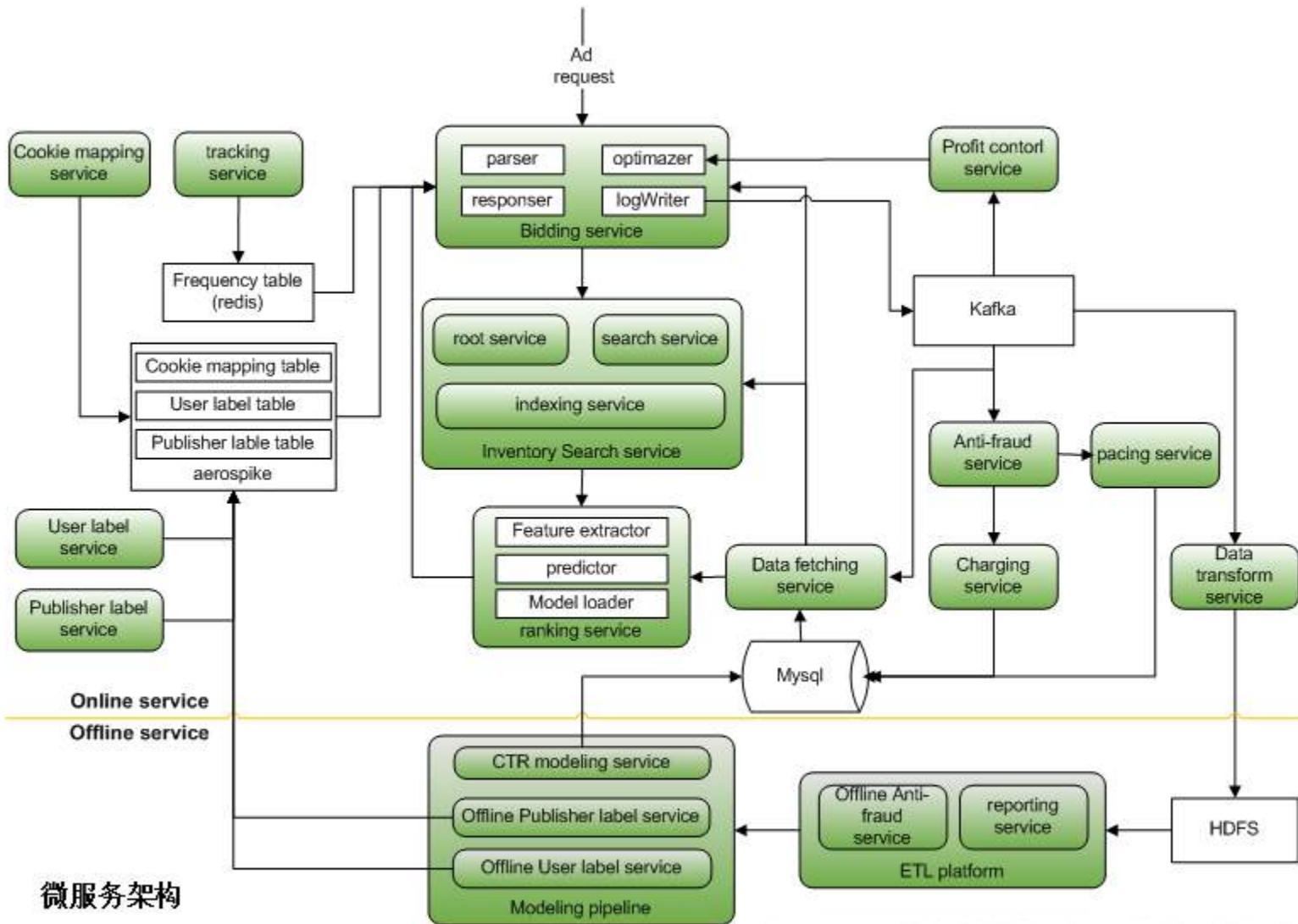
平均后端延迟<10 MS

| 360展示广告系统介绍

- 技术栈

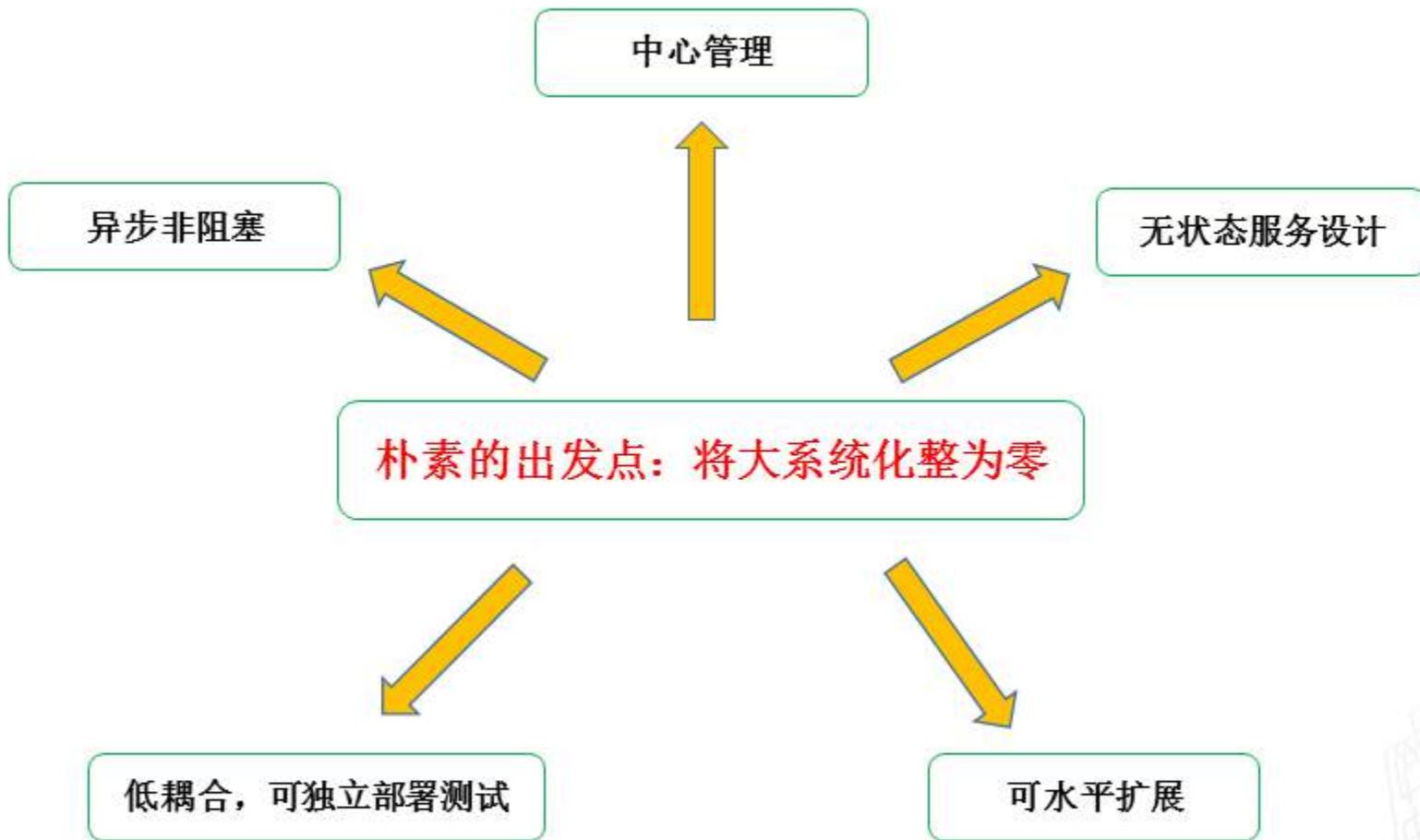


360点睛实效平台架构介绍



微服务架构

微服务架构



| 规模扩大的挑战

在40万的qps压力下，要在10ms内，从千万个广告投放中挑选出合适的广告

流量迅速增长

2014-2017增长**3倍+**



成本迅速增长

增长**? 倍**

投放量迅速增长

2014-2017增长**10倍+**

简单粗暴加机器会带来成本迅速增长需要更高效的解决方案

| 规模扩大的挑战-合理的拆分

- 合理拆分是分布式的核心
 - 业务拆分
 - 水平拆分
 - 垂直拆分
- 架构设计的重点：基于业务特点给出合适的设计方案

业务拆分+垂直拆分

VS

水平拆分



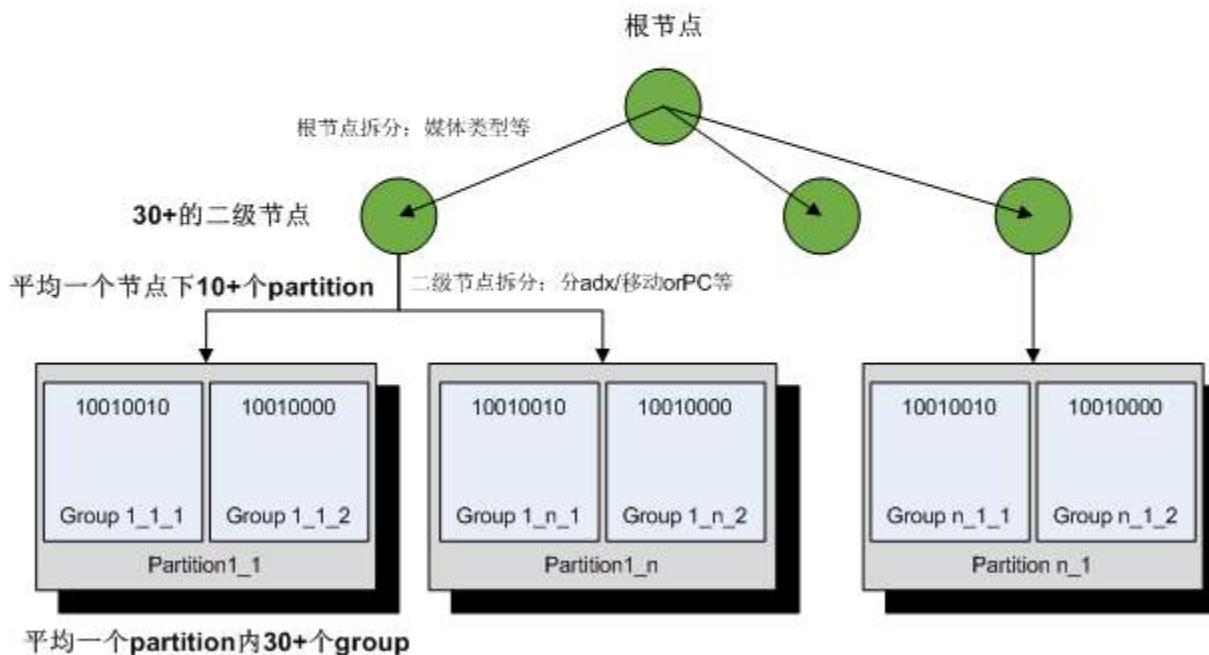
| 规模扩大的挑战-合理的拆分

- 展示广告业务的特点：
 - 广告投放指定的部分投放条件和媒体流量直接相关。
 - 例子1：投放指定了媒体类型（新闻网站/视频网站）等
 - 例子2：投放指定了媒体渠道（google adx/360MAX adx）
 - 例子3：投放指定了PC还是移动的流量
- 展示广告业务的拆分设计思路：
 - 根据流量特点，将投放索引进行拆分
 - 当特定流量发起广告请求，能直接走到对应的索引分支进行检索，而不是在全量投放数据上进行检索

| 规模扩大的挑战-合理的拆分

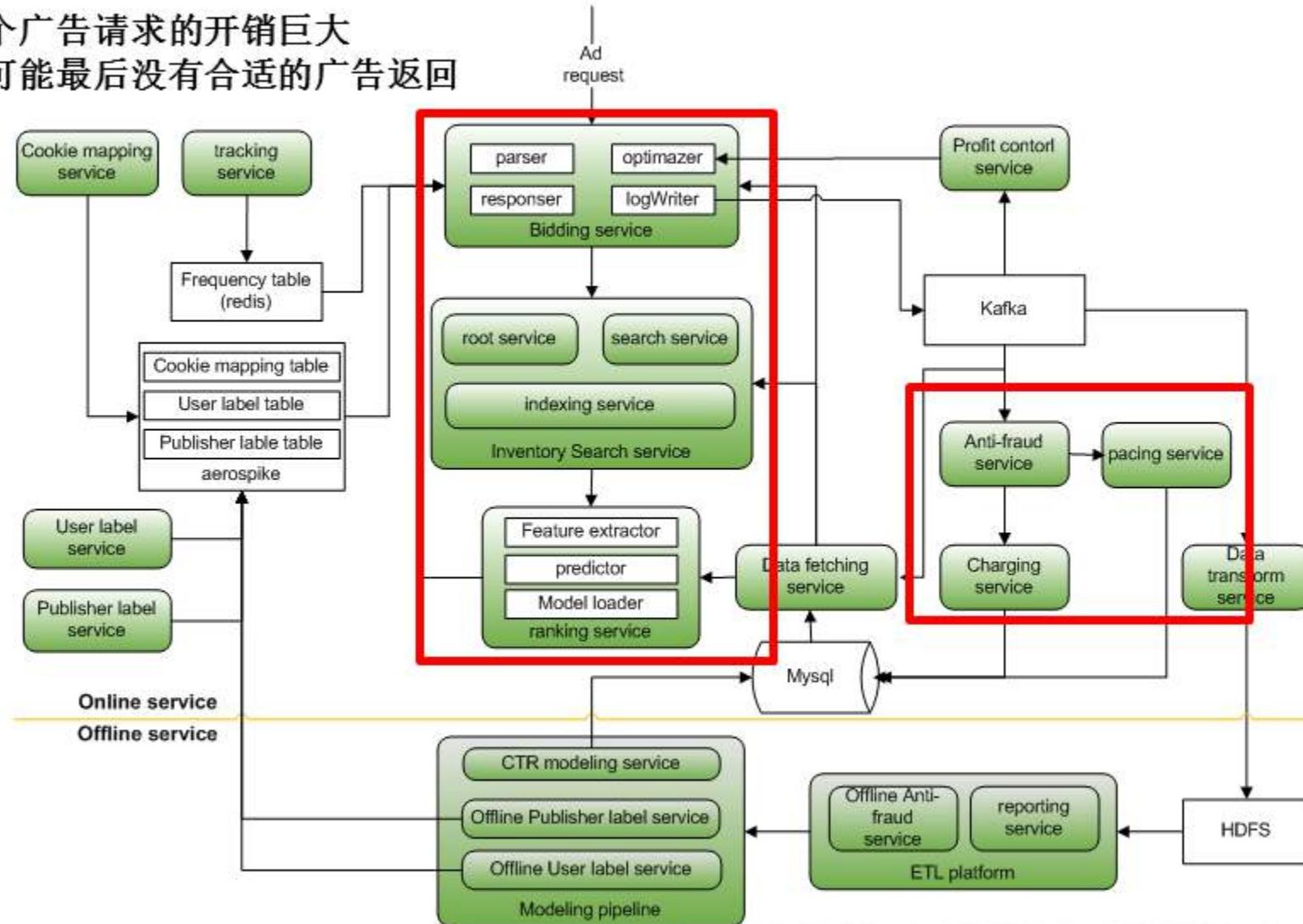
- 展示广告投放拆分方案：

- 根据投放和流量的关系，基于区分度做partition
- 允许同一个投放出现在不同partition内
- Partition内，根据广告的定向模式分group，group之间的投放不重复
- 每个group按定向模式算出自己的bitmap（8bit代表8种定向模式）



规模扩大的挑战-合理的分层处理

一个广告请求的开销巨大
但可能最后没有合适的广告返回



| 规模扩大的挑战-合理的分层处理

- 不是所有的流量都有一样的价值
- 不是所有的投放都有一样的价值
- 区分对待，不搞平均主义

分层处理

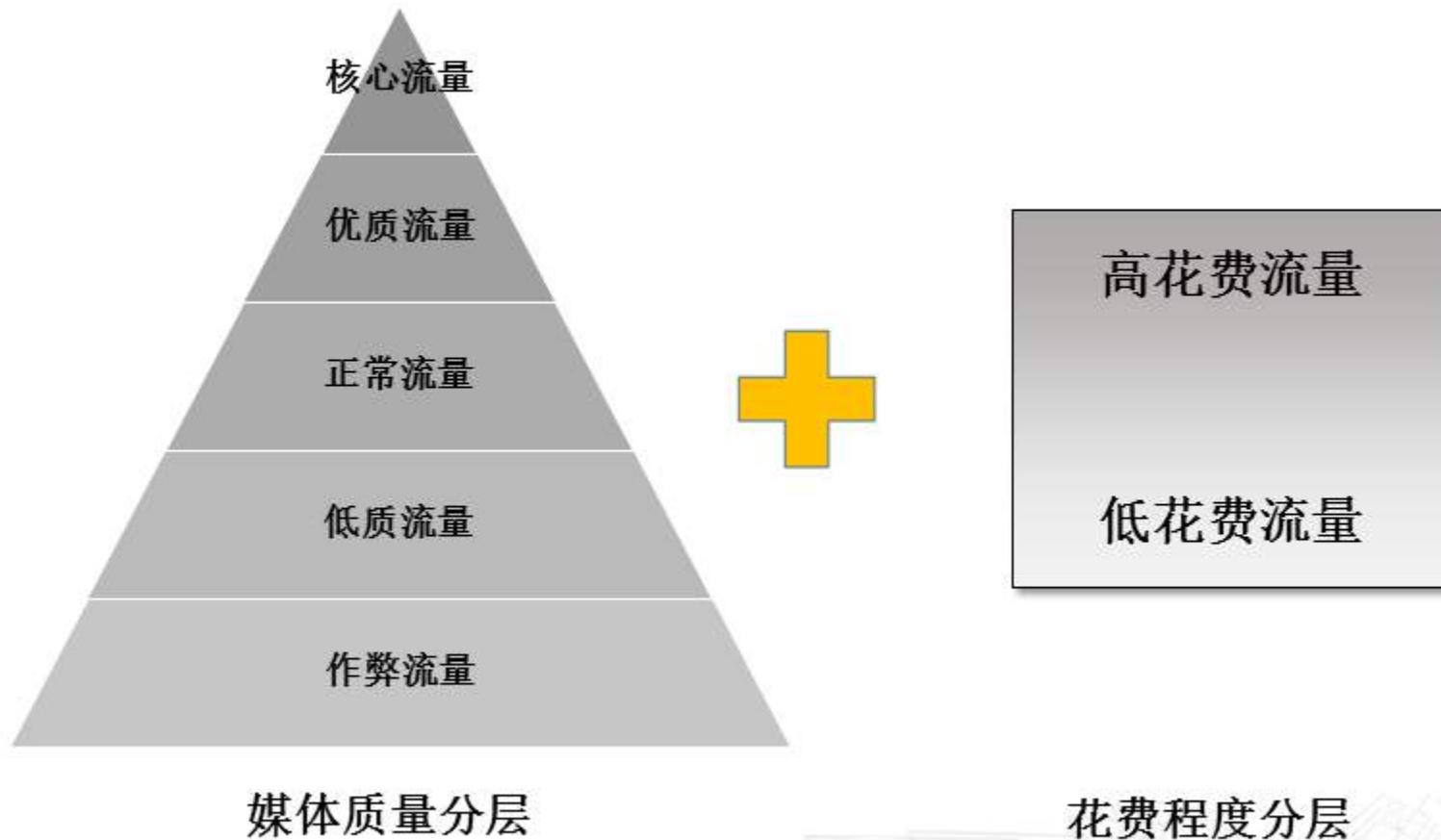


提前过滤



| 规模扩大的挑战-合理的分层处理

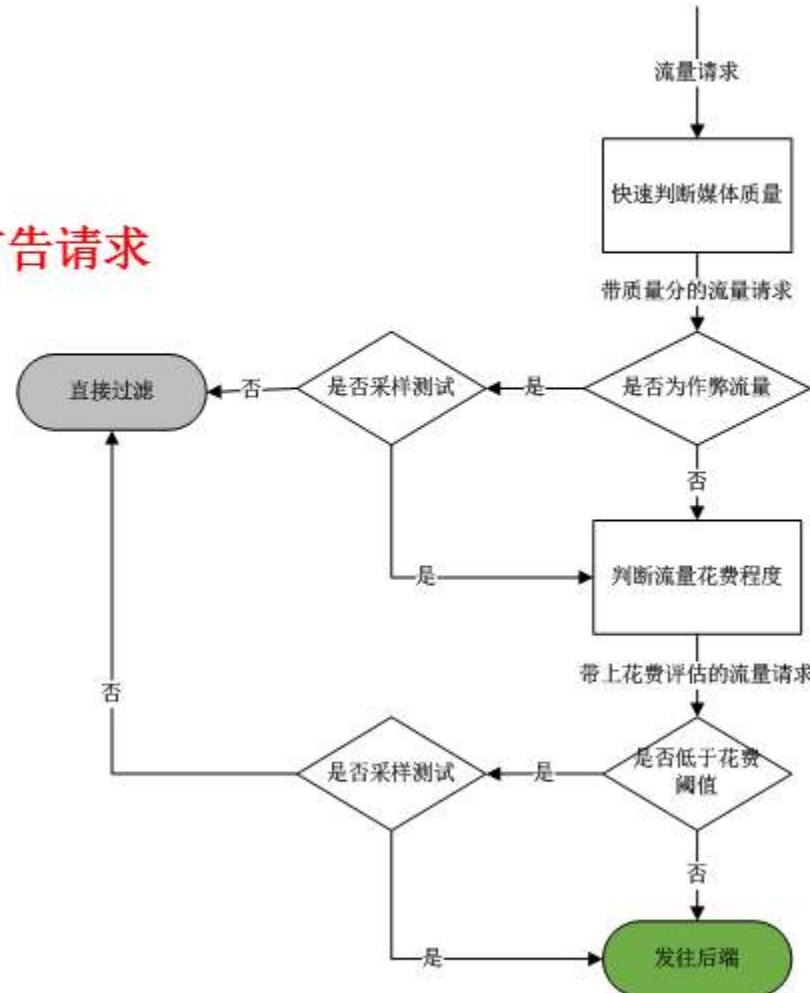
- 流量分层处理



| 规模扩大的挑战-合理的分层处理

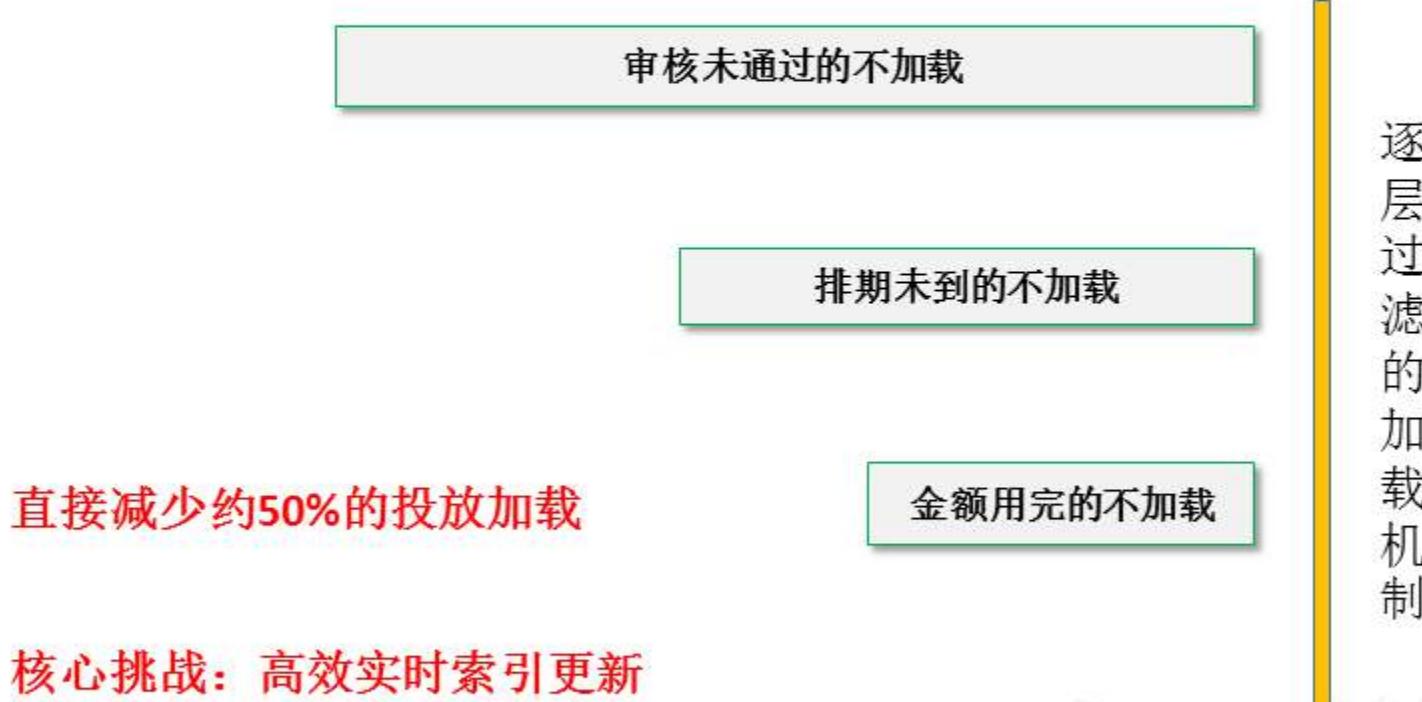
- 流量分层处理流程

直接减少约40%的广告请求
收入影响<3%



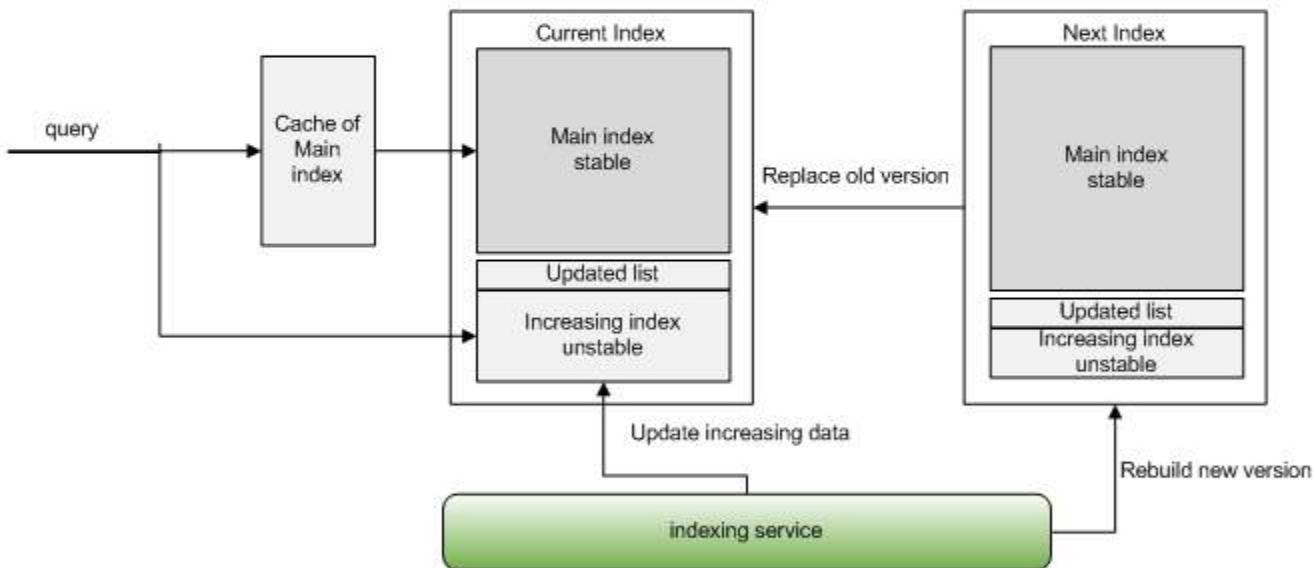
| 规模扩大的挑战-合理的分层处理

- 投放分层处理
 - 和搜索引擎不同，广告投放检索更看重实时性，不需要检索历史投放



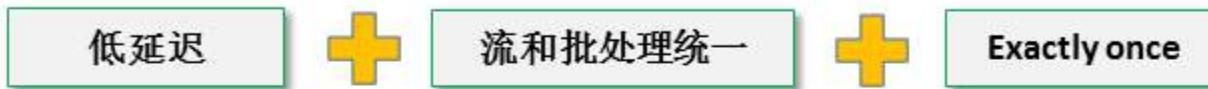
| 规模扩大的挑战-合理的分层处理

- 实时投放索引|更新
 - 索引分层：主索引 + 增量索引
 - 检索逻辑：主索引结果 - 更新列表 + 增量索引结果



| 规模扩大的挑战-实时处理

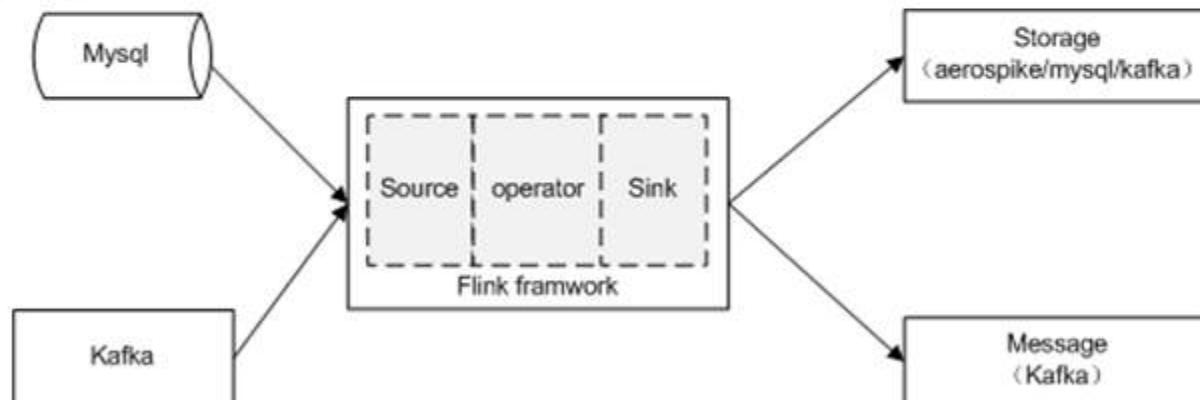
- 需求：



- 之前系统：

- Hadoop + storm + spark streaming

- 现在的方案：



| 高可用性管理

- 核心思路
 - 高可用性要在架构设计时就考虑进来
 - 基于微服务的架构，核心服务要保证自身高可用，做好隔离
 - 贯彻分层处理理念
 - 对于大型分布式系统，需要提供分布式debug能力和监控管理
- 方案：
 - 降级服务
 - 分布式debug系统
 - 分布式监控管理系统



| 高可用性管理-降级服务

- 自动降级 + 人工干预



| 高可用性管理-分布式debug系统

- 设计要点：

低侵入的debug lib

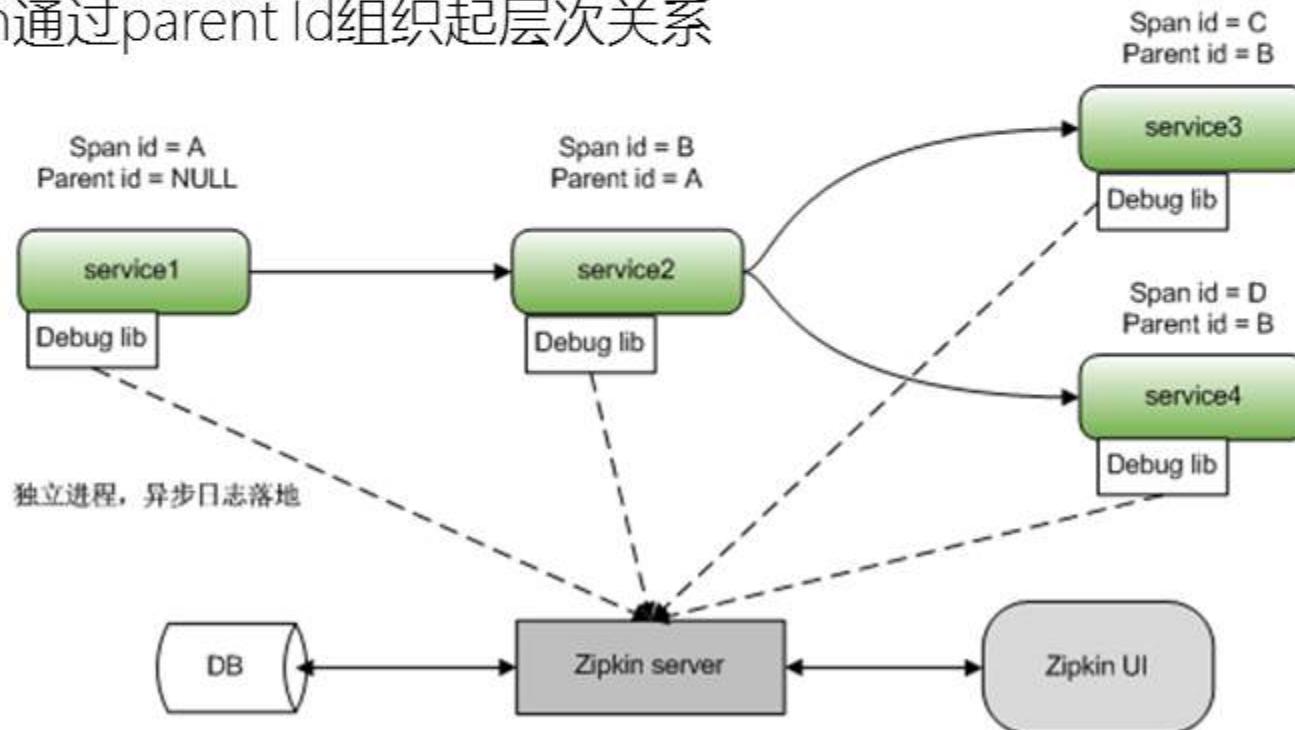


debug日志汇总串联



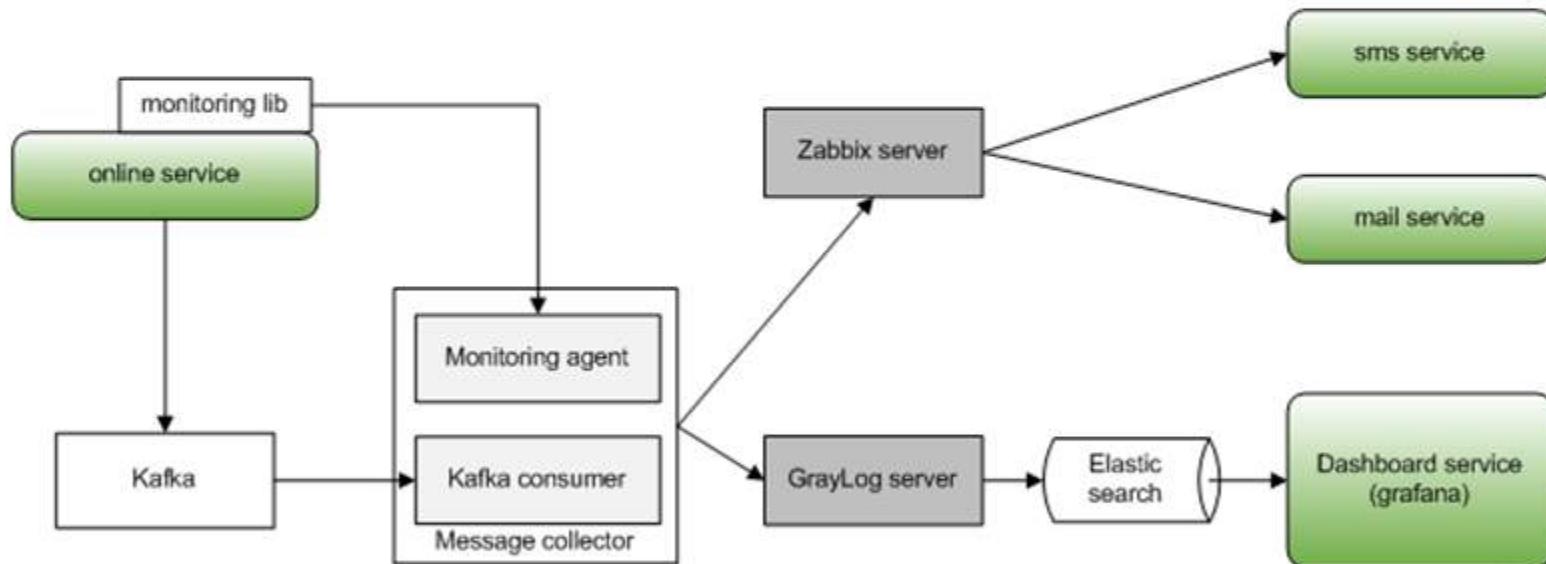
可视化展示

- 一条请求链路通过trace id唯一标识，每经过一个模块，建立一个span
- Span通过parent id组织起层次关系



| 高可用性管理-分布式监控管理系统

- 每个线上模块嵌入一个monitoring lib，启动独立的监控服务
- Monitoring agent通过API调用获得线上模块的监控指标
- 收集和处理：Graylog + elasticsearch + Grafana + zabbix



| 总结-分布式系统设计的一些经验

整体架构思路：化整为零的微服务架构

高效应对规模增长

- 理解业务，做合理的拆分
- 分层处理
- 实时处理框架

高可用性管理

- 降级服务能力
- 分布式debug系统
- 监控管理系统

Q&A



2017 Software Architecture Summit

正确性驱动建模：用TLA+设计系统

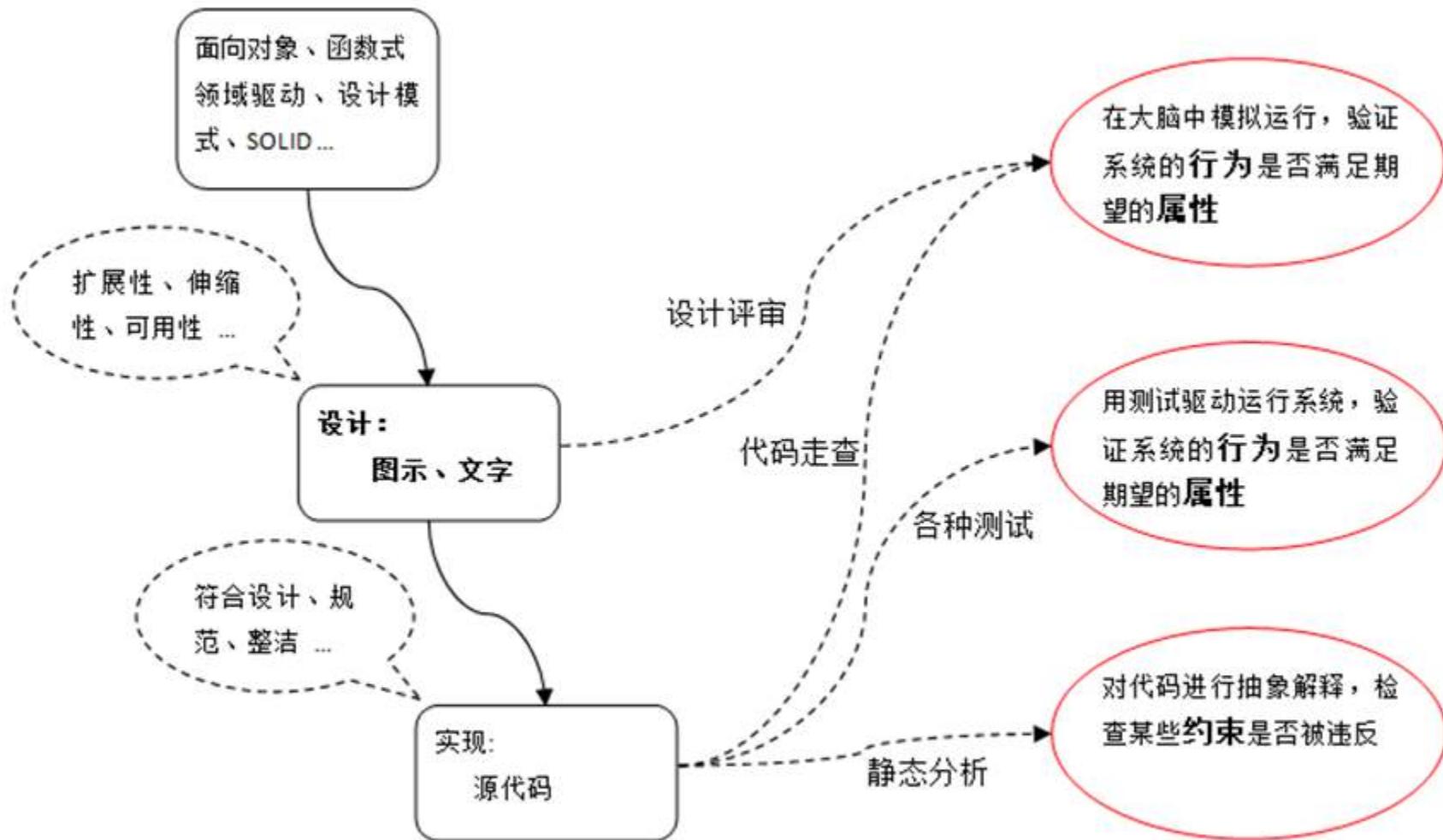
孙鸣

中兴通讯资深系统架构师，17年软件设计、架构经验。国内敏捷方法的早期践行者。《敏捷软件开发：原则、模式与实践（C#）》、《平衡敏捷与规范》和《Erlang趣学指南》等书的译者。曾领导过高性能海量数据存储引擎、下一代波分系统架构等设计工作。目前致力于TLA+的推广和应用，以提升设计和产品质量。



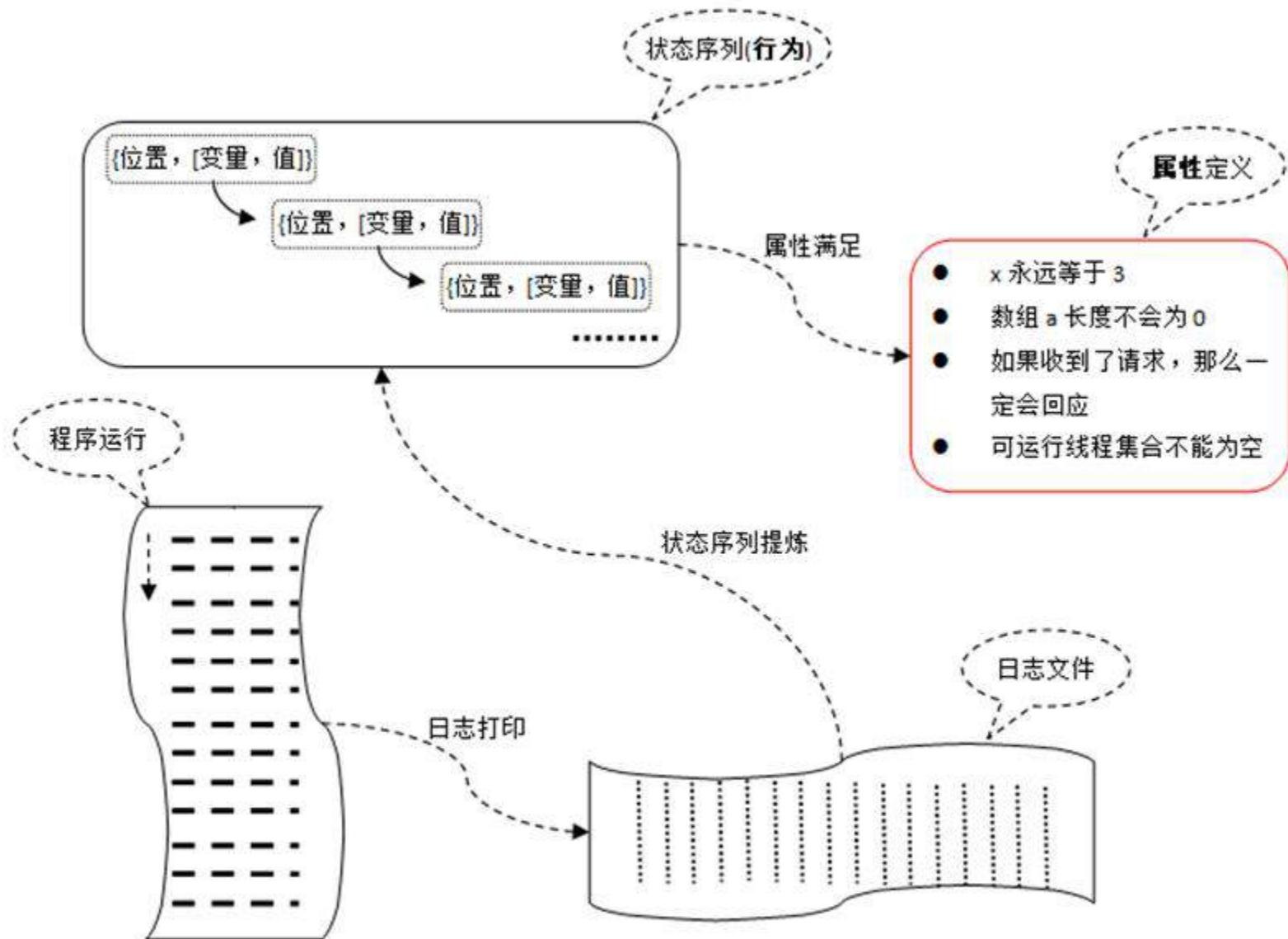
设计 & 正确性





"To a first approximation, we can say that accidents are almost always the result of incorrect estimates of the likelihood of one or more things." - C. Michael Holloway, NASA

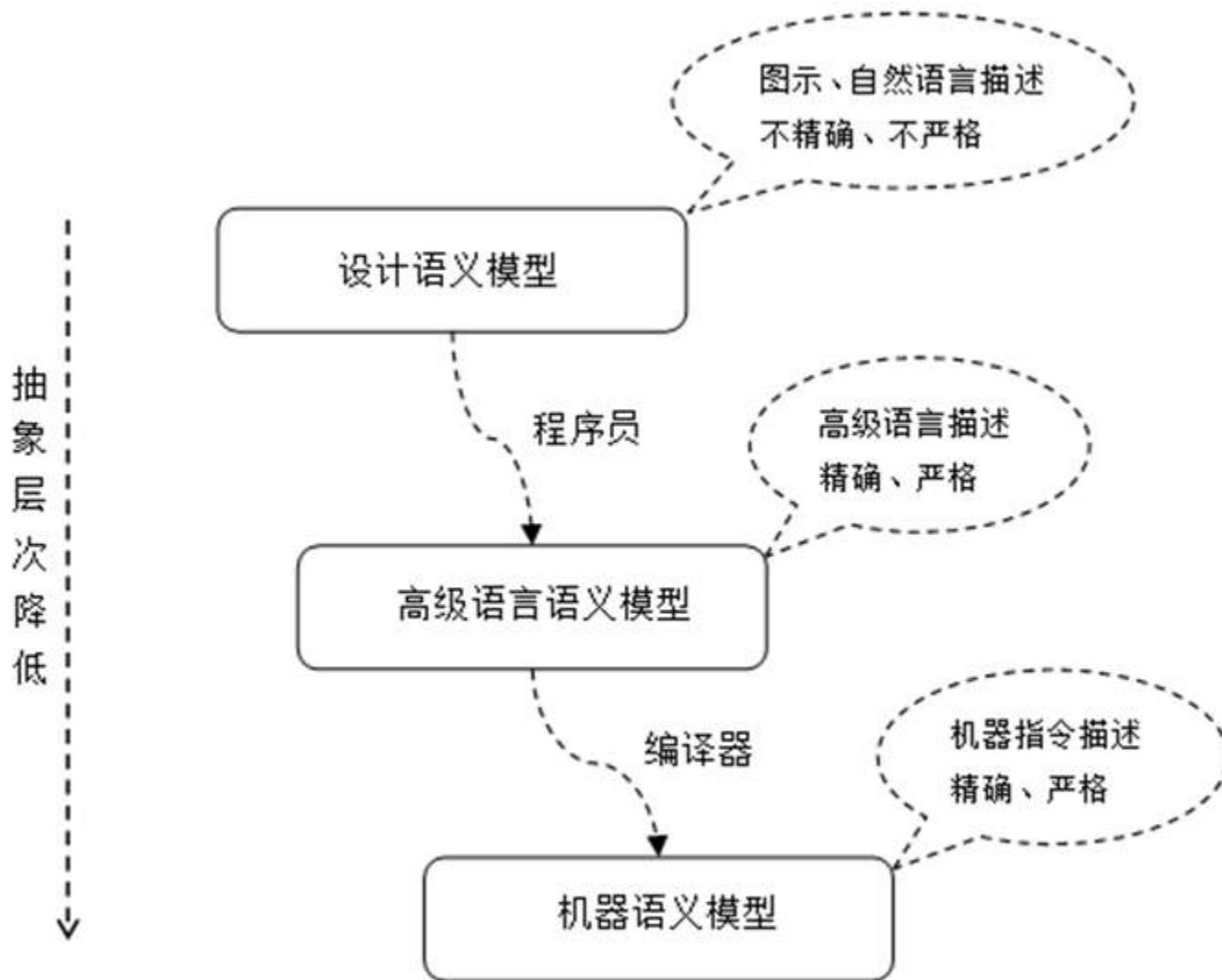
一般来讲，事故的发生几乎都是错误估计了某些事情出现可能性的结果。



繁琐、耗时、痛苦，还需要运气！

模型 & 抽象





低抽象语义模型的行为集合 是高抽象语义模型行为集合的子集

越抽象， 包含的行为越多。

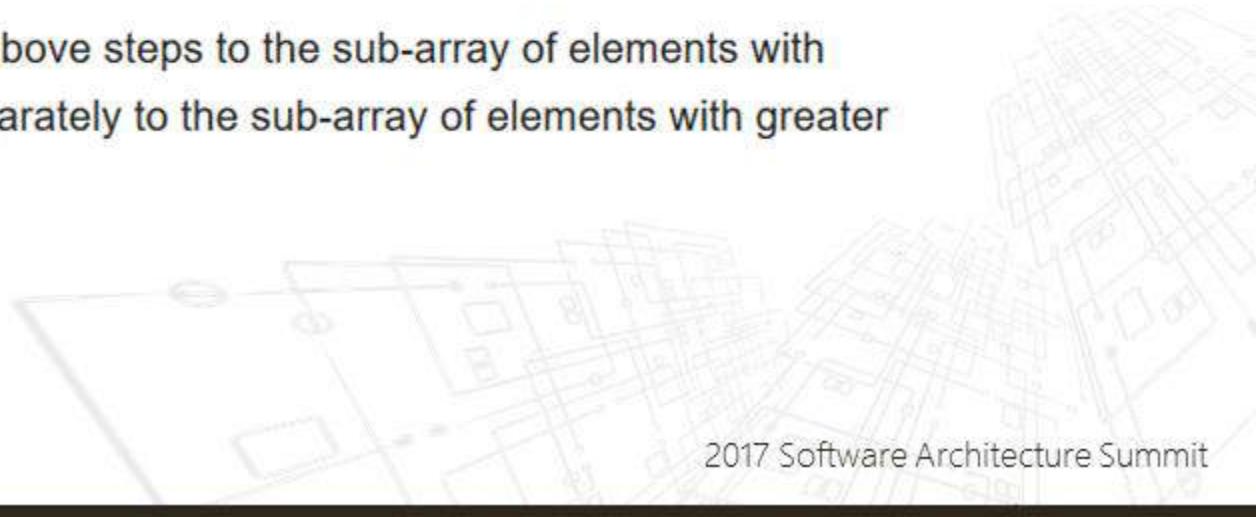
抽象是用不确定性忽略无关细节

Quicksort

From Wikipedia, the free encyclopedia

The steps are:

1. Pick an element, called a *pivot*, from the array.
2. *Partitioning*: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation.
3. Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.



系统蓝图

- 正确性属性
- 环境属性、环境事件
- 问题固有变量、状态、行为

架构设计

- 风格、模式选择
- 中间件选择
- 语言、范型选择

像科学家一样思考



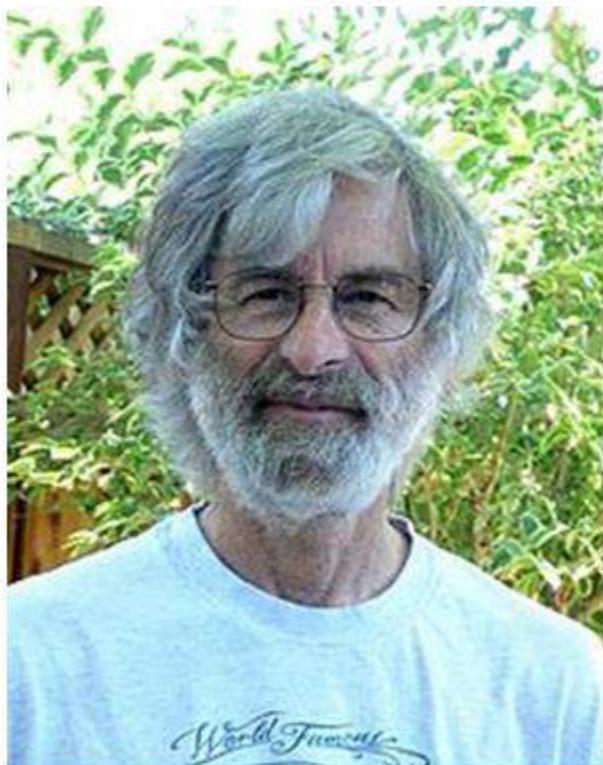
动态连续系统建模

提取问题**相关变量**，用**微分方程**描述变量随时
间变化关系

动态**离散**系统建模

提取问题**相关变量**，用**TLA+**描述变量每一步
的变化关系

Leslie Lamport



贡献：

TLA+

Paxos

Byzantine Generals

Digital Signatures

Sequential Consistency

LaTex

TLA+的设计哲学

实用主义而非理论完美

简单：一线工程师可以很快学习和使用

实用：能够应用于具有相当复杂程度的现实软件、硬件系统。

通用：以同样的形式应用于不同种类的系统：分布式、并发、顺序、反应式、批处理等等

We are motivated not by an abstract ideal of elegance, but by the practical problem of reasoning about real algorithms. Rigorous reasoning is the only way to avoid subtle errors... ... and we want to make reasoning as simple as possible by making the underlying formalism simple

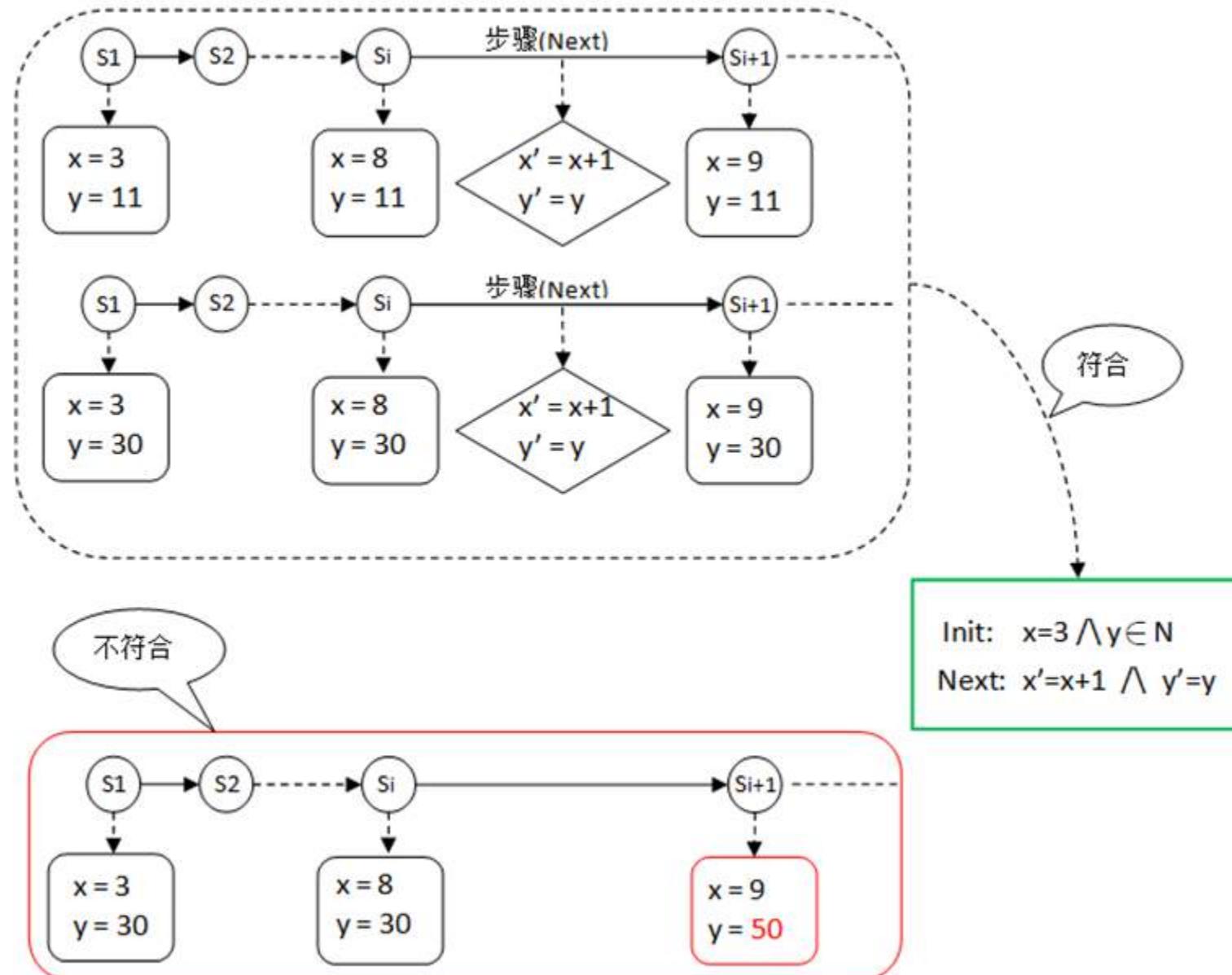
Leslie Lamport

行为 & 属性

The Standard Model An abstract system is described as a collection of behaviors, each representing a possible execution of the system, where a behavior is a sequence of states and a state is an assignment of values to variables.

The TLA+ Hyperbook, Leslie Lamport

标准模型：我们用行为集合描述抽象系统，每个行为都表示该系统的一个可能执行情况，行为是状态序列，状态是对变量的赋值。



系统模型(行为集合)
的严格、精确定义

告诉神灯你想
要什么

$s_1 \rightarrow s_2 \rightarrow s_3 \dots$

.....
.....

过滤出

行为宇宙



简单的数学

一阶命题逻辑 : $\wedge, \vee, \neg, \Rightarrow, \equiv$

基本的集合论 : $\cap, \cup, \subseteq, \setminus$

一阶谓词逻辑 : \forall, \exists

线性时态逻辑 : $\Box, \Diamond, \rightsquigarrow$

期望属性的定义

Safety属性：坏的事情一定不会发生

x 永远为偶数

工作线程数目永远大于0

临界区中最多只有一个线程

Liveness属性：好的事情最终会发生

如果收到用户请求，最终一定会回应

y 最终会大于100

如果线程尝试进入临界区，最终一定会进入

实战时间



The Science of Programming (Monographs in Computer Science)

by David Gries (Author)



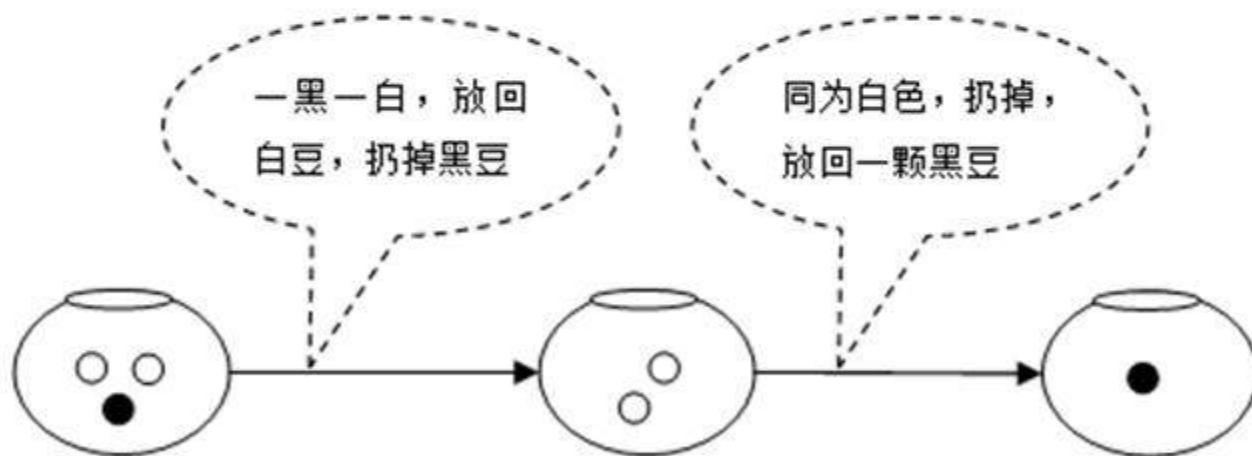
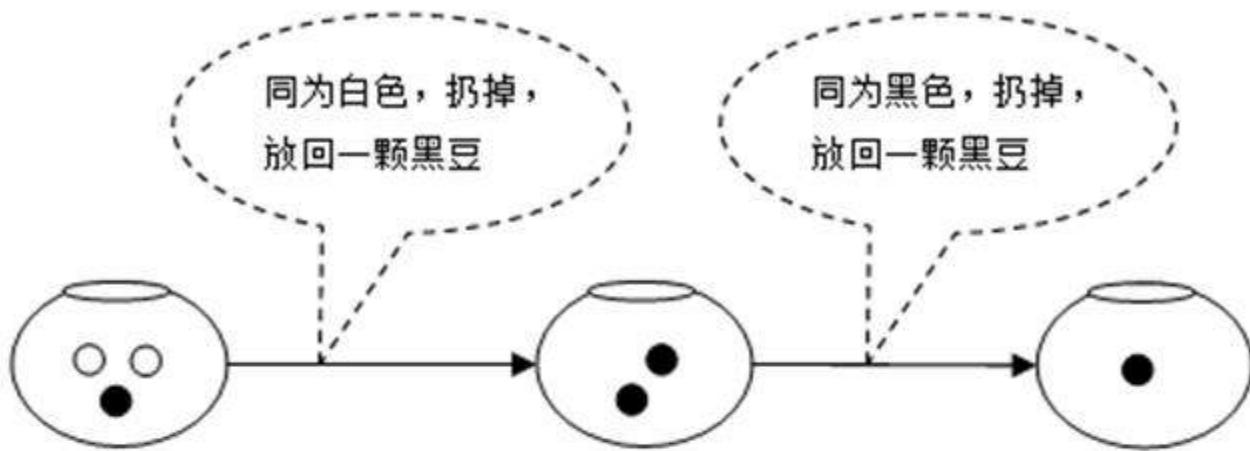
10 customer reviews

咖啡罐问题：

有一个装有黑色和白色咖啡豆的罐子，罐子的外面还有一堆“足够多”的黑色咖啡豆。重复如下过程，直到罐子中只剩下一颗咖啡豆：

随机从罐子中取出两颗豆。如果它们的颜色相同，就把它们扔掉，并向罐子中放入一颗黑豆。如果颜色不同，就把白豆放回罐中，黑豆扔掉。

证明这个过程一定会结束。结束时，罐中所剩豆子的颜色和最初罐子中白豆和黑豆数目之间有何函数关系？



EXTENDS Integers

CONSTANT M,N

VARIABLES b,w

Init == b = M /\ w = N

Next == $\vee \wedge b > 1$ /* same black
 $\wedge b' = b - 1$
 $\wedge w' = w$
 $\vee \wedge w > 1$ /* same white
 $\wedge b' = b + 1$
 $\wedge w' = w - 2$
 $\vee \wedge w > 0 \wedge b > 0$ /* not same
 $\wedge b' = b - 1$
 $\wedge w' = w$

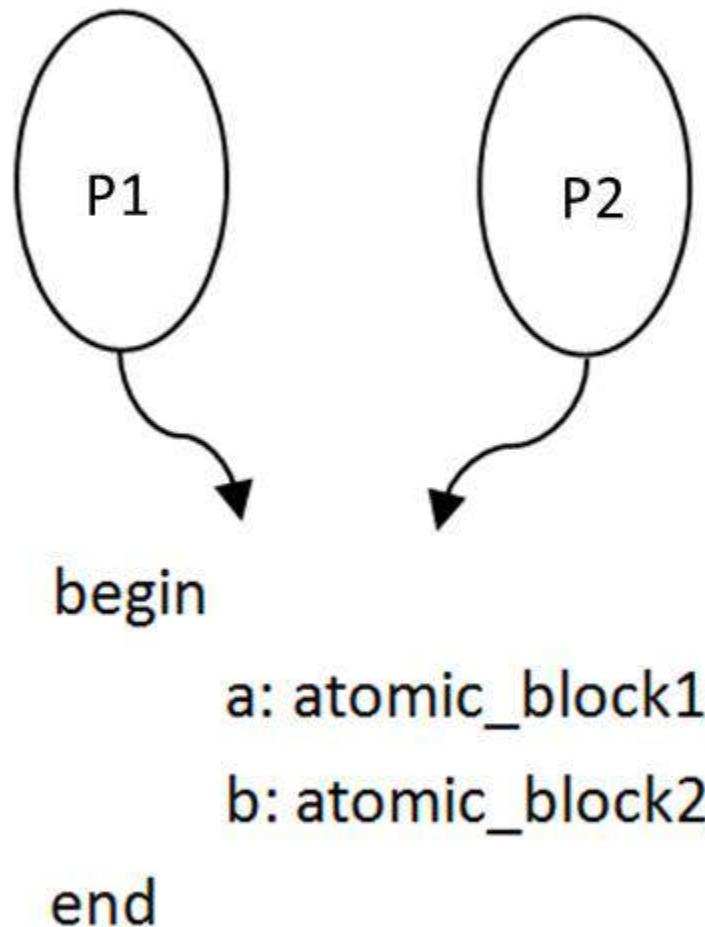
期望属性定义

不变性(invariant) : $(N-w) \% 2 = 0$

或者

时态公式 : $[](N-w) \% 2 = 0$

并发的本质



所有可能的行为

P1a -> P1b -> P2a -> P2b

P1a -> P2a -> P1b -> P2b

P1a -> P2a -> P2b -> P1b

P2a -> P2b -> P1a -> P1b

P2a -> P1a -> P2b -> P1b

P2a -> P1a -> P1b -> P2b

模型定义

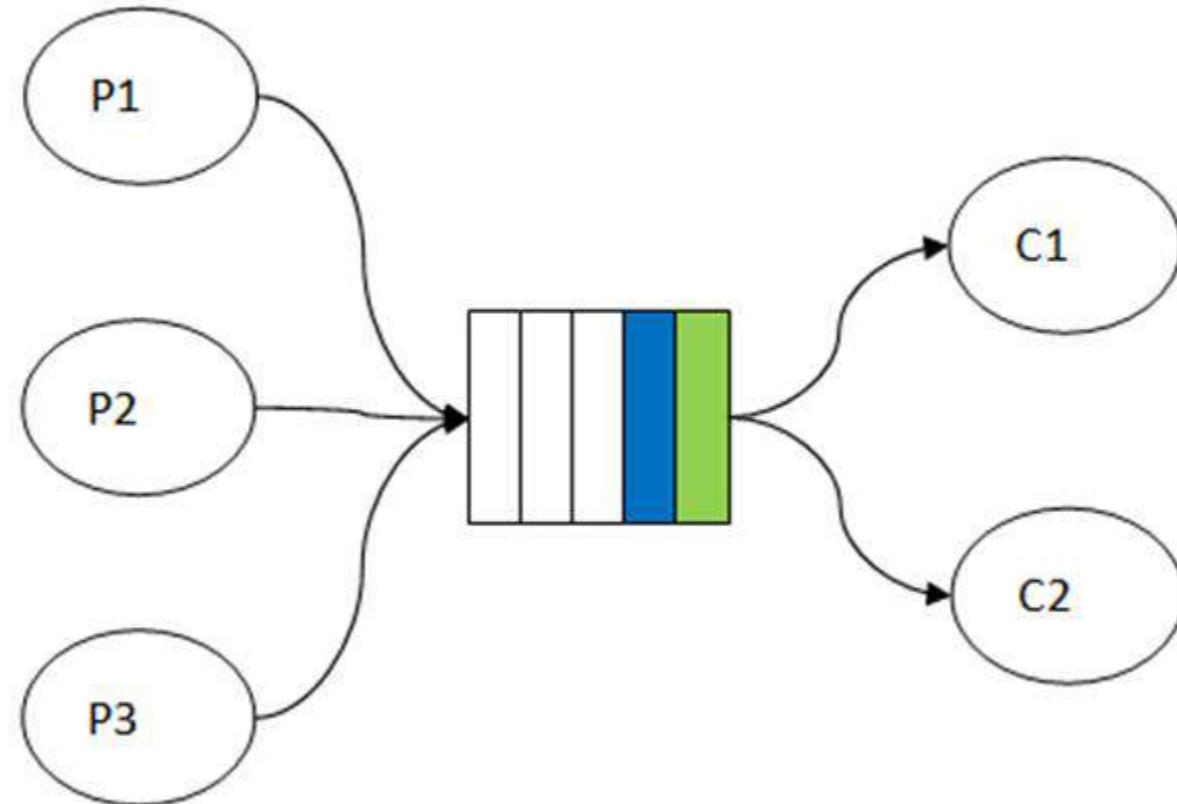
```
Init == P1 = "a" /\ P2 = "a"

Next == \/\ /\ P1 = "a"
          /\ AtomicBlock1
          /\ P1' = "b"
          /\ P2' = P2
          \/\ /\ P2 = "a"
              /\ AtomicBlock1
              /\ P2' = "b"
              /\ P1' = P1
          \/\ /\ P1 = "b"
              /\ AtomicBlock2
              /\ P1' = "done"
              /\ P2' = P2
          \/\ /\ P2 = "b"
              /\ AtomicBlock2
              /\ P2' = "done"
              /\ P1' = P1
```

把并发系统当成一组进(线)程是一种幻觉，进(线)程只是一种看待系统的视角，并不是系统的固有部分。

多进(线)程系统和具有非确定性行为的单进(线)程没有区别。

生产者消费者问题



折磨人的bug

```
public class Buffer<E> {  
    ...  
    public synchronized void put (E e) throws InterruptedException {  
        while (isFull())  
            wait();  
        notify();  
        //放入buffer  
        ...  
    }  
    public synchronized E get () throws InterruptedException {  
        while (isEmpty())  
            wait();  
        notify();  
        //从buffer取出  
        ...  
        return e;  
    }  
    ...  
}
```

"Java Concurrency In Practice" 第17页

"Correctness means a class *conforms to its specification*. A good specification defines *invariants* constraining an object's state, and *postconditions* describing the effects of its operations.

Since we often don't write adequate specifications for our classes, how can we possibly know they are correct? We can't, but that doesn't stop us from using them anyway, once we've convinced ourselves that "the code works".

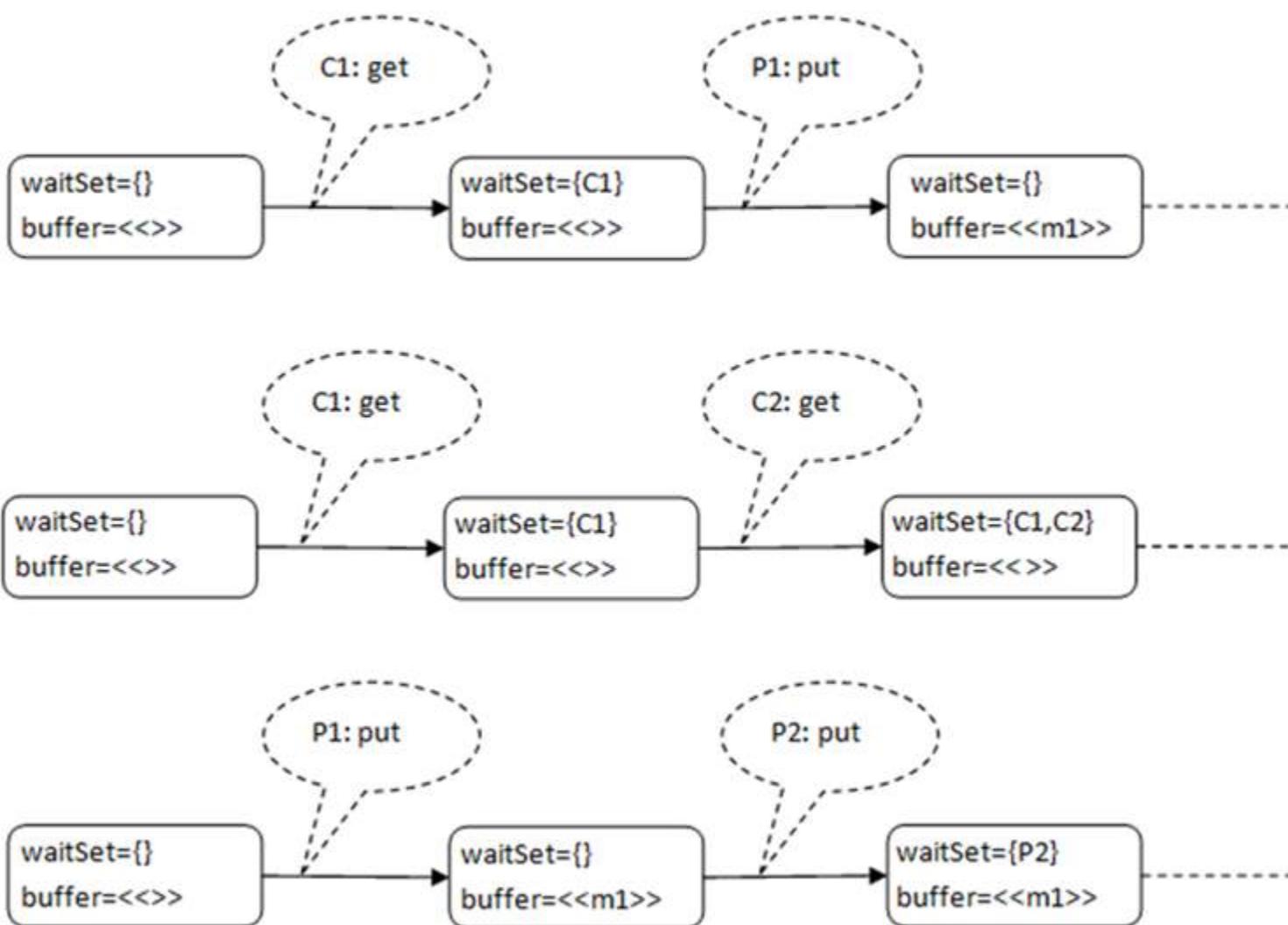
从“直觉正确”到严格正确

模型变量：

waitSet ----> {} 集合 (set)

buffer ----> << >> 元组 (tuple)

生产者={P1,P2,P3} 消费者={C1,C2}



Put和Get

```
Put(t) == IF Len(buffer) < BufCapacity  
        THEN /\ buffer' = Append(buffer, 1)  
              /\ Notify  
        ELSE /\ Wait(t)  
              /\ UNCHANGED buffer
```

```
Get(t) == IF Len(buffer) > 0  
        THEN /\ buffer' = Tail(buffer)  
              /\ Notify  
        ELSE /\ Wait(t)  
              /\ UNCHANGED buffer
```

Wait和Notify

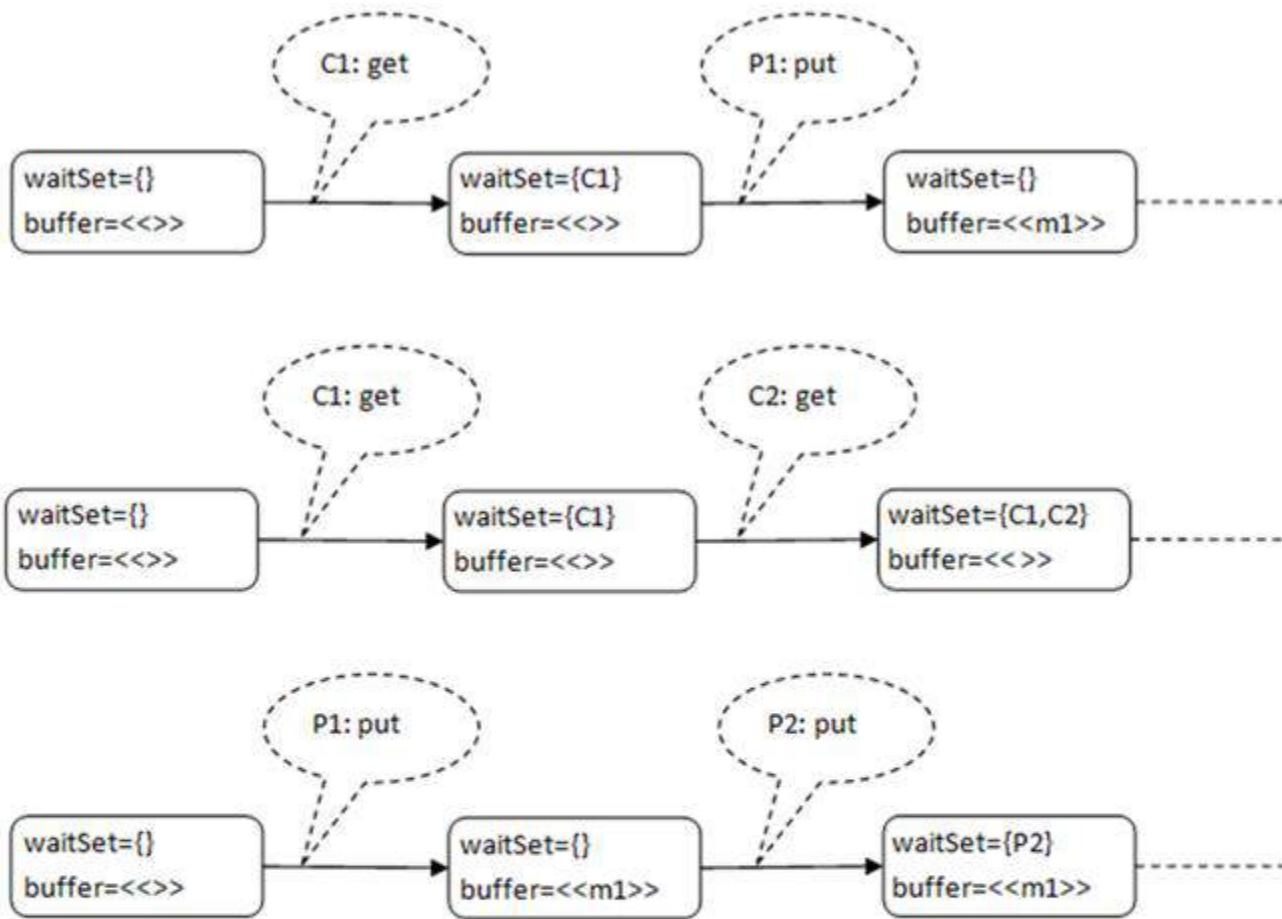
```
Notify == IF waitSet # {}
    THEN \E x \in waitSet : waitSet' = waitSet \ {x}
    ELSE UNCHANGED waitSet
```

```
Wait(t) == waitSet' = waitSet \union {t}
```

模型定义

```
Init == buffer = <<>> /\ waitSet = {}  
  
Next == \E t \in RunningThreads :  
    IF t \in Producers THEN Put(t)  
    ELSE Get(t)
```

```
Participants == Producers \union Consumers  
RunningThreads == Participants \ waitSet
```



```

Init == buffer = <<>> /\ waitSet = {}

Next == \E t \in RunningThreads :
    IF t \in Producers THEN Put(t)
    ELSE Get(t)
  
```

属性定义

不变性(invariant) : **RunningThreads # {}**

或者

时态公式 : **[](RunningThreads # {})**

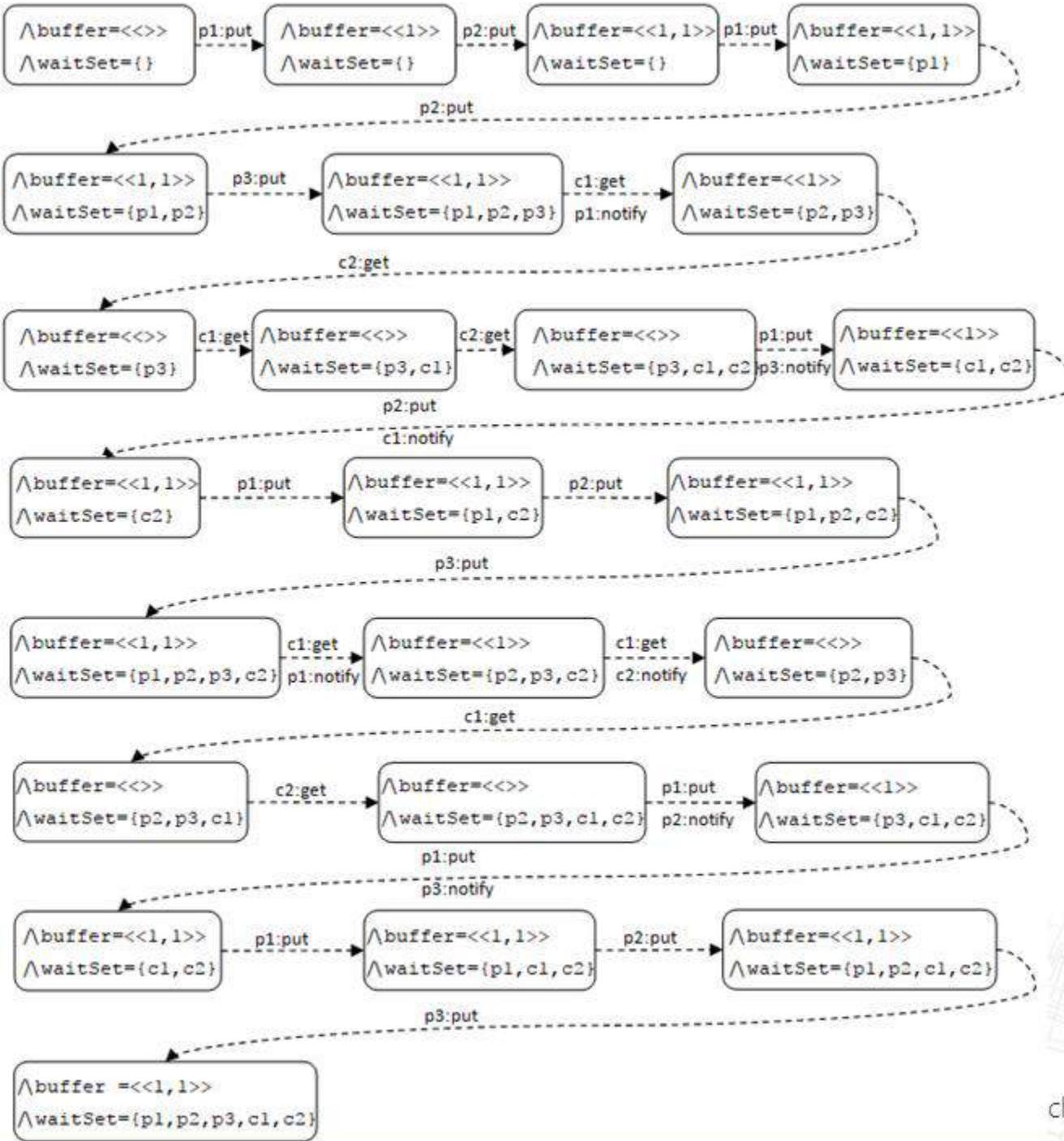
```

1 ----- MODULE Buffer -----
2 EXTENDS Naturals, Sequences
3
4 CONSTANTS Producers, Consumers, BufCapacity
5 VARIABLES buffer, waitSet
6
7 Participants == Producers \union Consumers
8 RunningThreads == Participants \ waitSet
9
10 Notify == IF waitSet # {}
11     THEN \E x \in waitSet : waitSet' = waitSet \ {x}
12     ELSE UNCHANGED waitSet
13 Wait(t) == waitSet' = waitSet \union {t}
14
15 Put(t) == IF Len(buffer) < BufCapacity
16     THEN /\ buffer' = Append(buffer, 1)
17         /\ Notify
18     ELSE /\ Wait(t)
19         /\ UNCHANGED buffer
20 Get(t) == IF Len(buffer) > 0
21     THEN /\ buffer' = Tail(buffer)
22         /\ Notify
23     ELSE /\ Wait(t)
24         /\ UNCHANGED buffer
25
26 Init == buffer = <>> /\ waitSet = {}
27 Next == \E t \in RunningThreads :
28     IF t \in Producers THEN Put(t)
29     ELSE Get(t)
30 Spec == Init /\ [] [Next]_<<buffer, waitSet>>
31 NoDeadlock == [] (RunningThreads # {})
32
33 =====

```

模型检查器： TLC

Error-Trace	
Name	Value
■ buffer	<< >>
> ■ waitSet	{"p1", "p3", "c1", "c2"}
▼ ▲ <Action line 43, col 9 tc State (num = 20)	
> ■ buffer	<<1>>
> ■ waitSet	{"p3", "c1", "c2"}
▼ ▲ <Action line 43, col 9 tc State (num = 21)	
> ■ buffer	<<1, 1>>
> ■ waitSet	{"c1", "c2"}
▼ ▲ <Action line 43, col 9 tc State (num = 22)	
> ■ buffer	<<1, 1>>
> ■ waitSet	{"p1", "c1", "c2"}
▼ ▲ <Action line 43, col 9 tc State (num = 23)	
> ■ buffer	<<1, 1>>
> ■ waitSet	{"p1", "p2", "c1", "c2"}
▼ ▲ <Action line 43, col 9 tc State (num = 24)	
> ■ buffer	<<1, 1>>
> ■ waitSet	{"p1", "p2", "p3", "c1", "c2"}



不中断业务多节点系统升级

有一款电信设备，其中有多块同样的业务单板，可以根据业务负载情况动态增加或者减少可服务的单板数目。我们要设计一个版本升级部署系统来为这些业务处理单板升级版本。

要求：在升级的过程中不能中断业务处理。（这个问题具有普适性。只要把单板换成服务器，就是通用的不间断业务服务器升级部署系统）。

模型变量

- board : 要升级的单板集合
- working_board : 当前正在提供服务的单板集合
- upgrading_board : 当前正在升级的单板集合
- version : 函数 , 定义域为board , 值域为{v1,v2} , 用于记录单板当前版本号
- stage : 函数 , 定义域为board ,
值域为{"idle","propose","upgrading","upgraded"}
用于记录单板的升级状态

正确性属性定义

- **升级过程结束，所有单板的版本一致**：如果升级过程结束，那么对于board集合中的任意两个元素m和n， $version[m]$ 等于 $version[n]$
- **在升级过程的任意时刻，至少有一块单板在提供服务**：在升级过程中的任意时刻，working_board集合不为空
- **在升级过程的任意时刻，提供服务的单板版本一样**：在升级过程中的任意时刻，对于working_board集合中的任意两个元素m和n， $version[m]$ 等于 $version[n]$ 。

基本模型

```
fair process ( Board ∈ board )
{
    l1: await stage[self] = "propose" ;
        stage[self] := "upgrading" ;
    l2: version[self] := "v2" ;
        stage[self] := "upgraded" ;
}
```

```
fair process ( Controller = "c1" )
    variable k ;
{
    c1: k := upgrading_board ;
    c2: SetStage("propose", k) ;
    c3: await ∀ p ∈ upgrading_board : stage[p] = "upgraded" ;
        SwitchBoard() ;
        k := upgrading_board ;
    c4: SetStage("propose", k) ;
    c5: await ∀ p ∈ upgrading_board : stage[p] = "upgraded" ;
        upgrading_board := {} ;
        working_board := board ;
}
```

正确性属性严格定义和验证

- 升级过程一定会结束：

$$\text{Termination} \triangleq \diamond(\forall self \in \text{ProcSet} : pc[self] = \text{"Done"})$$

- 升级过程结束，所有单板的版本一致：

$$\text{AllDone} \triangleq \forall self \in \text{ProcSet} : pc[self] = \text{"Done"}$$

$$\text{PartialCorrectness} \triangleq \text{AllDone} \Rightarrow \forall m, n \in \text{board} : \text{version}[m] = \text{version}[n]$$

- 在升级过程的任意时刻，至少有一块单板在提供服务：

$$\text{WorkingSetNotEmpty} \triangleq \text{working_board} \neq \{\}$$

- 在升级过程的任意时刻，提供服务的单板版本一样：

$$\text{SameWorkingVer} \triangleq \forall m, n \in \text{working_board} : \text{version}[m] = \text{version}[n]$$

模型正确性验证

What to check?

Deadlock

Invariants

Formulas true in every reachable state.

<input checked="" type="checkbox"/> PartialCorrectness	Add
<input checked="" type="checkbox"/> SameWorkingVer	Edit
<input checked="" type="checkbox"/> WorkingSetNotEmpty	Remove

Properties

Temporal formulas true for every possible behavior.

<input checked="" type="checkbox"/> Termination	Add
	Edit
	Remove

模型扩展：单板升级失败

```
fair process ( Board ∈ board )
{
l1: await stage[self] = "propose" ∨ stage[self] = "abort" ;
    if ( stage[self] = "propose" ) {
l2:      stage[self] := "upgrading" ;
l3:      either {
            version[self] := "v2" ;
            stage[self] := "upgraded" ;
        }
        or {
            stage[self] := "failed" ;
        } ;
l4:      await stage[self] = "abort" ∨ stage[self] = "commit" ;
            if ( stage[self] = "abort" ) {
                version[self] := "v1" ;
                stage[self] := "aborted" ;
            }
            else {
                stage[self] := "committed" ;
            }
        }
        else {
            stage[self] := "aborted" ;
        }
    }
}
```

主控算法自然语言描述

- 升级 upgrading_board 集合中的单板
- 如果其中有一块单板升级失败
 - 回退 upgrading_board 中的已经升级的单板
 - 通知 working_board 集合中单板的升级终止
 - 整个升级过程结束（失败）
- 如果全部单板升级成功
 - 切换 working_board 和 upgrading_board
 - 升级切换后的 upgrading_board 集合中的单板
 - 如果其中一块单板升级失败
 - ◆ 回退 upgrading_board 中的已经升级的单板
 - ◆ 再次切换 working_board 和 upgrading_board
 - ◆ 回退 upgrading_board 集合中的单板版本
 - ◆ 整个升级过程结束（失败）
 - 如果全部单板升级成功
 - ◆ 通知 board 集合中的单板整个升级成功
 - ◆ 整个升级过程结束（成功）

```

fair process ( Controller = "c1" )
    variable k, aborted = FALSE; step = 1;
{
c1: k := upgrading_board;
c2: SetStage("propose", k);
    k := upgrading_board;
c3: while ( k ≠ {} ) {
    with ( p ∈ k ) {
        await stage[p] = "upgraded" ∨ stage[p] = "failed" ;
        if ( stage[p] = "failed" ) {
            aborted := TRUE;
        }
        k := k \ {p};
    }
}
if ( aborted = TRUE ) {
    k := upgrading_board;
c4: SetStage("abort", k) ;
c5: await ∀ p ∈ upgrading_board : stage[p] = "aborted" ;
    if ( step = 1 ) {
        k := working_board;
        SetStage("abort", k);
c6: await ∀ p ∈ working_board : stage[p] = "aborted" ;
        upgrading_board := {};
        working_board := board;
    }
    else {
        SwitchBoard();
        k := upgrading_board;
c8: SetStage("abort", k) ;
c9: await ∀ p ∈ upgrading_board : stage[p] = "aborted" ;
        upgrading_board := {};
        working_board := board;
    }
}
else {
}

```

```

if ( step = 1 ) {
    SwitchBoard();
    step := 2;
    goto c1;
}
else {
c10: k := board;
c11: SetStage("commit", k);
c12: await ∀ p ∈ board : stage[p] = "committed" ;
    upgrading_board := {};
    working_board := board;
}
}

```



用TLA+设计系统

1. 直接定义safety属性和系统模型，然后用TLC验证系统的所有行为都满足所定义的属性
2. 在非常抽象的层次上定义系统模型和正确性属性，然后验证实际可实现的系统设计和这个抽象设计间存在“abstraction/refinement”关系，也就是实际系统的每个行为都是抽象设计的行为
3. 找到蕴含safety属性的系统不变性(invariant)，验证这个不变性是系统设计的“归纳不变性”(inductive invariant)



感受到的TLA+好处

1. 清晰的思考和表达
2. 在更高的抽象层次上严格精确地定义系统行为和正确性，自动化验证
3. 保证设计正确性
4. 有助于对问题和设计的理解、团队沟通、知识传递，提升研发效率
5. 有益于提高代码质量（更系统的断言）

业界案例：Amazon

System	Components	Line count	Benefit
S3	Fault-tolerant low-level network algorithm	804 PlusCal	Found 2 design bugs. Found further design bugs in proposed optimizations.
	Background redistribution of data	645 PlusCal	Found 1 design bug, and found a bug in the first proposed fix.
DynamoDB	Replication and group-membership systems (which tightly interact)	939 TLA ⁺	Found 3 design bugs, some requiring traces of 35 steps.
EBS	Volume management	102 PlusCal	Found 3 design bugs.
EC2	Change to fault-tolerant replication, including incremental deployment to existing system, with zero downtime	250 TLA ⁺ 460 TLA ⁺ 200 TLA ⁺	Found 1 design bug.
Internal distributed lock manager	Lock-free data structure	223 PlusCal	Improved confidence. Failed to find a liveness bug as we did not check liveness.
	Fault tolerant replication and reconfiguration algorithm	318 TLA ⁺	Found 1 design bug. Verified an aggressive optimization.

Fig. 1. Examples of applying TLA⁺ to some of our more complex systems

Chris Newcombe, Principal Engineer at Amazon

TLA⁺ is the most valuable thing that I've learned in my professional career. It has changed how I work, by giving me an immensely powerful tool to find subtle flaws in system designs. It has changed how I think, by giving me a framework for constructing new kinds of mental-models, by revealing the precise relationship between correctness properties and system designs, and by allowing me to move from 'plausible prose' to precise statements much earlier in the software development process.

在我的职业生涯中，学到的最有价值的东西就是TLA+。它是一个异常强大的发现系统设计中难查bug的工具，有了这个工具，我的工作方式改变了。它赋予我一个构造新思维模型的框架，它揭示了系统设计和正确性属性间的精确关系，它让我在软件开发过程的早期就可以进行精确而非“似是而非”的描述，它改变了我的思考方式

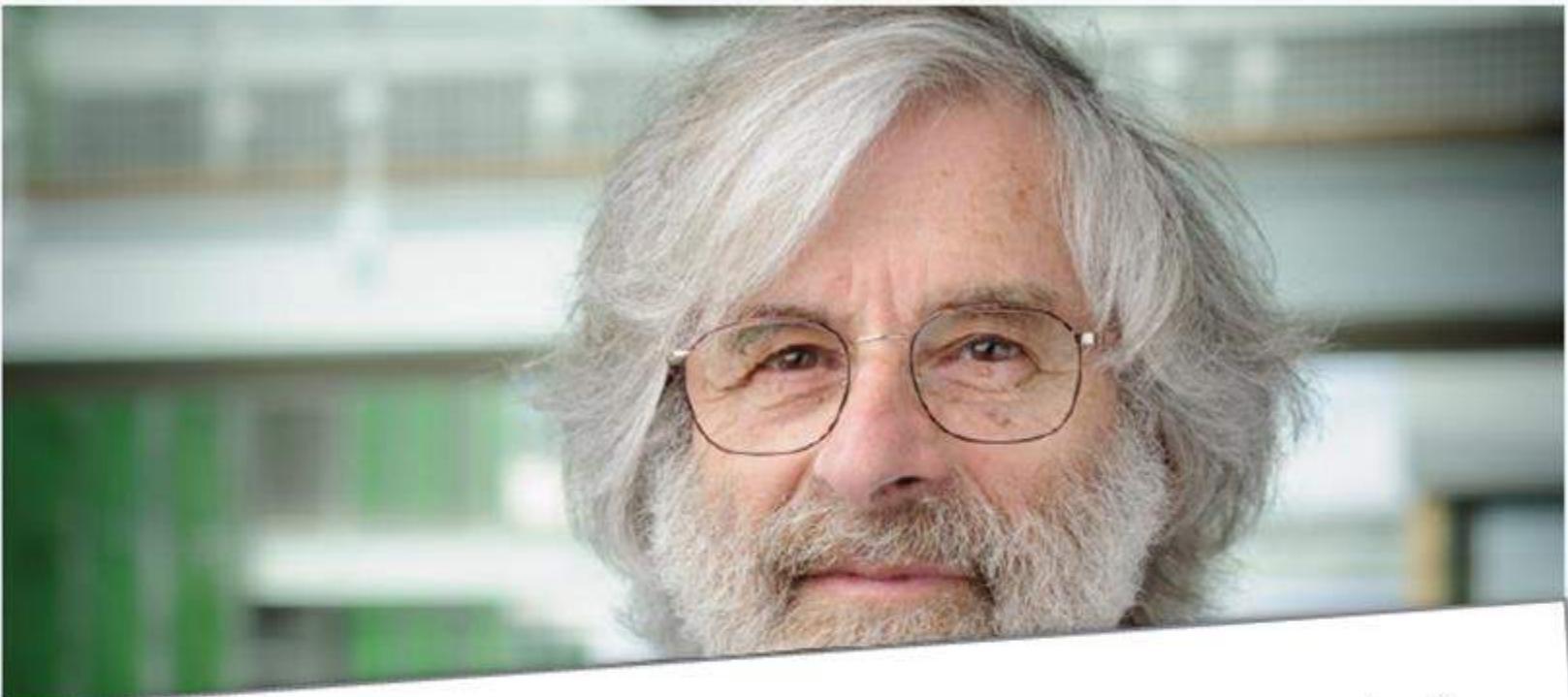
<https://groups.google.com/forum/#!searchin/tlaplus/most%20people%7Csort:relevance/tlaplus/ZJCI-UF31fc/Mawvwi6U1CYJ>

Computation and State Machines

Leslie Lamport 19 April 2008

一直以来，我对计算机科学过于强调编程语言感到焦虑。这种强调的结果是：即使专家级C++程序员也不能写出满足期望行为的程序。对此的回应要么是使用其它更合适的“编程/规格说明/开发”语言，要么是使用更好的调试工具。

我认为写出好程序最好的方法是教授程序员**更好的思考**。思考是一种操纵**概念**而非语言的能力。但是，在学习概念时如何不被表达概念的语言干扰呢？我的答案是：使用和其他所有工程科学分支一样的语言——也就是——**数学**。



“...the TLA+ specification language [...] represents a Quixotic attempt to overcome engineers' antipathy towards mathematics.”

Q & A



2017 Software Architecture Summit

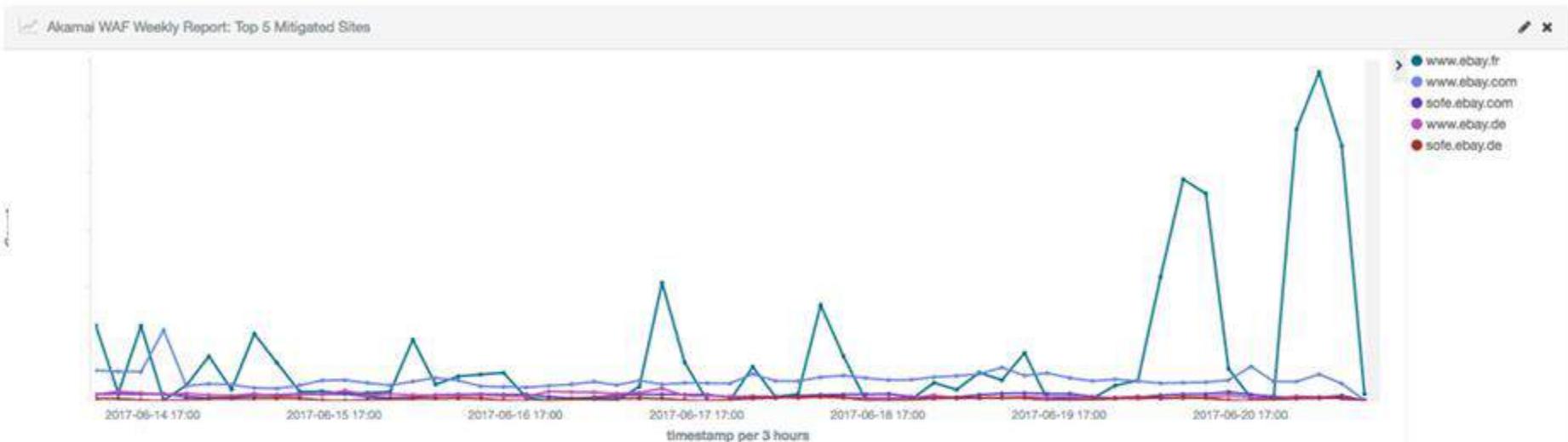
RateLimiter解决方案在eBay中的应用

许云

2005年北京理工大学博士毕业。具有 10 年以上 的研发经验，在产品架构和算法方面具有丰富的经验。2009年任职于HP负责USLAM产品的设计开发，解决了诸多算法与性能问题，该产品为许多国外运营商提供SLA实时评价和预警。2014年加入eBay 基础平台部门负责产品设计开发，目前主要负责安全类平台如eBay Rate Limiter。



| Attack



I Agenda

- Introduction

- Purpose
- Requirement

- Related Work

- Algorithm
- Implementation

- Solution

- Architecture
- Metering
- Process Flow
- Reporting

- Next Work

- Summary



Why

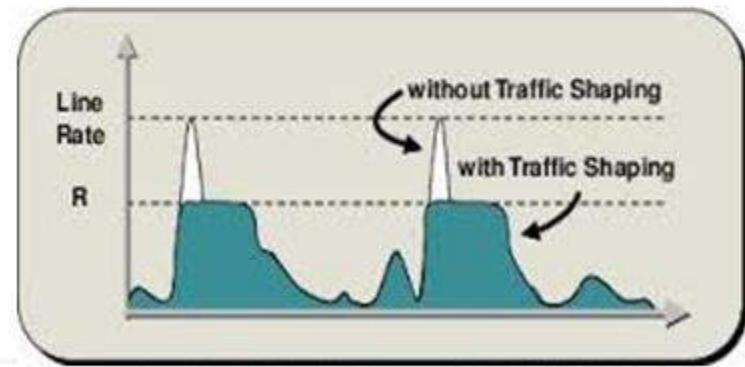
Overload



DoS



Captcha



Traffic Shaping

What

What it is

Policy-based traffic rate controls operating at Layer 7.

Based on **subjects**

- Authenticated user
- End user IP address
- HTTP headers (like application ID)

What we use it for

- Prevent **application level** overuse and DoS
 - Brute force: Signin
 - Expensive operations: Search, promoted listings
 - **Network DoS**
- eBay Developer Program **tiered usage limits**
- Business policies
 - 5 Feedback requests per seller per hour

Requirement - Infrastructure

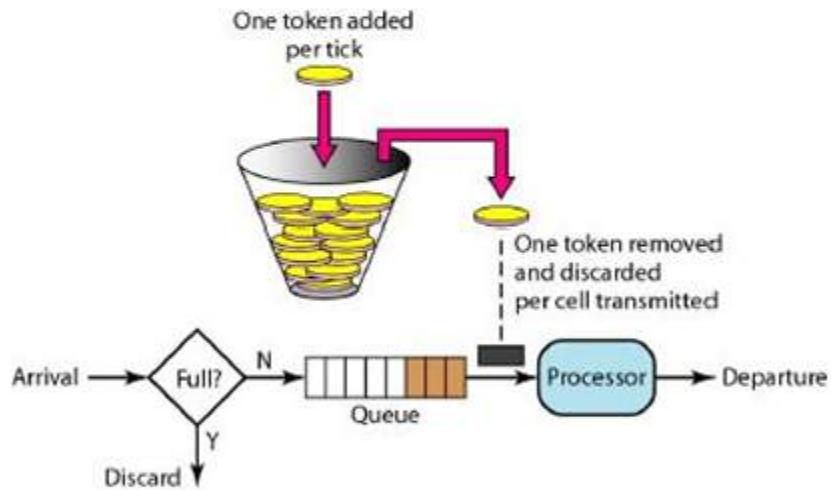
- Reliability / DR
- Scalability
- Low Data Loss
- Low Latency
- Open Architecture
- Cloud Native

Requirement - Function

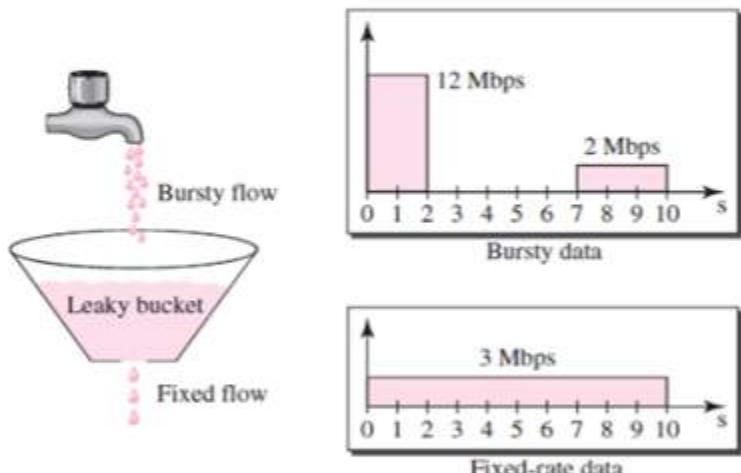
- Multi Calculation Periods
- Complex Policy
- Effect Duration
- Captcha Service
- DoS Resistant
- Monitoring / Reporting

Related Work - Algorithm

Token Bucket



Leaky Bucket

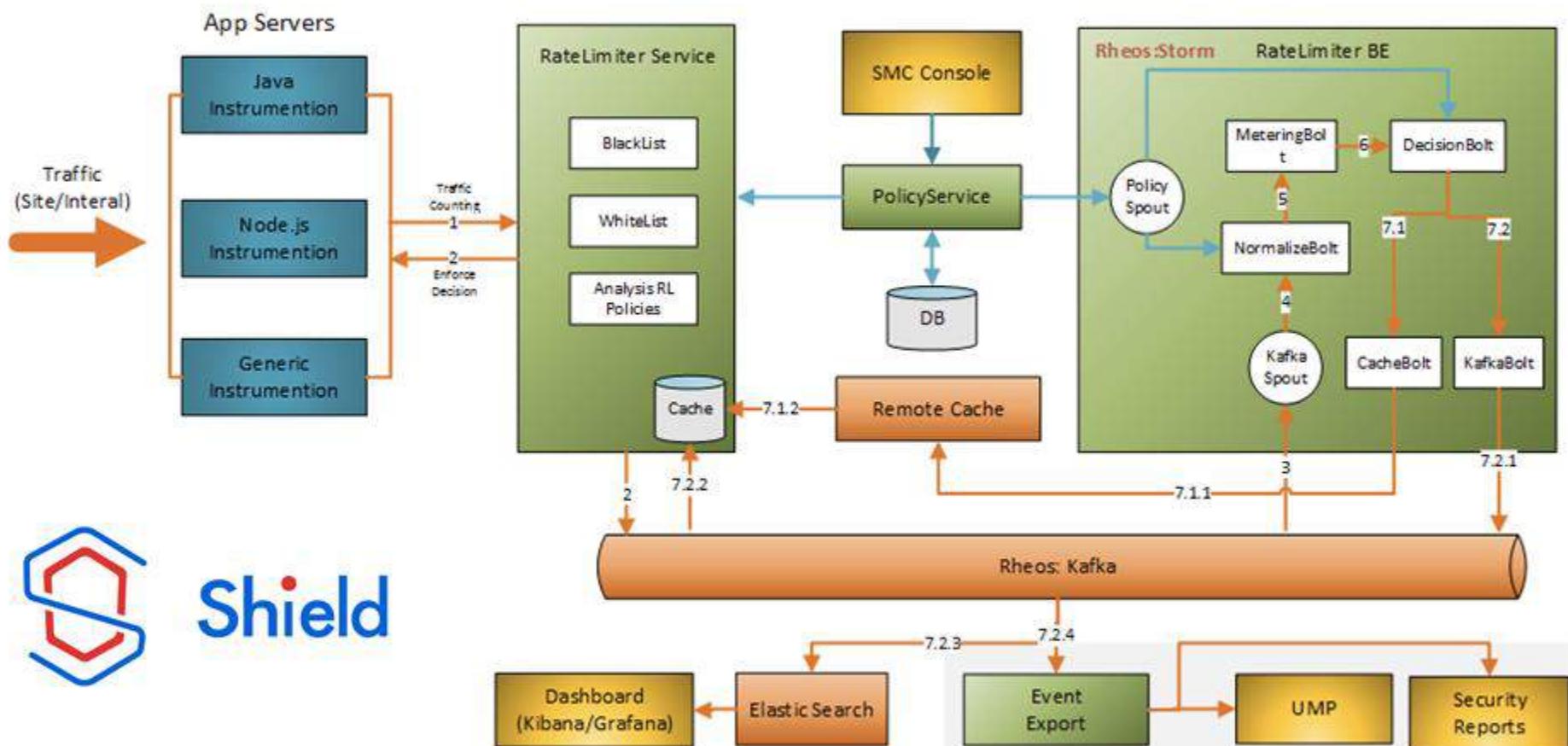


Related Work - Implementation

Guava	Redis	Hystrix
😊 Traffic Shaper	😊 Multi Processes	😊 Multi Processes
😊 Sliding Window	😢 Remote Call	😢 Complex Rules
😢 Multi Processes	😢 Complex Rules	😢 Effect Duration
😢 Complex Rules	😢 Effect Duration	
😢 Effect Duration		



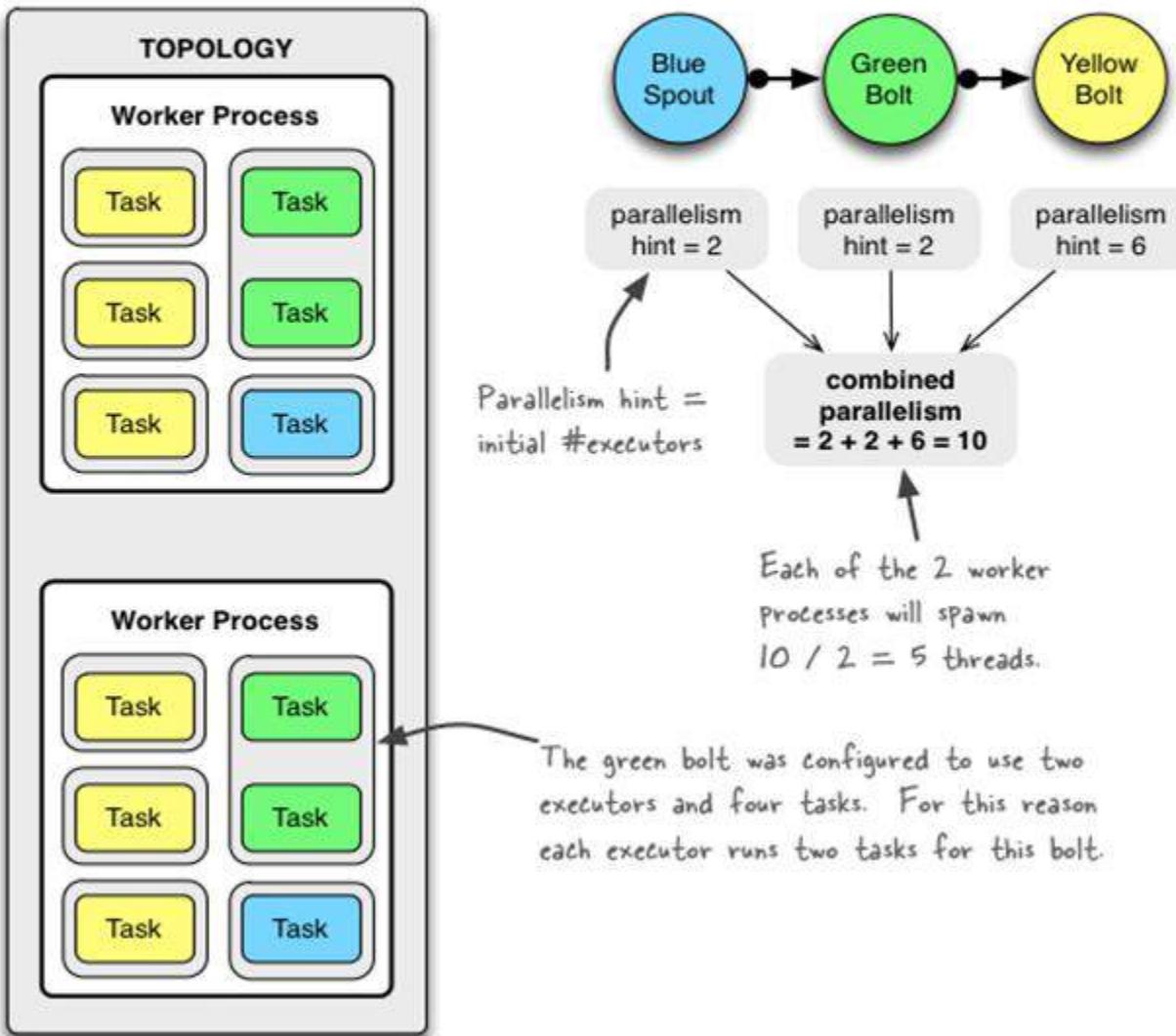
Architecture



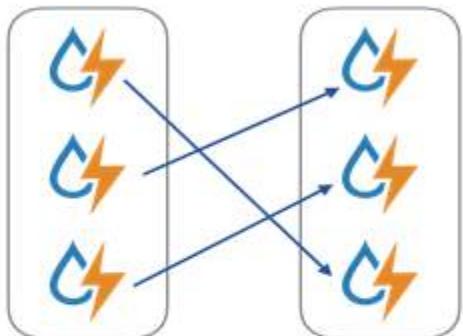
Storm vs Spark Streaming vs Flink

	Storm	Spark Streaming	Flink
Streaming Model	Native	Micro-Batching	Native
Guarantees	At-Least-Once	Exactly-Once	Exactly-Once
Back Pressure	No	Yes	Yes
Latency	Very Low	Medium	Low
Throughput	Low	High	High
Fault Tolerance	Record ACKs	RDD Based CheckPointing	CheckPointing
Stateful	No	Yes (DStream)	Yes (Operators)

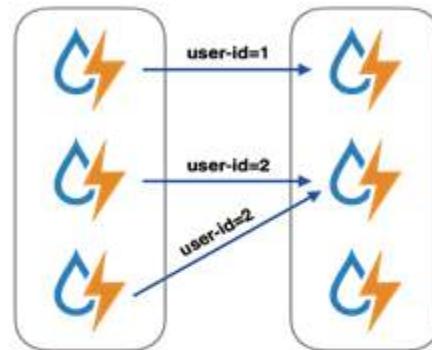
Storm Parallelism



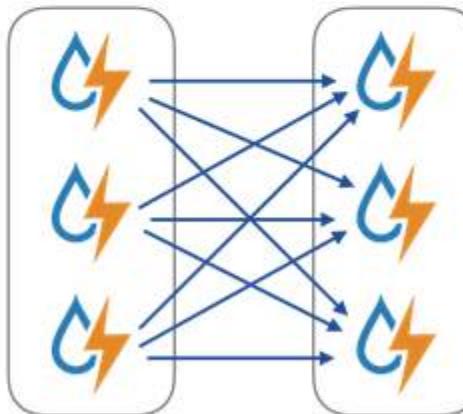
Stream Grouping



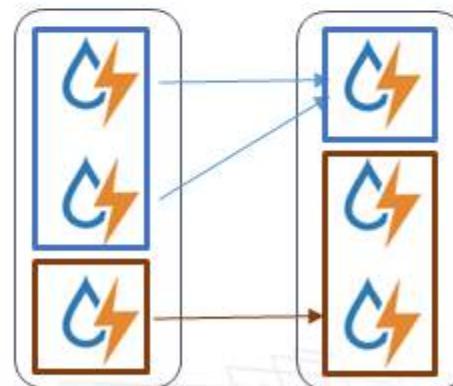
Shuffle Grouping



Fields Grouping

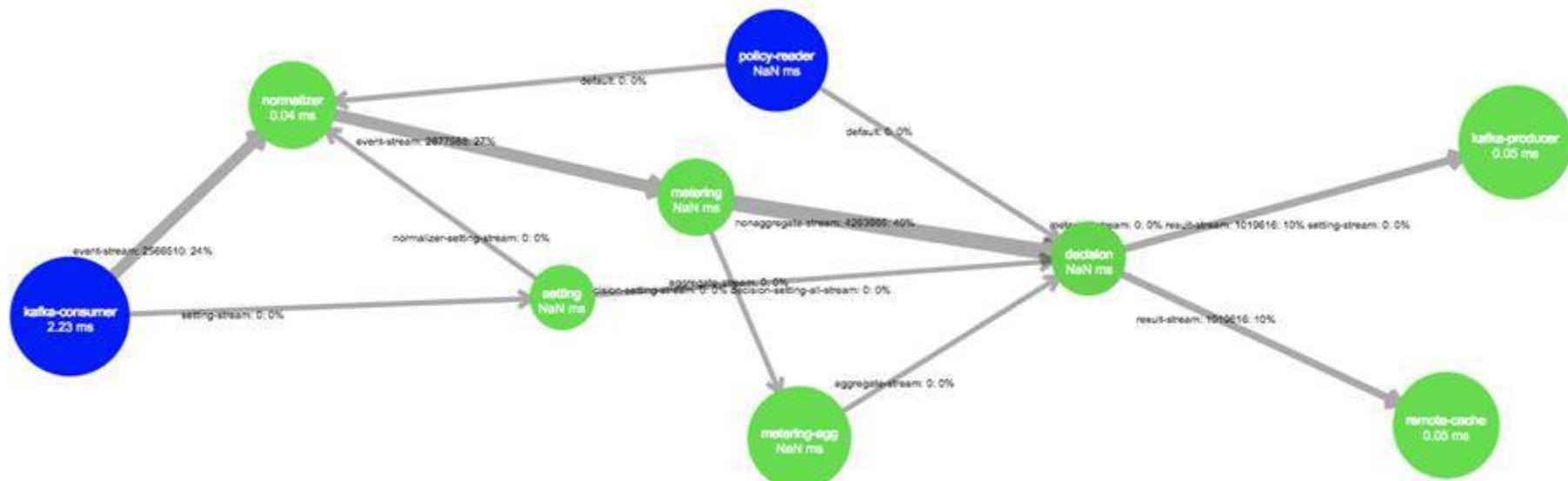


All Grouping



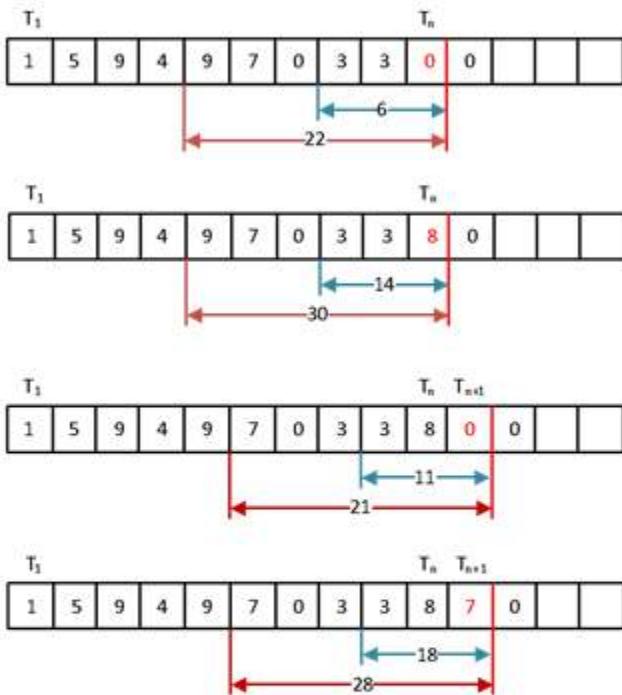
Local Field Grouping

Topology

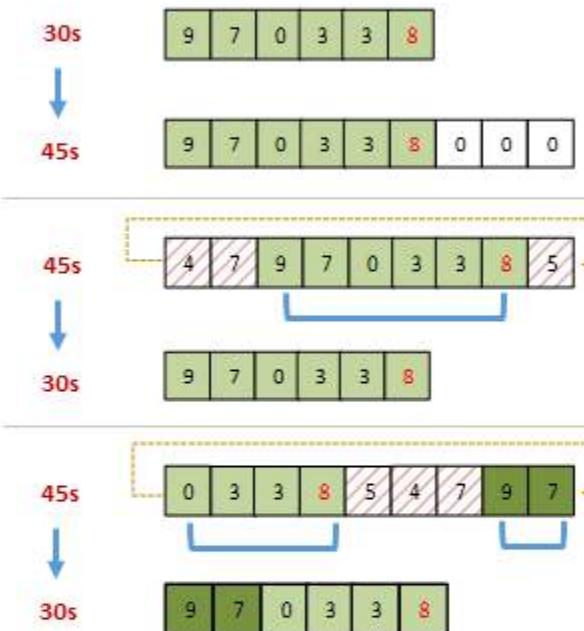


Metering

Counting

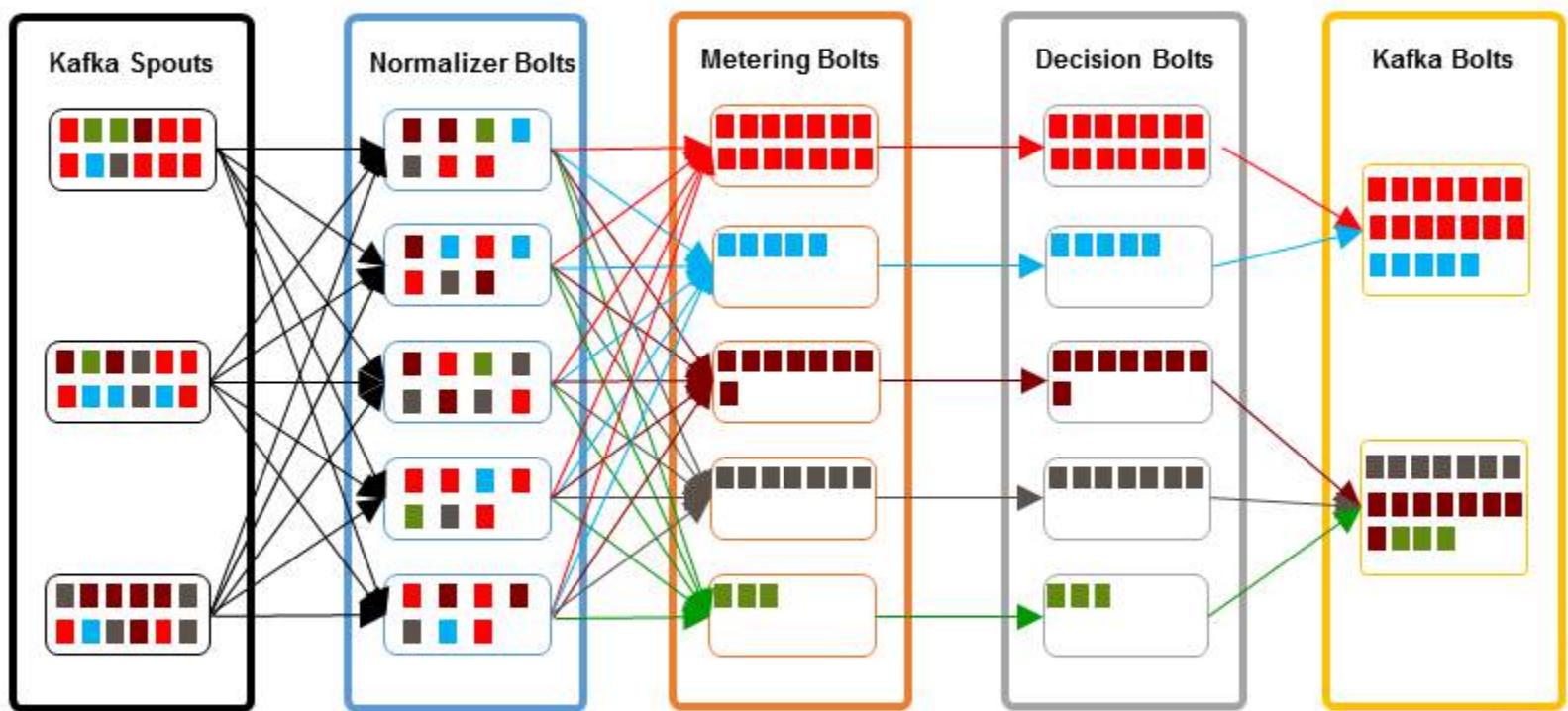


Resizing

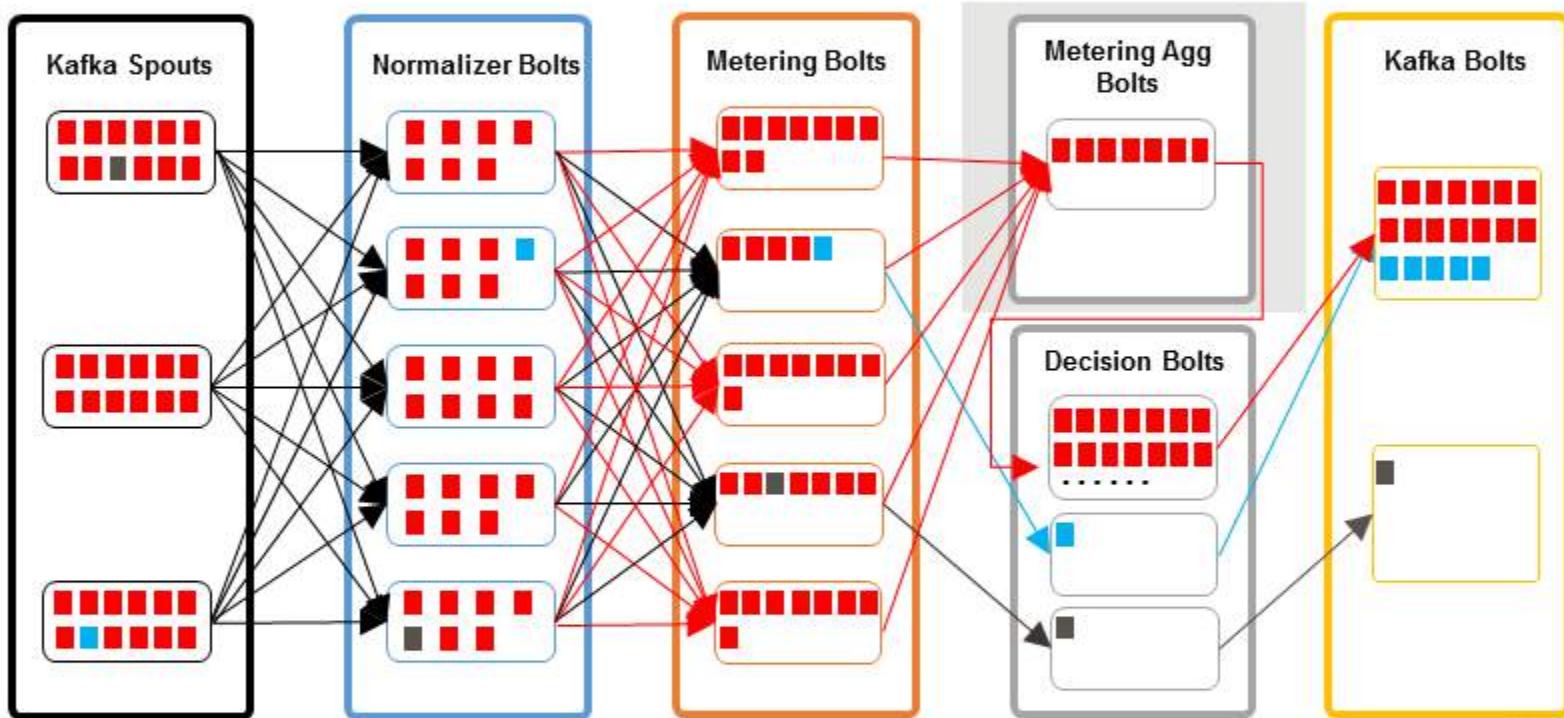


- Support **multi period** at same time
- Computation complexity from **$O(n)$** \rightarrow **$O(1)$** by caching sum value
- **Recycle** the arrays if all their values are zero

Process Flow - Normal Case



Process Flow - Attack Case



Best Practice

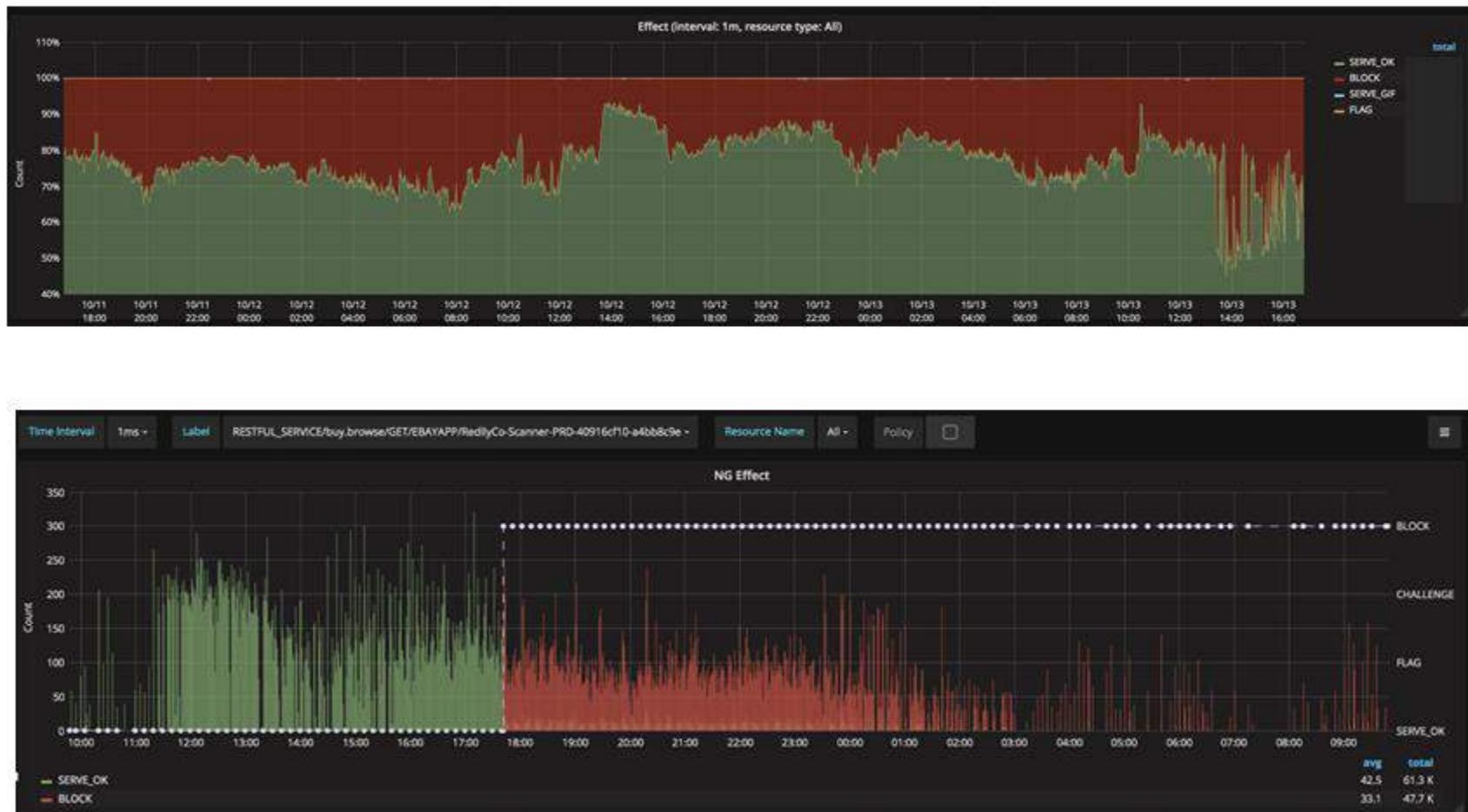
- **Storm**

- Number of Worker
- JVM Memory
- Network
- Serializer (kryo)

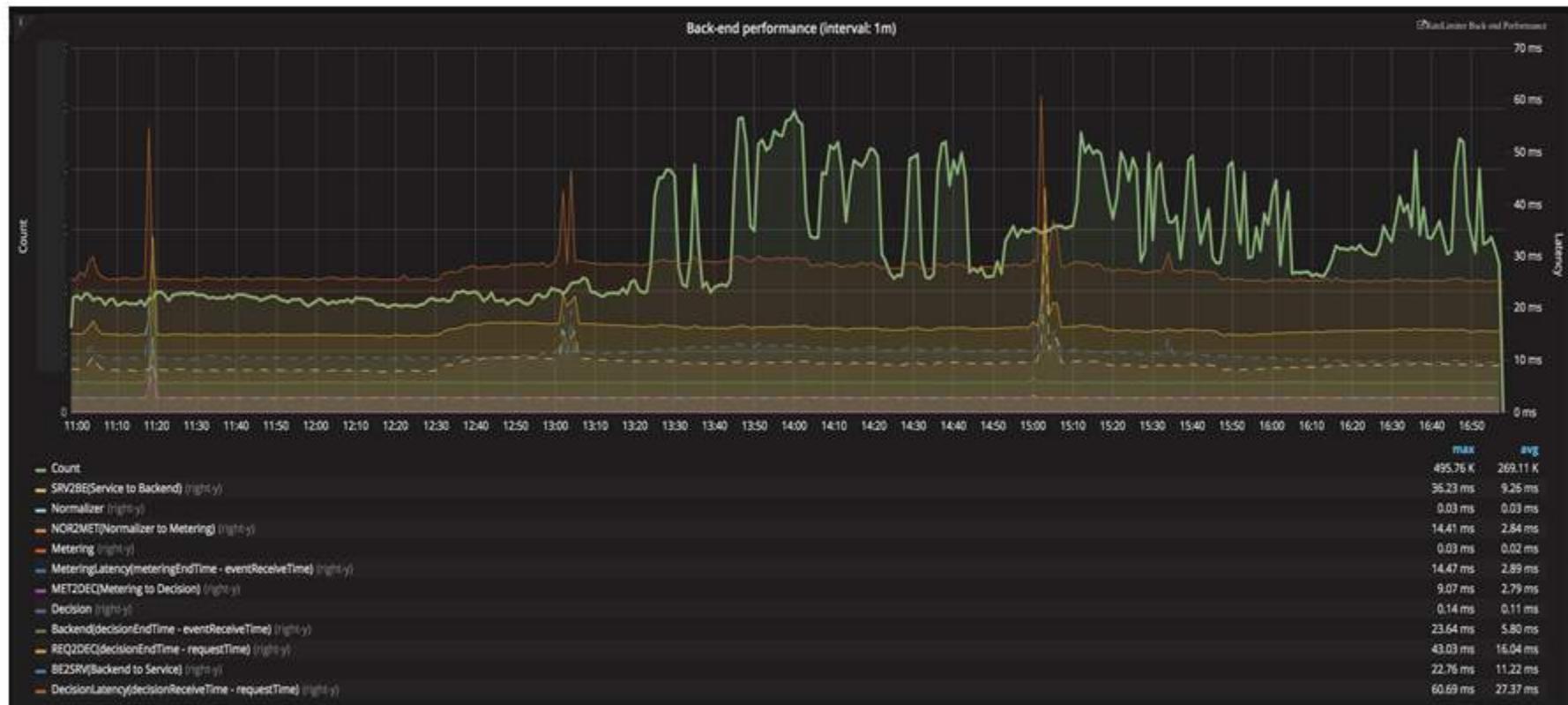
- **Kafka**

- Number of Kafka Spout
- Order of Messages
- LATEST offset

Reporting



Reporting



Future Work

- Resource Recognition
- Policy Recommendation

Summary

- Common Solution
- Asynchronized Mechanism
- Multi Calculation Periods
- Complex Rules
- Low Latency
- Self Protection

Q & A



2017 Software Architecture Summit

Scaling 100PB Data Warehouse in Cloud

Changshu LIU

Software Engineer @ Pinterest

10/2017

Pinterest: The World's Catalog of Ideas





Scale@Pinterest

Biz Scale

- 200M+ MAUs
- 120B+ Pins
- 3B+ Boards

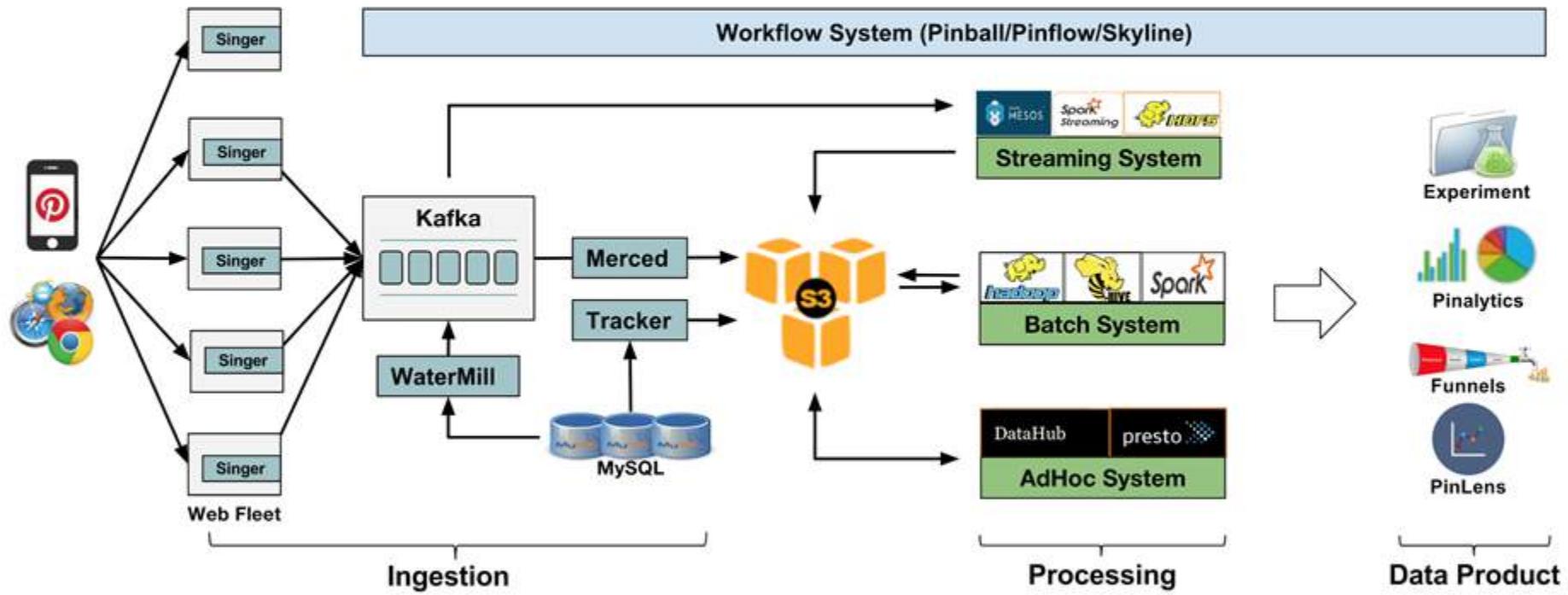
Data Scale

- 140+ PB @ S3
- 6000+ Hive/Hadoop nodes
- 400+ Presto nodes





Data@Pinterest

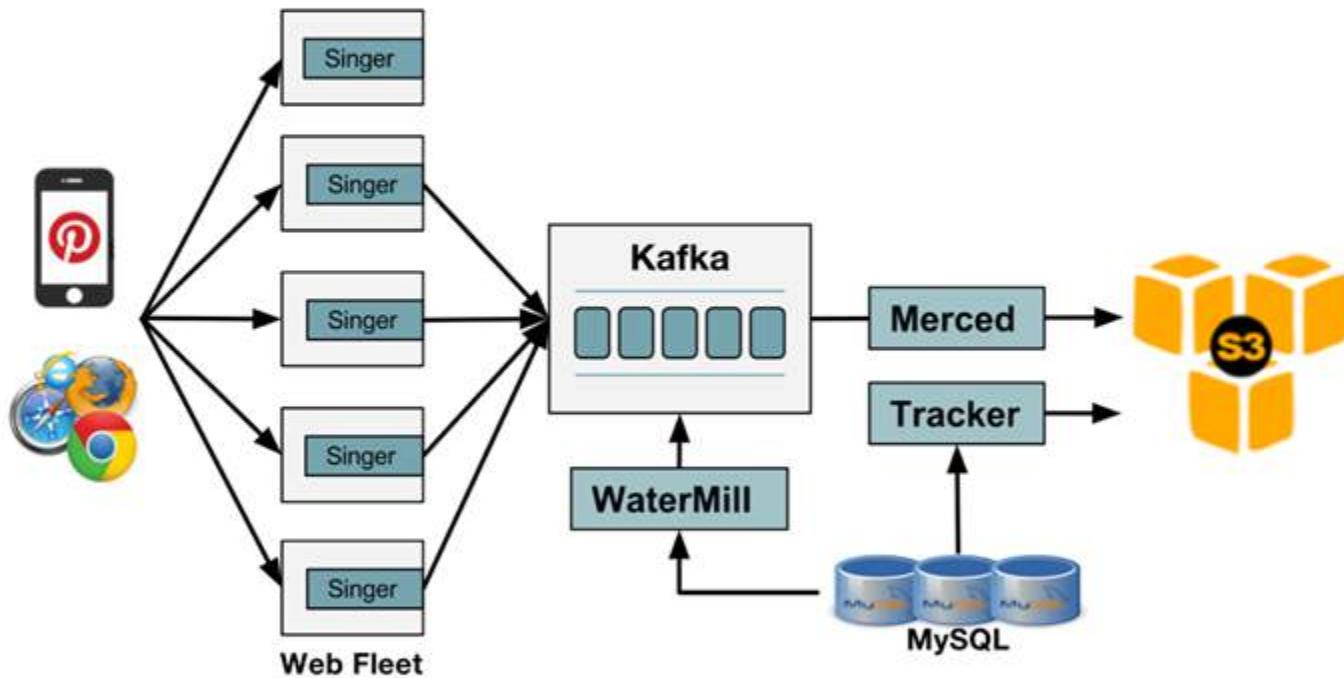


Agenda

- 1 Data Ingestion
- 2 Streaming
- 3 Hive
- 4 Presto
- 5 Workflow

1

Data Ingestion - Overview

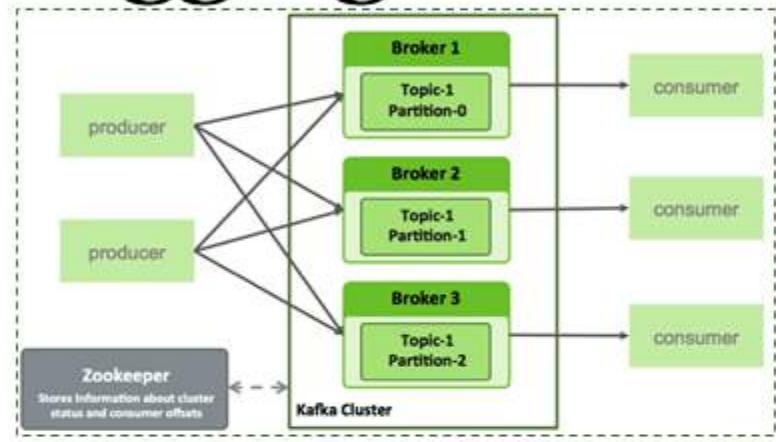


1

Data Ingestion - Logging

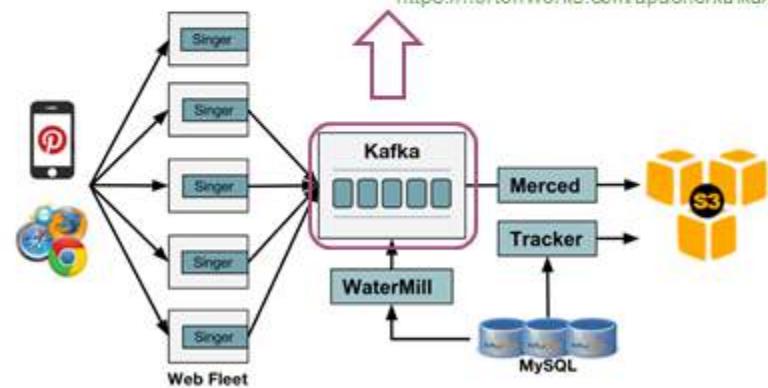
Singer

- Logging Agent for Kafka
- Decouple logging and kafka
 - Failure Isolation
 - Buffering, Pooling

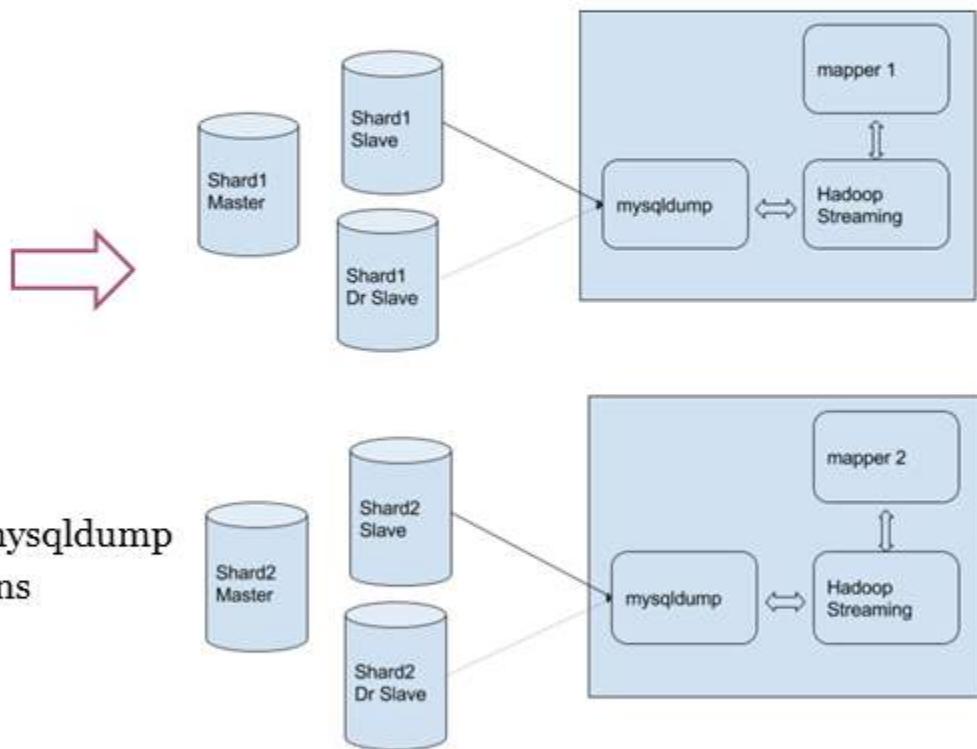
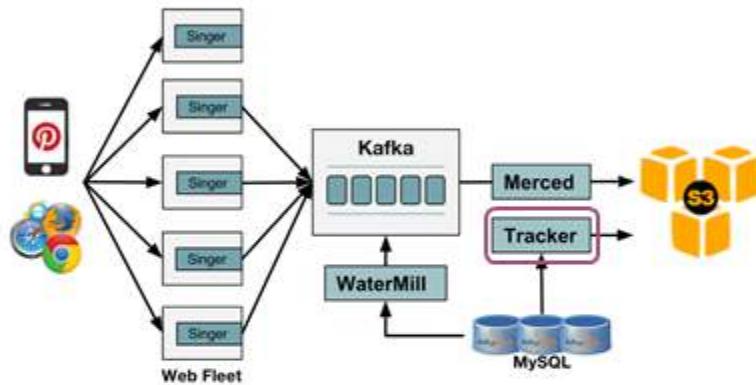


Merced

- Log Persistence and Transportation Service
- Kafka -> Storage Destinations
 - S3, HBase and SQS
- Auto rebalancing, failure handling



1 Data Ingestion - Full DB Dump

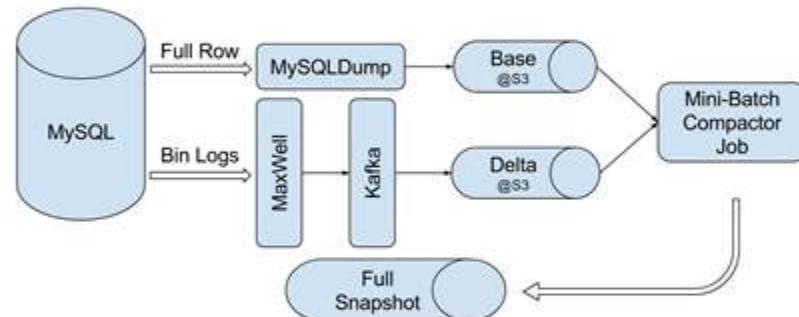
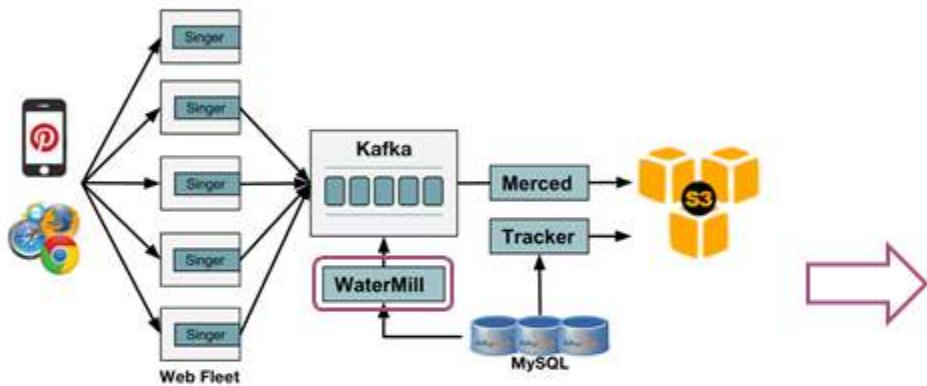


Tracker

- Daily db dump: Hadoop Streaming with mysqldump
- Hadoop node <-> MySQL node traffic pains
 - Balancing & Throttling
- Performance pains
 - Remote mysqldump
 - Python streaming

1

Data Ingestion - Inc DB Dump



WaterMill

- Maxwell tails MySQL bin logs and writes to Kafka
 - Tail bin logs and write to kafka as json
- Mini-batch job to apply updates on base version
 - Distribute records by db shard id
 - Small (update) table **join** Big (base) table
- Alias table points to the latest snapshot

1

Data Ingestion - Stats

Kafka Stats

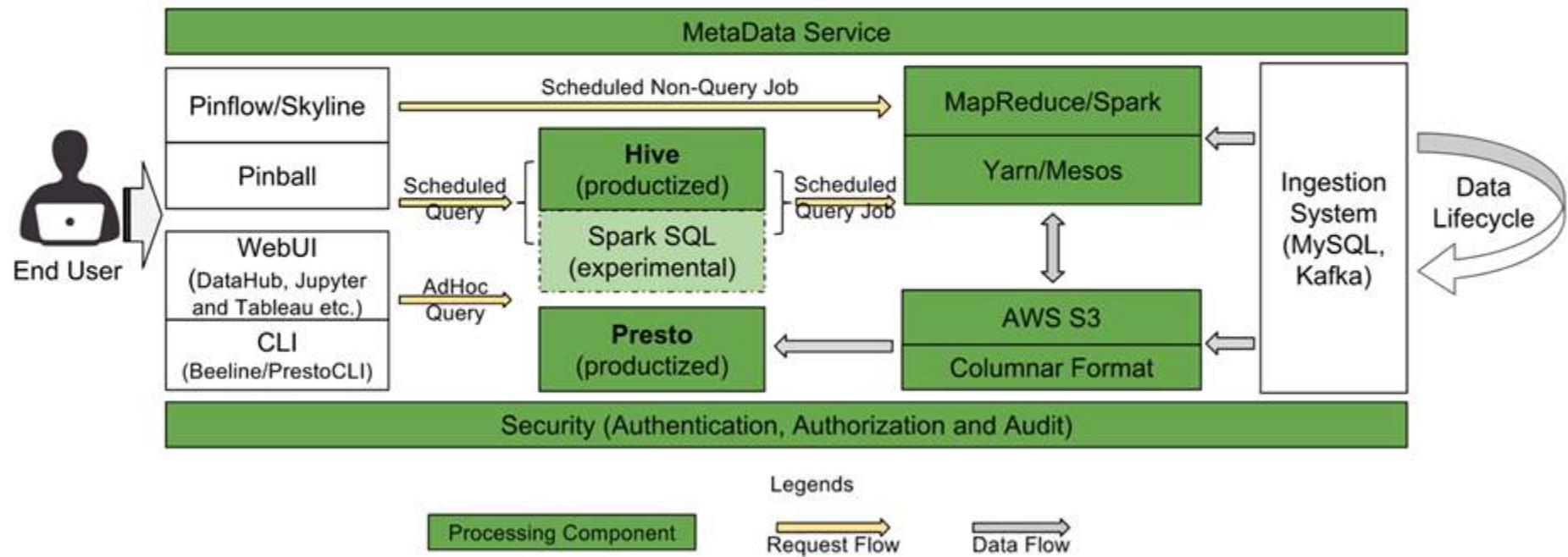
- ~150B Messages/Day
- ~400T Bytes/Day
- ~1000 Kafka Topics
- ~1400 Kafka Brokers

DB Dump Stats

- ~80 Table/Day
- ~100T Bytes/Day

Doctor Kafka <https://github.com/pinterest/doctorkafka>

Data Processing Overview

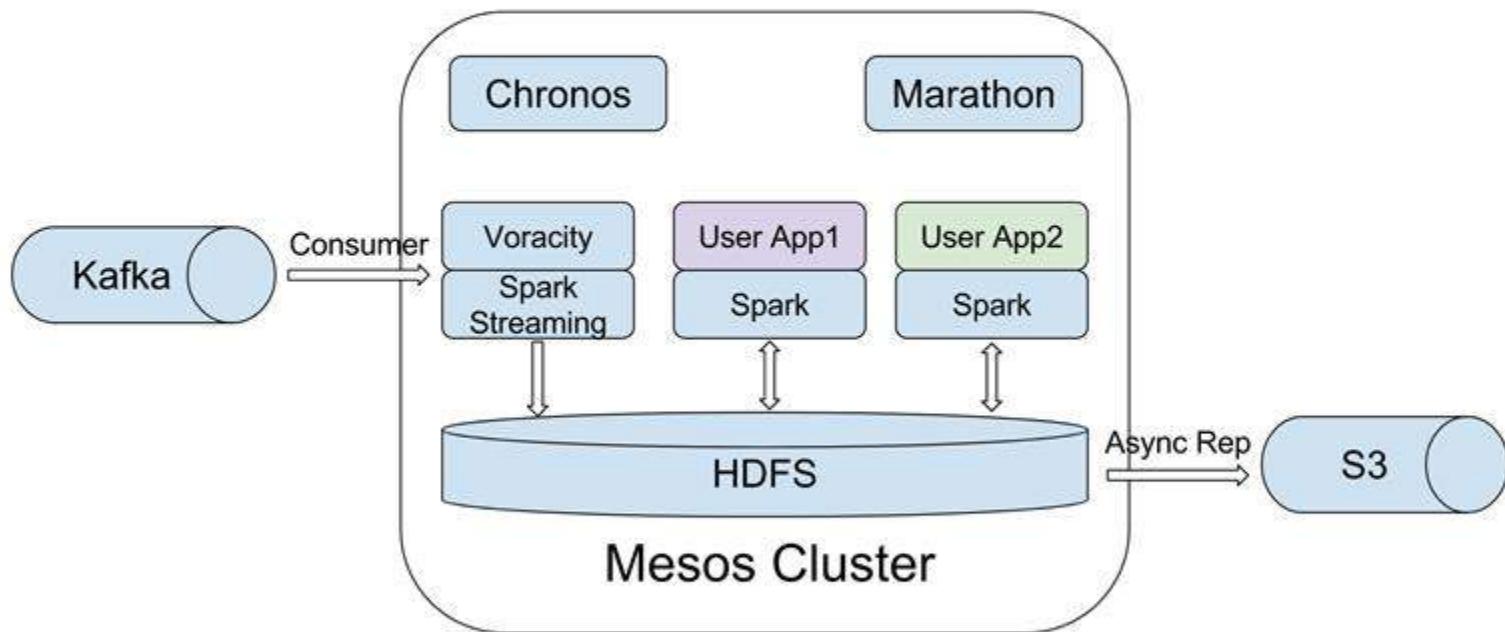


Agenda

- 1 Data Ingestion
- 2 Streaming
- 3 Hive
- 4 Presto
- 5 Workflow

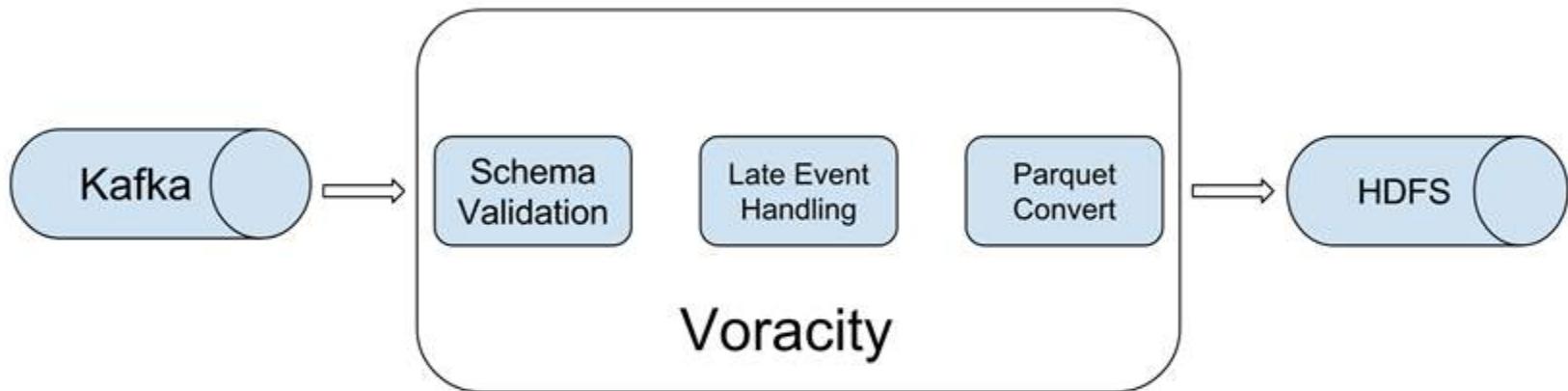
2

Streaming - Overview



2

Streaming - Voracity



- Schema Validation
 - Filter out malformed records
- Late Arrival Event Handling
 - Move late event to proper bucket
- Parquet Conversion
 - Json/Thrift -> Parquet

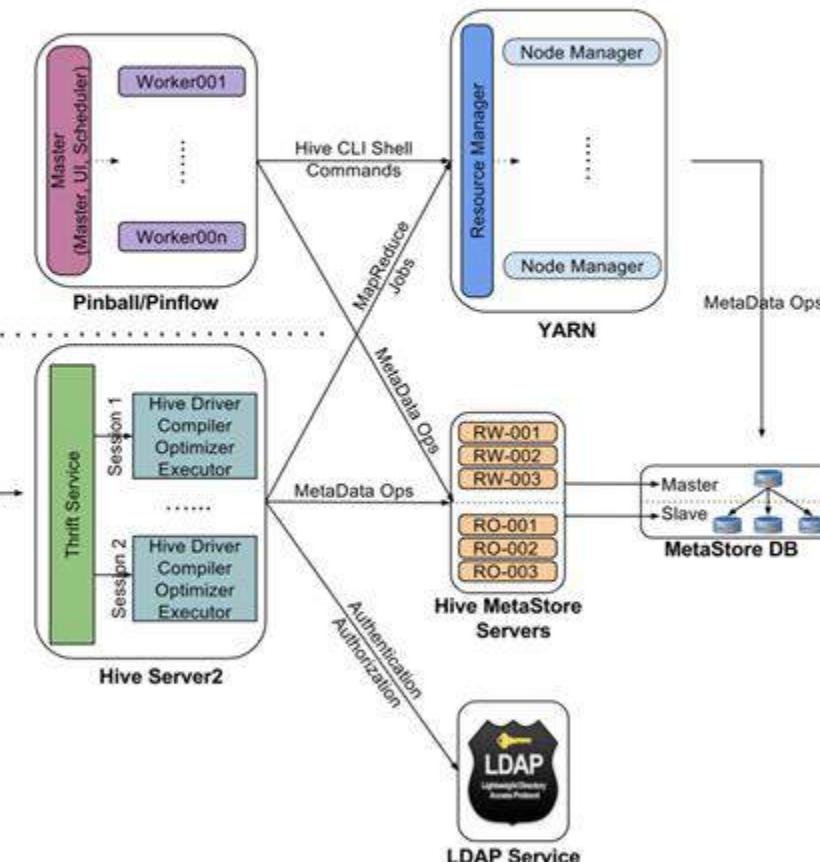
Agenda

- 1 Data Ingestion**
- 2 Streaming**
- 3 Hive**
- 4 Presto**
- 5 Workflow**

3

Hive - Overview

Schedule Query

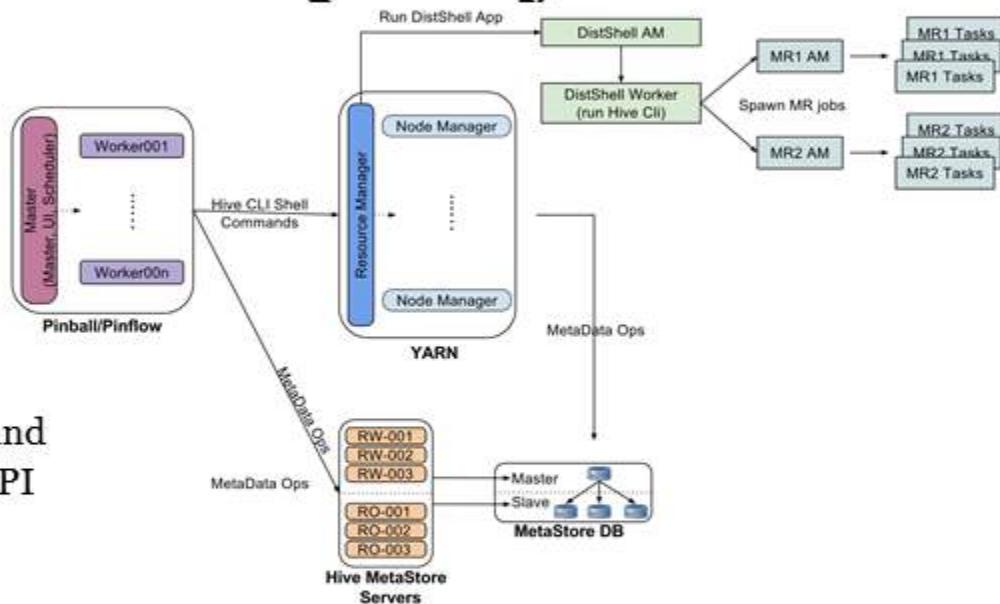


AdHoc Query

3

Hive - Scheduled Query

- Query is submitted by System
- Based on Hive CLI
- Talking to MetaStore (MySQL)
- YARN REST API
 - DistShell: AppMaster + Worker
 - Hive CLI as pure shell command
 - Get stdout/stderr from NM REST API
 - Rewrite select query to insert to s3
 - Container log is not stable



```
SELECT country, count(*)
FROM db_users
GROUP BY country
```

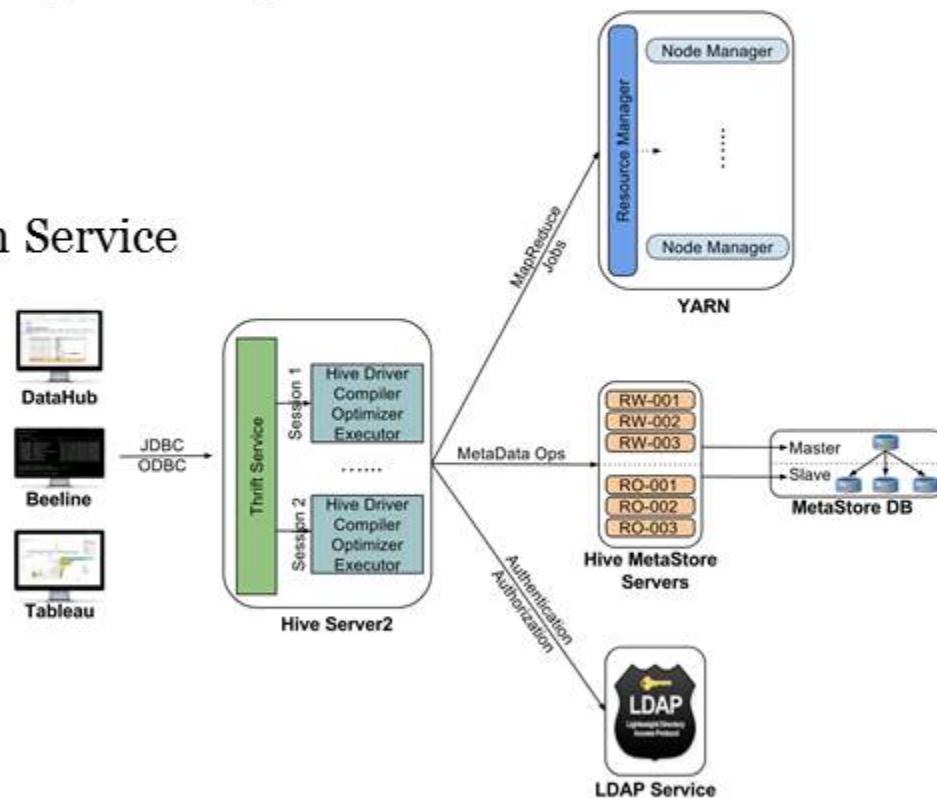
Rewrite

```
INSERT OVERWRITE DIRECTORY 's3://hive_query_output/cluster_name/20171019/application_1508263025827_2301/0'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
COLLECTION ITEMS TERMINATED BY '\002'
MAP KEYS TERMINATED BY '\003'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
SELECT country, count(*)
FROM db_users
GROUP BY country
```

3

Hive - AdHoc Query

- Query is Submitted by Human
 - Through beeline/tableau etc
- HiveServer2 is the Query Submission Service
 - via JDBC/ODBC
 - Zookeeper based HA
- Talking to Hive MetaStore Server
 - HiveServer2 is a shared resource
- LDAP for Authentication
 - Impersonation for 3rd party tool
- Hive hook to Audit all queries



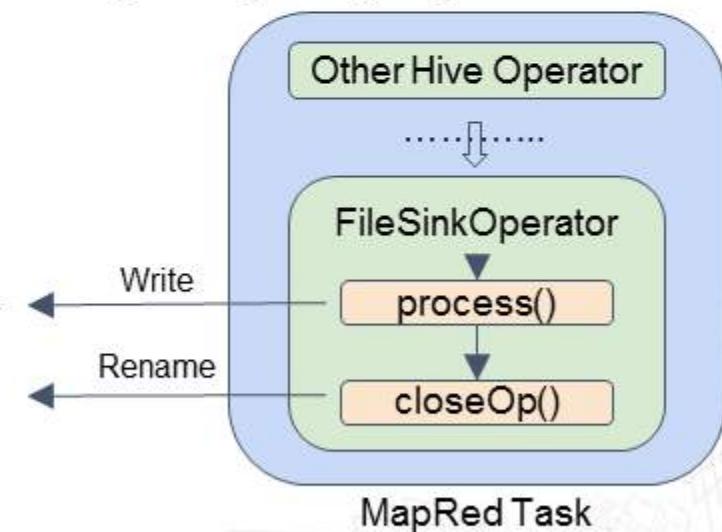
3

Hive - Cloud Integration

FileSinkOperator - Where S3 doesn't fit

- Kind of “2-phase commit”
- Phase 1 - each task attempt will write to a unique tmp output path

```
hdfs://${staging_dir}_${query_id}/  
| _task_tmp.-ext-${stage_id}/_tmp.${task_id}_${attempt_id}  
| _tmp.-ext-${stage_id}/${task_id}_${attempt_id}
```

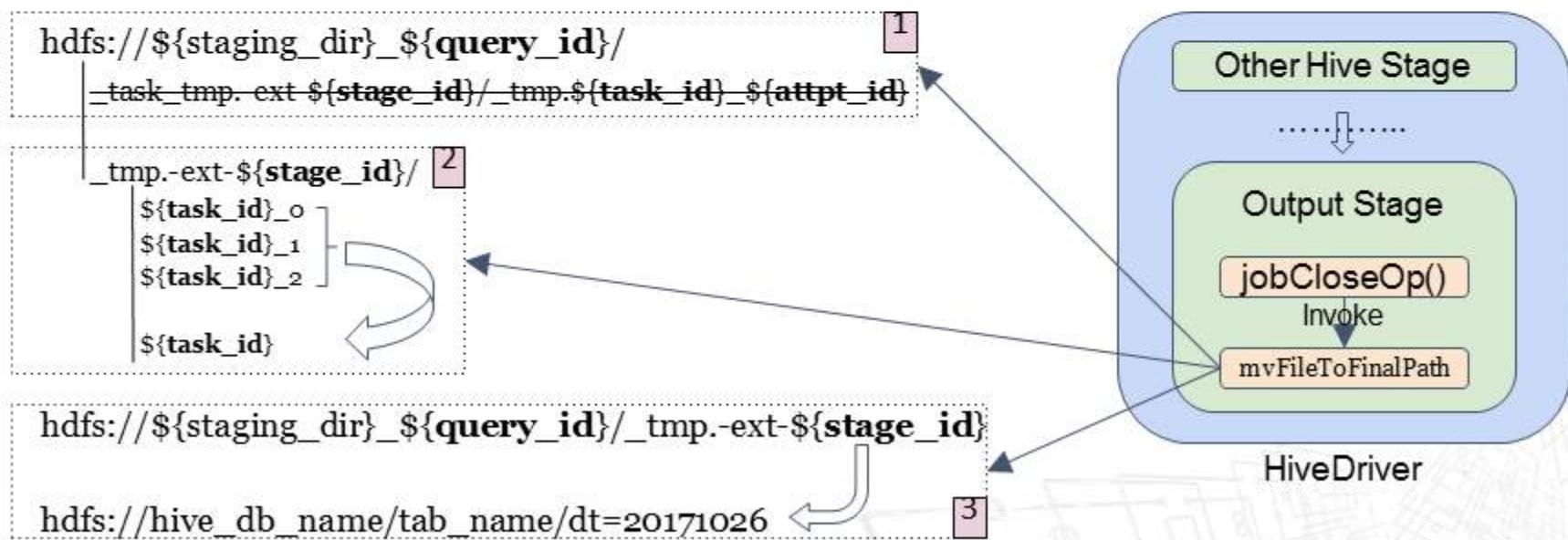


3

Hive - Cloud Integration

FileSinkOperator - Where S3 doesn't fit

- Kind of “2-phase commit” protocol
- Phase 2 - stage controller (HiveDriver) rename query temp to final path



3

Hive - Cloud Integration

FileSinkOperator - Where S3 doesn't fit

- Fast & Atomic file/dir **rename** operation is the key
- Unfortunately, rename on s3 is copy & delete

3

Hive - Direct Committer @ S3

Direct Output Committer

- Each task attempt writes to final output directly
 - {target folder}/{task_id}
- Consequences
 - Query Level Failure -> partial output in target folder
 - Cleanup target folder in the very beginning of each query
 - Disable INSERT INTO (using INSERT OVERWRITE Q1 UNION ALL Q2)
 - Task Level Failure -> FileAlreadyExistsException
 - File open mode CREATE -> OVERWRITE (ORC & Parquet)
 - Speculative Execution -> corrupted task output
 - Cause: outWriter.close() in killed task attempt
 - S3A write to local disk first and then upload to s3 in close()
 - Solution: Disable speculative execution

3

Hive - Speculative Exec @ S3

Direct Committer with Speculative Execution

- S3A Staging Committer (originated from Netflix)
 - Based on S3 multipart uploading [1](#)
 - Decouple upload & commit, simulate write to tmp & rename to final
 - Being actively worked in Apache: [HADOOP-13786](#)
- S3A Hive Committer (originated from Pinterest, on-going)
 - Ignore the MapReduce commit protocol
 - Allow different task attempts to commit at the same time
 - Stage controller (HiveDriver) dedup multiple attempts for the same task

3 Hive - Dynamic Partition @ S3

New Challenges

- Target folder path is only known at runtime
- Race condition among multiple FinkSinkOperators
- Discover new dynamic partitions to update Metastore

Solutions

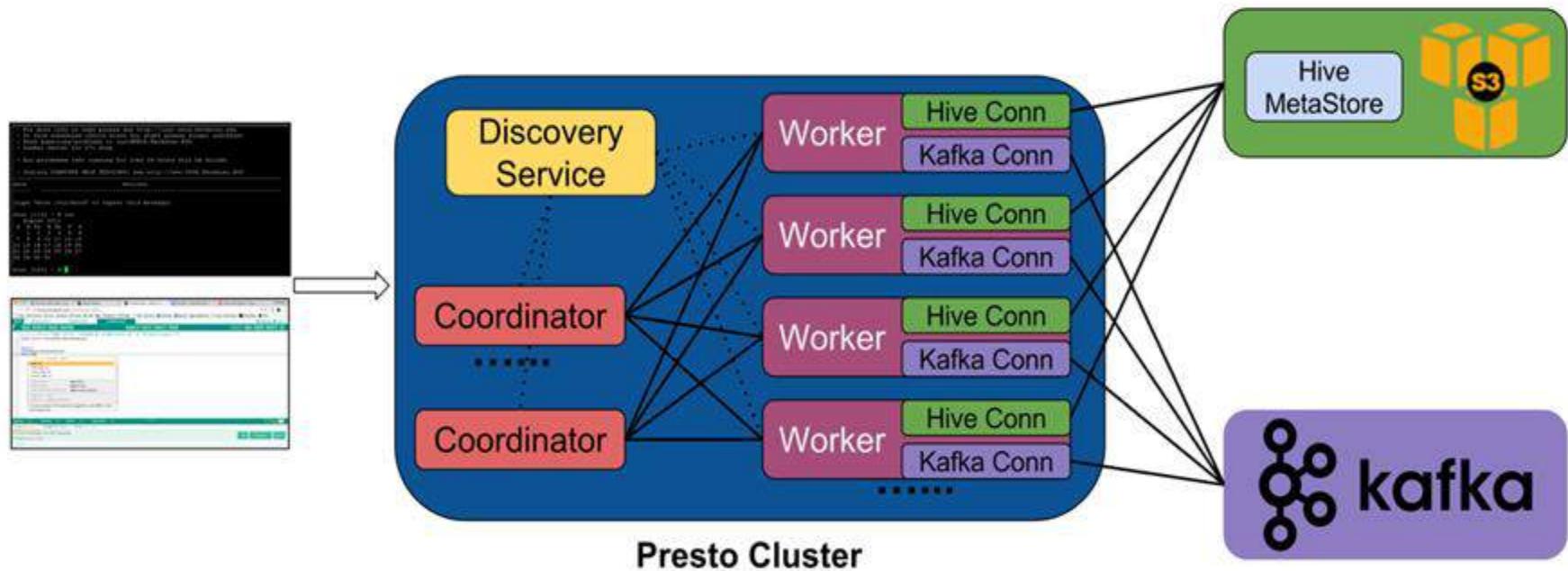
- Encode the ts into output file name: {target_part_folder}/{dpCtx.queryStartTs}_{task_id}
- New dynamic partition: folders that
 - Modification time > query start time
 - Contains files starting with query_start_ts

Agenda

- 1 Data Ingestion**
- 2 Streaming**
- 3 Hive**
- 4 Presto**
- 5 Workflow**

4

Presto - Overview



4

Presto - Cluster & Stats

Cluster Management

- Instance type: r3.8xl
- Computing only, no local data
- Using AWS ASG
- Managed by Terraform

Presto Stats

- ~800 users
- ~10k queries daily
- 5X ~ 50X faster than Hive
- AdHoc (250 node): P90: 4min, P99: 30min
- App (150 node): P90: 2min, P99: 20min

4

Presto - Feature Challenges

Concurrency Bug in ObjectInspector library

- Hive runtime is for single thread env
- Presto is highly concurrent and shared

Thrift Table Support

- Presto Hive MetaStore client doesn't support "dynamic" table
- Deeply nested table will hang in planning phase

Schema Evolution Support

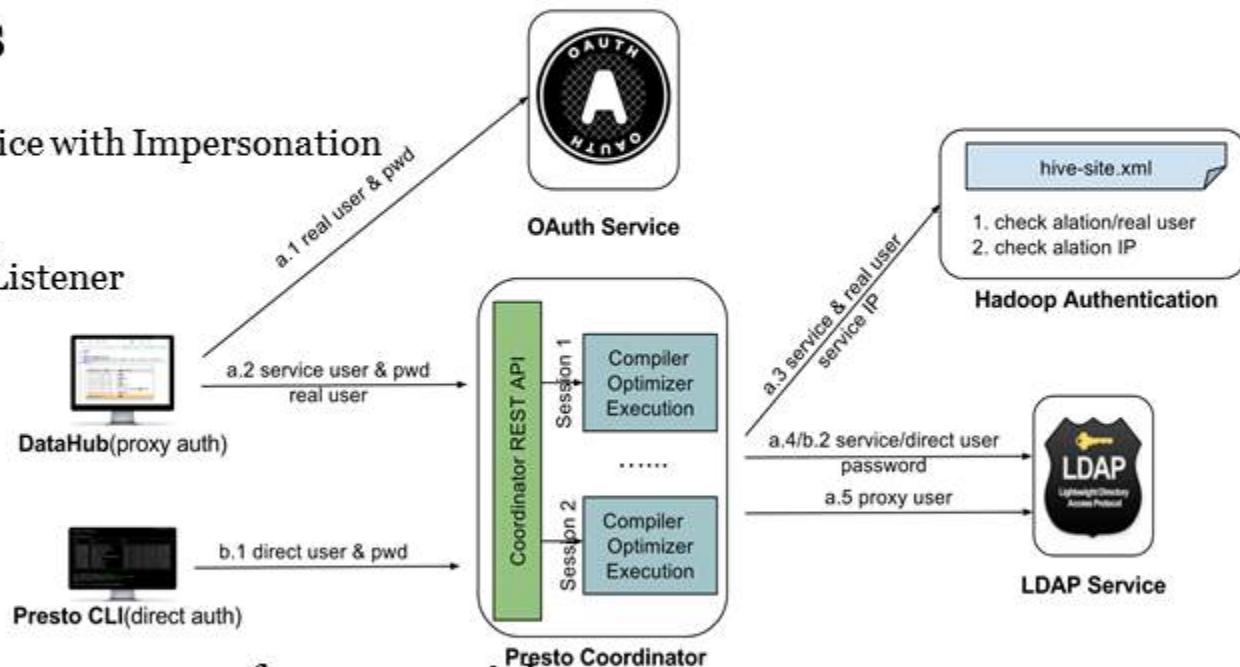
- Historical partition schema mismatch
- Small int -> big int
- Parquet table column rename

4

Presto - Security Enhancement

Security Features

- Authentication
 - Integrated middle service with Impersonation
- Audit
 - Every Query is Logged
 - Plugin based on EventListener



Access Control

- Check LDAP group for proxy user for access rights
- Data Level Access is controlled by IAM role

4

Presto - Cluster Operations

Separate AdHoc & Application Cluster

- Adhoc cluster is generally less stable
- App cluster only run well known queries

Bad Query Monitor

- Detected by bot based on runtime and splits #
- Kill the query and send slack message to end user

Bad Node Monitor

- Detected by bot based on query failure due to node issue stats
 - Bad node - 3 query failed on it in last 5 min
- Specially important when the cluster is scaled up & down on a daily basis

Agenda

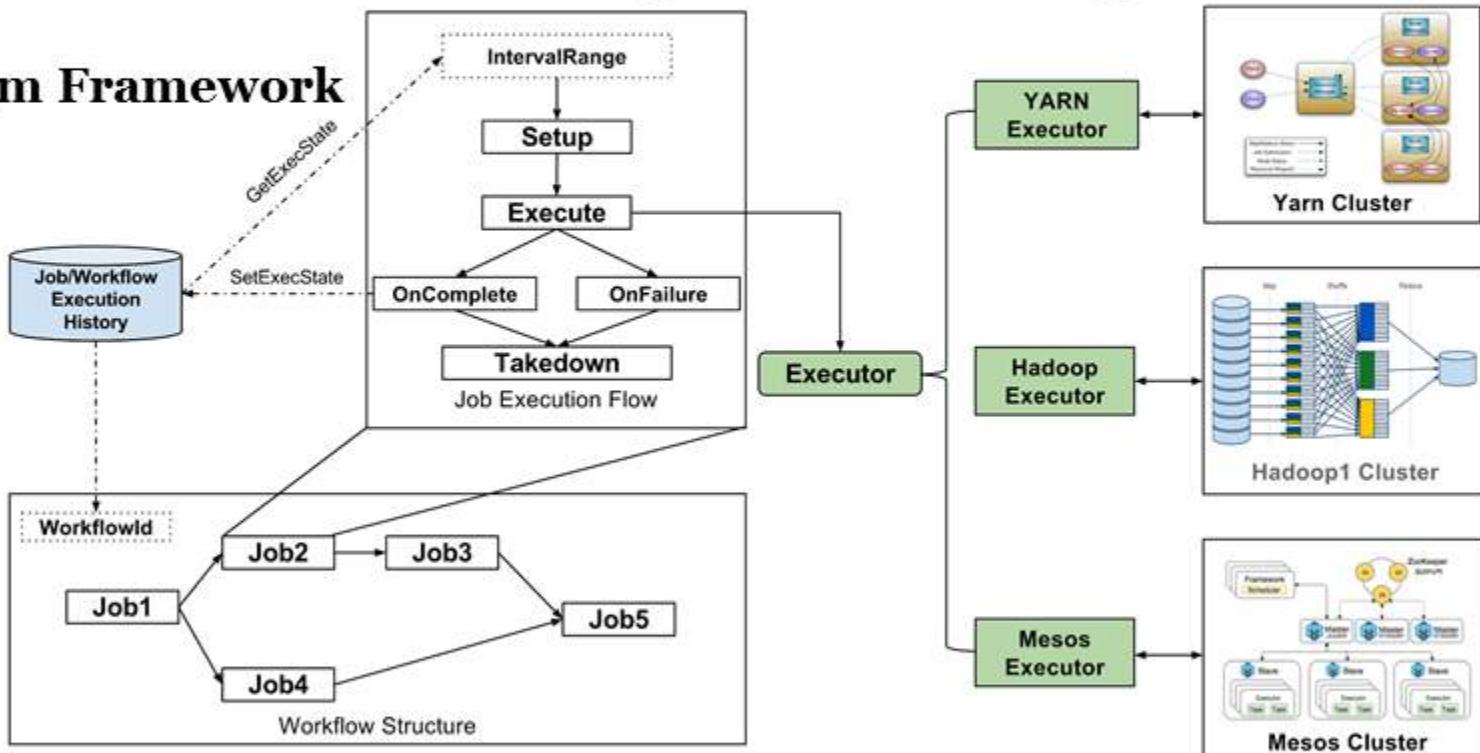
- 1 Data Ingestion**
- 2 Streaming**
- 3 Hive**
- 4 Presto**
- 5 Workflow**

5

Workflow - Programming

Workflow Prgm Framework

- Skyline
- DataJob
- Pinflow



5

Data Architecture - Manager

Workflow Manager

- Pinball
- <https://github.com/pinterest/pinball>

Workflow Stats

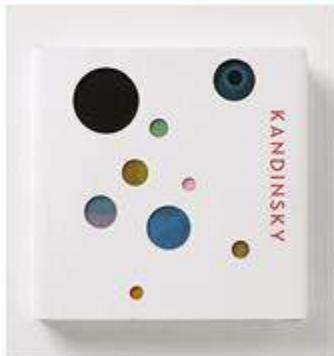
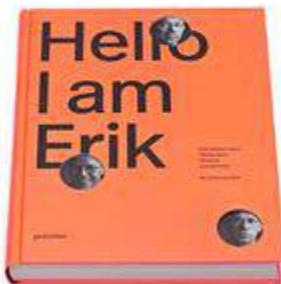
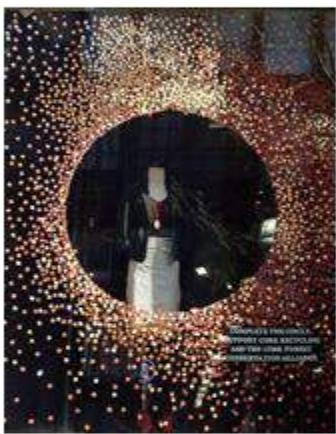
- 2000+ Workflows
- 60k+ Jobs
- ~80% Hive
- ~15% Cascading/Spark
- ~5% Python





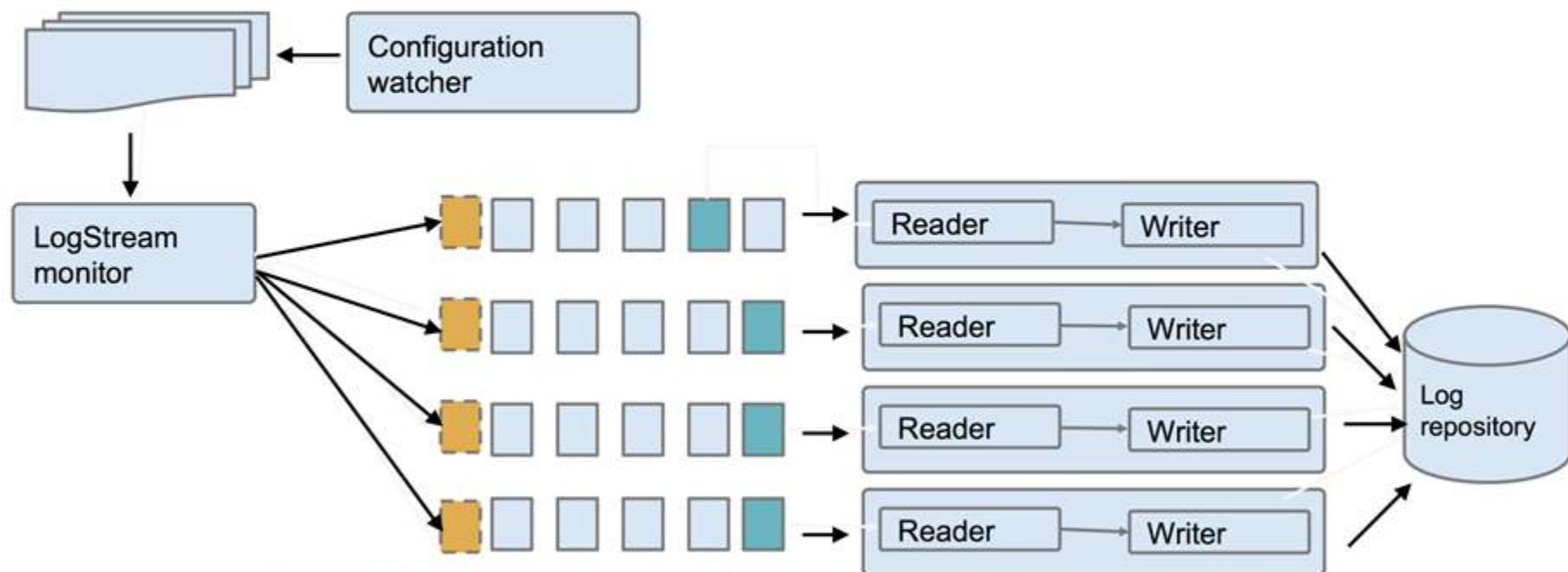
The World's Catalog of Ideas

Discover Ideas



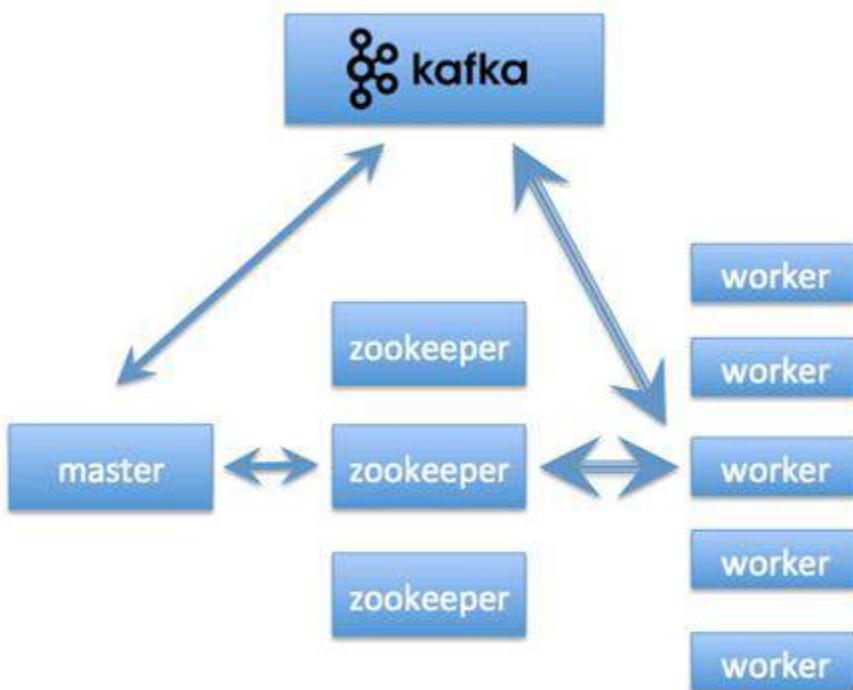
Singer Architecture

Log configuration



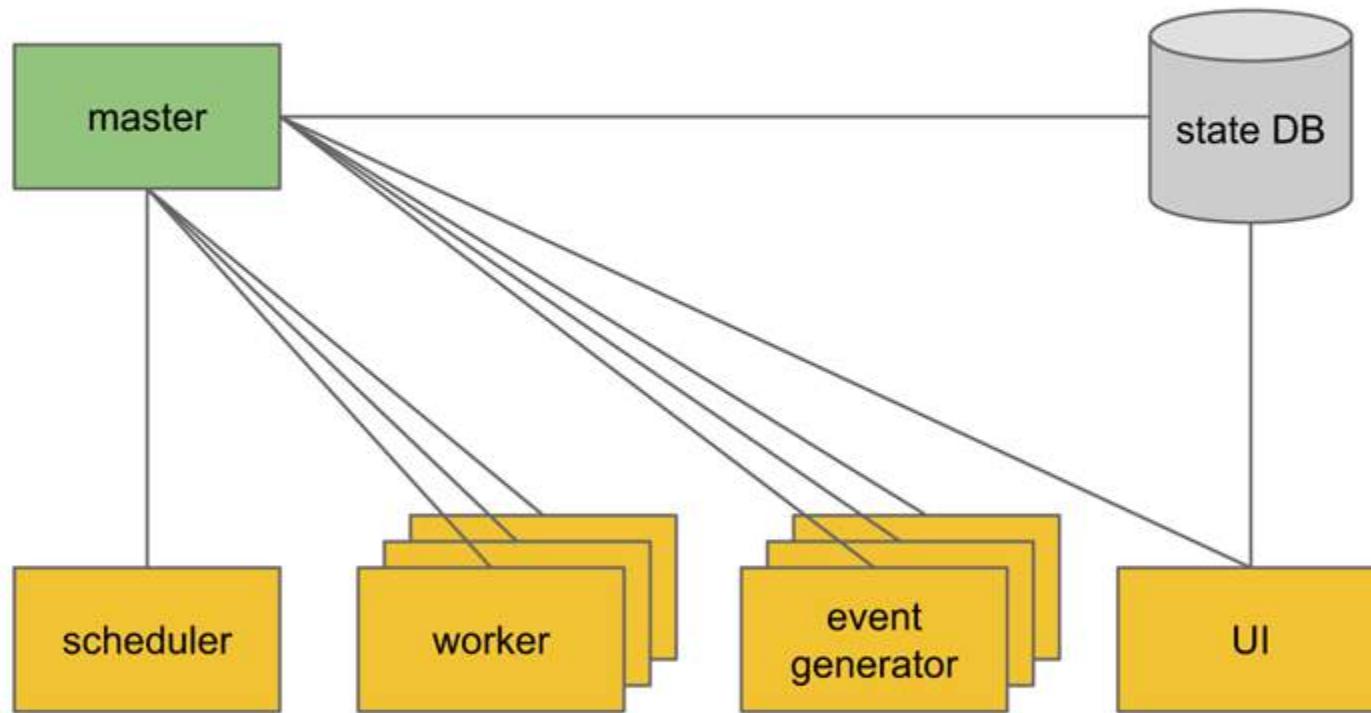
<https://www.slideshare.net/DiscoverPinterest/singer-pinterests-logging-infrastructure>

Merced Architecture



<https://github.com/pinterest/secor>

Pinball Architecture



<https://github.com/pinterest/pinball>

Uber实时处理系统的演化

袁泳@Uber

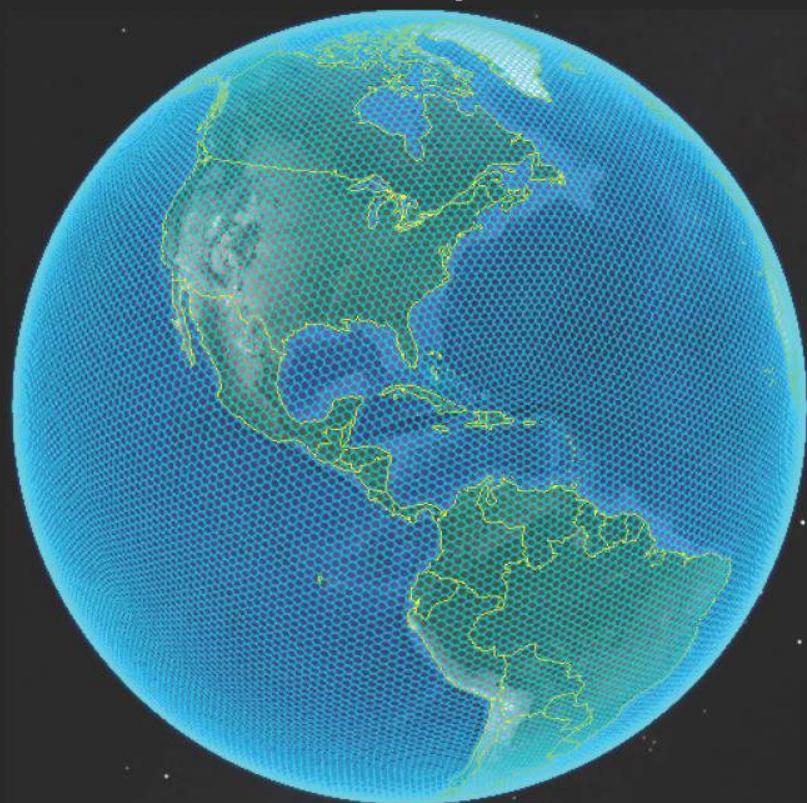


Four Kinds of Analytics

- On demand aggregation and pattern detection
- Clustering
- Forecasting
- Pattern detection on geo-temporal data

Two Ingredients

Geo/Spatial



Time



Real-time aggregation and pattern matching

Complex Event Processing

Examples

How many cars enter and exit a **user defined area** in past 5 minutes

CEP with full historical context

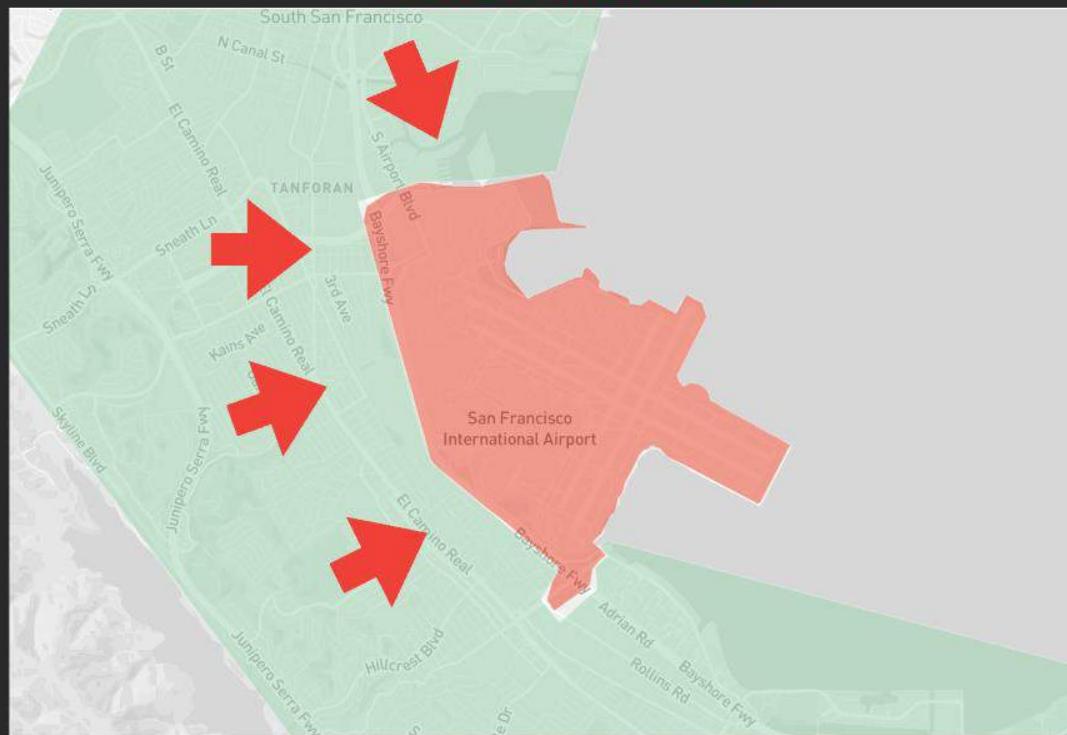
Notify me if a partner completed her 100th trip in a given area just now?

Patterns in the future

How many **first-time riders** will be **dropped off** in a given area in the next 5 minutes?

Patterns in the future

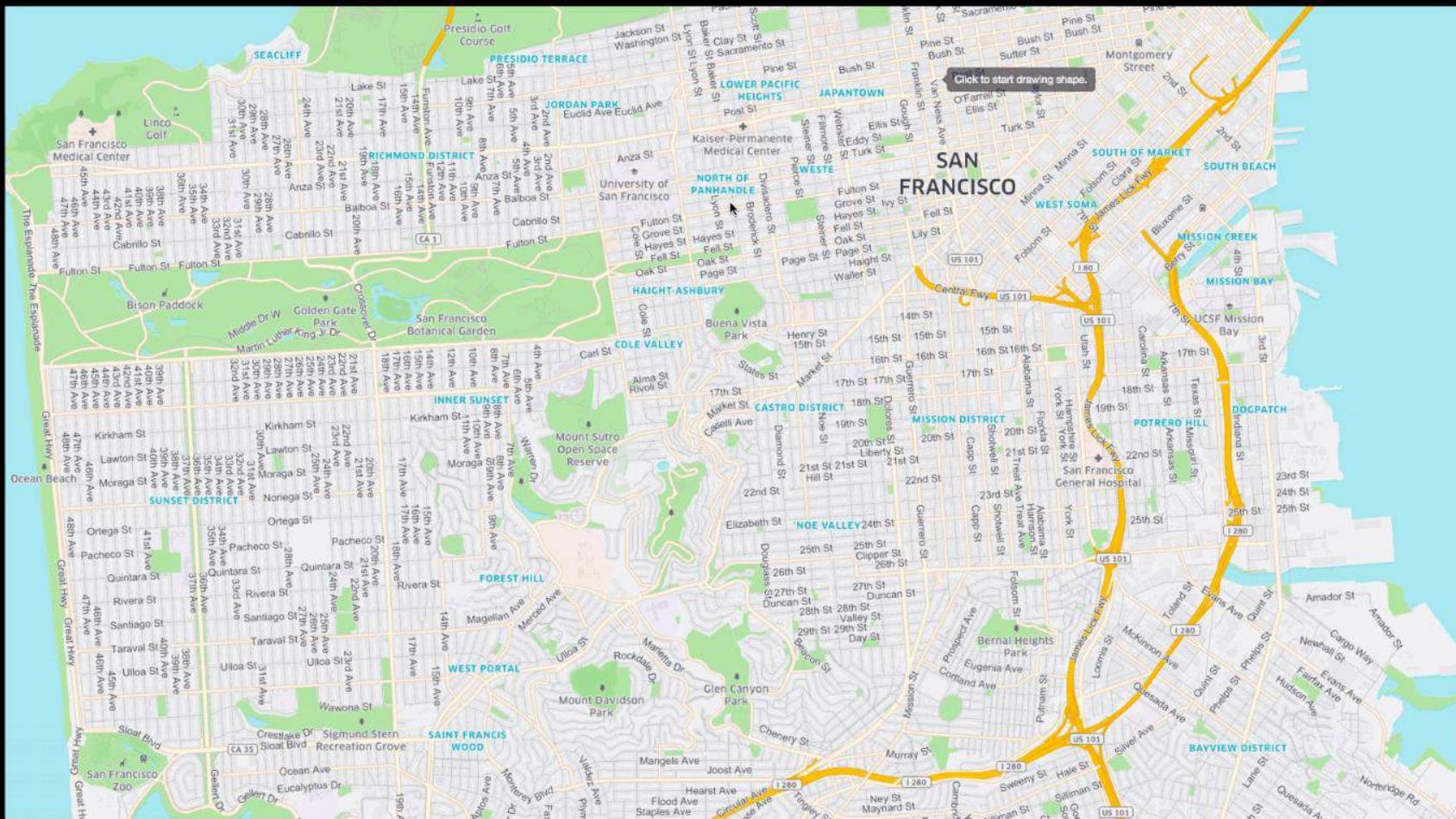
How many **first-time riders** will be **dropped off** in a given area in the **next 5 minutes?**



Geo: user flexibility is important



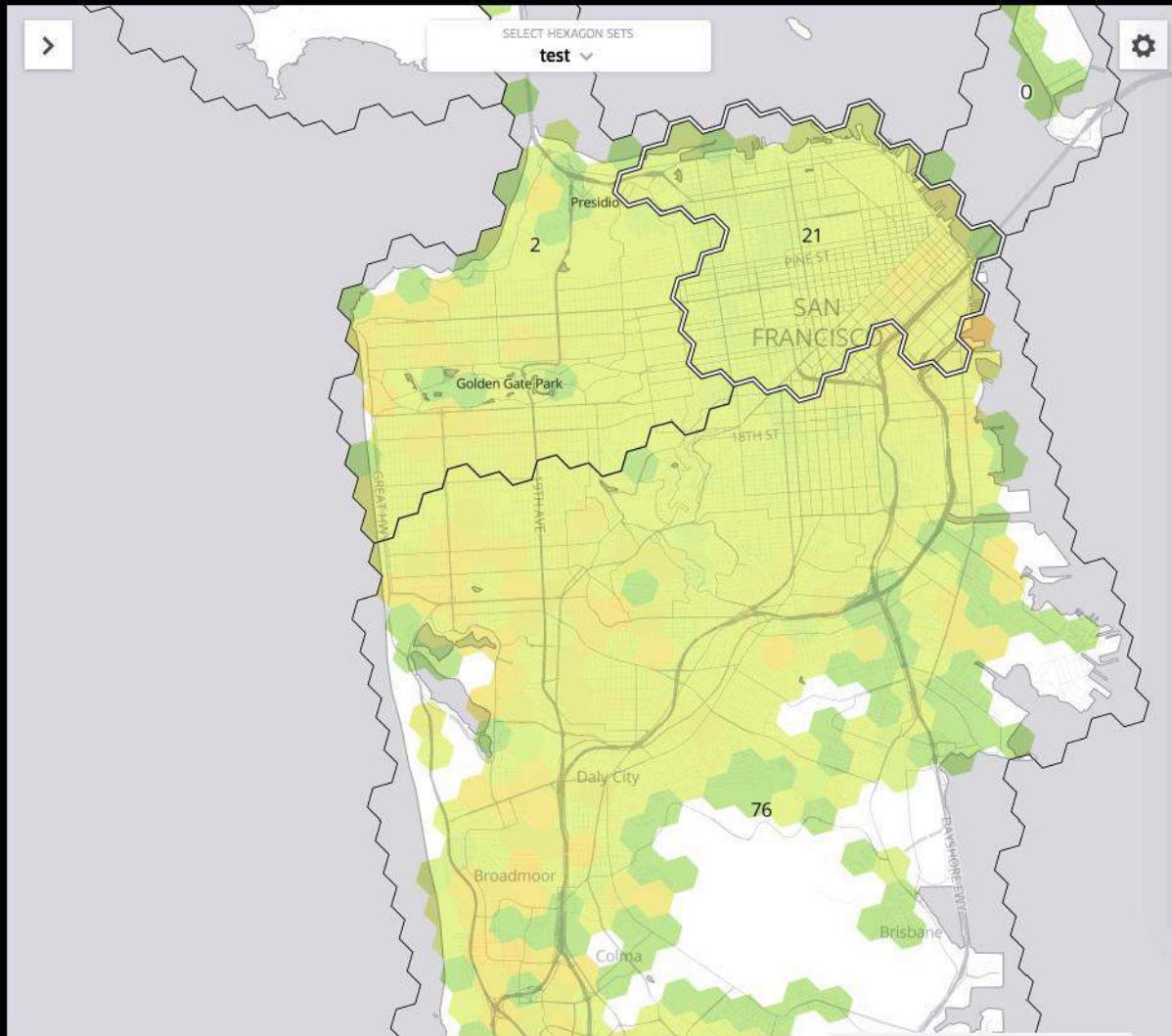
Geo: user flexibility is important



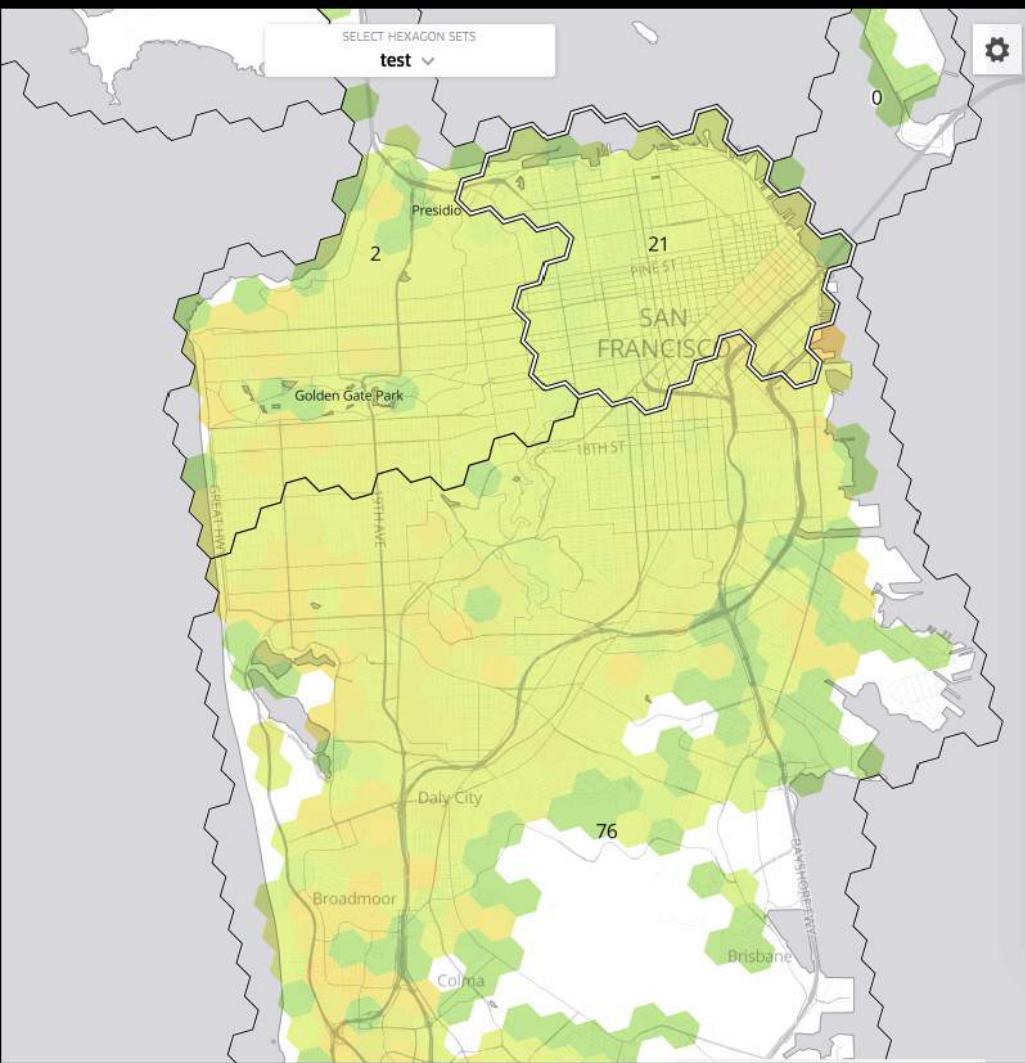
It needs to be scalable



It needs to be scalable



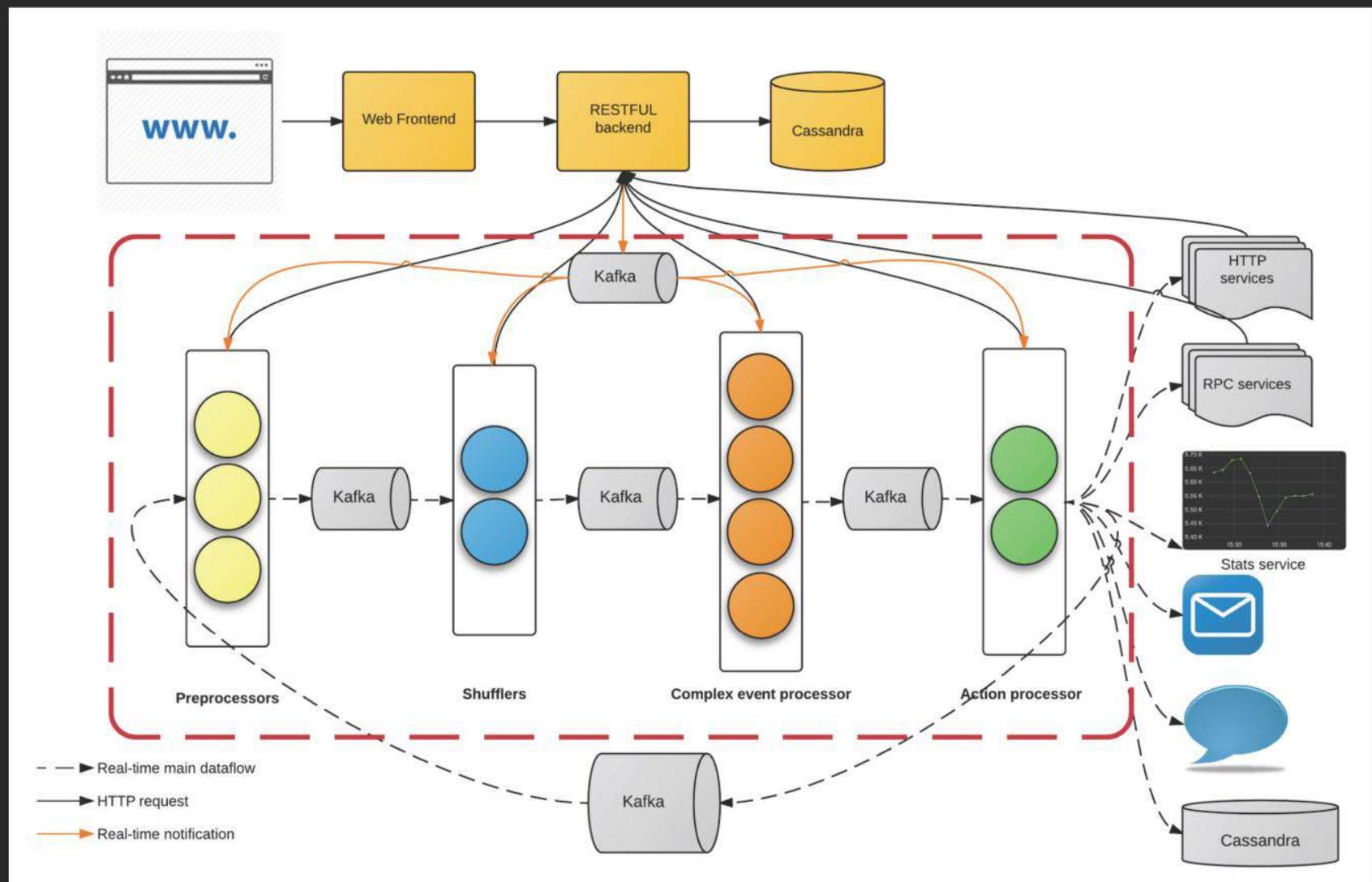
It needs to be scalable

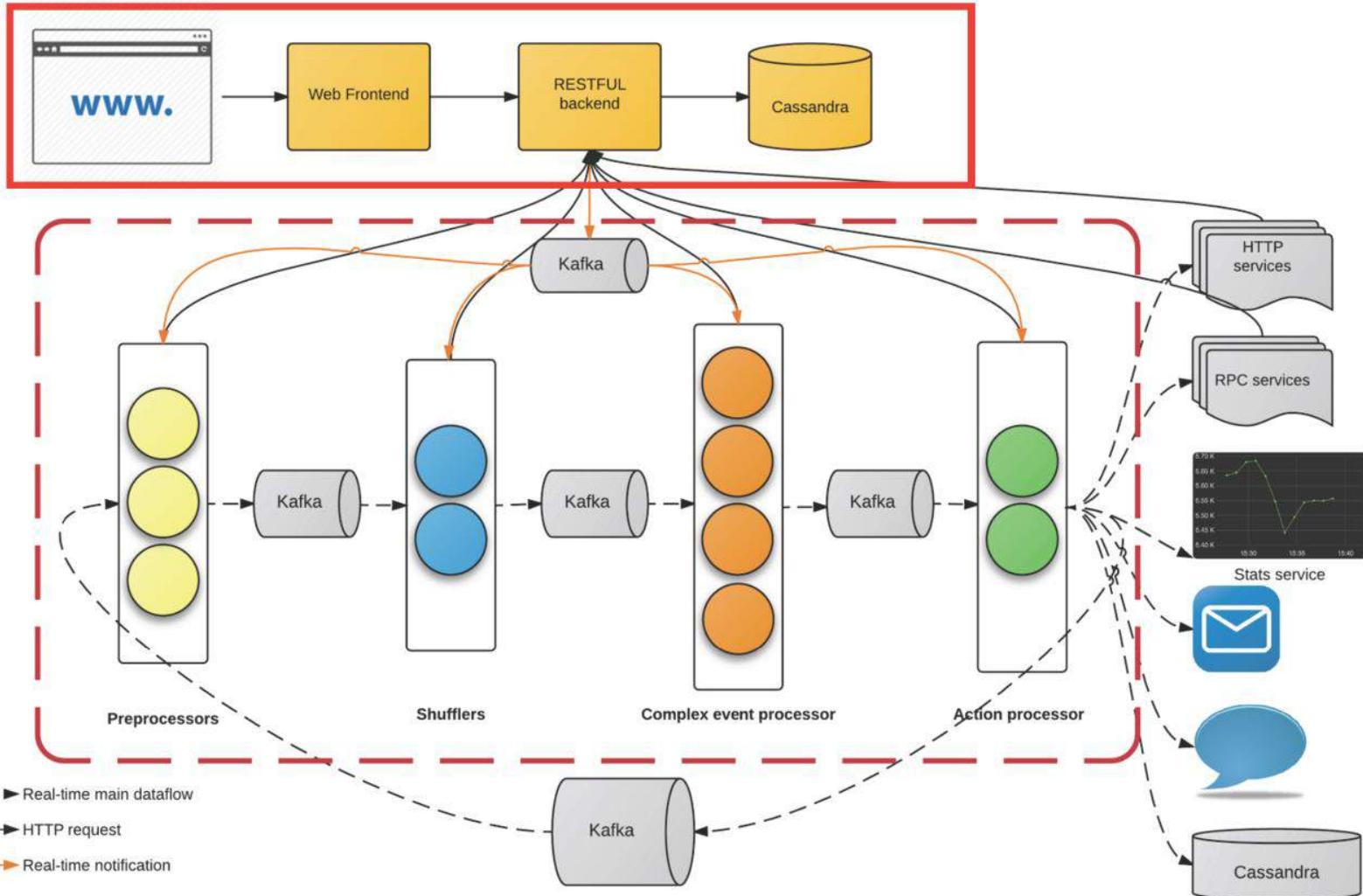


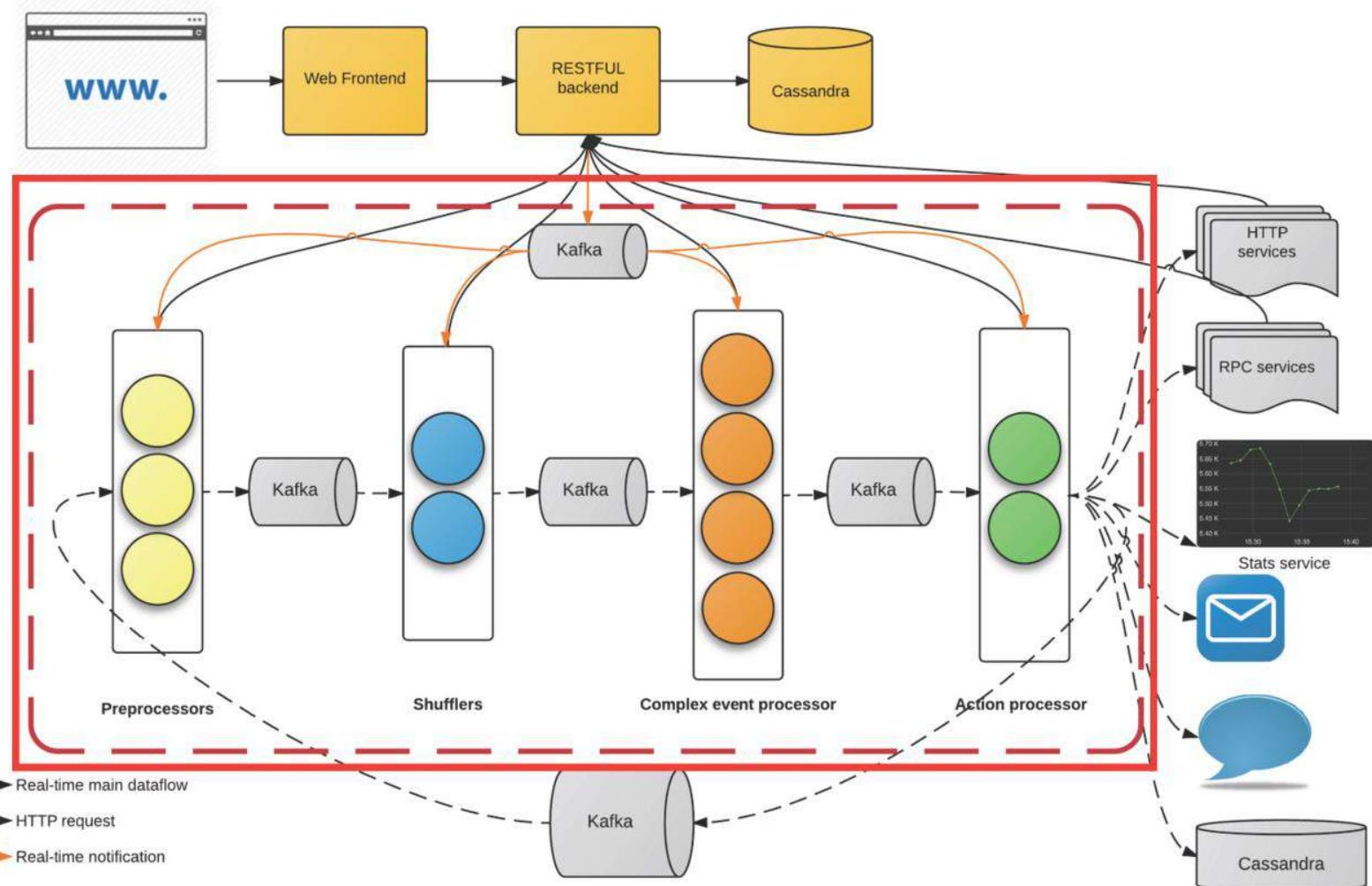
- Every hexagon
- Every driver/rider

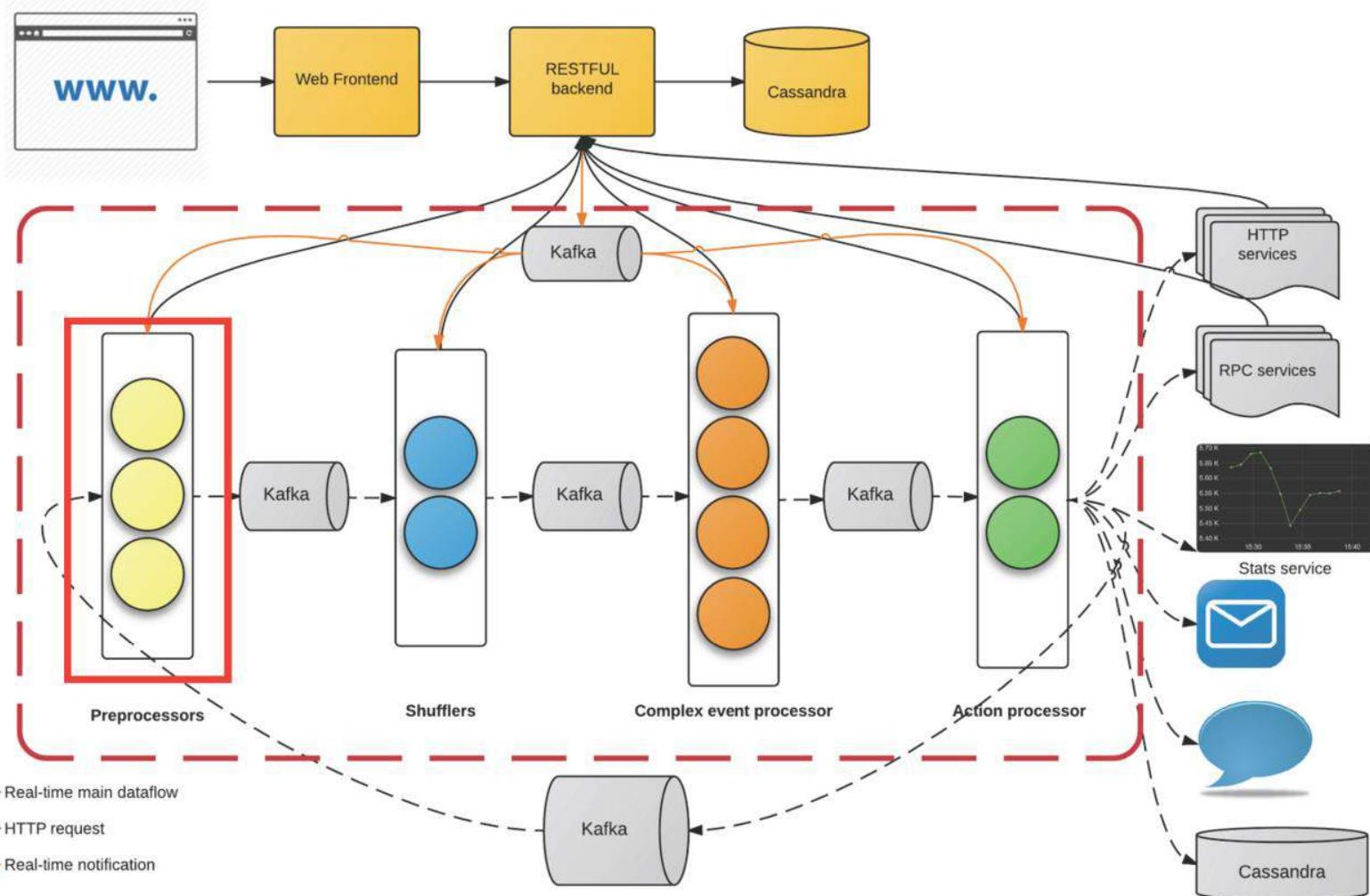
CEP Pipeline Built on Samza

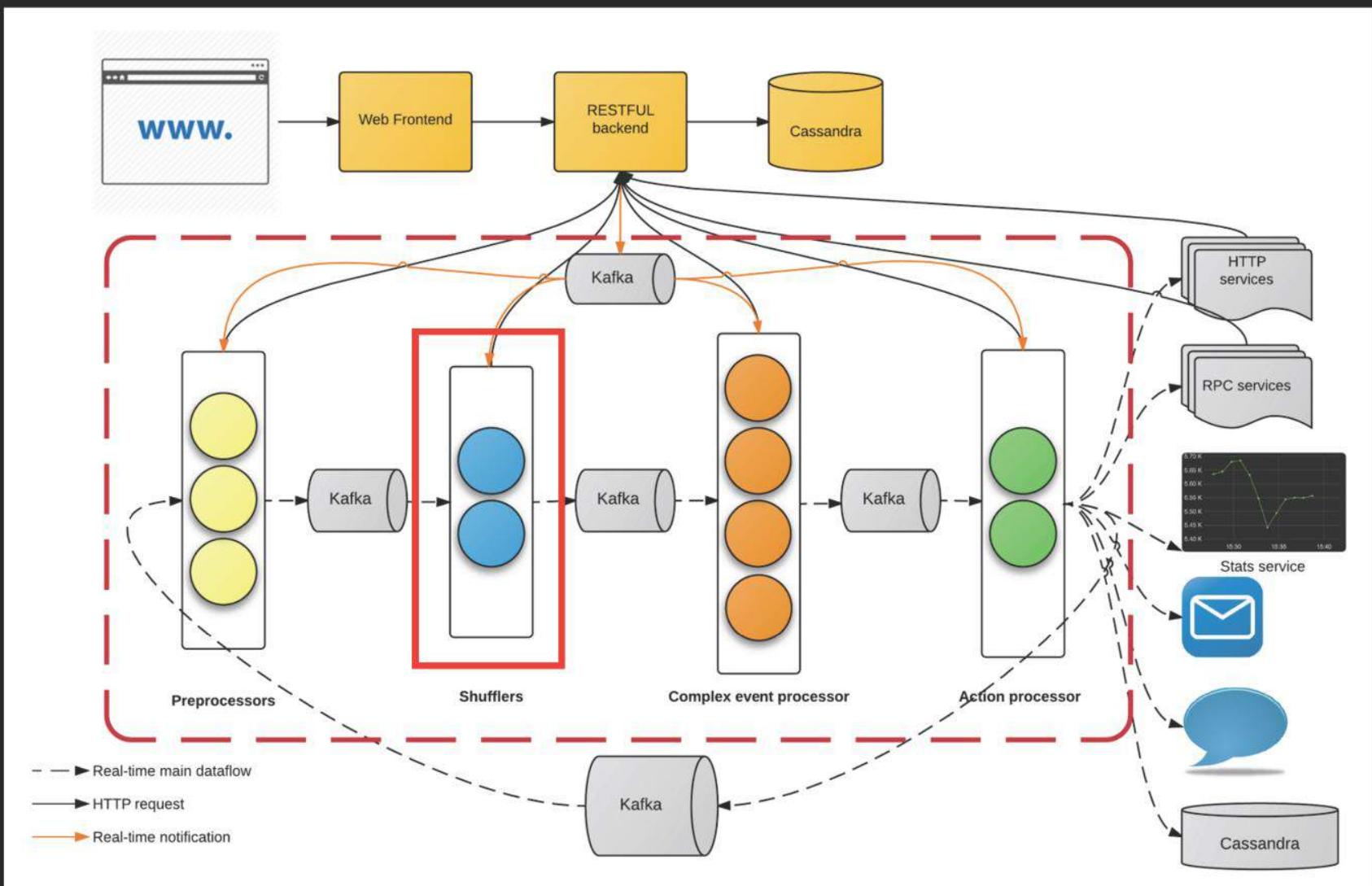
- No hard-coded CEP rules
- Applying CEP rules per individual entity: topic, driver, rider, cohorts, and etc
- Flexible checkpointing and statement management

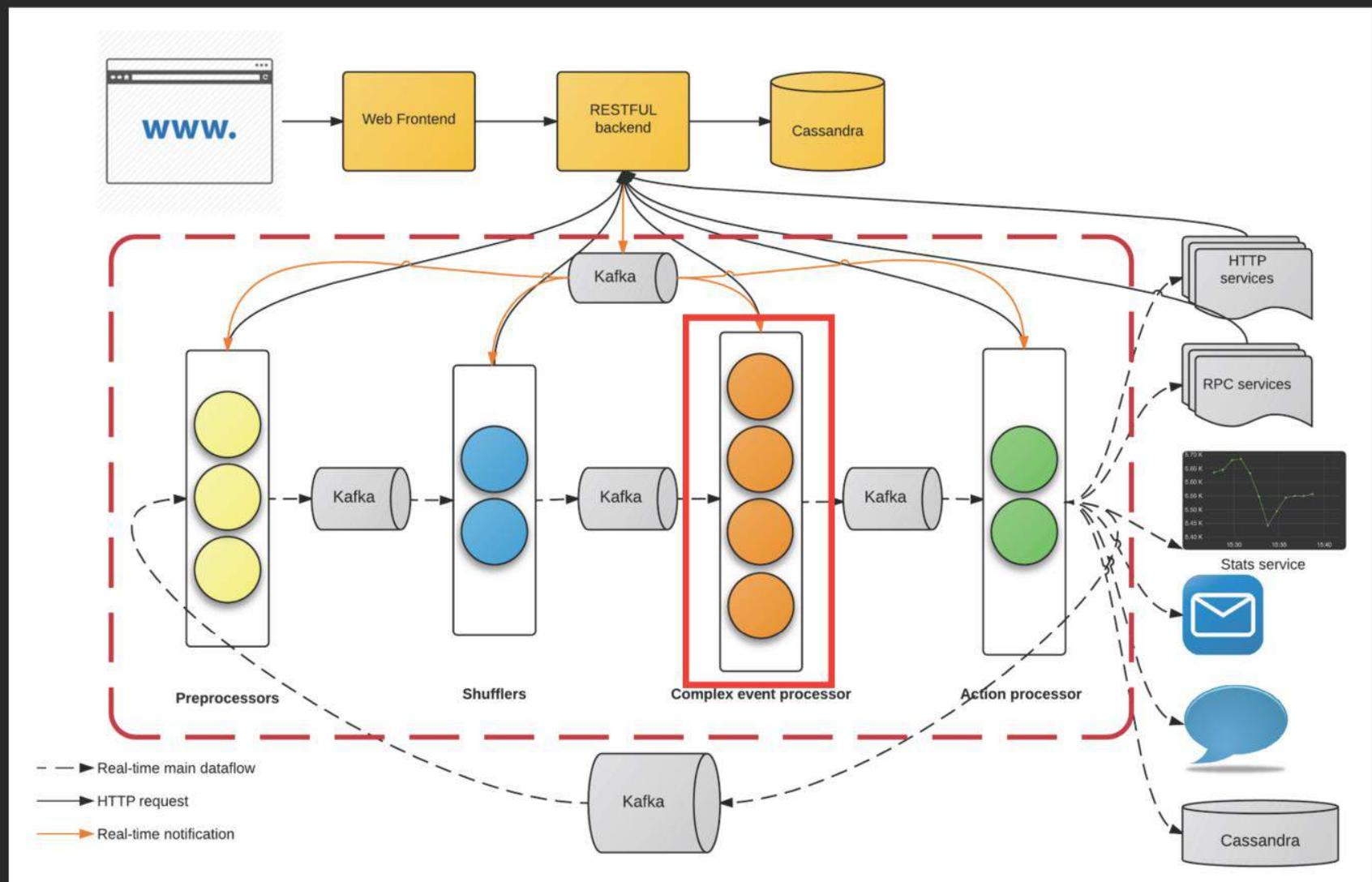


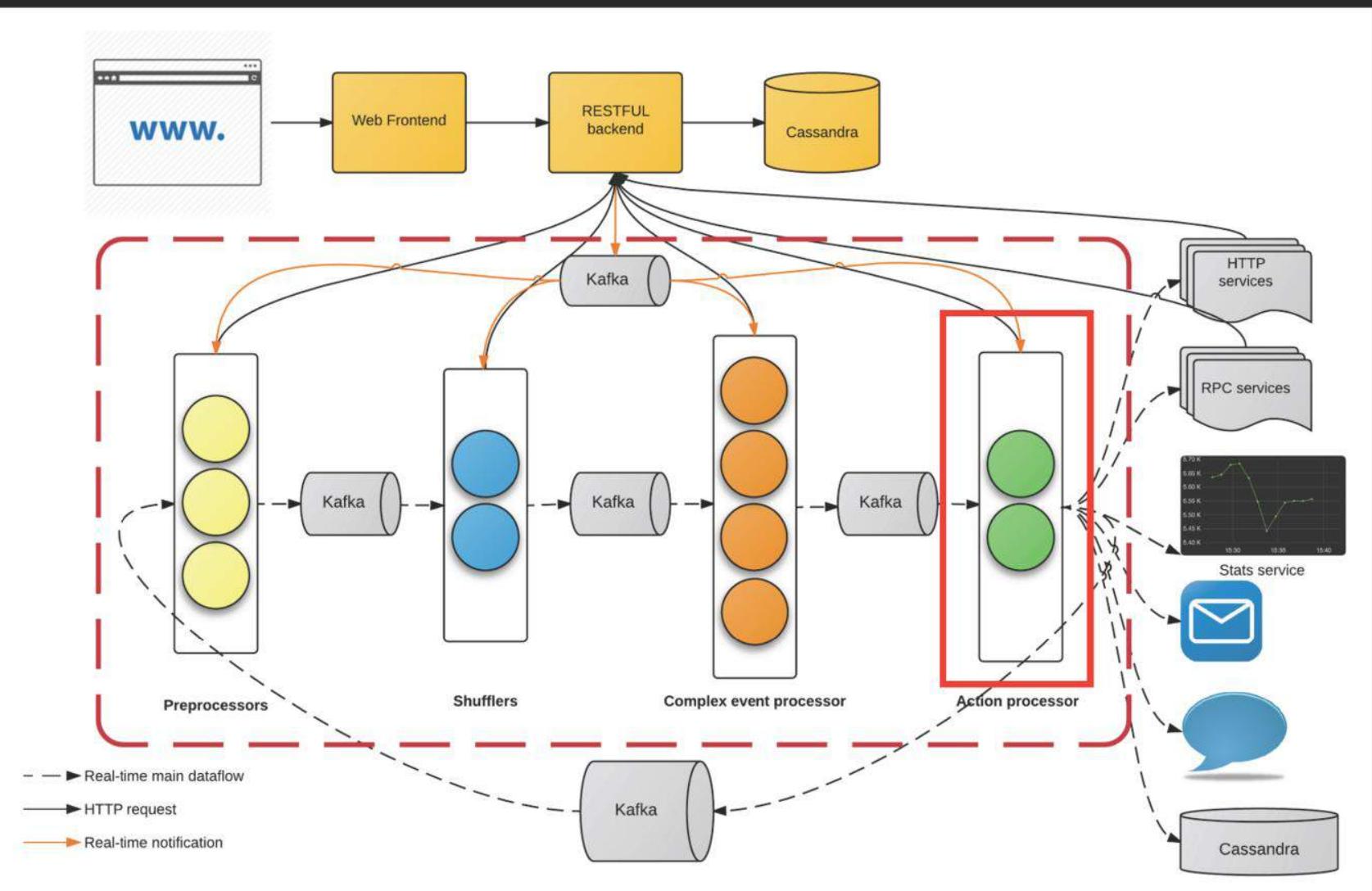


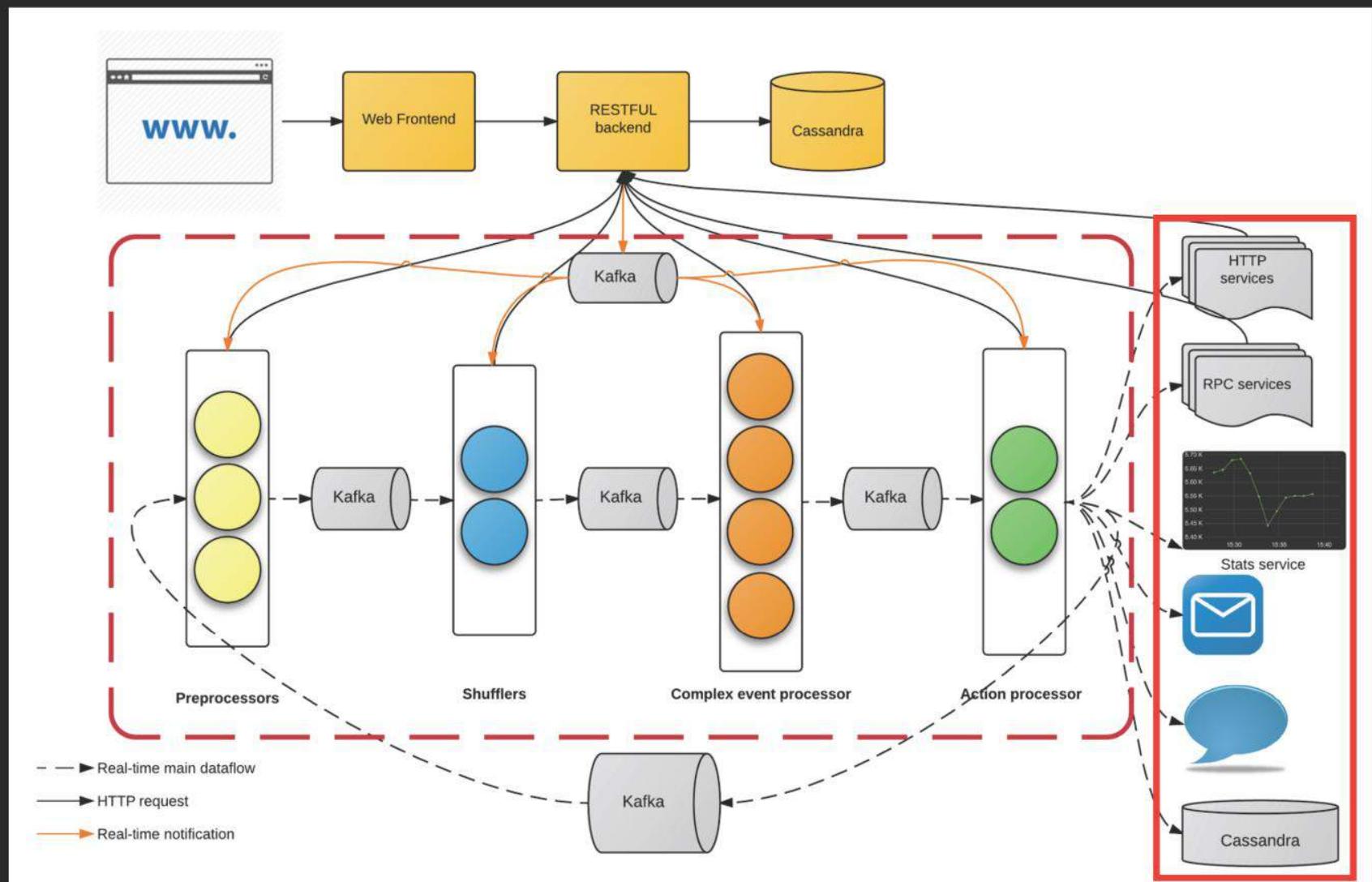








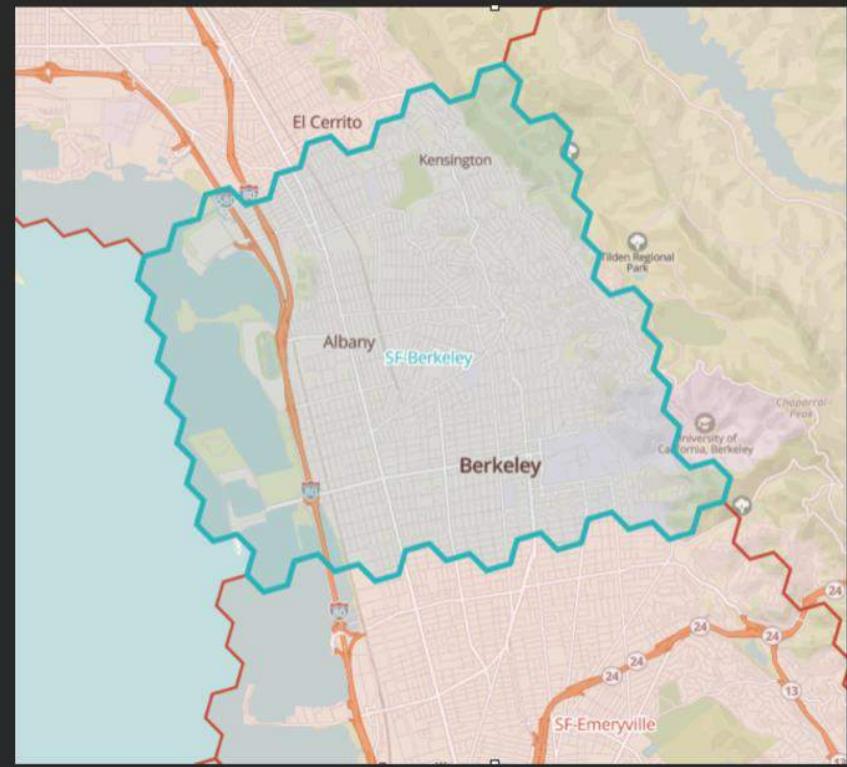
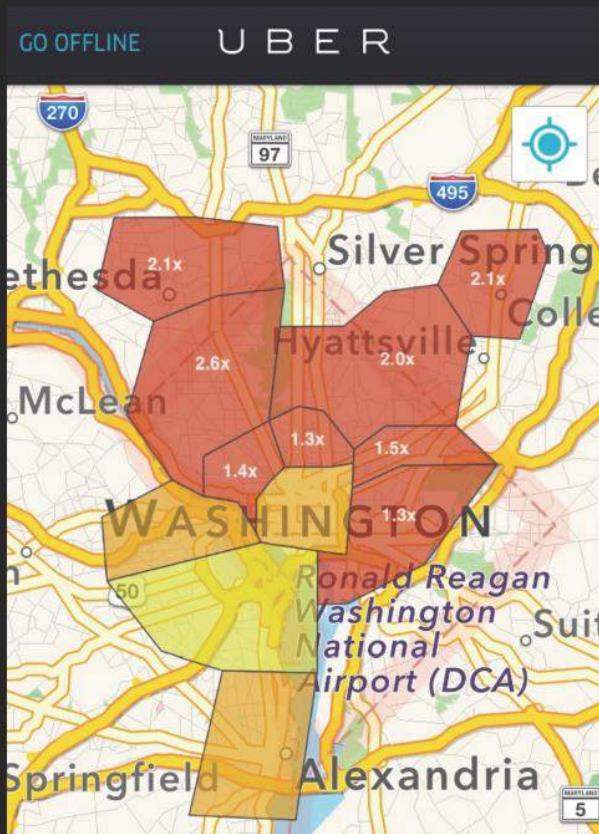




We need to evolve our architecture for other
analytics

Clustering

Manually Created Cluster

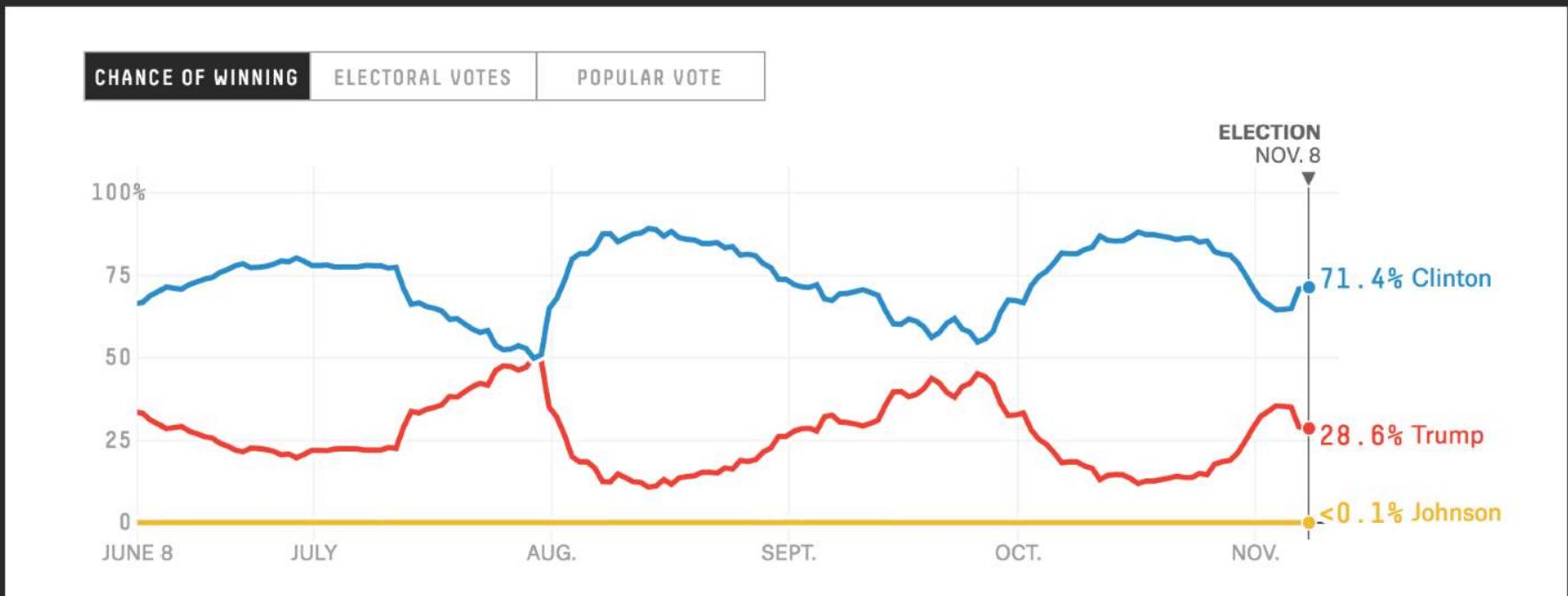


Call for algorithmically created clusters

- Clustering based on key performance metrics

Call for algorithmically created cluster

- Clustering based on key performance metrics
- Continuously measure the clusters



Call for algorithmically created clusters

- Clustering based on key performance metrics
- Continuously measure the clusters
- **Different** clustering for **different** business needs

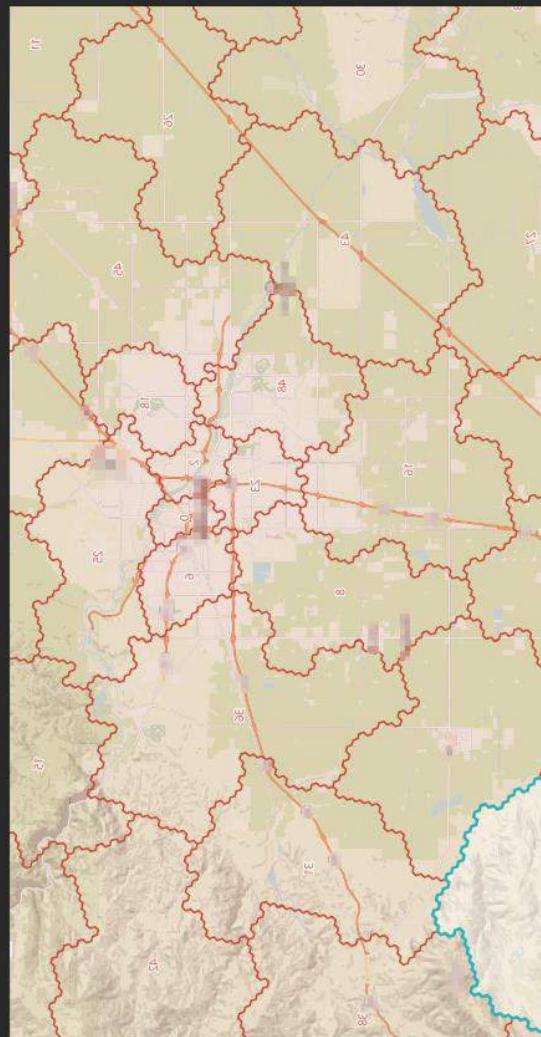
Call for algorithmically created clusters

- Clustering based on key performance metrics
- Continuously measure the clusters
- Different clustering for different business needs
- Create clusters in minutes for **all cities**

Call for algorithmically created clusters

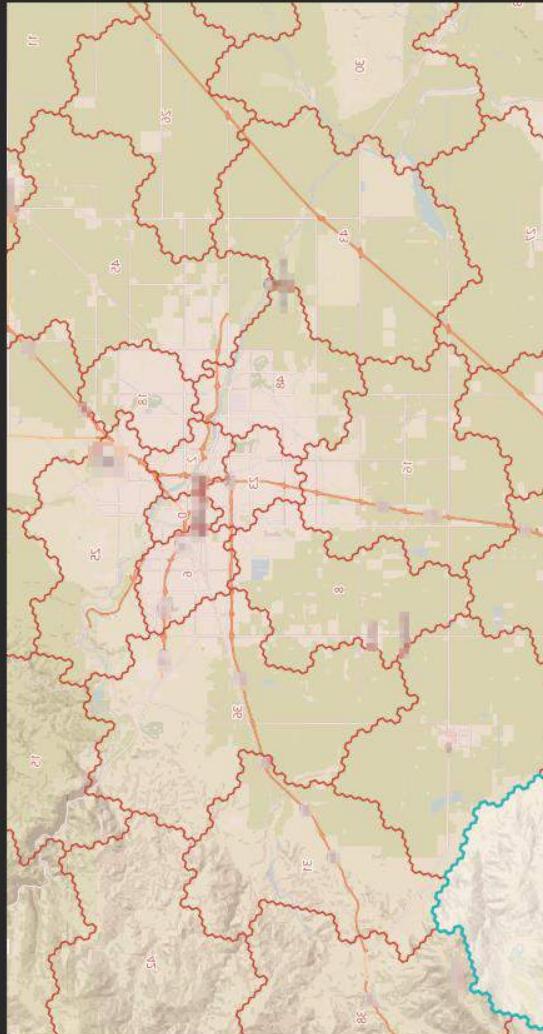
- Clustering based on key performance metrics
- Continuously measure the clusters
- Different clustering for different business needs
- Create clusters in minutes for all cities
- **Foundation** for other stream analytics

Home-grown Clustering Service

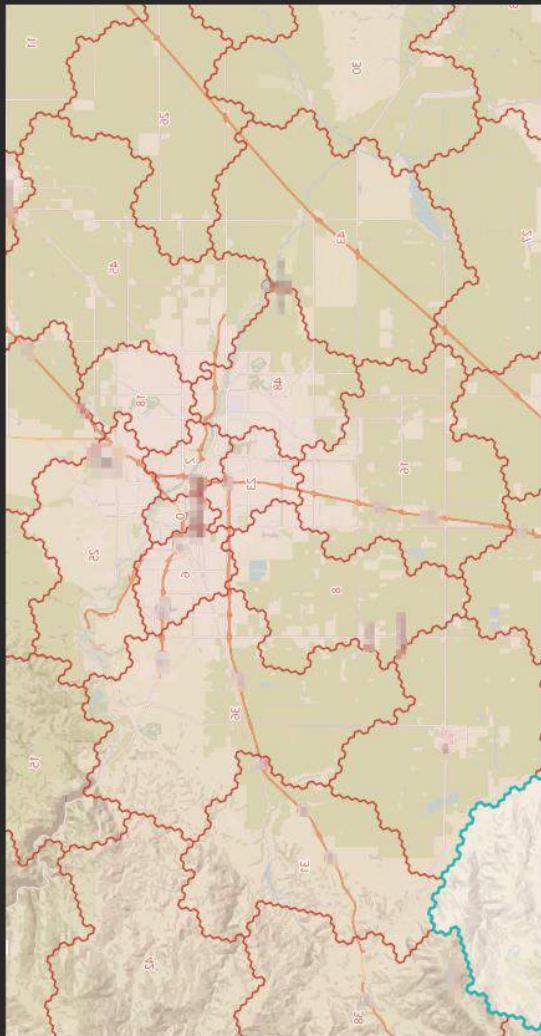


Home-grown Clustering Service

- All cities under 3 minutes

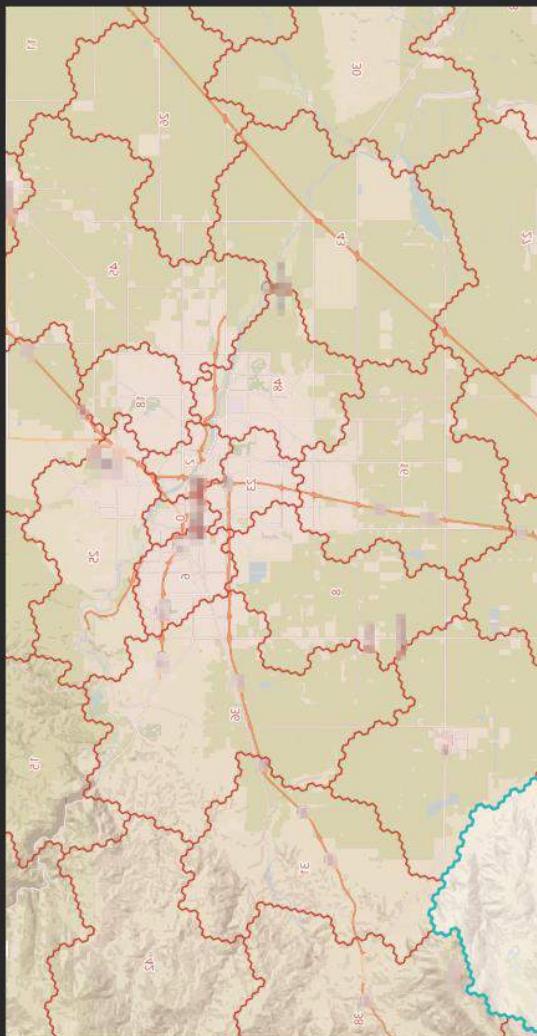


Home-grown Clustering Service



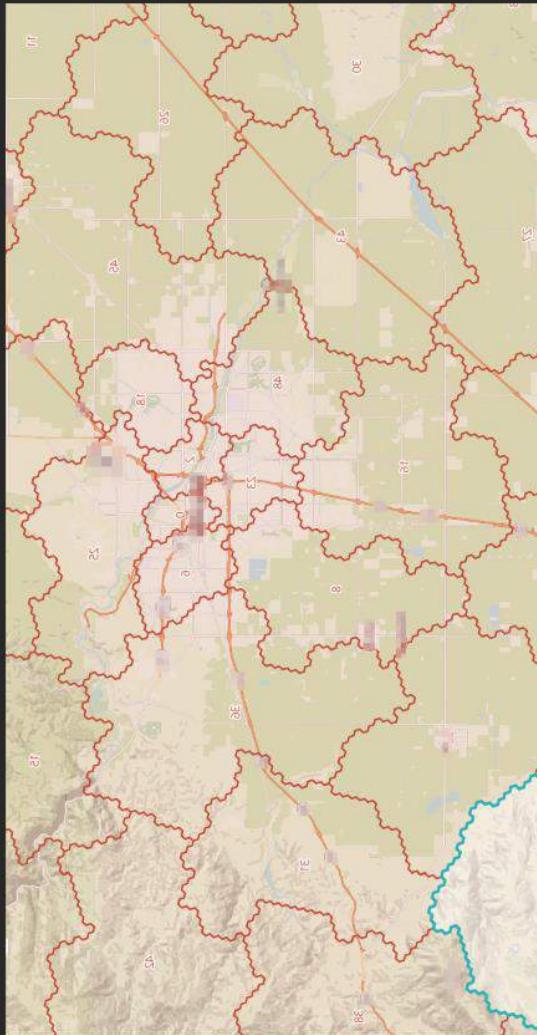
- All cities under 3 minutes
- **Pluggable** algorithms and measurements

Home-grown Clustering Service



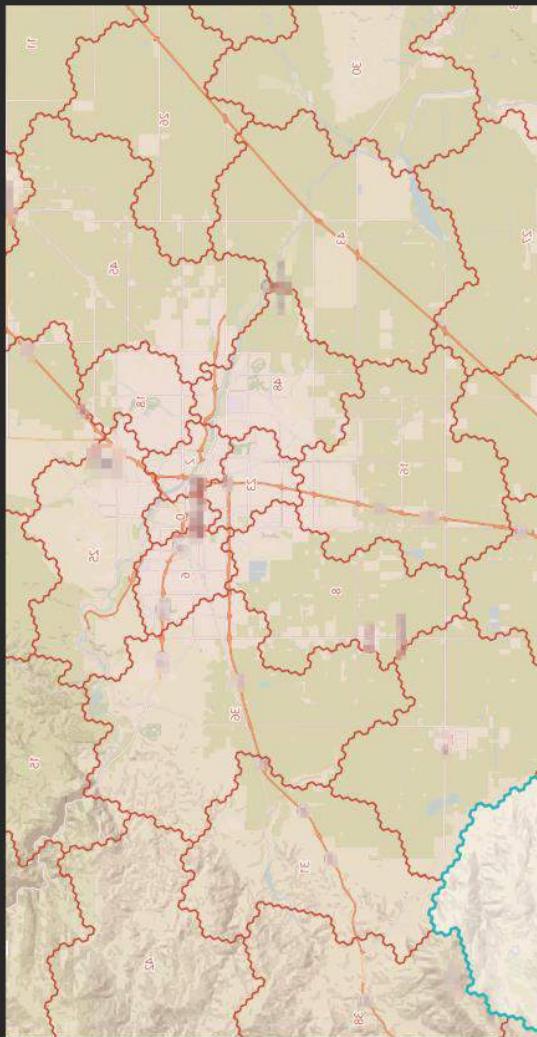
- All cities under 3 minutes
- Easily pluggable algorithms and measurements
- Historical geo-temporal data for clustering

Home-grown Clustering Service



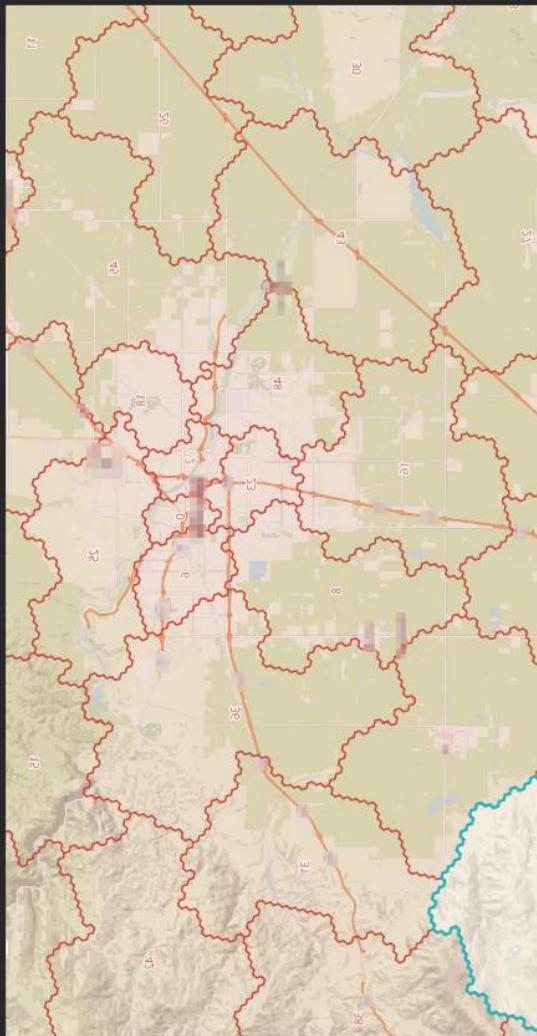
- All cities under 3 minutes
- Easily pluggable algorithms and measurements
- Historical geo-temporal data for clustering
- Real-time geo-temporal data for measurement

Home-grown Clustering Service



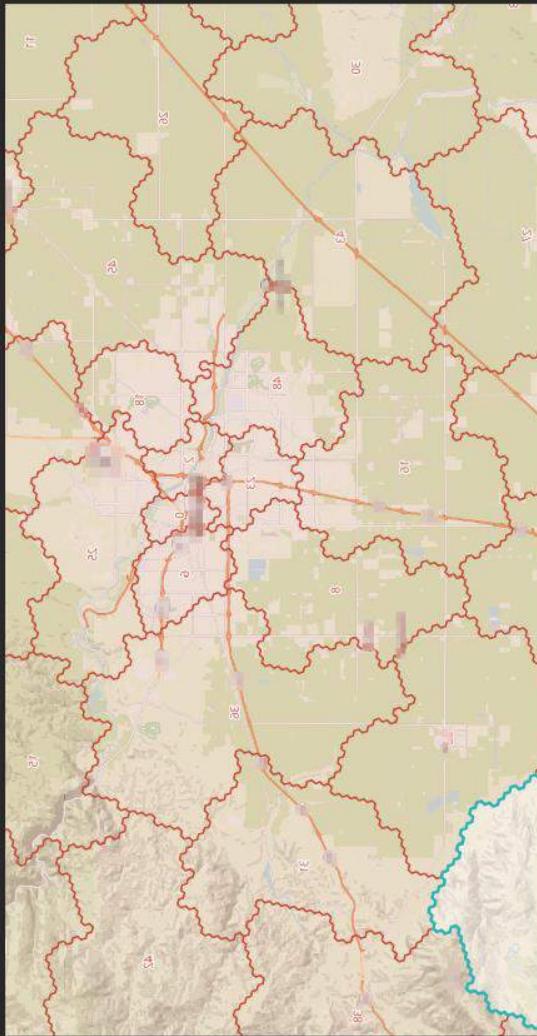
- All cities under 3 minutes
- Easily pluggable algorithms and measurements
- Historical geo-temporal data for clustering
- Real-time geo-temporal data for measurement
- Shared optimizations

Home-grown Clustering Service



- All cities under 3 minutes
- Easily pluggable algorithms and measurements
- Historical geo-temporal data for clustering
- Real-time geo-temporal data for measurement
- Shared optimizations. To put things in perspective:
 - 70,000 hexagons in SF
 - Naive distance function requires at least $70,000 \times 70,000 = 4.9 \text{ billion}$ pairs!

Home-grown Clustering Service



- All cities under 3 minutes
- Easily pluggable algorithms and measurements
- Historical geo-temporal data for clustering
- Real-time geo-temporal data for measurement
- Shared optimizations
 - Incremental updates
 - Compact data representation

Forecasting

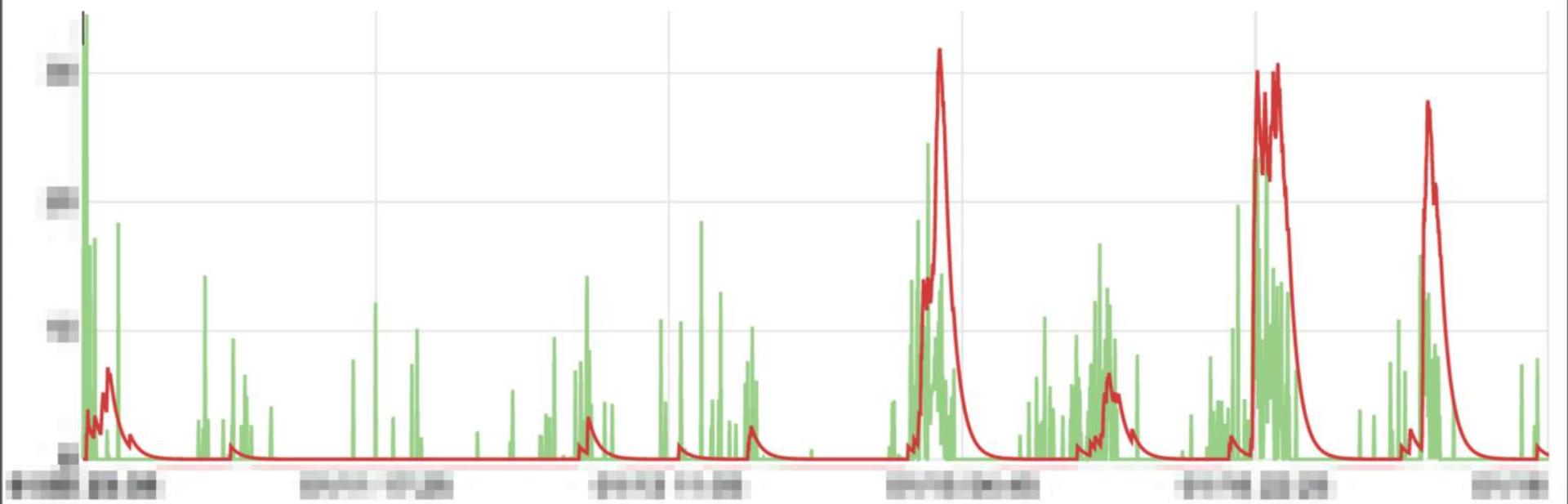
- Every decision is based on forecasting

Forecasting

- Forecasting based on both **historical** data and **stream** input

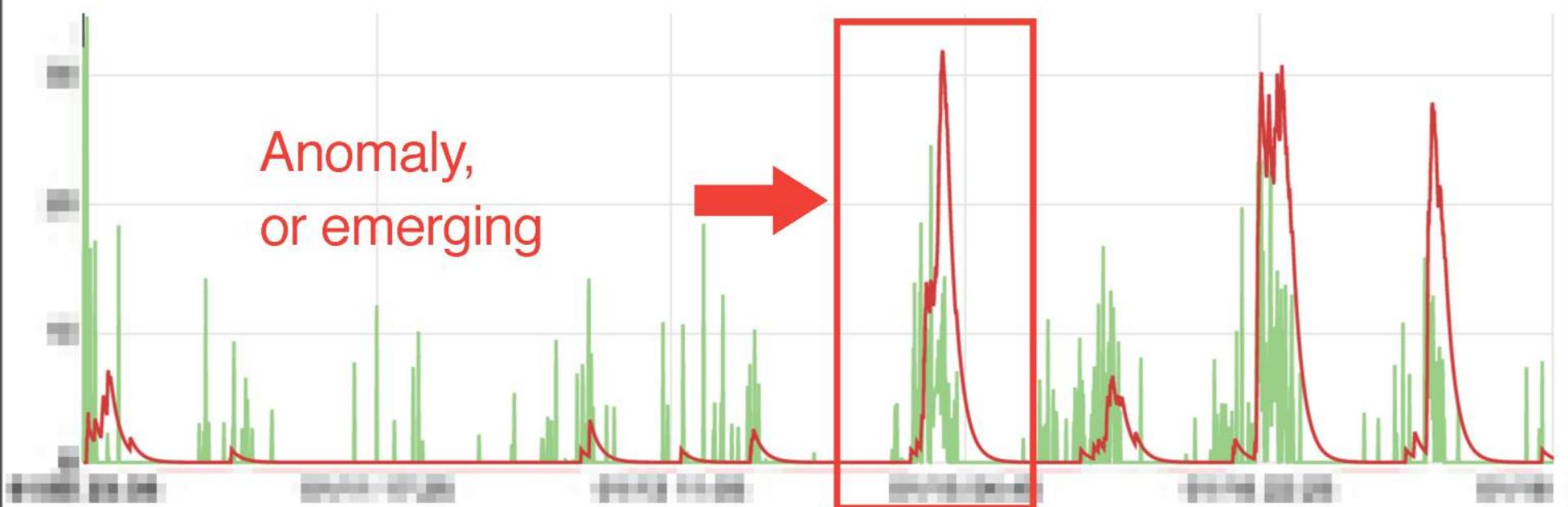
Forecasting

- Forecasting based on both **historical** data and **stream** input



Forecasting

- Forecasting based on both historical data and stream input



Forecasting

- Spatially granular forecasting - down to every hexagon

Forecasting

- Spatially granular forecasting - down to every hexagon

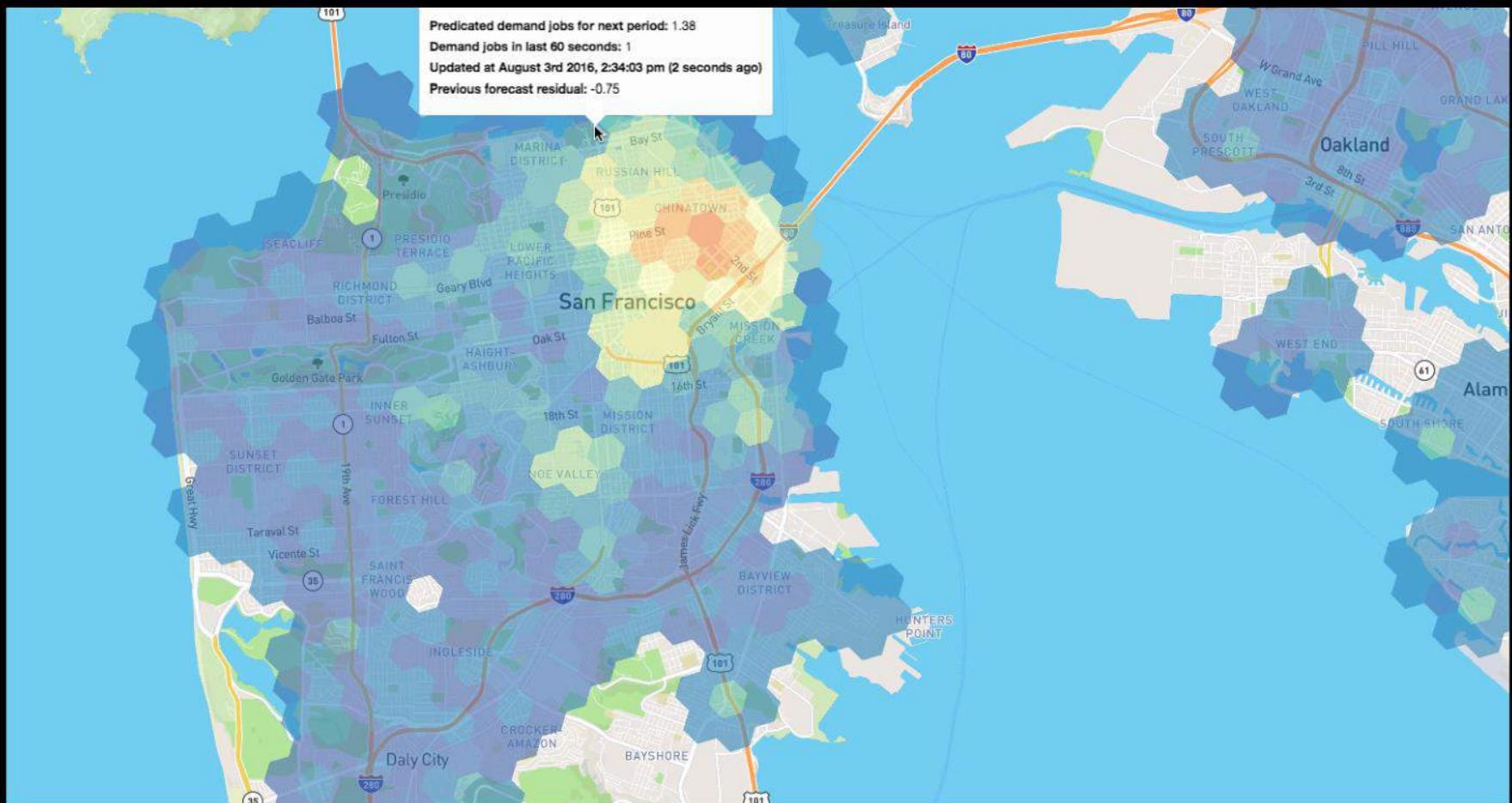


Forecasting

- Temporally granular forecasting - down to every minute

Forecasting

- Temporally granular forecasting - down to every minute



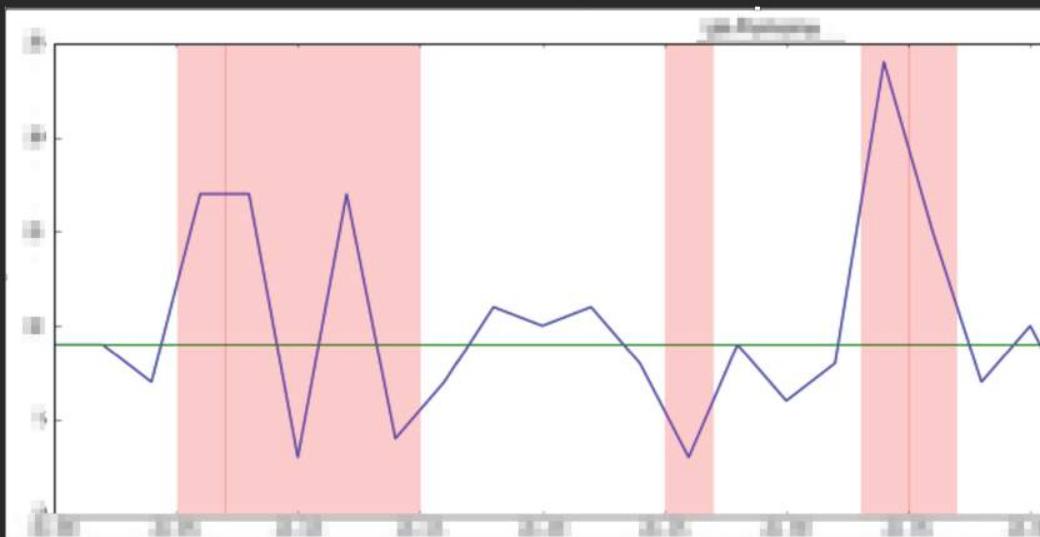
Pattern Detection

- Similarity of different metrics across geolocation and time
- Metric outliers across geolocations and time
- Frequent occurrences of certain patterns
- Clustered behavior
- Anomalies

Common Requirements in Pattern Detection

- Not just traditional time series analysis
- Incorporating insights on marketplace data
- Required both historical data and real-time input
- Spatially granular patterns - down to every hexagon
- Temporally granular patterns - down to every minute

Example: Anomaly Detection



- Simple time series analysis
- For a single geo area
- Can be noisy

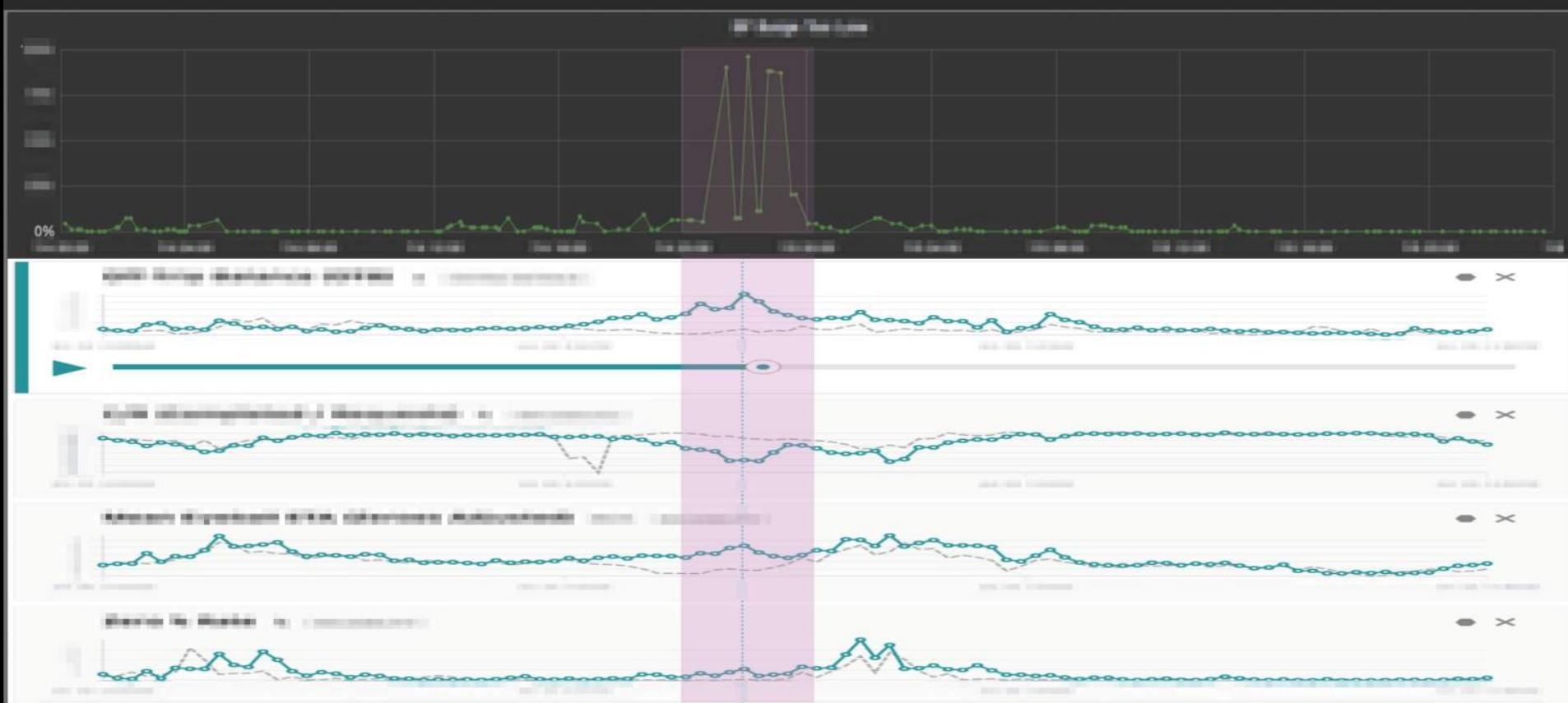
A More Realistic Anomaly Detection



Example: Anomaly Detection



Example: Anomaly Detection



What's the right architecture to support the analytics use cases?

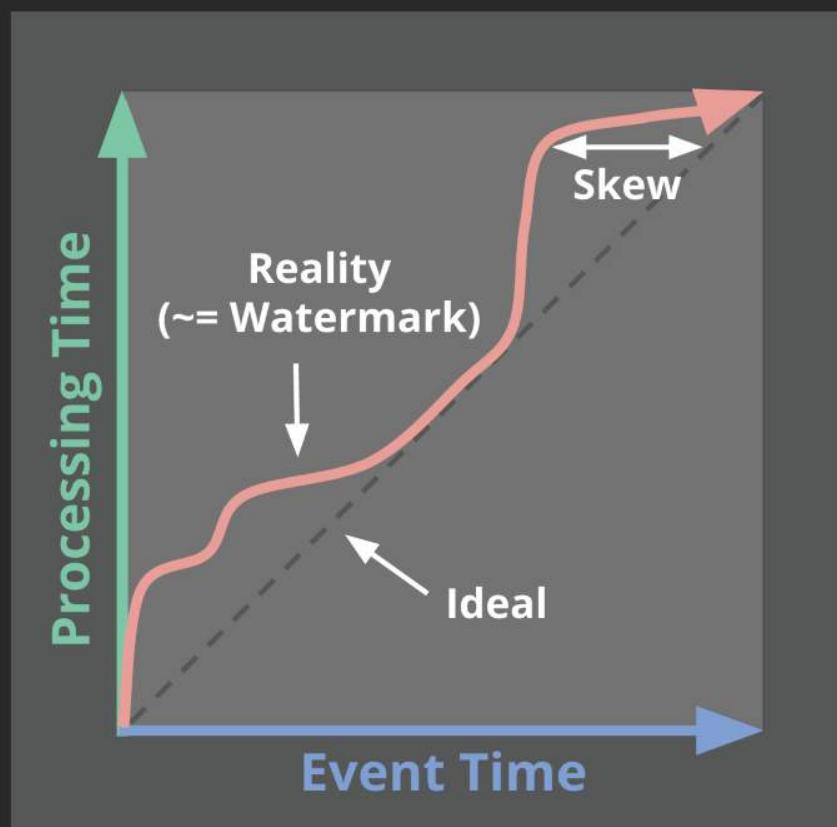
Shared abstraction: multi-dimensional geo-temporal
data

Shared abstraction: multi-dimensional geo-temporal data

- Time series by **event time**

Shared abstraction: multi-dimensional geo-temporal data

- Time series by **event time**

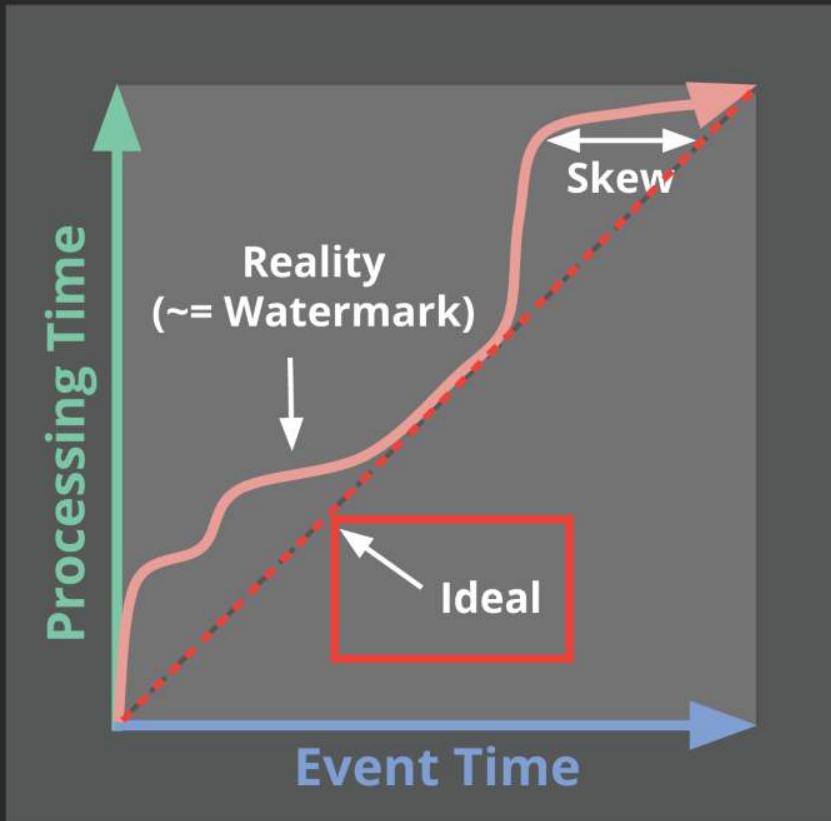


<https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101>

Shared abstraction: multi-dimensional geo-temporal data

- Time series by event time

<https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101>



Shared abstraction: multi-dimensional geo-temporal data

- Time series by **event time**



<https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101>

Shared abstraction: multi-dimensional geo-temporal data

- Time series by event time
- Flexible **windowing** - tumbling, sliding, conditionally triggered

Shared abstraction: multi-dimensional geo-temporal data

- Time series by event time
- Flexible **windowing** - tumbling, sliding, conditionally triggered
- e.g. event-based triggers

Shared abstraction: multi-dimensional geo-temporal data

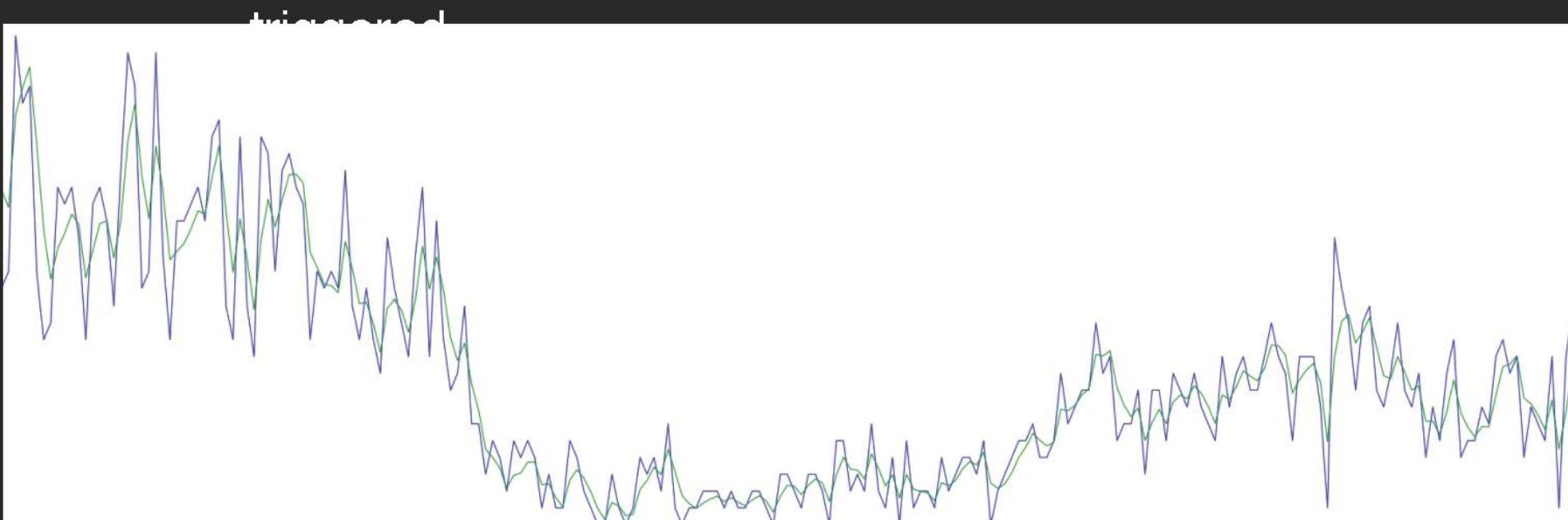
- Time series by event time
- Flexible **windowing** - tumbling, sliding, conditionally triggered
- e.g. event-based triggers
- e.g., triggers of computation results

Shared abstraction: multi-dimensional geo-temporal data

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- **Stateful** processing

Shared abstraction: multi-dimensional geo-temporal data

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally



Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- **Stateful** processing. E.g.,

$$Y(t) = \alpha X(t) + (1 - \alpha)Y(t - 1)$$

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- **Stateful** processing. E.g.,

$$Y(t) = \boxed{\alpha X(t)} + (1 - \alpha)Y(t - 1)$$

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- **Stateful** processing. E.g.,

$$Y(t) = \alpha X(t) + (1 - \alpha) Y(t - 1)$$

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- **Stateful** processing. E.g.,

$$Y(t) = \alpha X(t) + (1 - \alpha) Y(t - 1)$$



State

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered

$$Y(t) = \alpha X(t) + (1 - \alpha) Y(t - 1)$$



State **per key**

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- Stateful processing
- **Unified stream**

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- Stateful processing
- **Unified stream**
 - Real-time streams: **unbounded streams**

Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- Stateful processing
- **Unified stream**
 - Real-time streams: **unbounded streams**
 - Batch: **bounded streams**

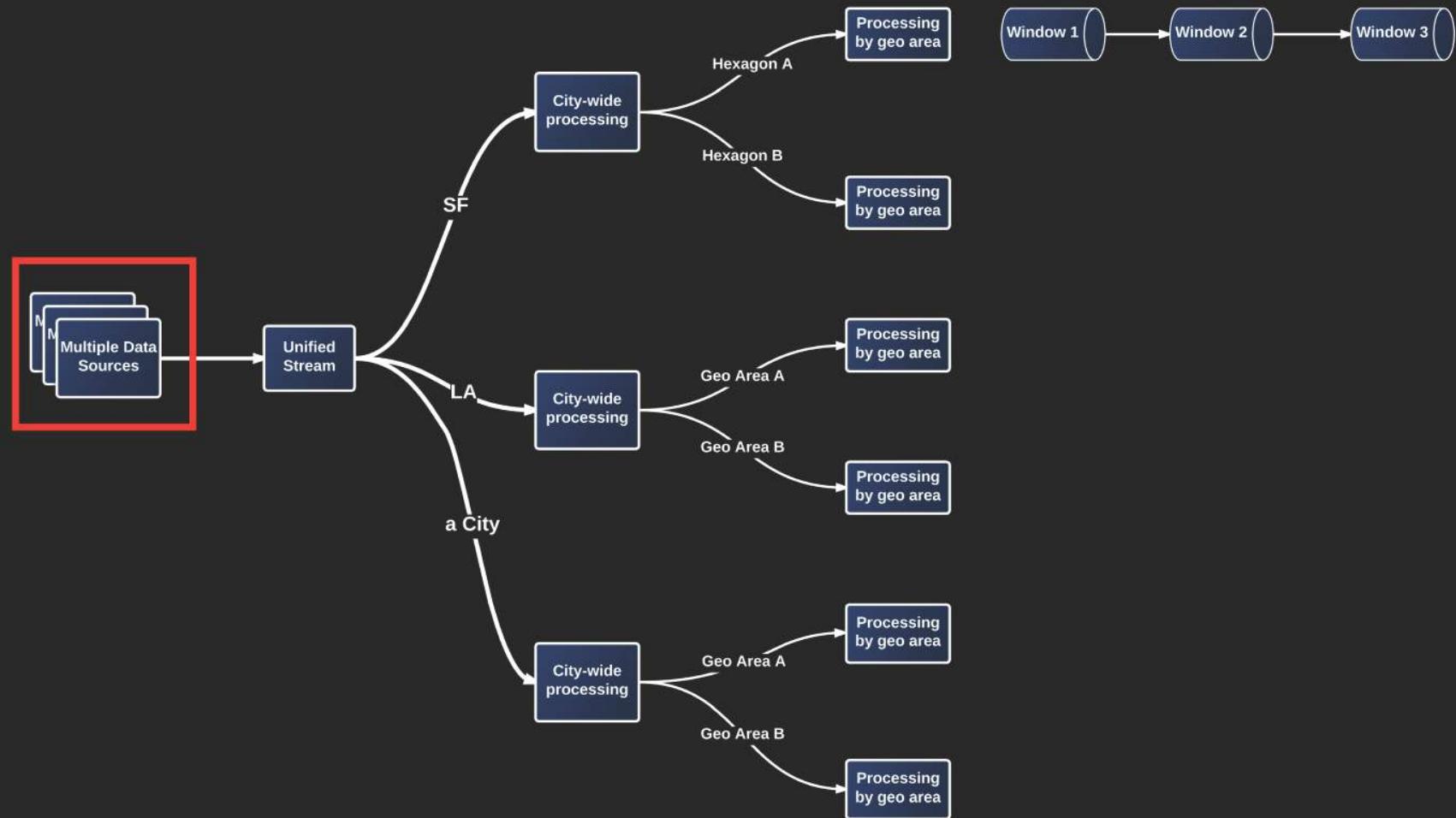
Shared abstraction: multi-dimensional geo-temporal

- Time series by event time
- Flexible windowing - tumbling, sliding, conditionally triggered
- Stateful processing
- **Unified stream**
 - Real-time streams: **unbounded streams**
 - Batch: **bounded streams**

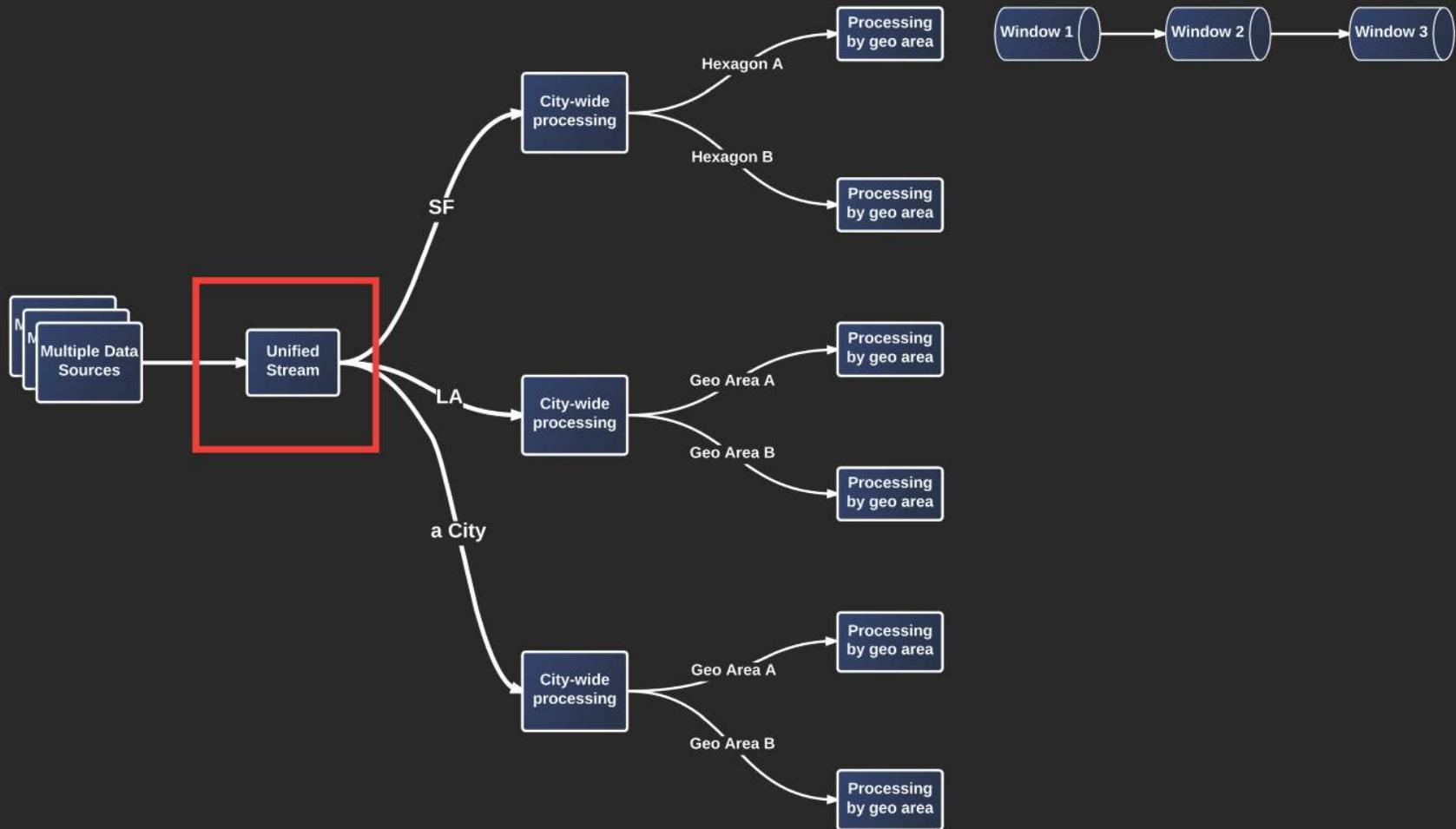
Apache Flink

- Ordering by event time
- Flexible windowing with watermark and triggers
- Exactly-once semantics
- Built-in state management and checkpointing
- Nice data flow APIs

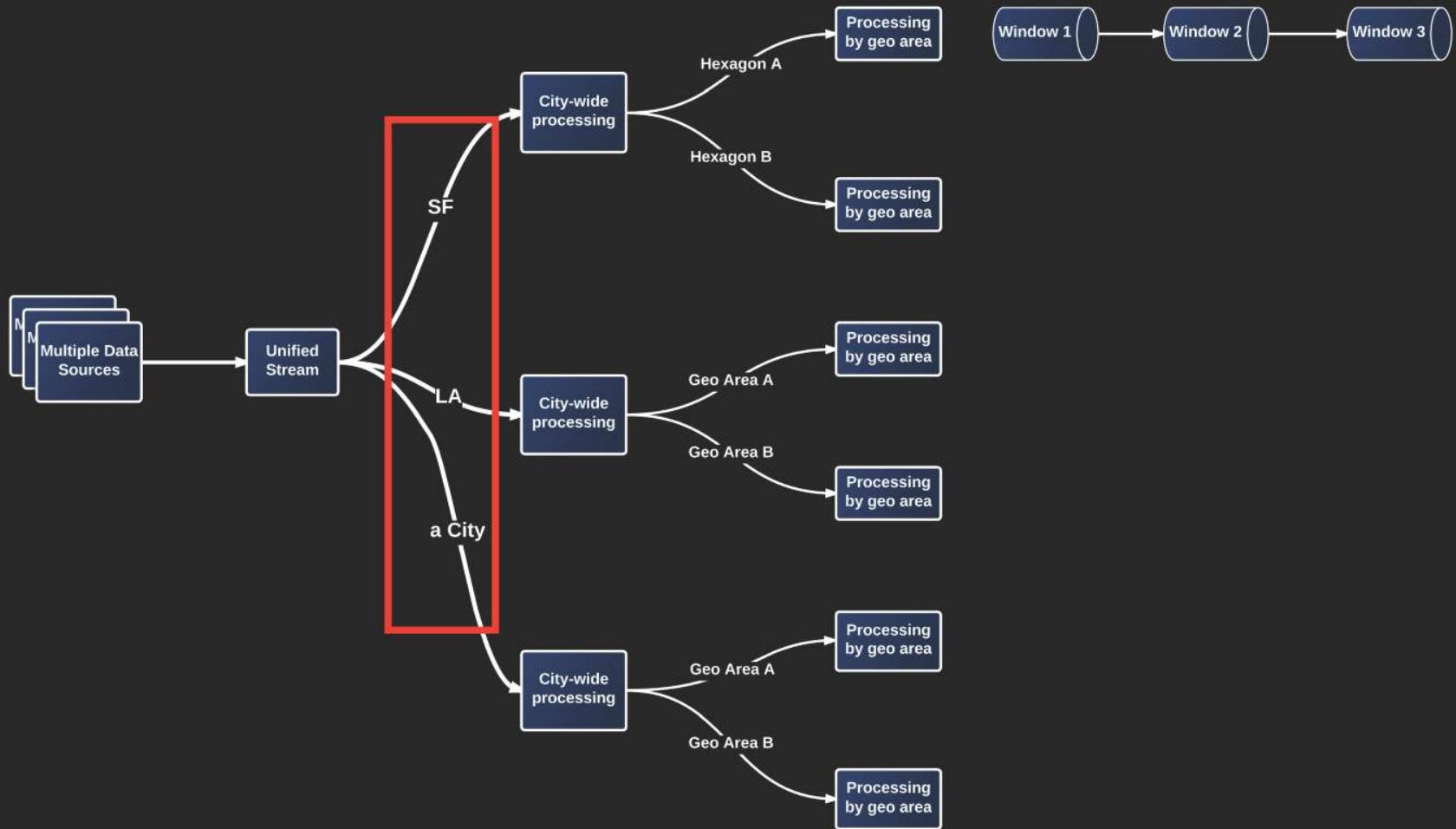
Mental Picture for Processing Geo-temporal Data



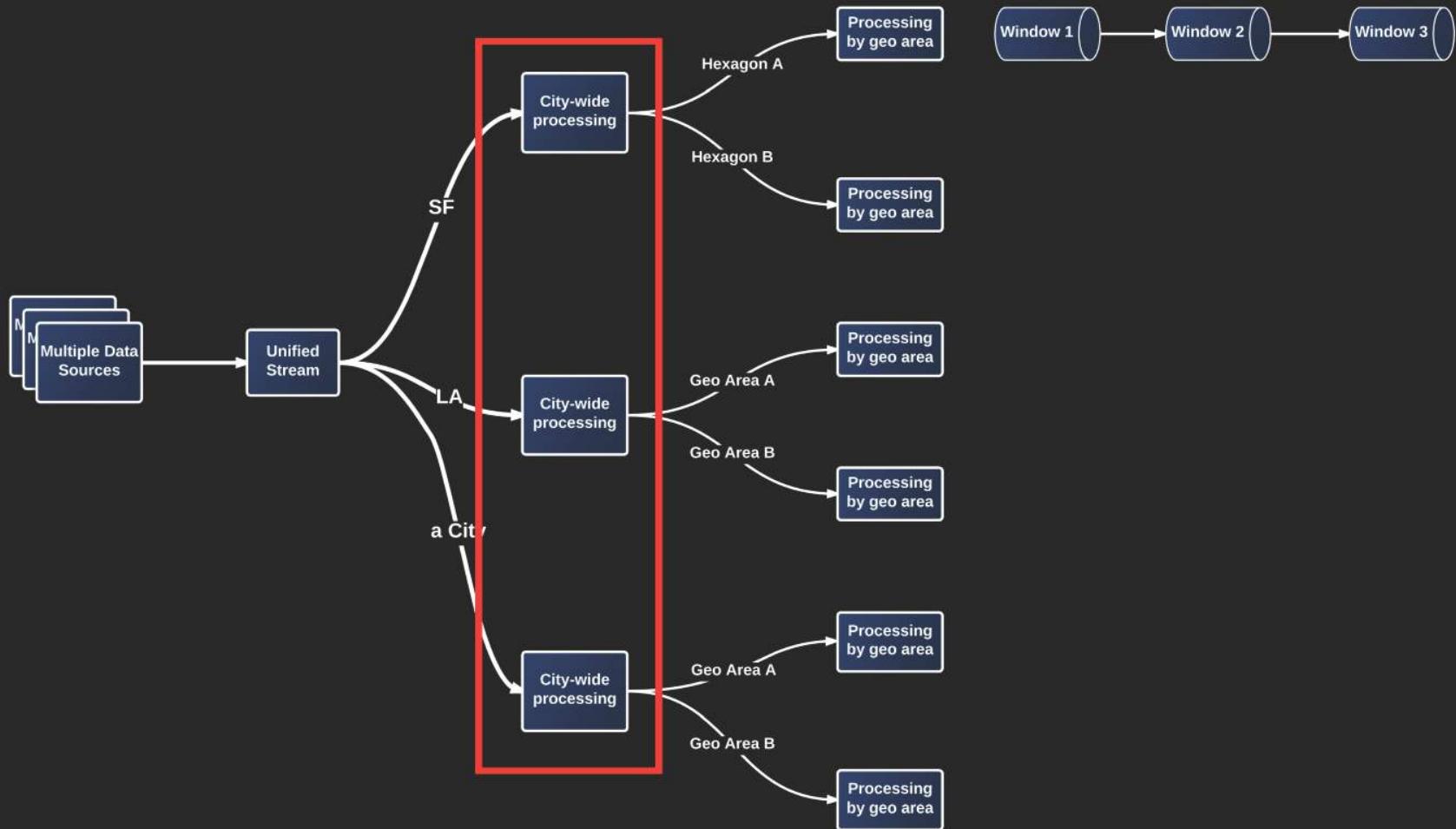
Mental Picture for Processing Geo-temporal Data



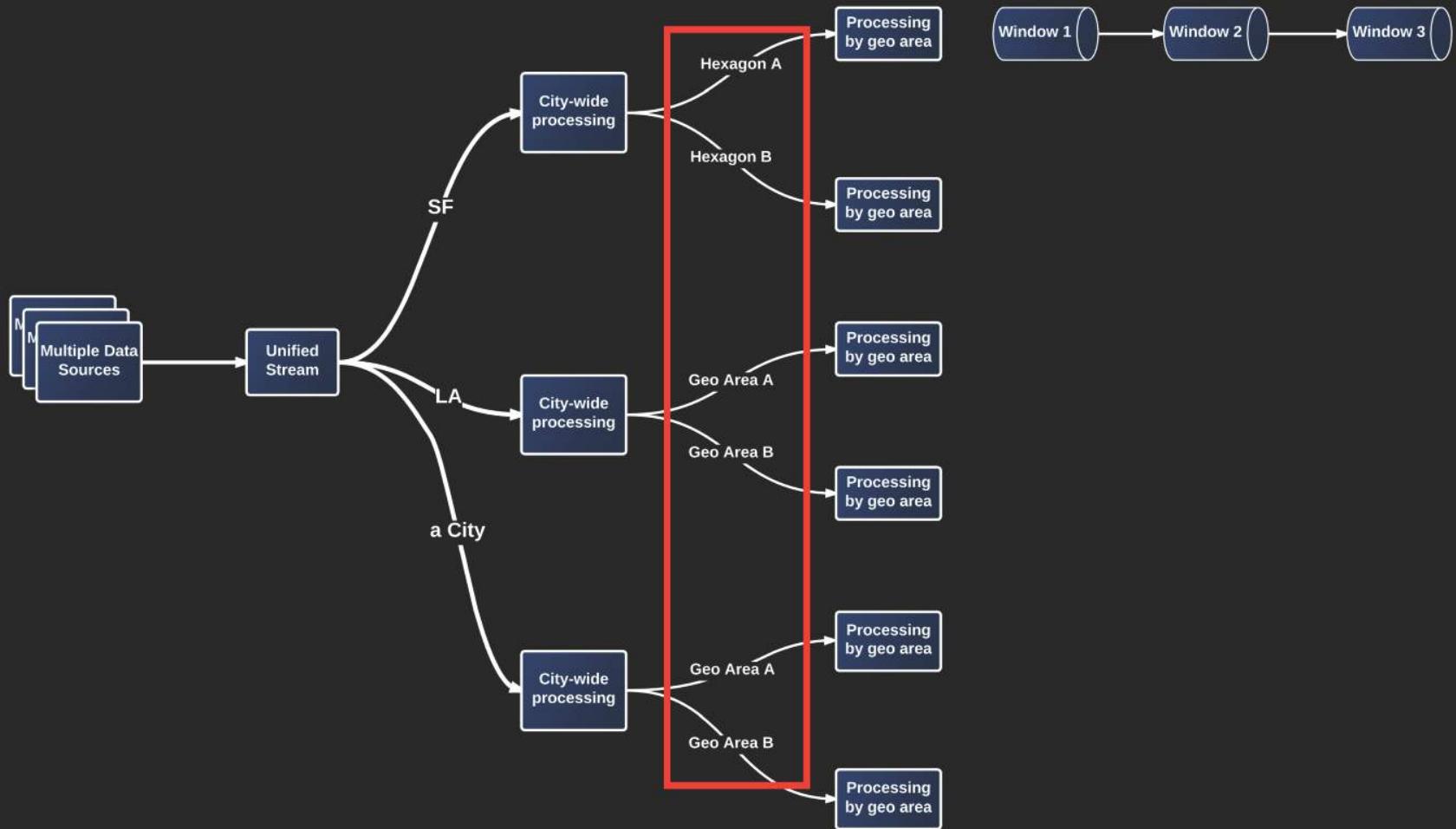
Mental Picture for Processing Geo-temporal Data



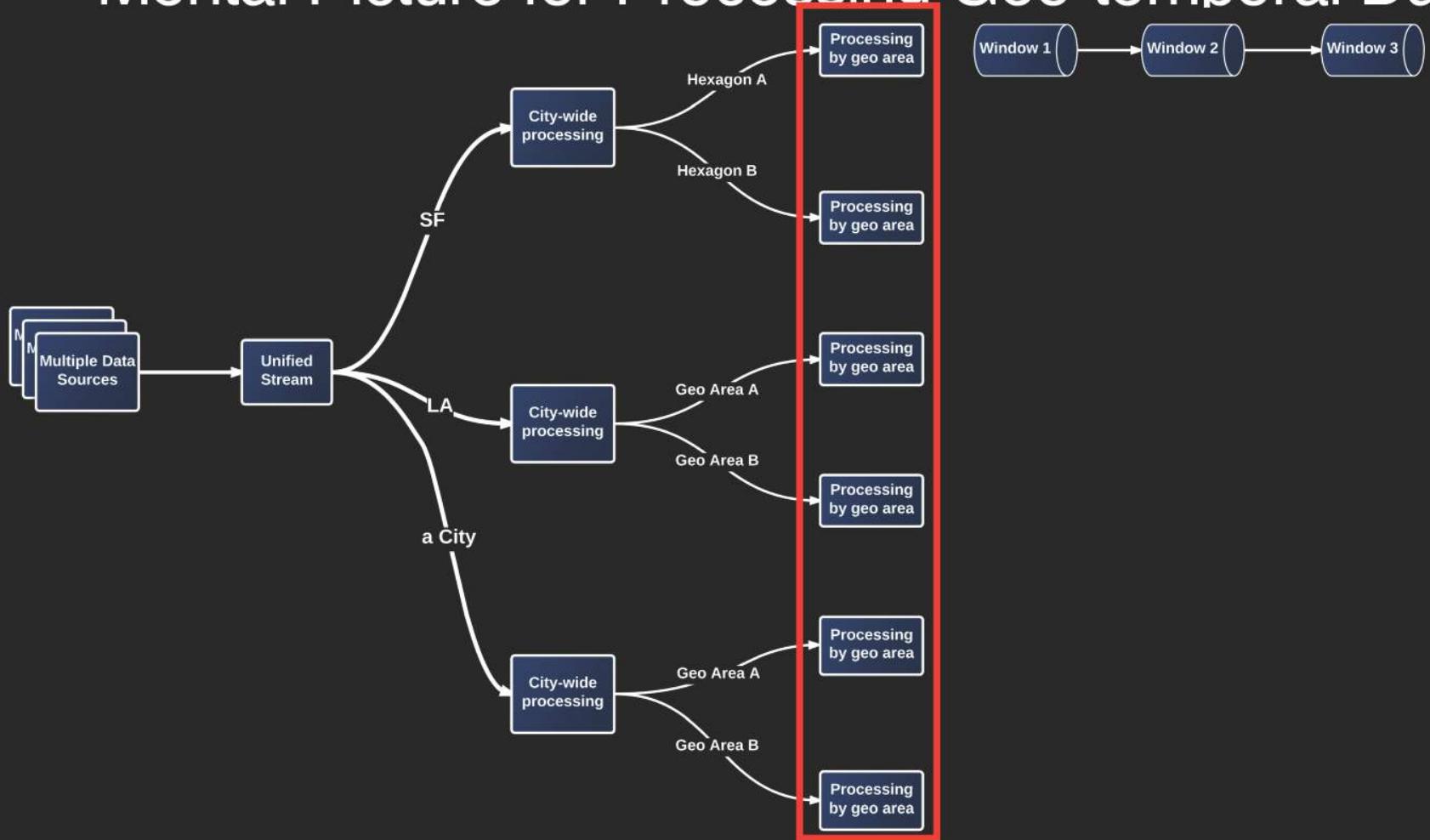
Mental Picture for Processing Geo-temporal Data



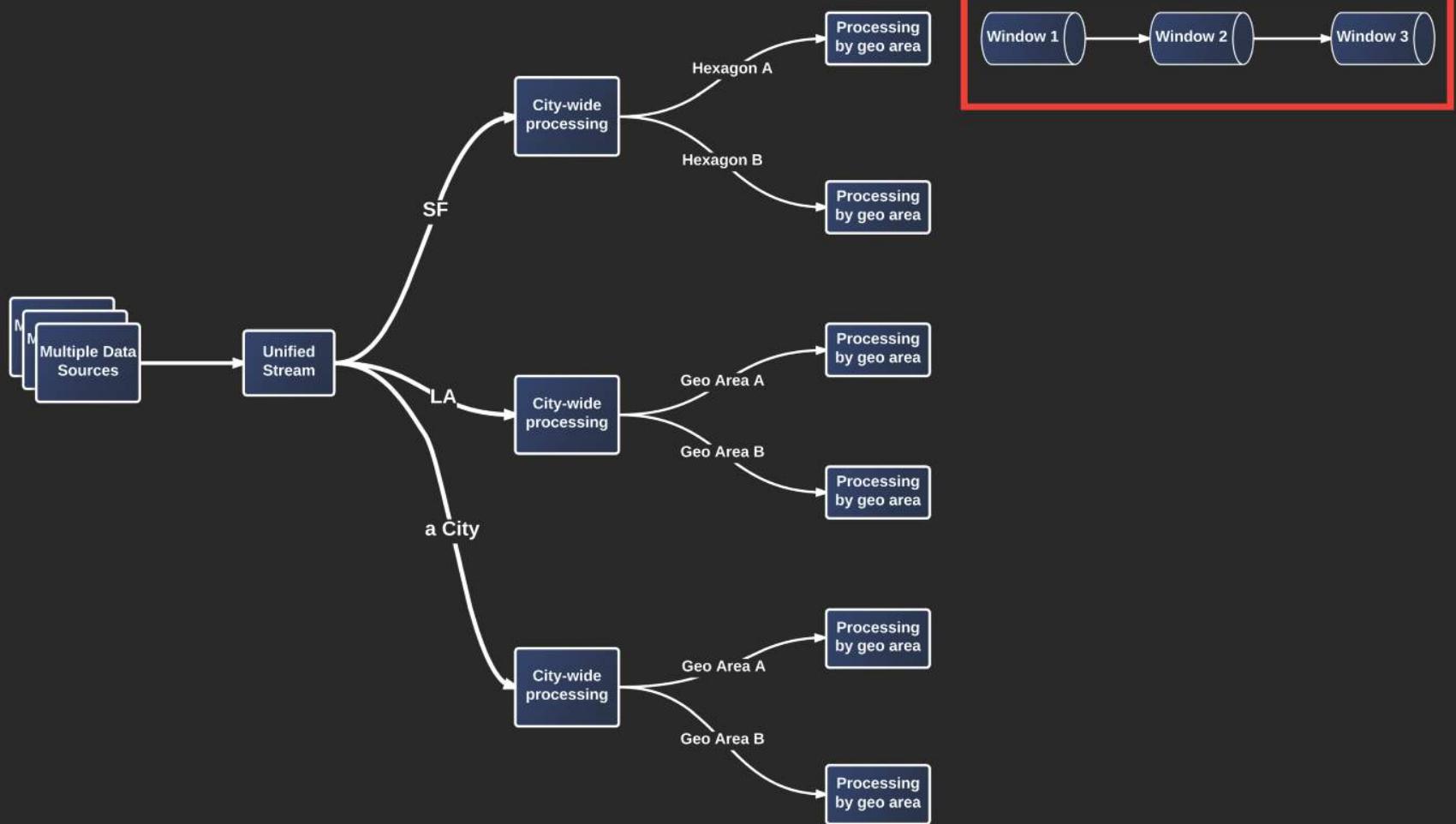
Mental Picture for Processing Geo-temporal Data



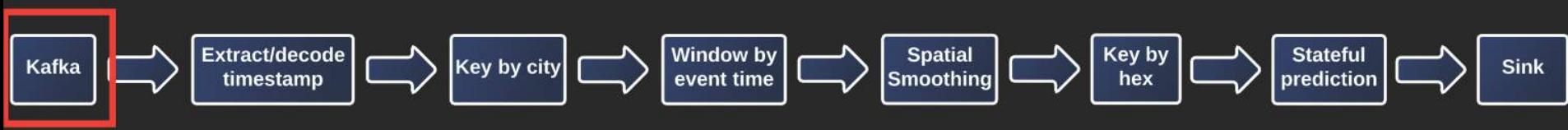
Mental Picture for Processing Geo-temporal Data



Mental Picture for Processing Geo-temporal Data



A Simple Example: simple prediction



Sources

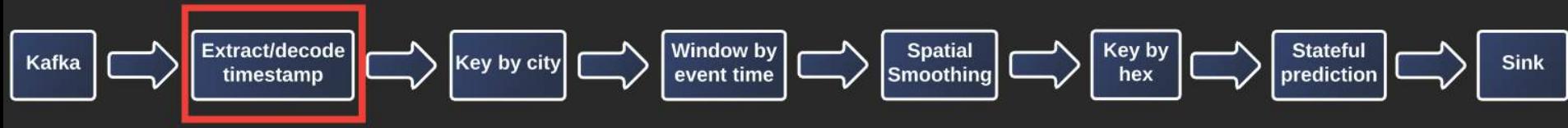
`.fromKafka()`

`.config(config)`

`.cluster(aCluster)`

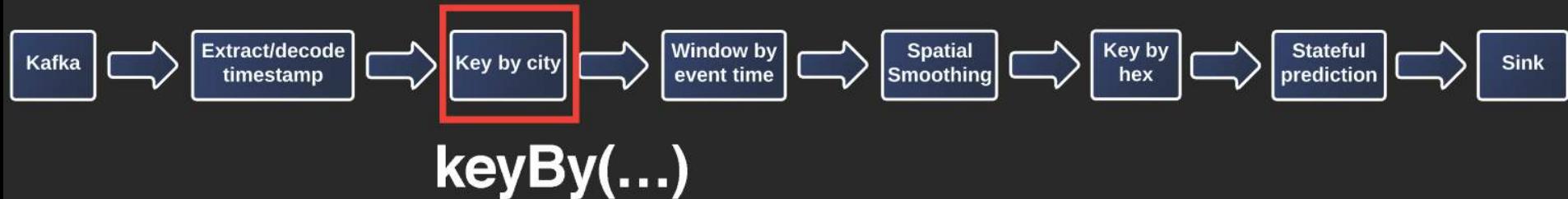
`.topics(topicList)`

A Simple Example



assignTimestampsAndWatermarks

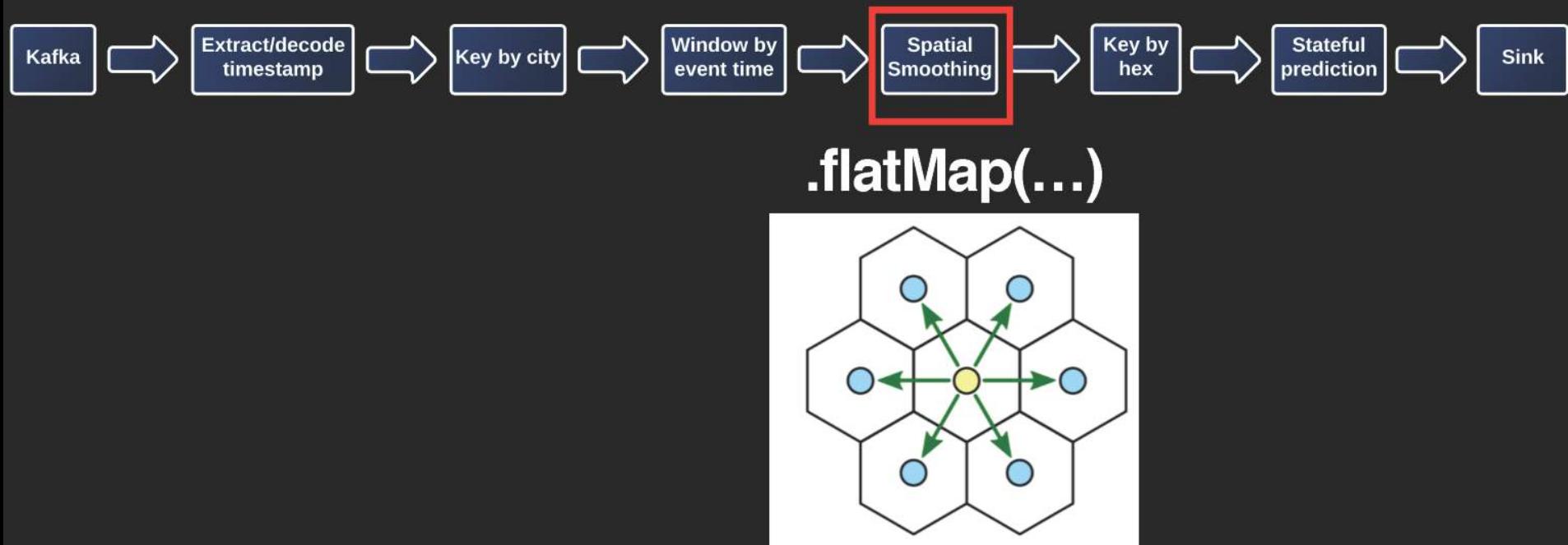
A Simple Example



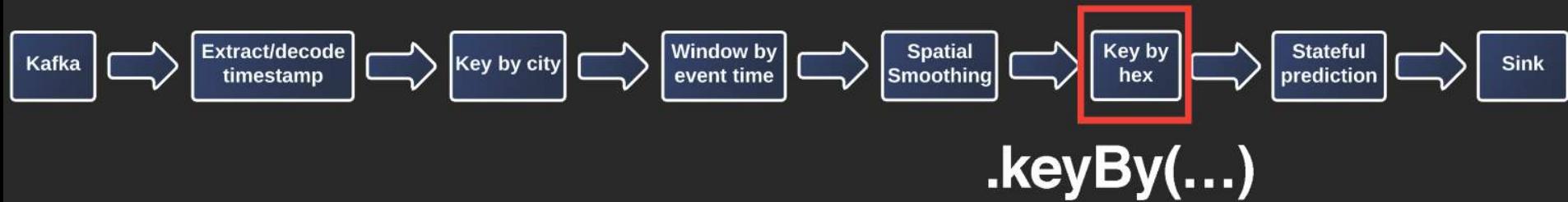
A Simple Example



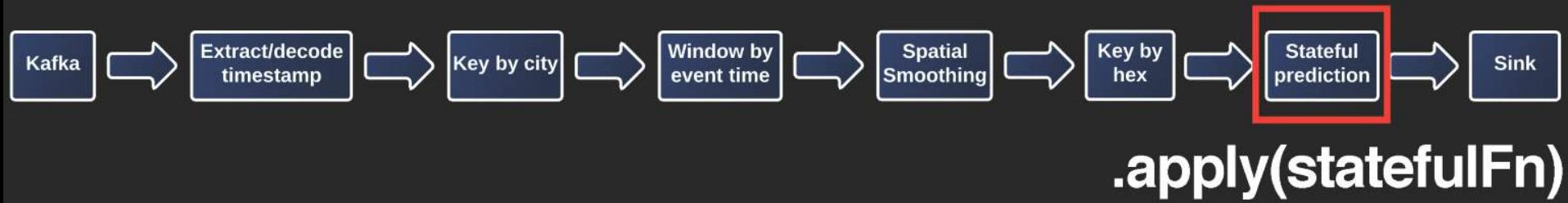
A Simple Example



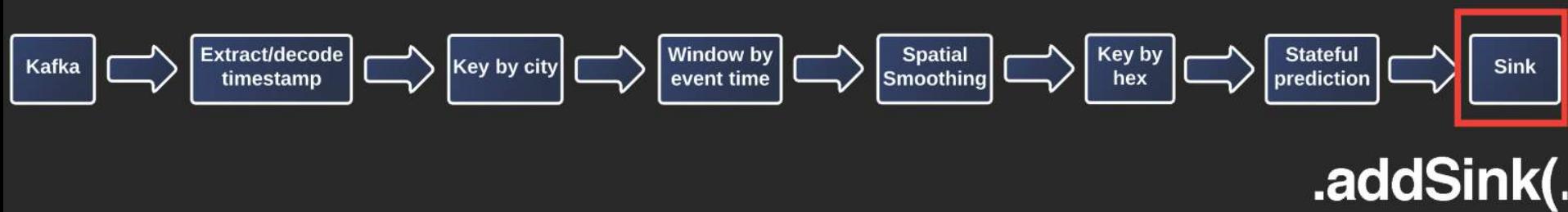
A Simple Example



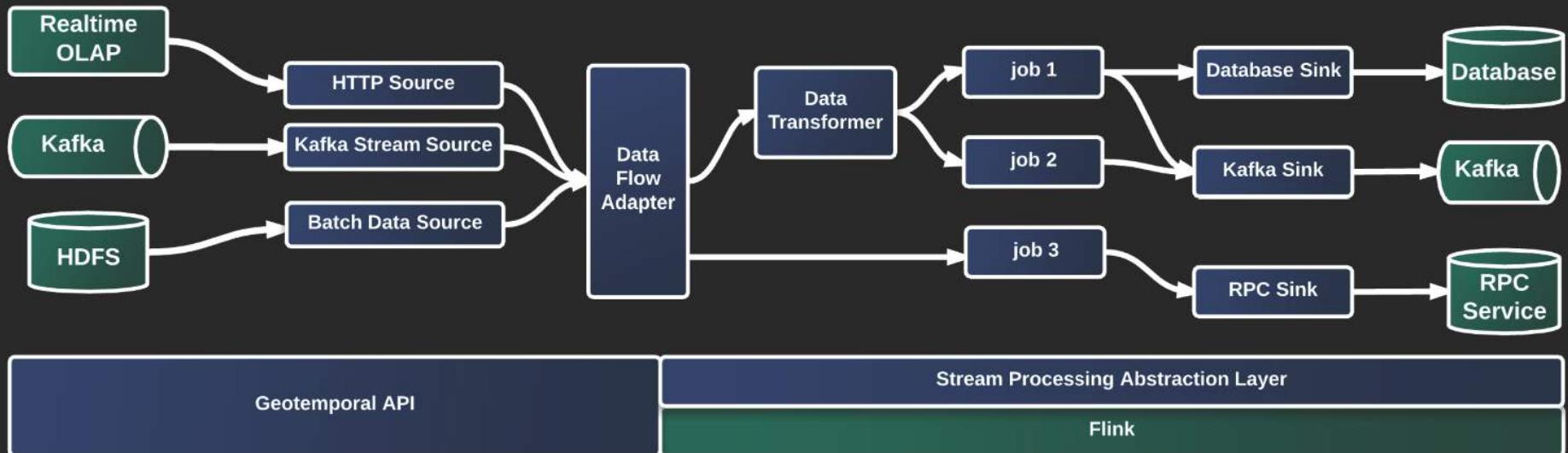
A Simple Example



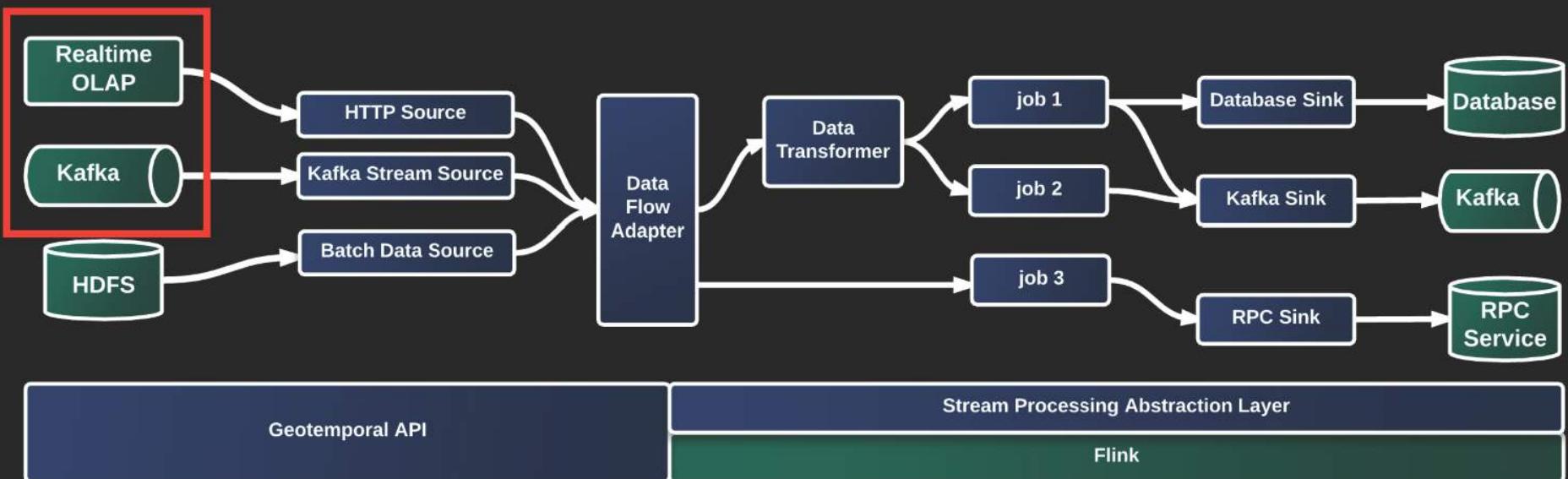
A Simple Example



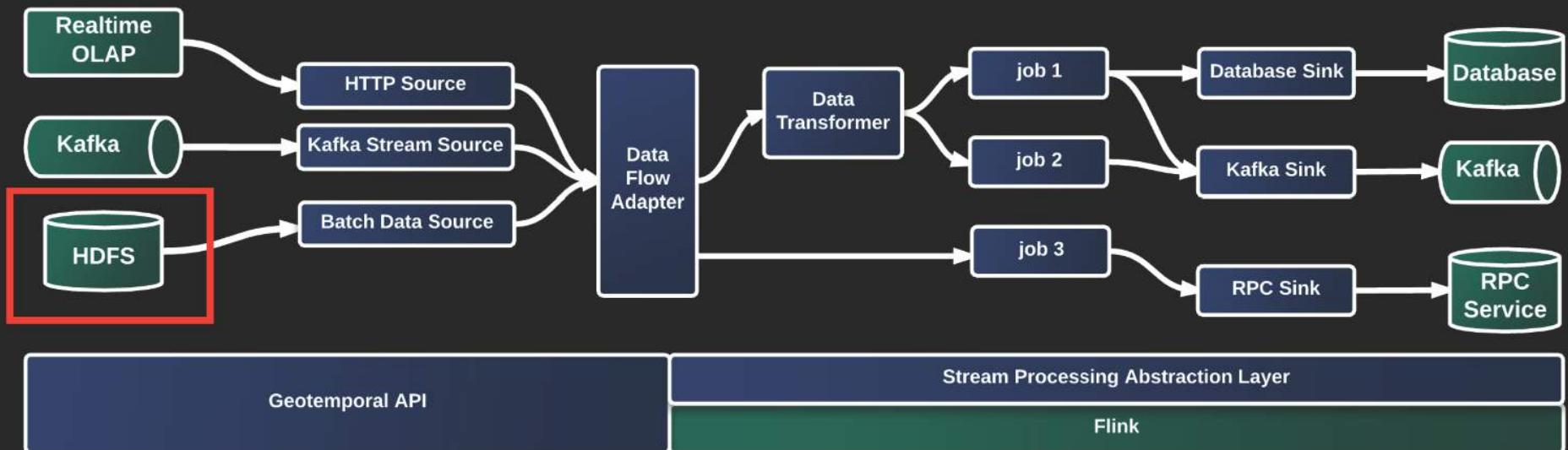
High Level Data Flow



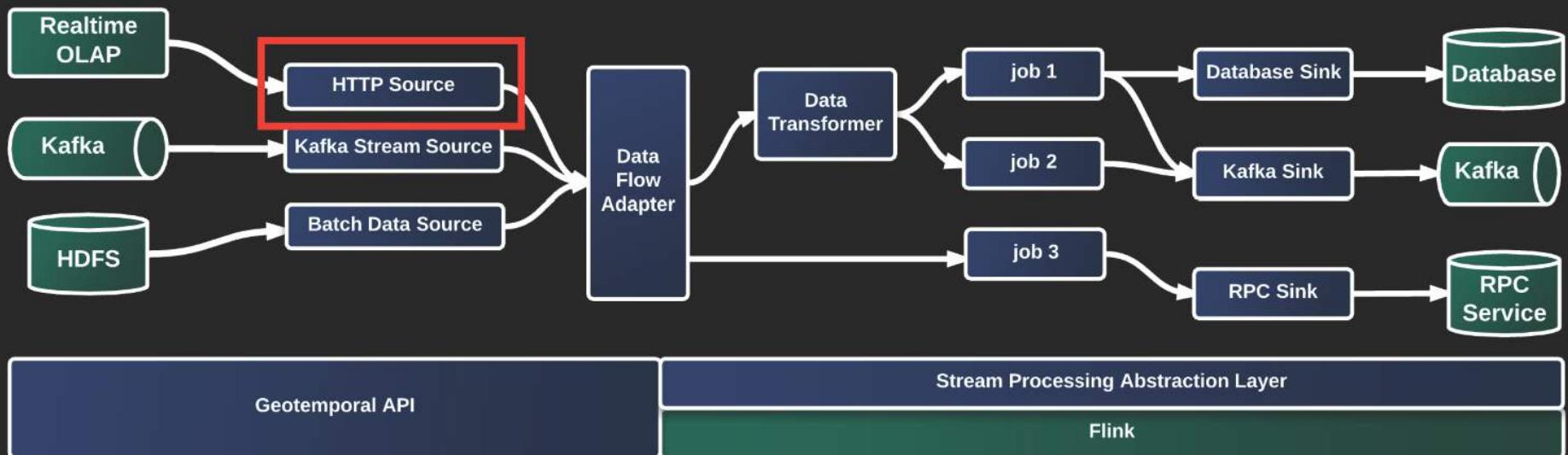
High Level Data Flow



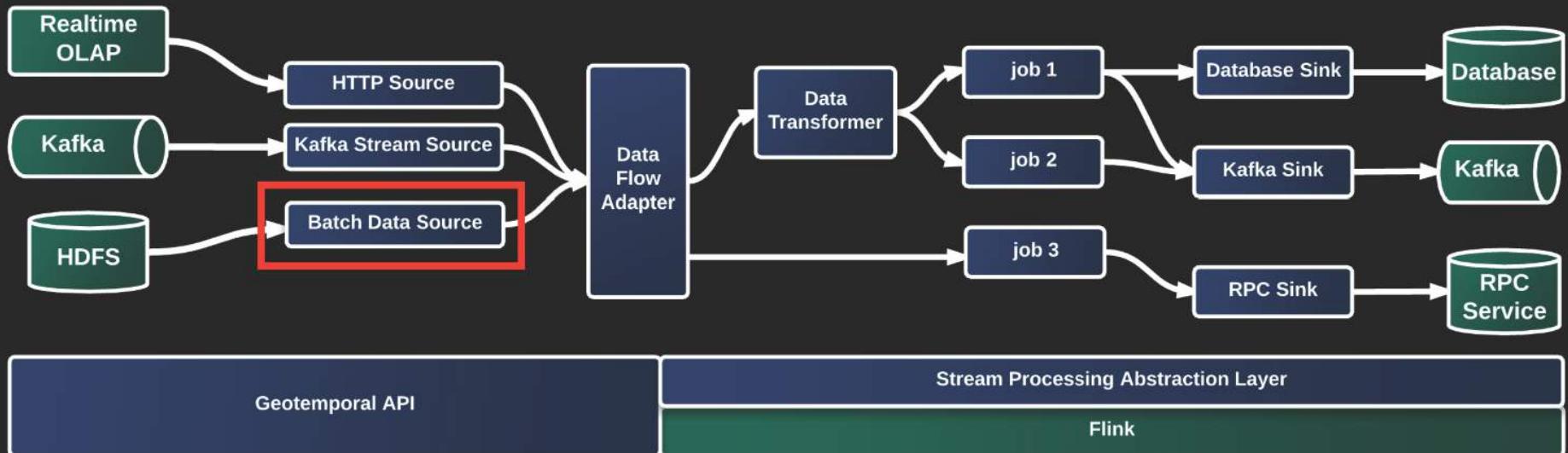
High Level Data Flow



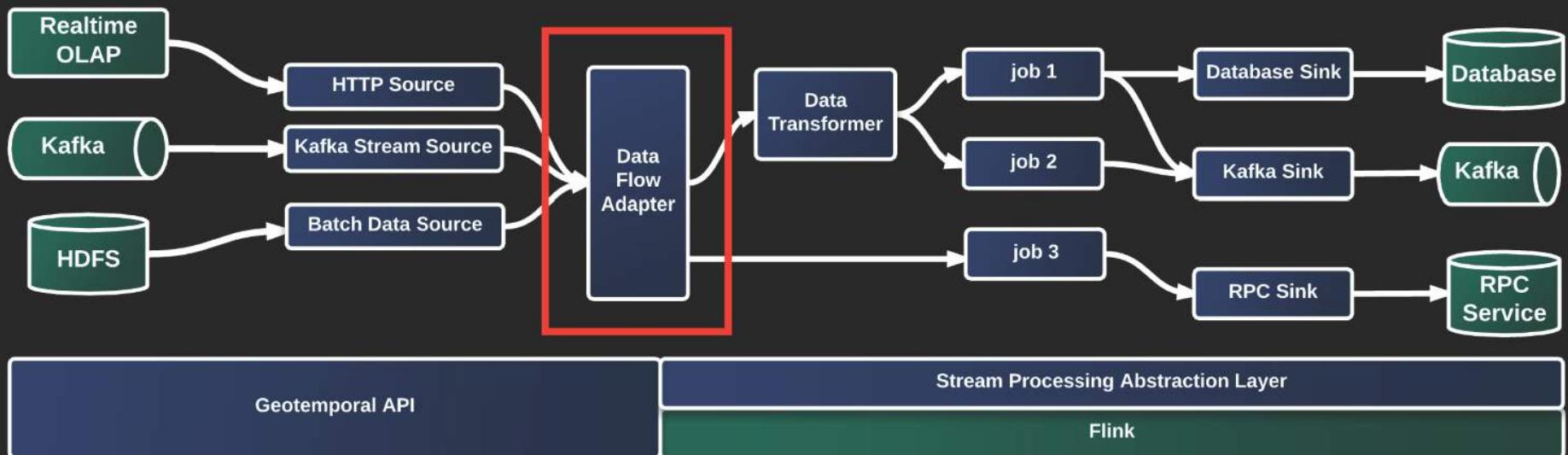
High Level Data Flow



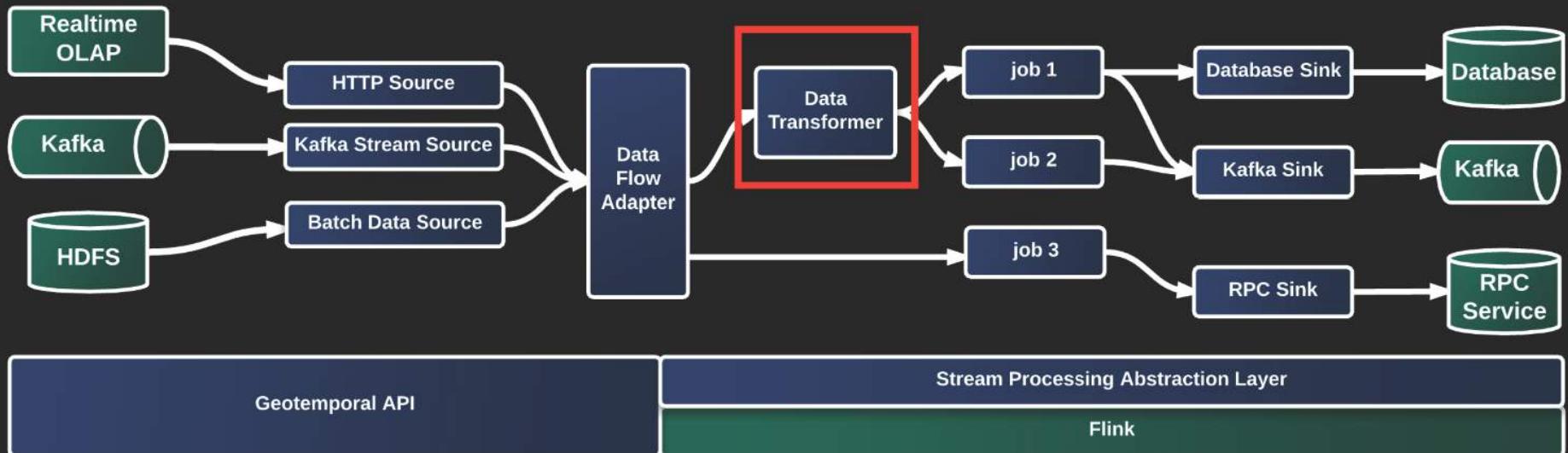
High Level Data Flow



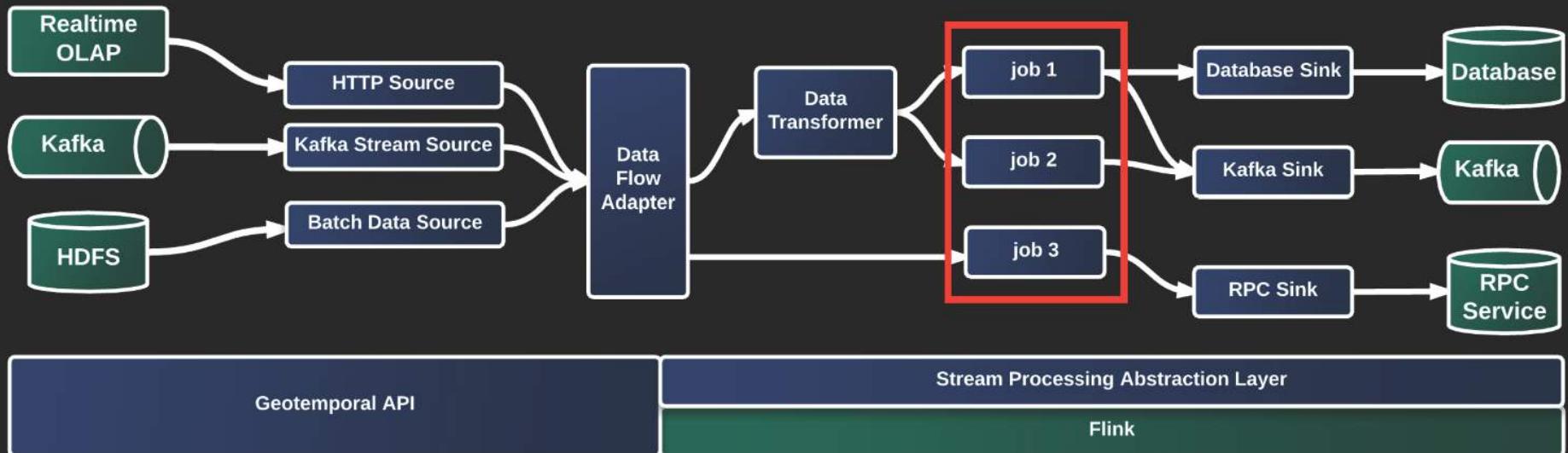
High Level Data Flow



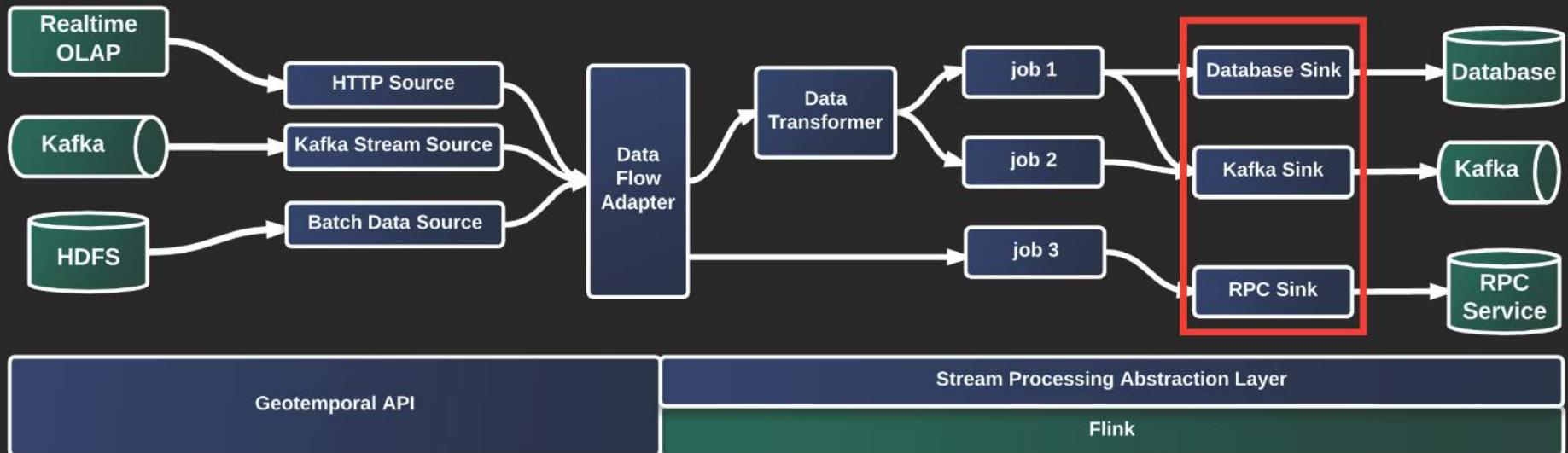
High Level Data Flow



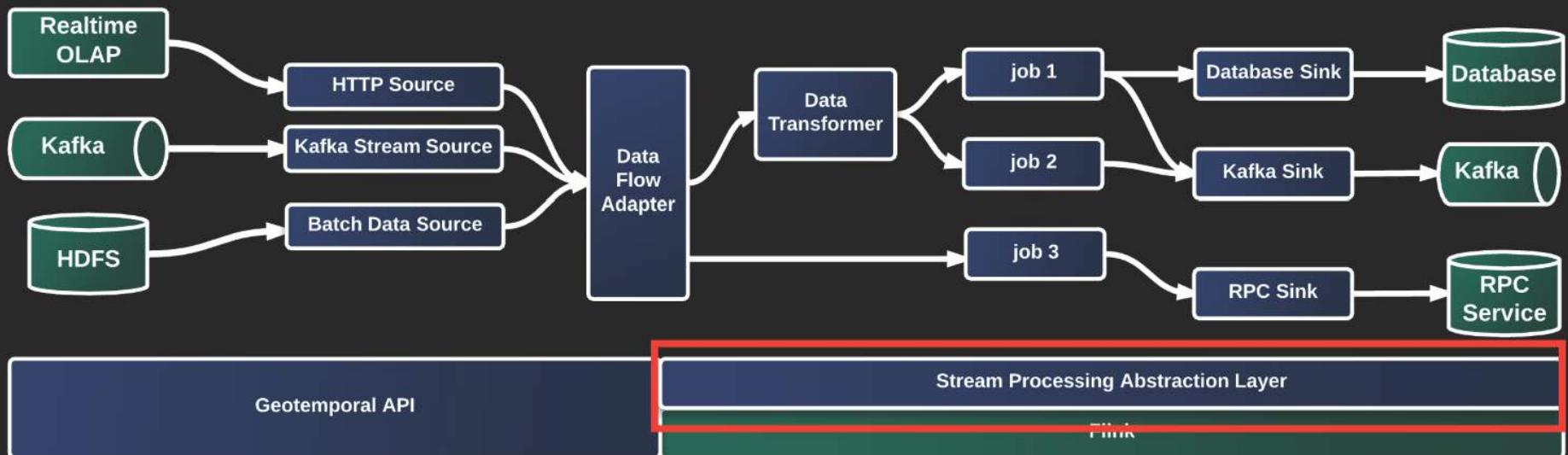
High Level Data Flow



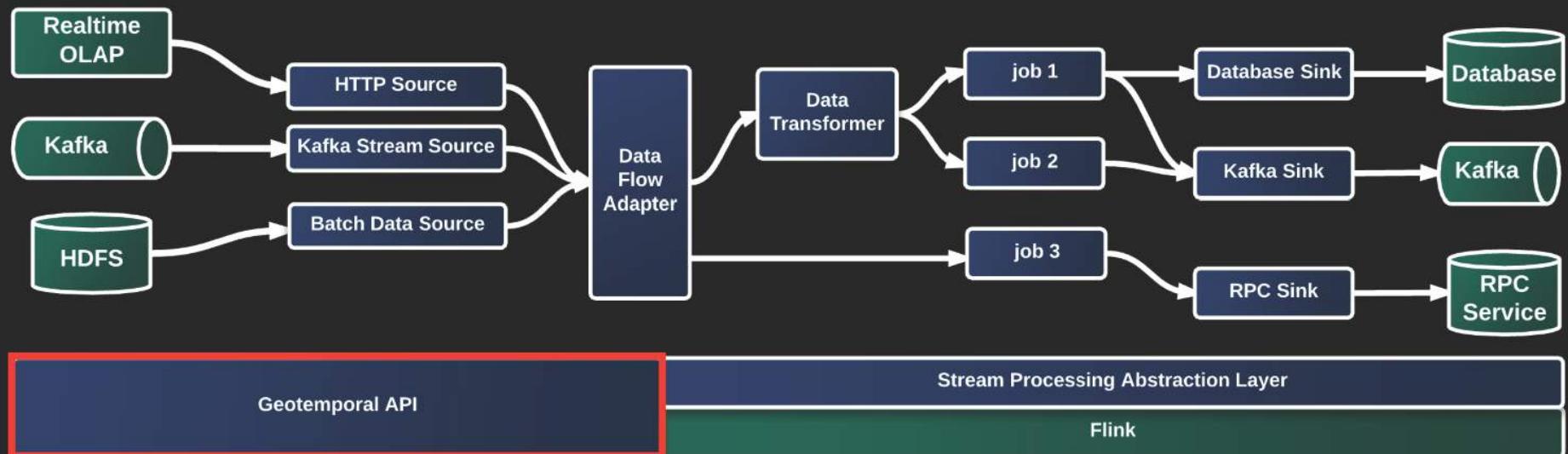
High Level Data Flow



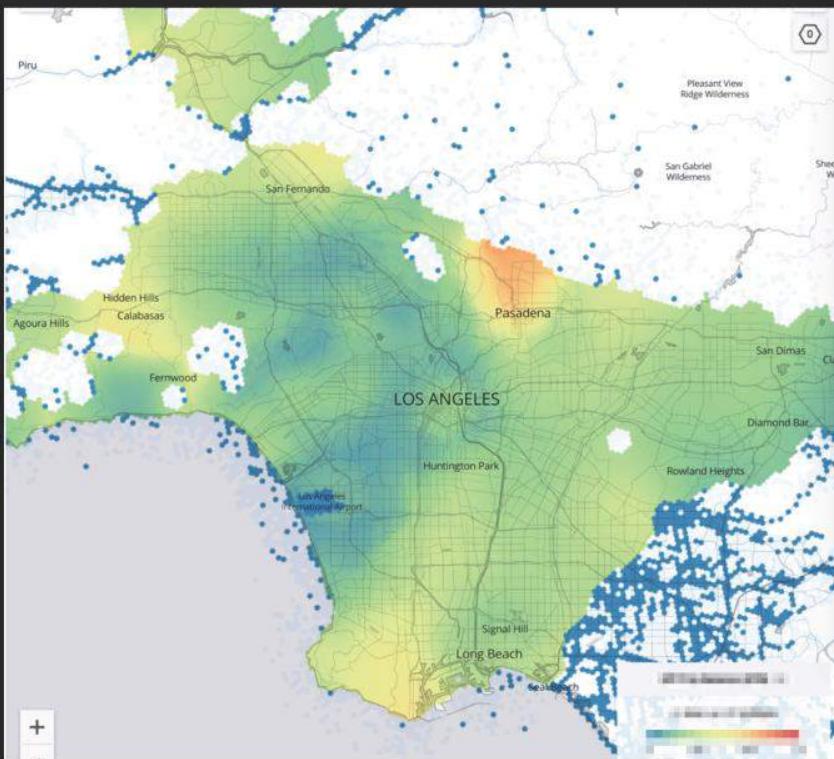
High Level Data Flow



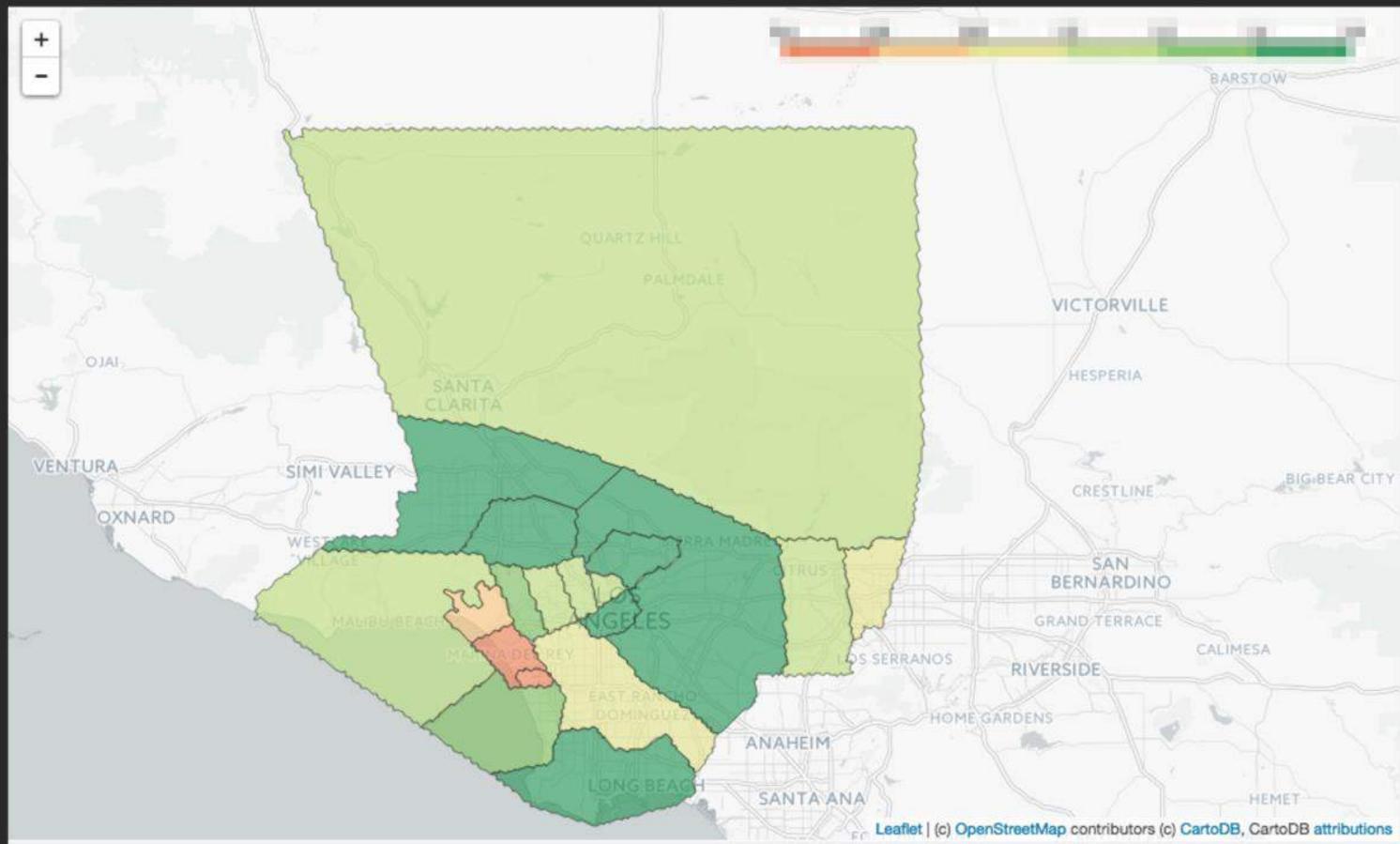
High Level Data Flow



Geotemporal API for efficiency



Geotemporal API for efficiency



Geotemporal API for productivity

```
private static ForkJoinPool fjPool = new ForkJoinPool();

@Override
public void postProcessResult(QueryResult result) {
    ImmutableMap<HexagonCoord, BucketWrapper> hexagons = HexagonAggregationUtility.buildHexagonMap(result, hexField);

    List<BucketWrapper> buckets = Lists.newArrayList(hexagons.values());

    fjPool.invoke(new KRingProcessor(SEQUENTIAL_THRESHOLD, hexagons, buckets, 0, buckets.size()));
}

private class KRingProcessor extends RecursiveTask<List<BucketWrapper>> {
    private int sequentialThreshold;
    private int low;
    private int high;

    private ImmutableMap<HexagonCoord, BucketWrapper> data;
    private List<BucketWrapper> buckets;

    KRingProcessor(int sequentialThreshold,
                  ImmutableMap<HexagonCoord, BucketWrapper> data,
                  List<BucketWrapper> buckets,
                  int low, int high) {
        this.sequentialThreshold = sequentialThreshold;
        this.data = data;
        this.buckets = buckets;
        this.low = low;
        this.high = high;
    }

    @Override
    protected List<BucketWrapper> compute() {
        if (high - low <= sequentialThreshold) {
            for (int i = low; i < high + 1; i++) {
                BucketWrapper bucket = buckets.get(i);
                Map<String, Object> values = bucket.getBucket().getValues();
                if (values.containsKey(hexField) && values.containsKey(metric)) {
                    processBucket(data, bucket);
                }
            }
        } else {
            int mid = low + (high - low) / 2;
            KRingProcessor left = new KRingProcessor(sequentialThreshold, data, buckets, low, mid);
            KRingProcessor right = new KRingProcessor(sequentialThreshold, data, buckets, mid, high);
            left.fork();
            right.compute();
            left.join();
        }
        return buckets;
    }

    private void processBucket(Map<HexagonCoord, BucketWrapper> hexagons, BucketWrapper bucket) {
        HexagonCoord hexCoord = bucket.getCoord();
        
        value();
    }
}
```

Geotemporal API for productivity

```
private static ForkJoinPool fjPool = new ForkJoinPool();

@Override
public void postProcessResult(QueryResult result) {
    ImmutableMap<HexagonCoord, BucketWrapper> hexagons = HexagonAggregationUtility.buildHexagonMap(result, hexField);

    List<BucketWrapper> buckets = Lists.newArrayList(hexagons.values());

    fjPool.invoke(new KRingProcessor(SEQUENTIAL_THRESHOLD, hexagons, buckets, 0, buckets.size()));
}

private class KRingProcessor extends RecursiveTask<List<BucketWrapper>> {
    private int sequentialThreshold;
    private int low;
    private int high;

    private ImmutableMap<HexagonCoord, BucketWrapper> data;
    private List<BucketWrapper> buckets;

    KRingProcessor(int sequentialThreshold,
                  ImmutableMap<HexagonCoord, BucketWrapper> data,
                  List<BucketWrapper> buckets,
                  int low, int high) {
        this.sequentialThreshold = sequentialThreshold;
        this.data = data;
        this.buckets = buckets;
        this.low = low;
        this.high = high;
    }

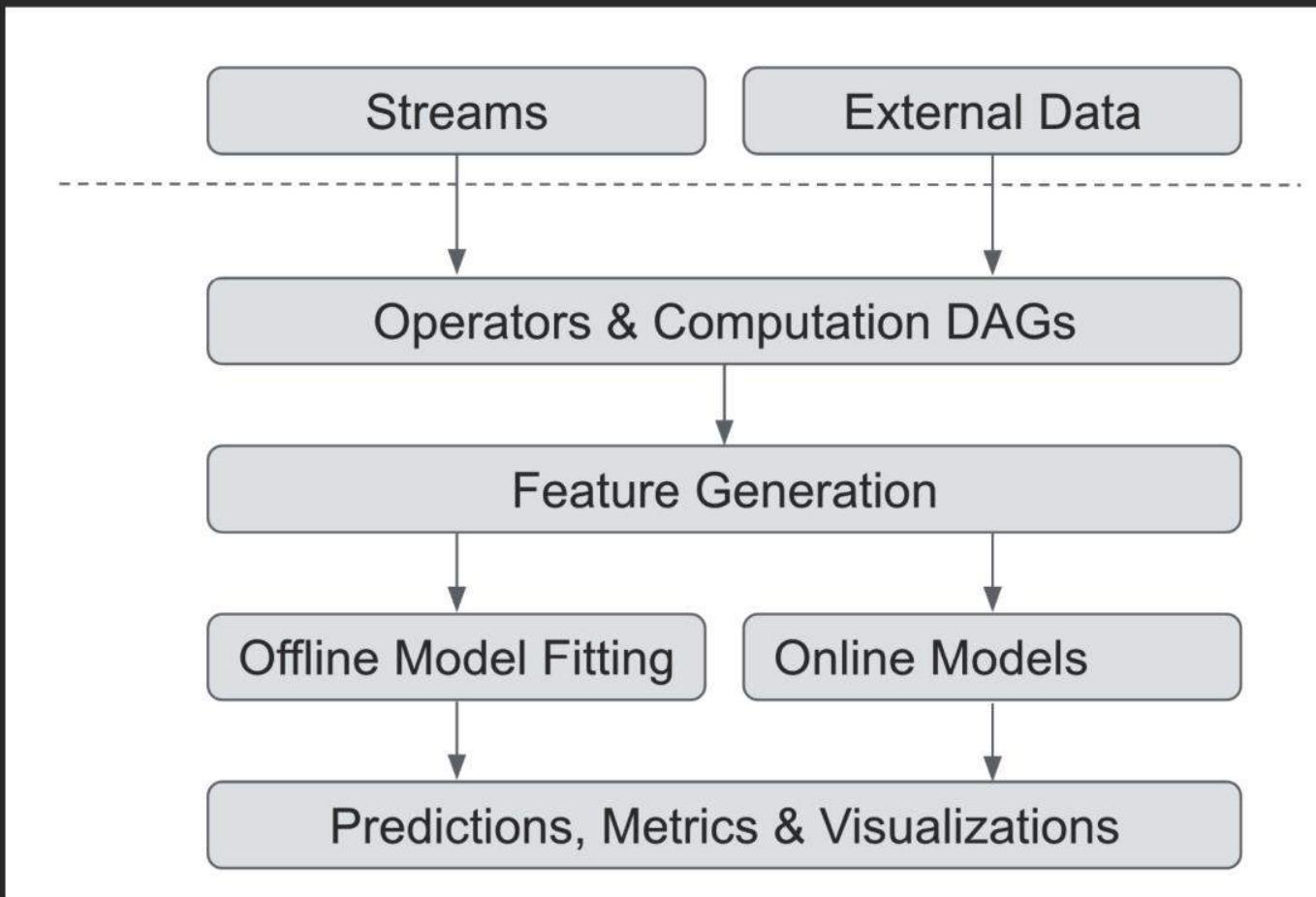
    @Override
    protected List<BucketWrapper> compute() {
        if (high - low <= sequentialThreshold) {
            for (int i = low; i < high + 1; i++) {
                BucketWrapper bucket = buckets.get(i);
                Map<String, Object> values = bucket.getBucket().getValues();
                if (values.containsKey(hexField) && values.containsKey(metric)) {
                    processBucket(data, bucket);
                }
            }
        } else {
            int mid = low + (high - low) / 2;
            KRingProcessor left = new KRingProcessor(sequentialThreshold, data, buckets, low, mid);
            KRingProcessor right = new KRingProcessor(sequentialThreshold, data, buckets, mid, high);
            left.fork();
            right.compute();
            left.join();
        }
        return buckets;
    }

    private void processBucket(ImmutableMap<HexagonCoord, BucketWrapper> data, BucketWrapper bucket) {
        HexagonCoord hexCoord = bucket.getCoord();
        if (!hexCoord.hasCoord()) {
            return;
        }
        Map<String, Object> values = bucket.getBucket().getValues();
        if (values.containsKey(hexField) && values.containsKey(metric)) {
            value();
        }
    }
}
```

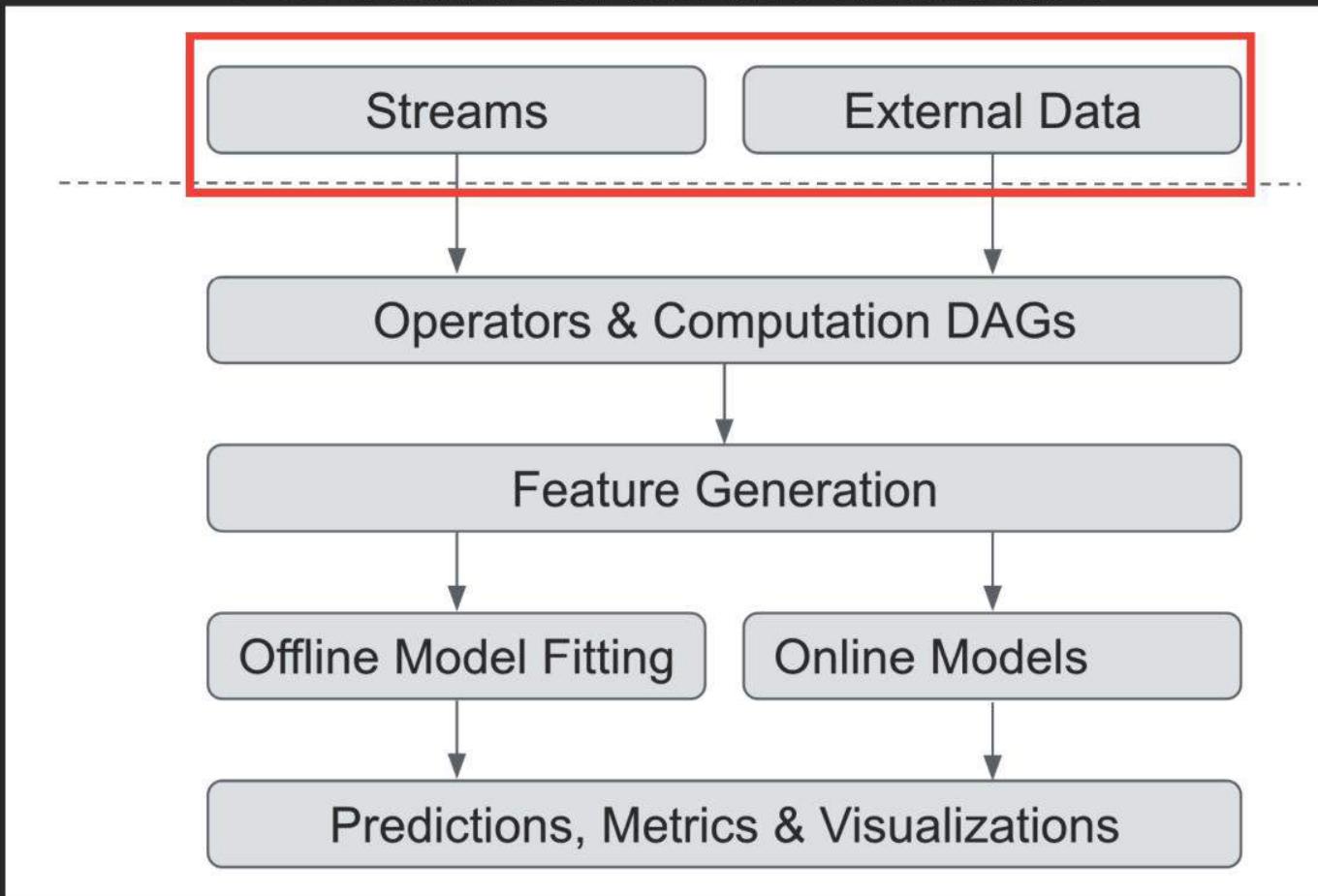


Geotemporal API for productivity

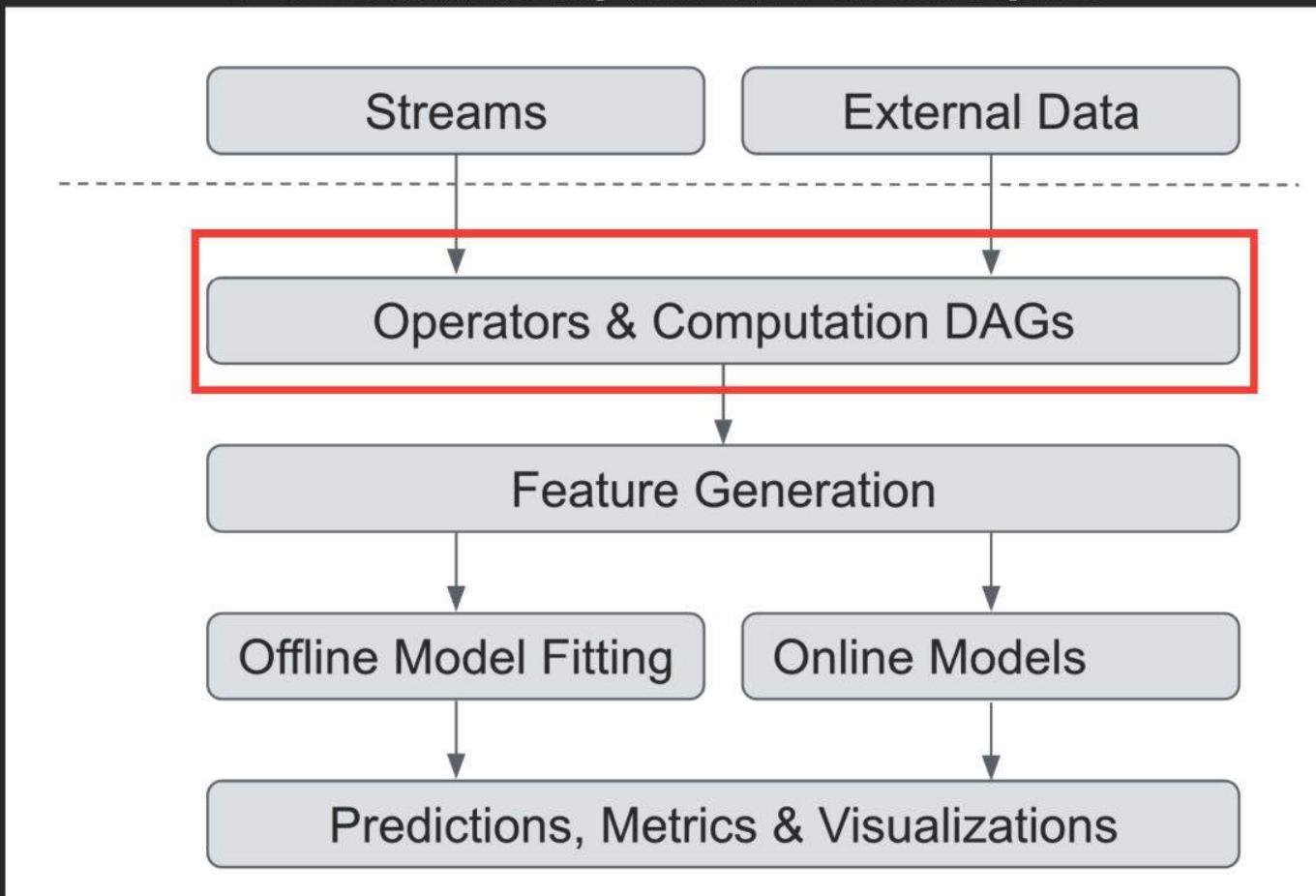
Forecasting as an example



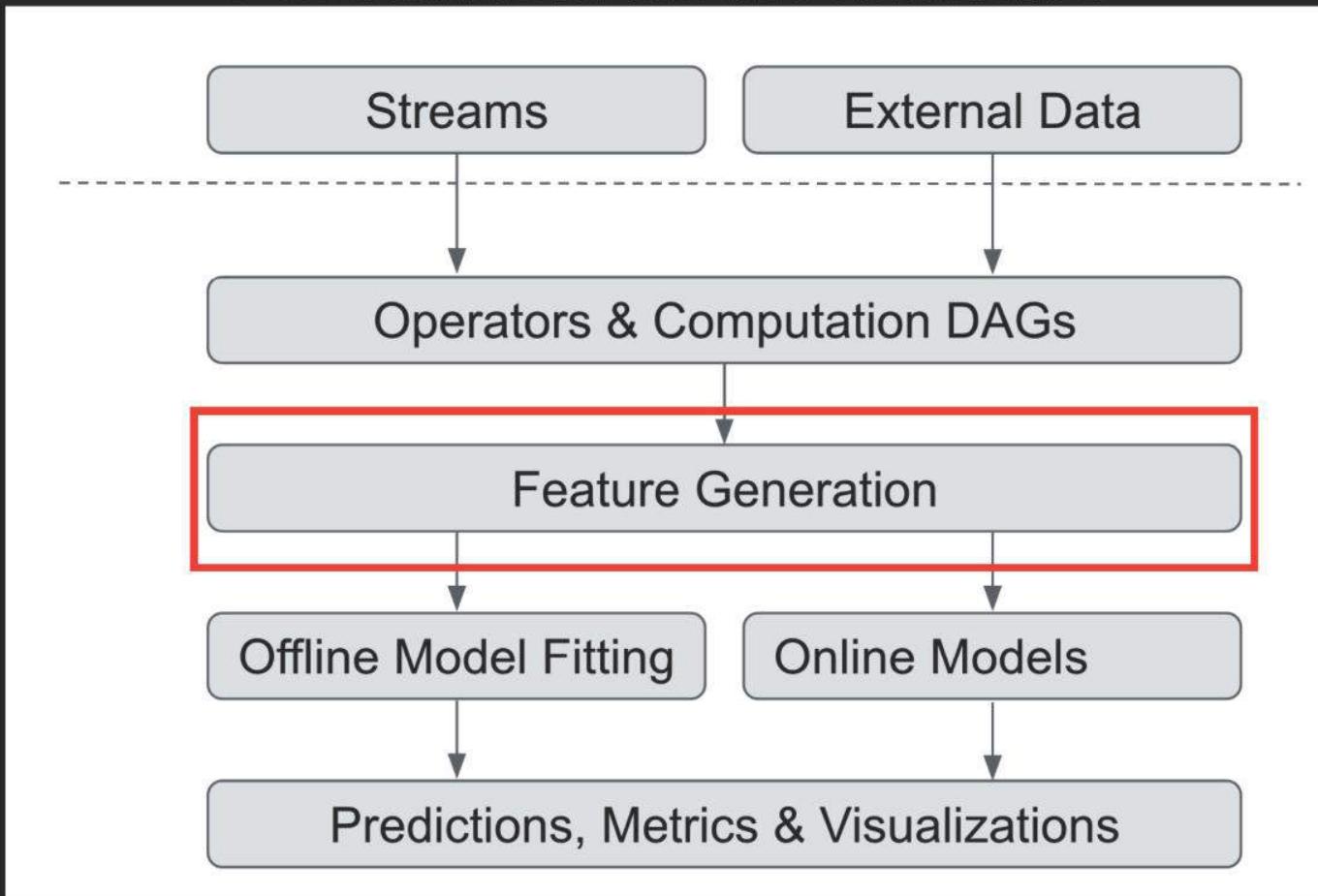
Forecasting as an example



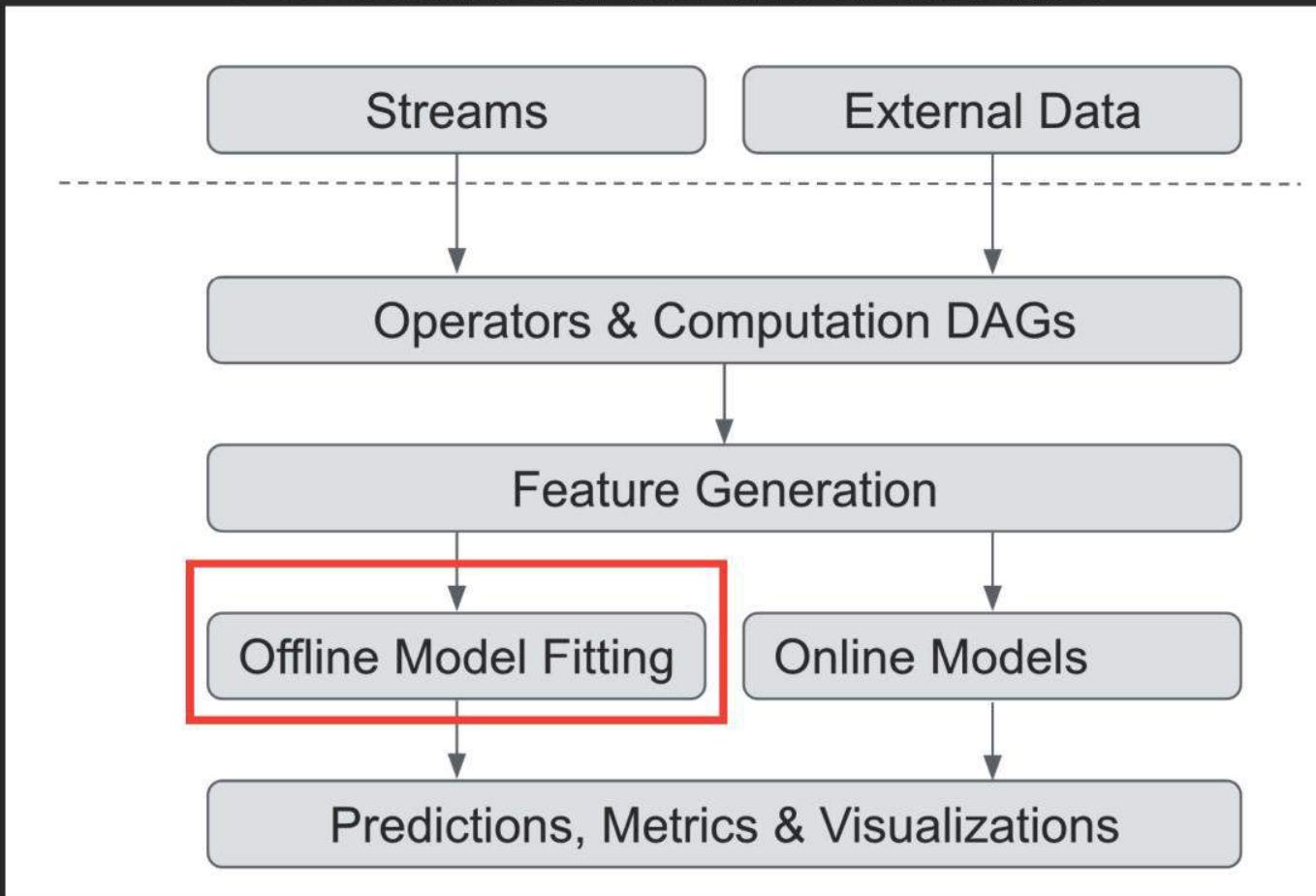
Forecasting as an example



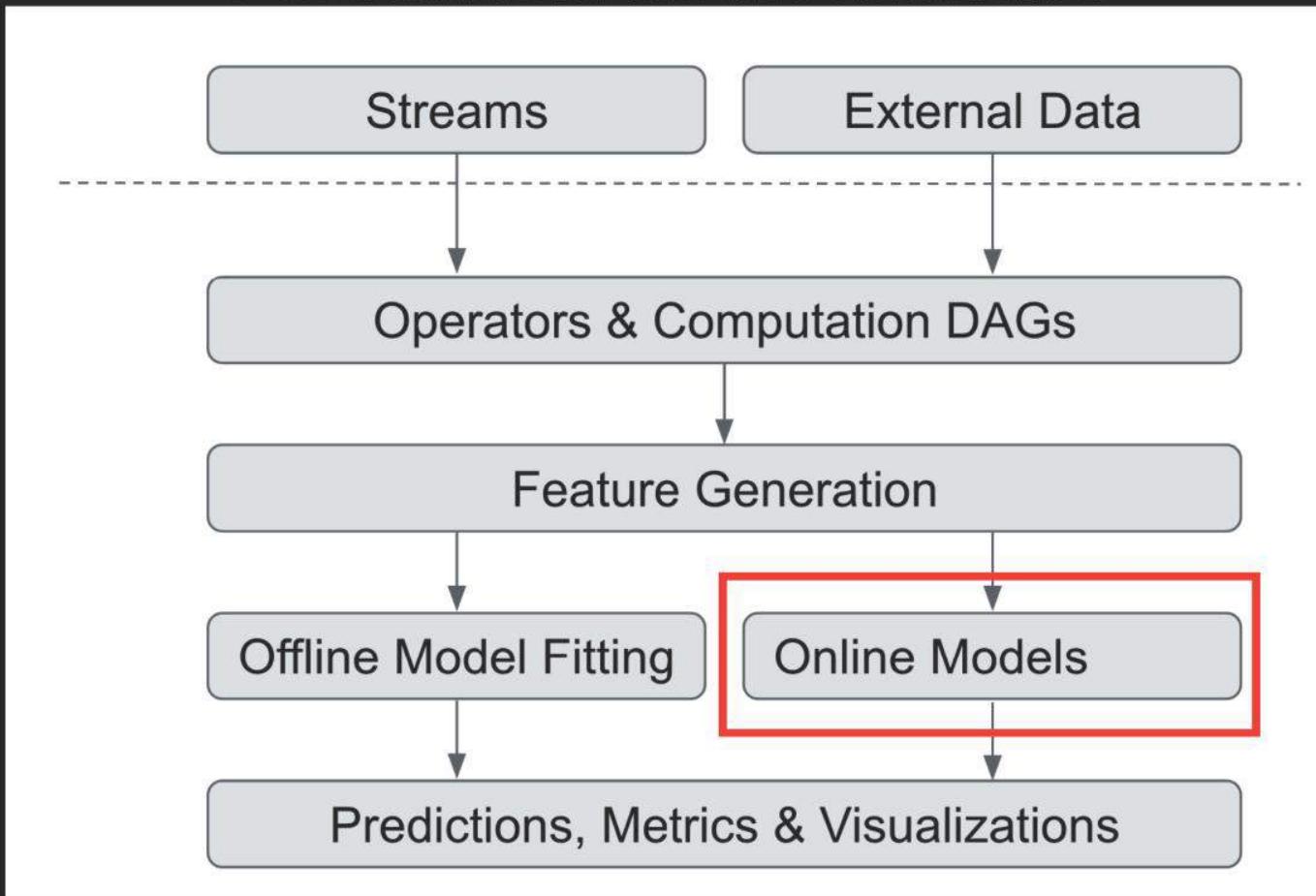
Forecasting as an example



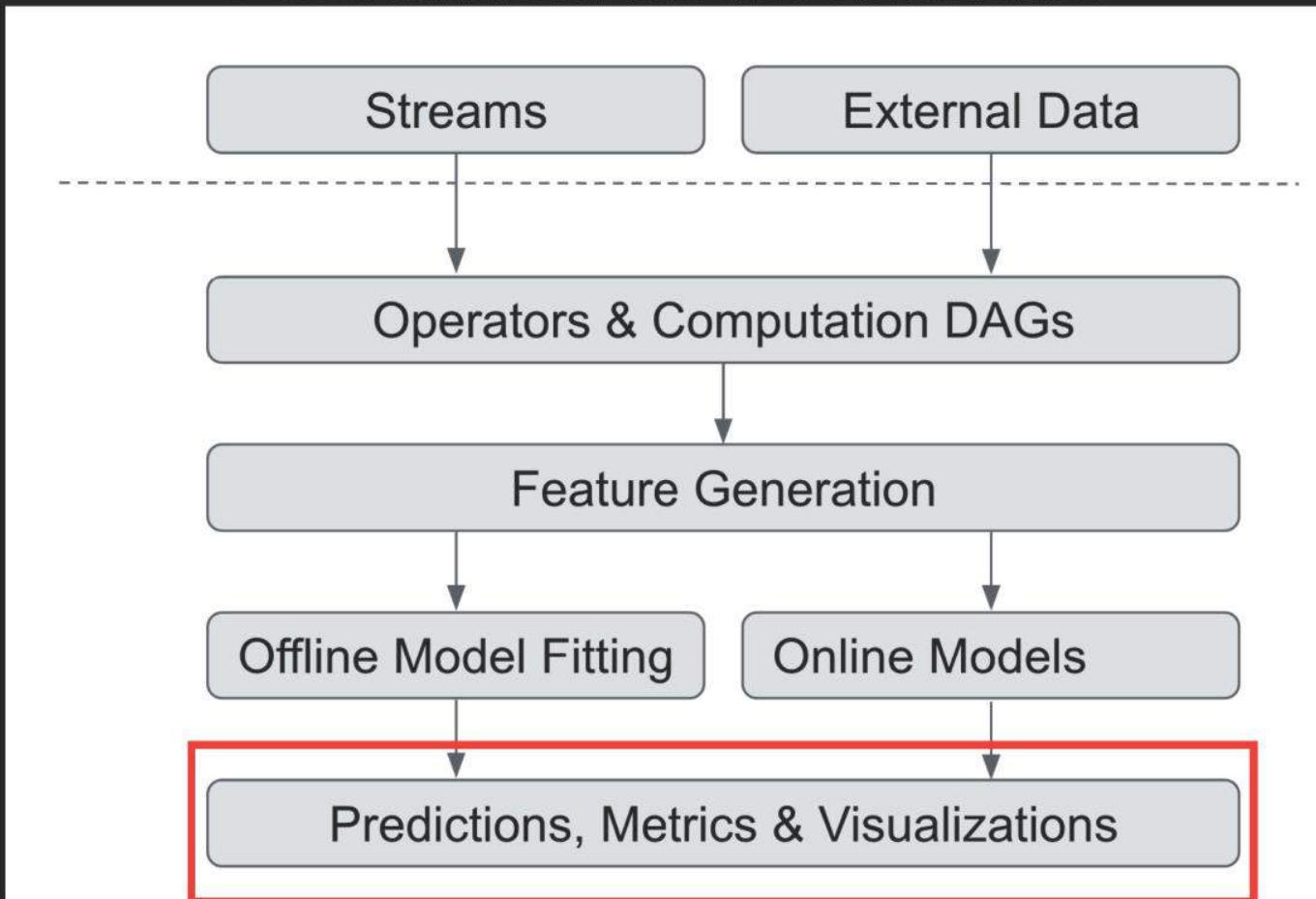
Forecasting as an example



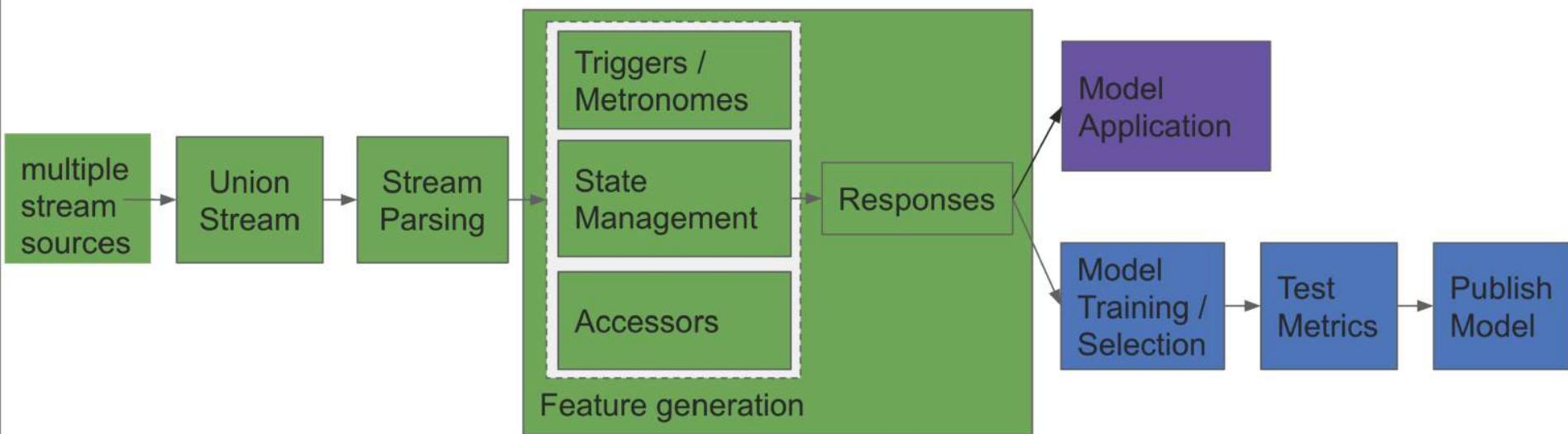
Forecasting as an example



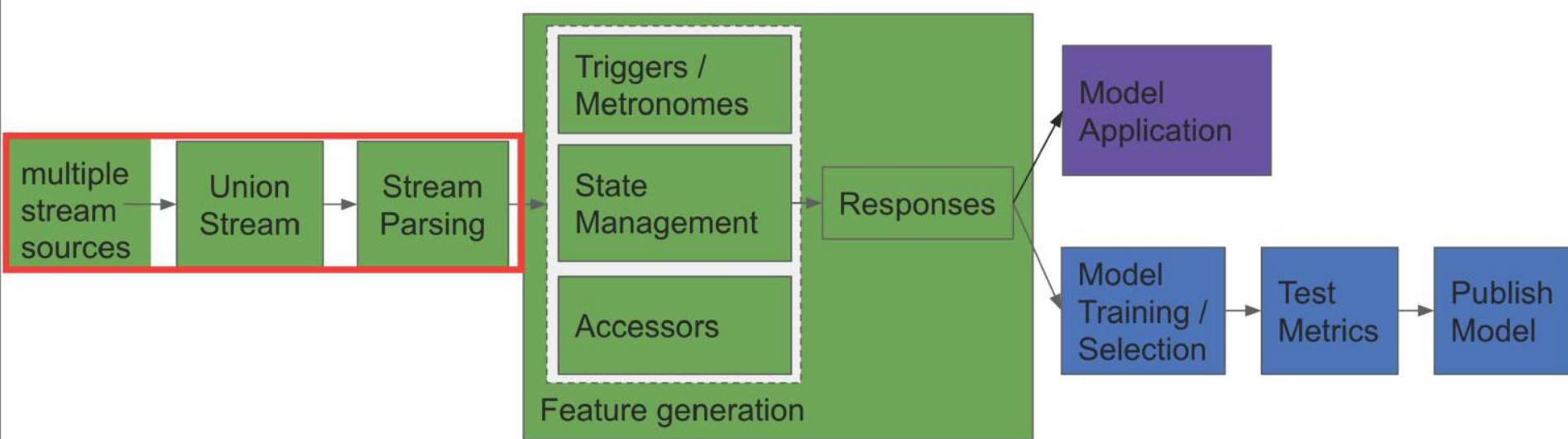
Forecasting as an example



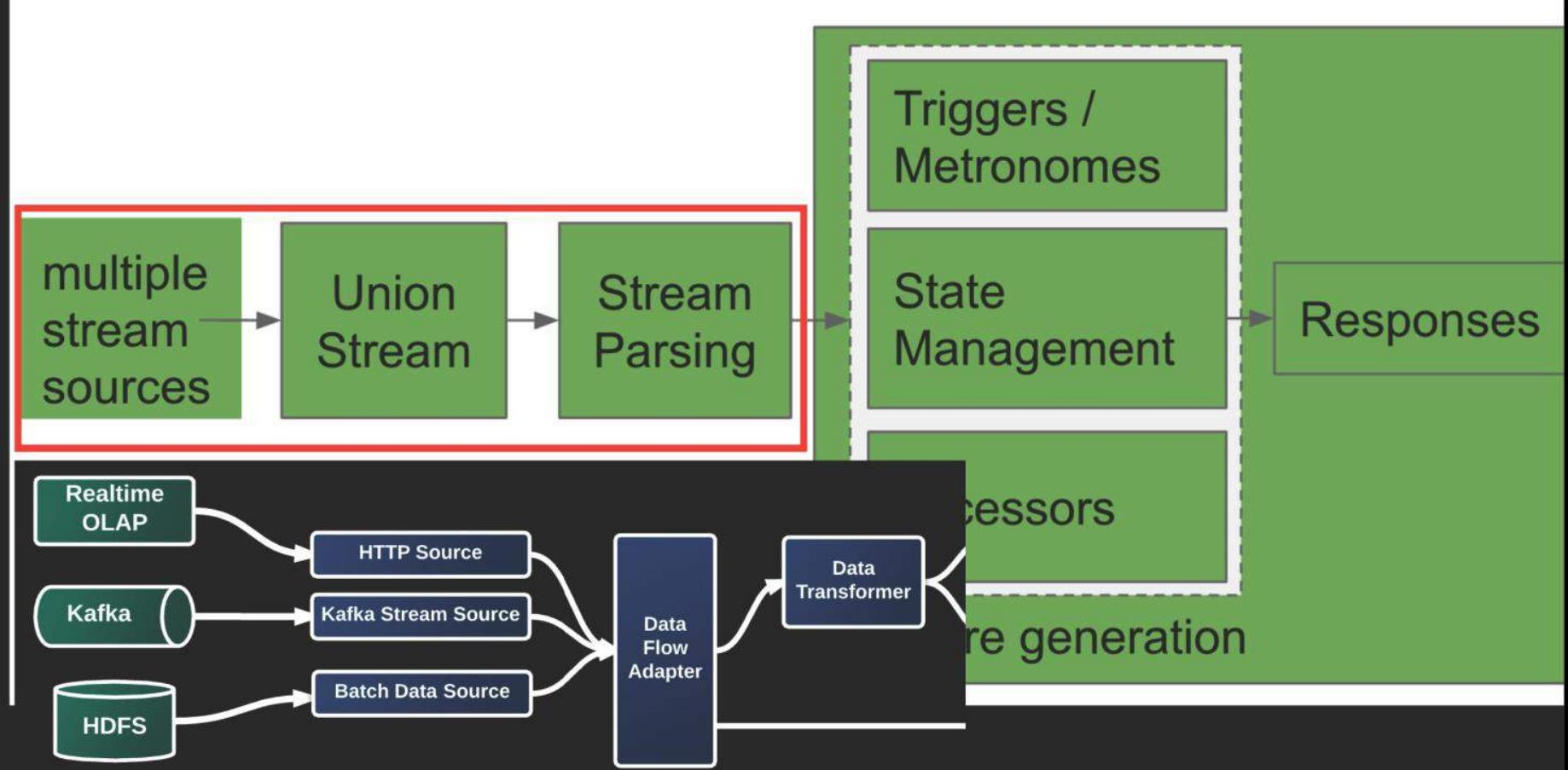
Forecasting as an example



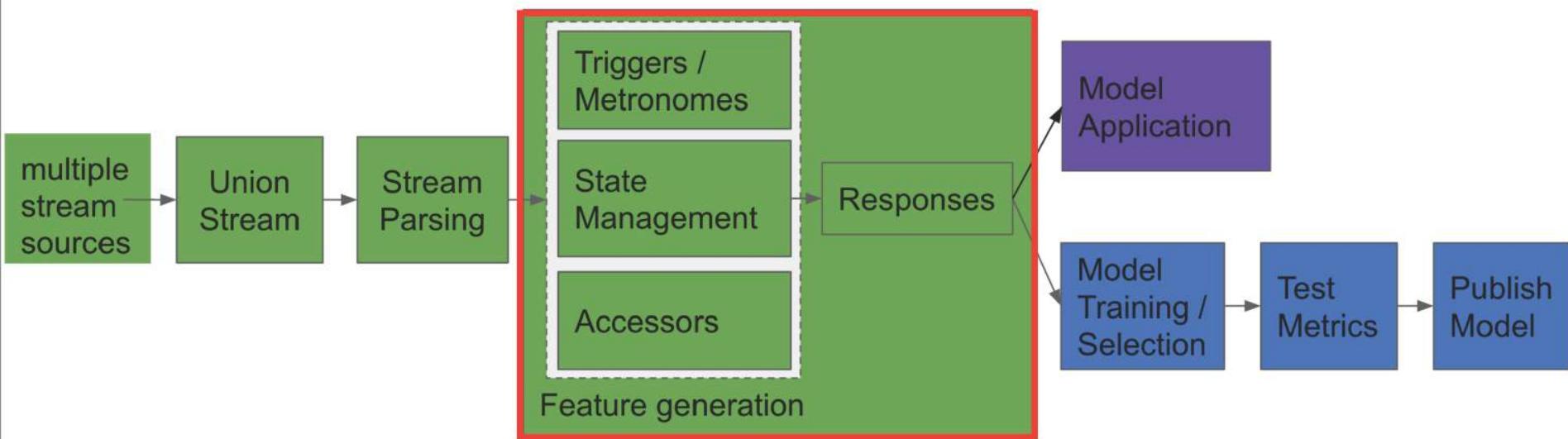
Forecasting as an example



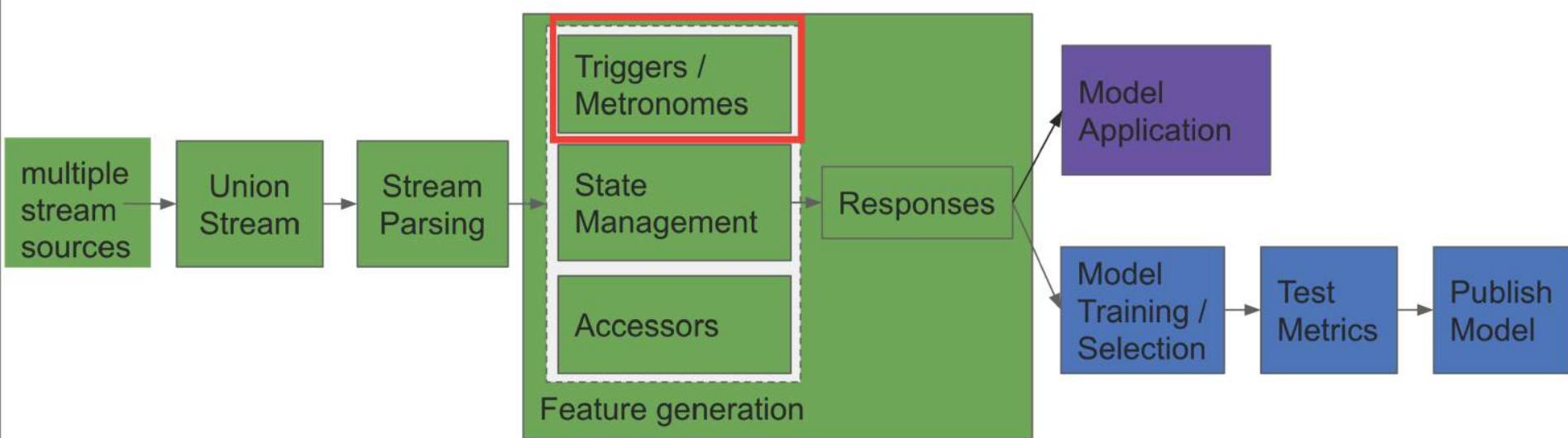
Forecasting as an example



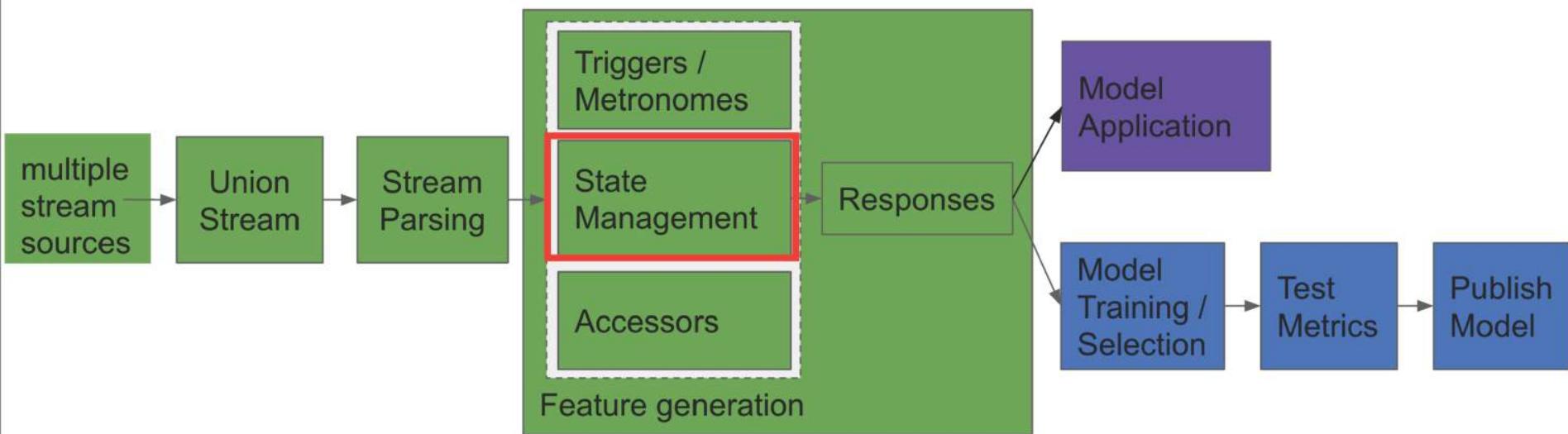
Forecasting as an example



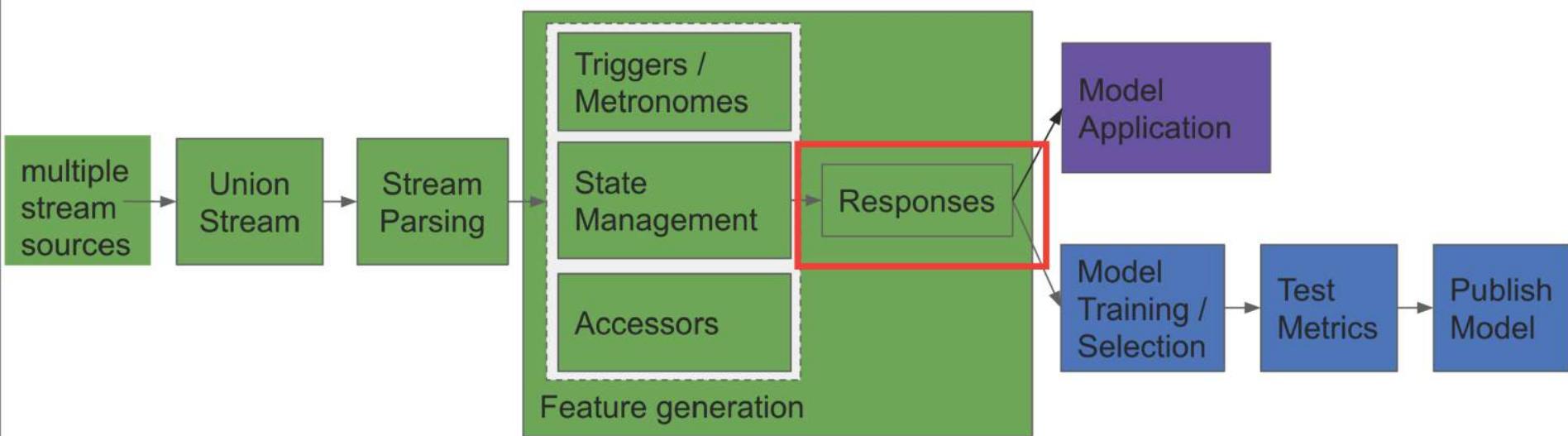
Forecasting as an example



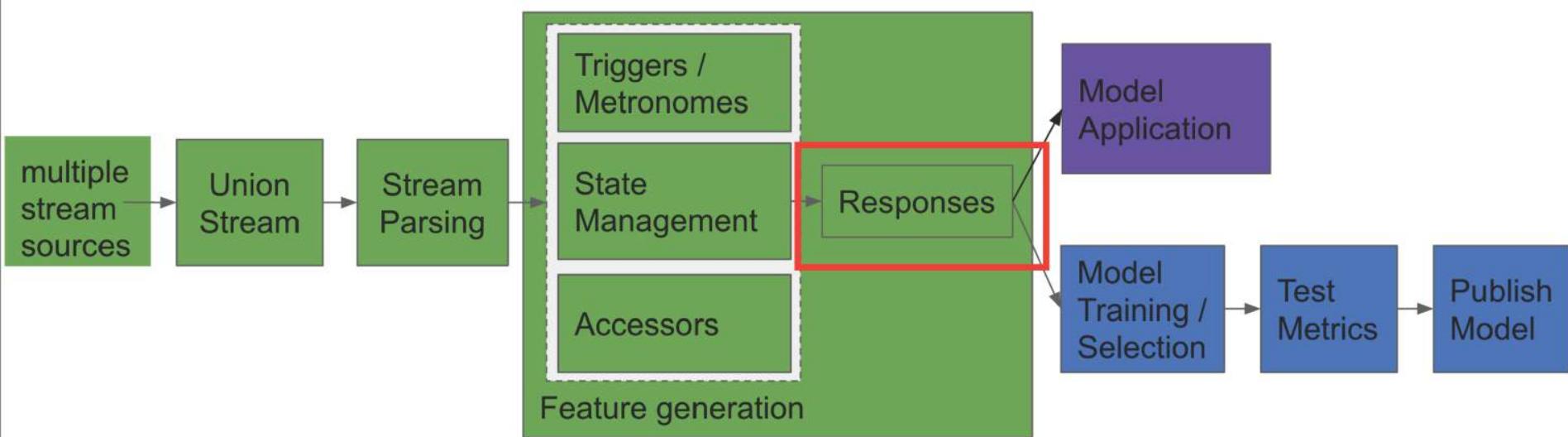
Forecasting as an example



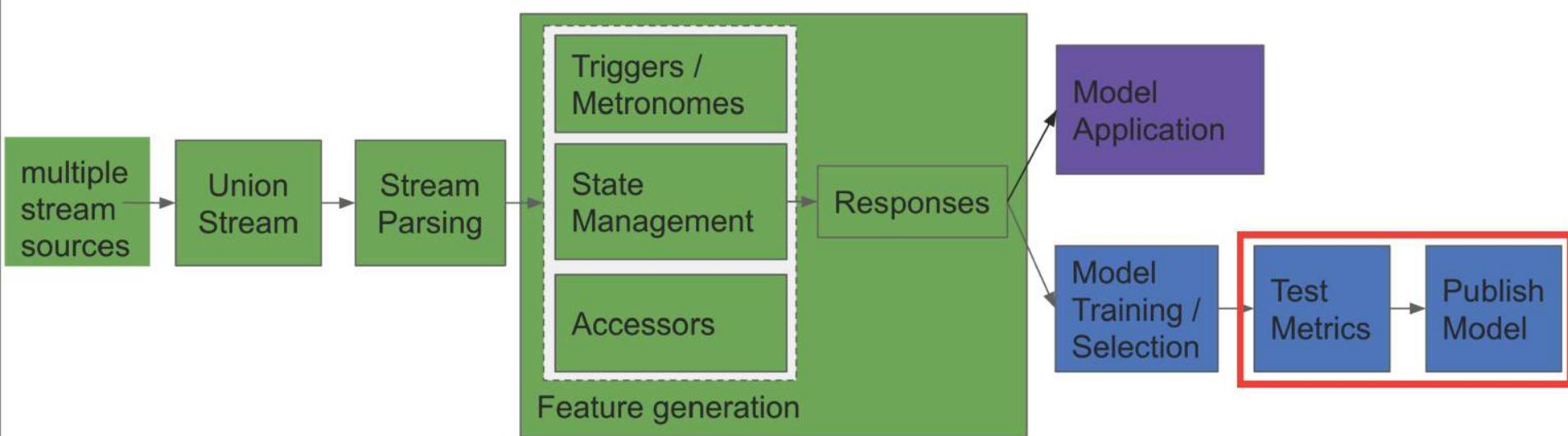
Forecasting as an example



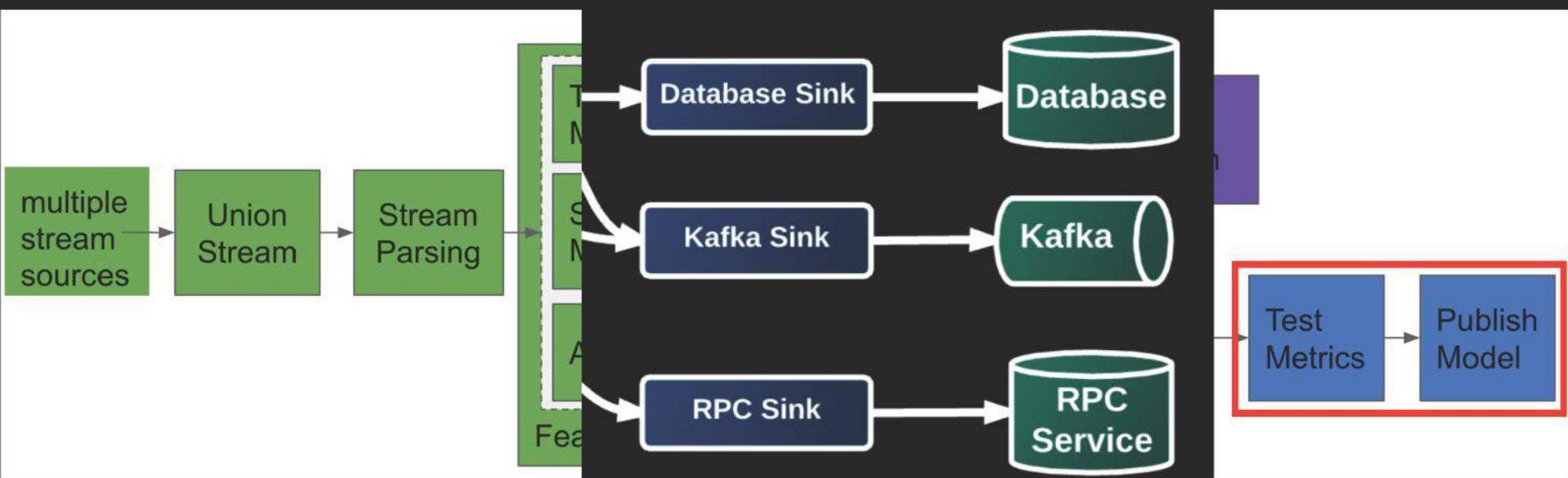
Forecasting as an example



Forecasting as an example



Forecasting as an example



Lessons Learned

- Make sure you have robust infrastructure support
- Scaling up, namely single-node optimization matters
- Ensure exactly-once by proper data modeling
- Use external state store to avoid too much snapshotting
- Standardize monitoring and data validation

Lessons Learned

- Make sure you have robust infrastructure support

Stream System

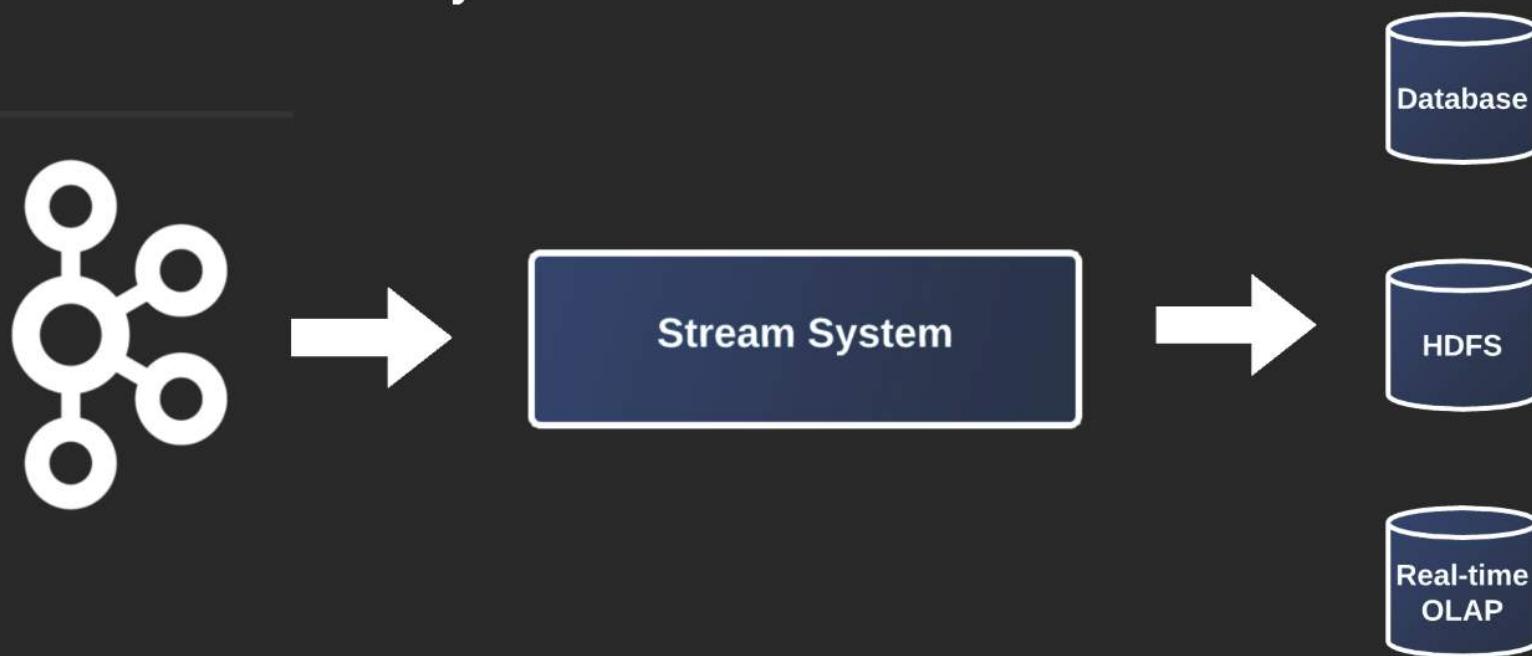
Lessons Learned

- Make sure you have robust infrastructure support



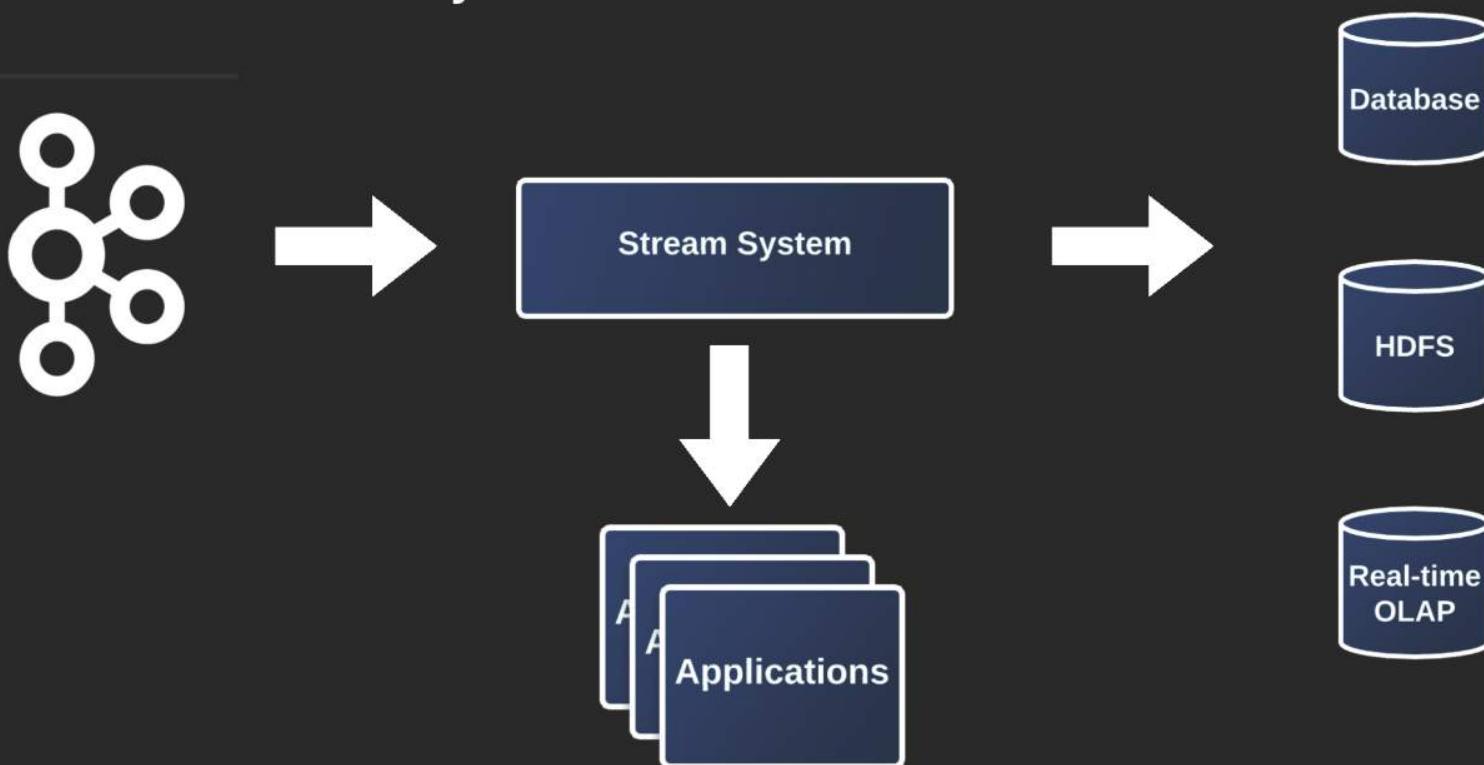
Lessons Learned

- Make sure you have robust infrastructure support



Lessons Learned

- Make sure you have robust infrastructure support



Lessons Learned

- Make sure you have robust infrastructure support
- Scaling up, namely single-node optimization matters

Lessons Learned

- Make sure you have robust infrastructure support
- Scaling up, namely single-node optimization matters
- Ensure exactly-once by proper data modeling

Lessons Learned

- Make sure you have robust infrastructure support
- Scaling up, namely single-node optimization matters
- Ensure exactly-once by proper data modeling
- Use external state store to avoid too much snapshotting
- Standardize monitoring and data validation

Lessons Learned

- Make sure you have robust infrastructure support
- Scaling up, namely single-node optimization matters
- Ensure exactly-once by proper data modeling
- Standardize monitoring and data validation

Choose a Stream Processing Platform

samza



Spark



beam



Thank You

2017 Software Architecture Summit

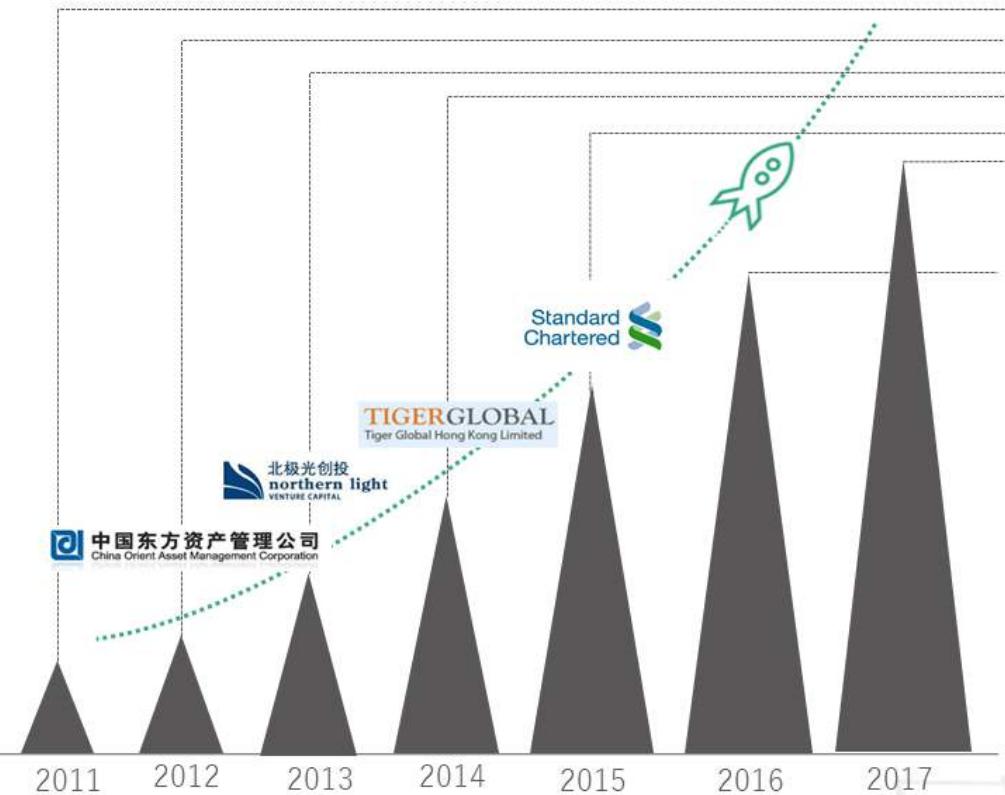
点融金融科技大数据架构

单忆南

高级数据总监



Dianrong History

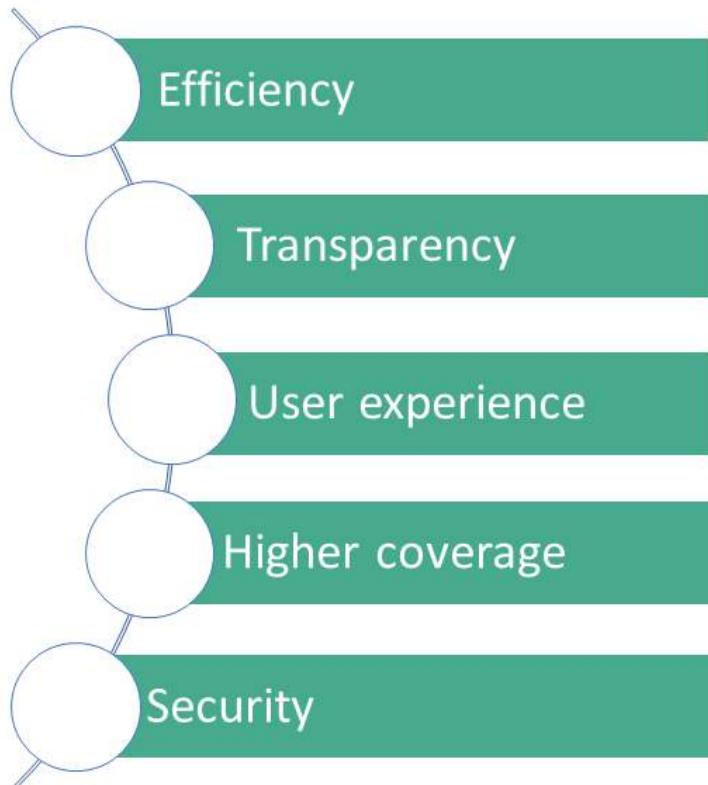


- Founded in Shanghai, China with technology platform from **Lending Club**
- We are **Six** years old
- VC investment from **Standard Northern Light Venture Capital, Tiger Global, Standard Chartered Private Equity, China Internet Fintech Fund, Bohai Financial, and Max Giant**
- Over **2,600** employees and **28** offices across China
- Launched China's First **Block chain platform** with **Foxconn**
- As of end of Sep 2017, total lender volume reached **37 billion RMB**, accumulated interest for lenders exceeded **1.4 billion RMB**.

Contents

- **Fintech Overview**
- Loan business
- Lender management
- Intelligent business management
- Big data solution architecture overview

Fintech Overview



A Range of Technology Strengths

- Payments: Digital wallets, peer-to-peer payments
- Investment: Peer-to-peer lending
- Insurance: Risk management
- Intelligent Advisory: Robo-advisor
- Retail Banking and Supply Chain Finance
- Financing: Crowdfunding, micro-loans and credit facilities
- Block Chain: Bitcoin

Contents

- Fintech Overview
- **Loan business**
- Lender management
- Intelligent business management
- Big data solution architecture overview

Big Data Applications

➤ Loan Business

- Fraud Detection
- Credit worthiness
- Intelligent collection
- Risk based pricing

Data requirement	Engineering	Algorithm
Borrower Profile Social network data PBOC Loan status	Third party data integration Extensible/flexible Near real time processing Support ML model	LR scorecard GBDT scorecard
Sessions, Views and Actions Lender profile	Behavior data extracting and cleaning Large scale data mining real time calculation/searching	Clustering Categorizing
Business data Business relations	OLAP Dashboard Large scale data simulation	Statistic model Monte Carlo

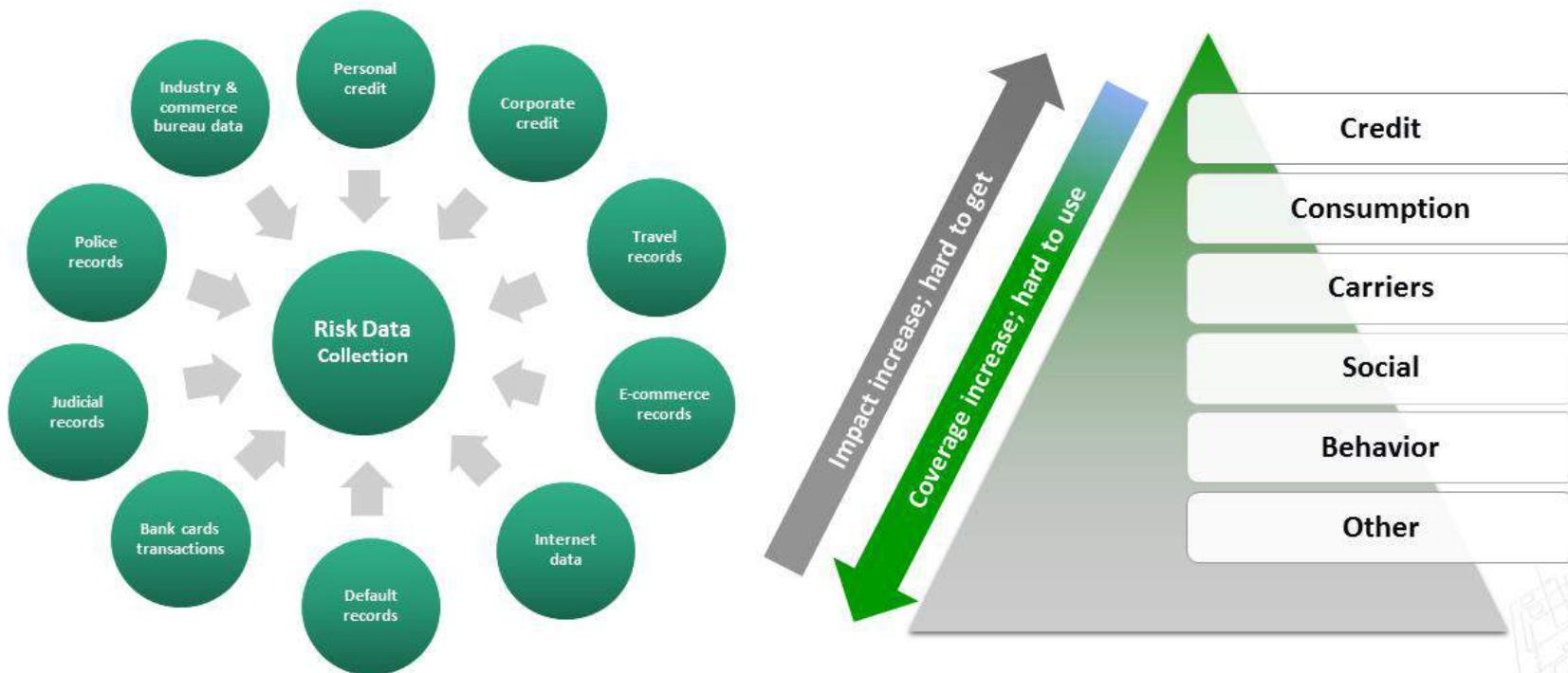
➤ Customer Management

- User profile
- Behavior analysis
- Behavior prediction
- Call center operation
- User segmentation
- Anti Econnoisseur

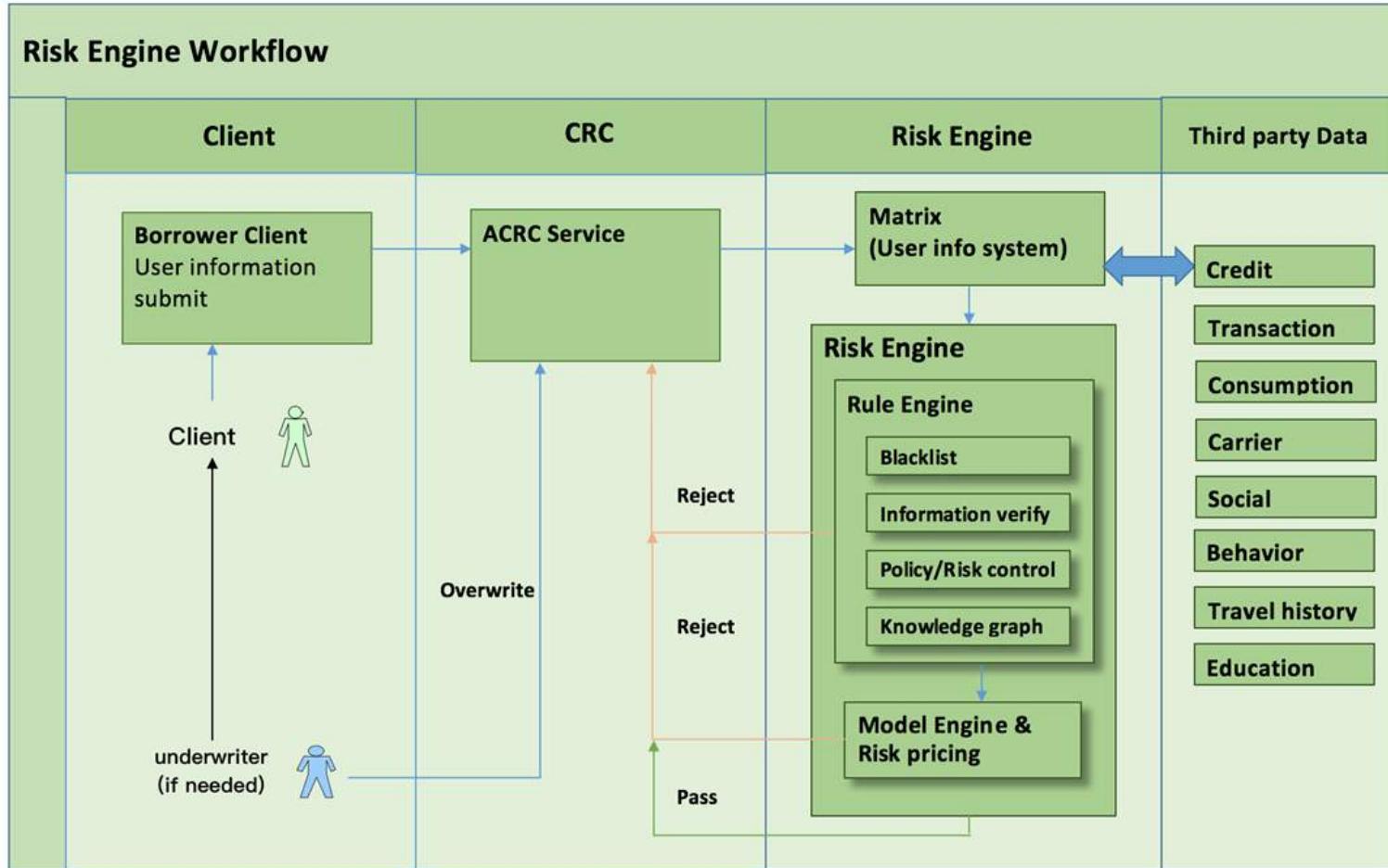
➤ Business Management

- Risk management
- Portfolio management

Loan Business – Data Collection



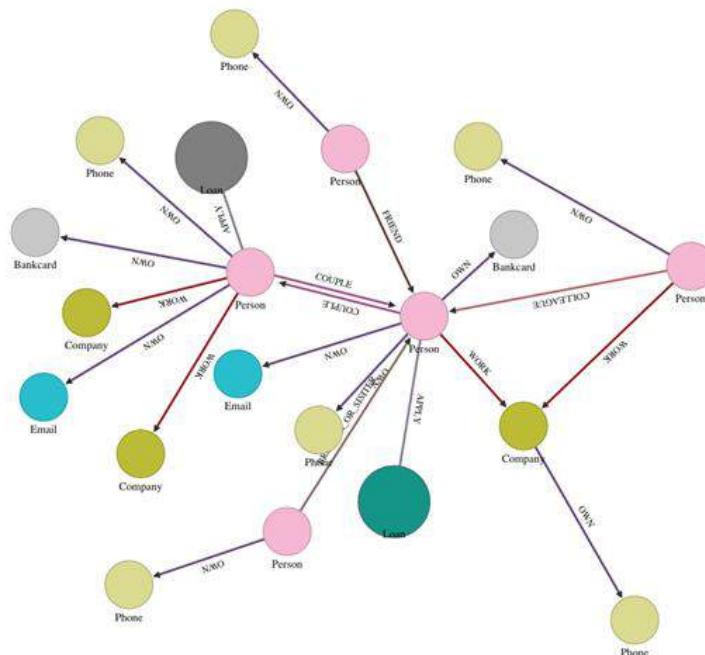
Loan Business- Online Risk Engine Workflow



Loan Business – Knowledge graph

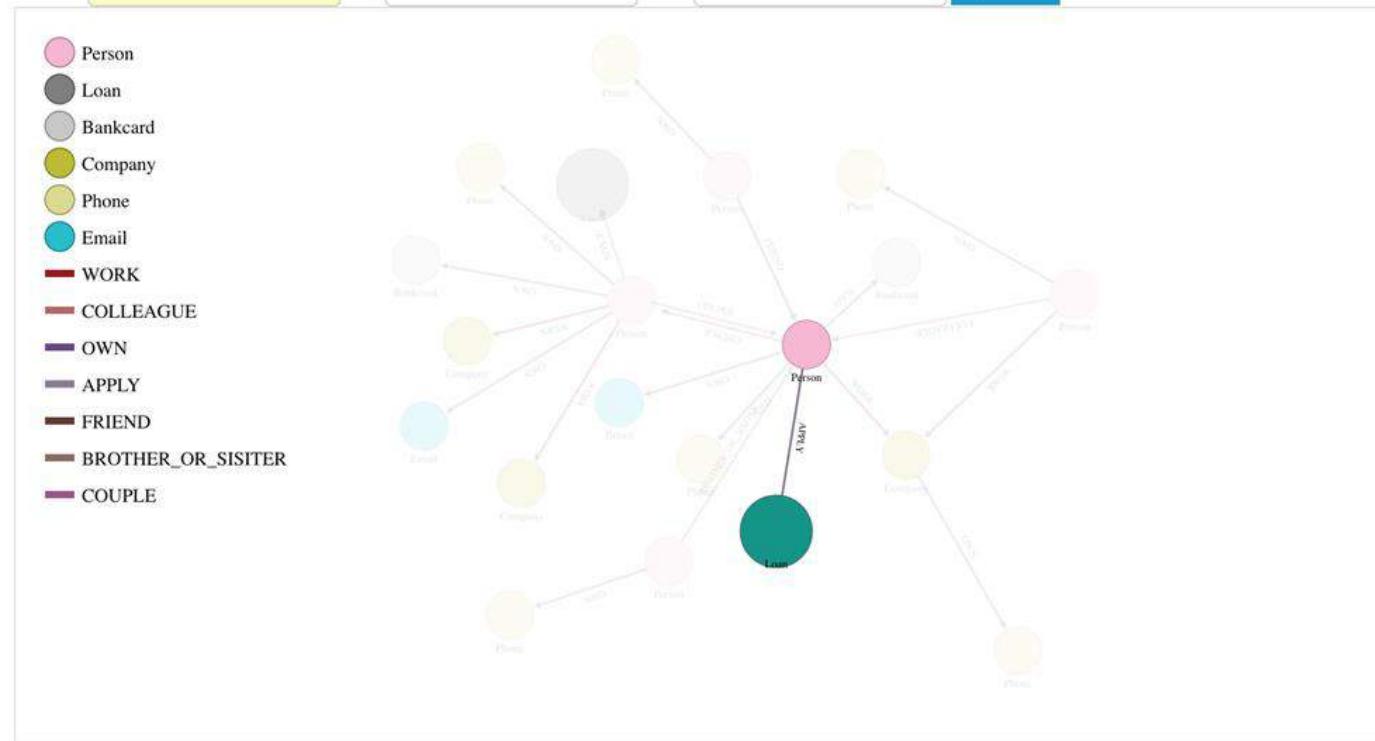
Loan ID: 280229 Hop: 5 Limit: 20 **QUERY**

- Person
- Loan
- Bankcard
- Company
- Phone
- Email
- WORK
- COLLEAGUE
- OWN
- APPLY
- FRIEND
- BROTHER_OR_SISITER
- COPUPLE



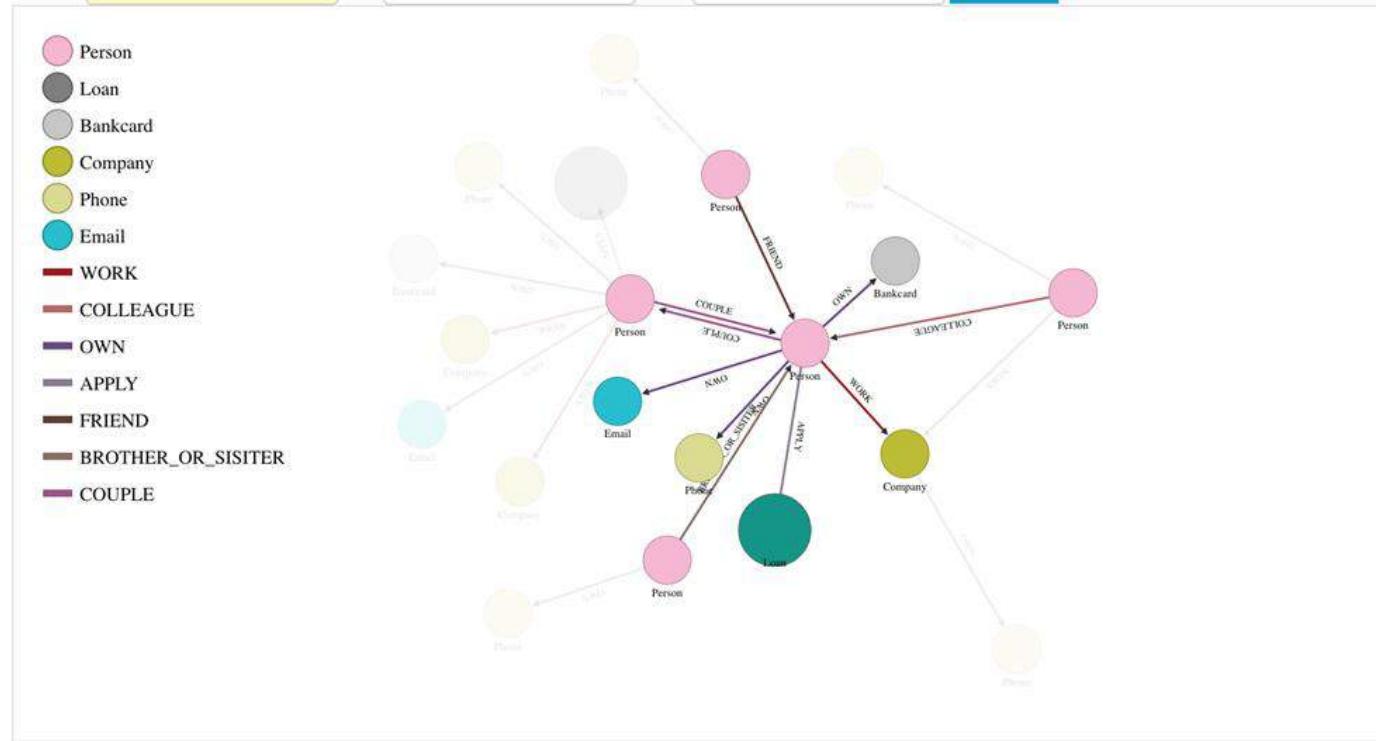
Loan Business – Knowledge graph

Loan ID 280229 Hop 5 Limit 20 **QUERY**



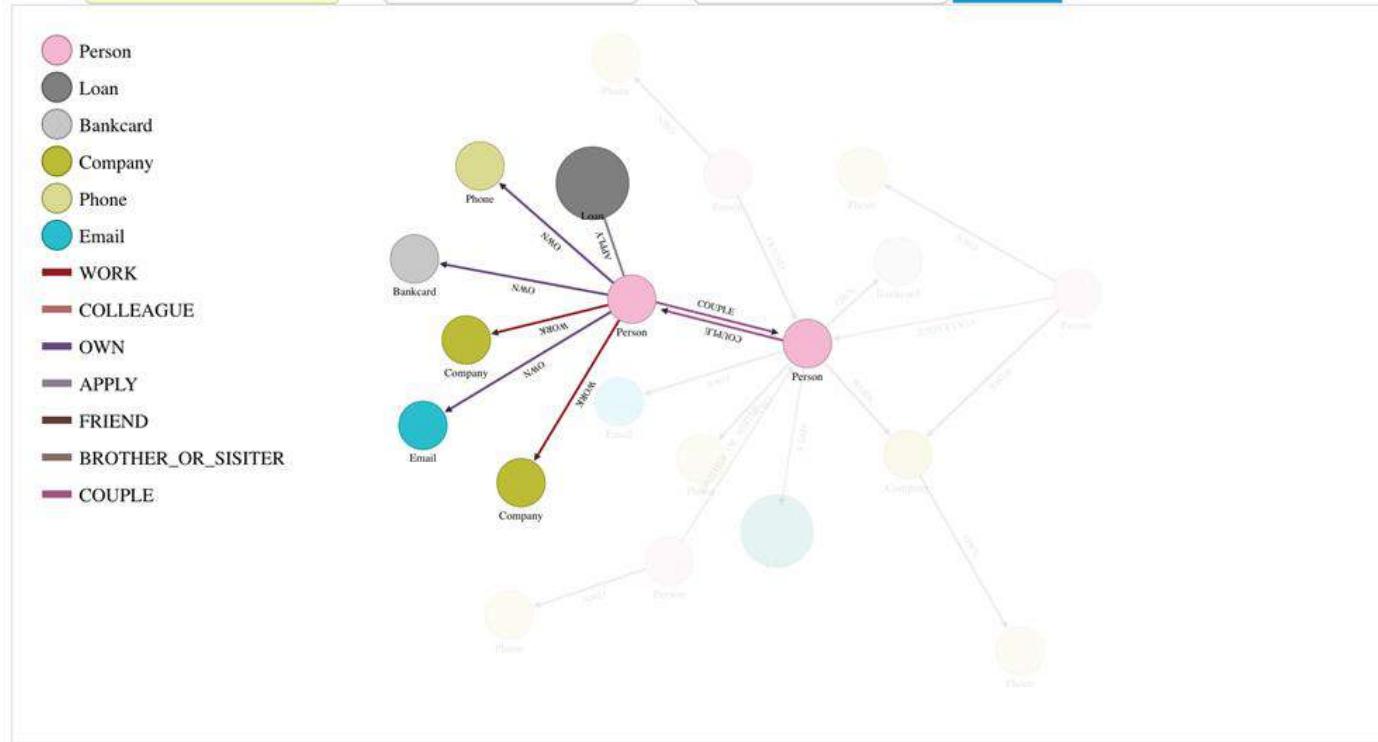
Loan Business – Knowledge graph

Loan ID 280229 Hop 5 Limit 20 QUERY

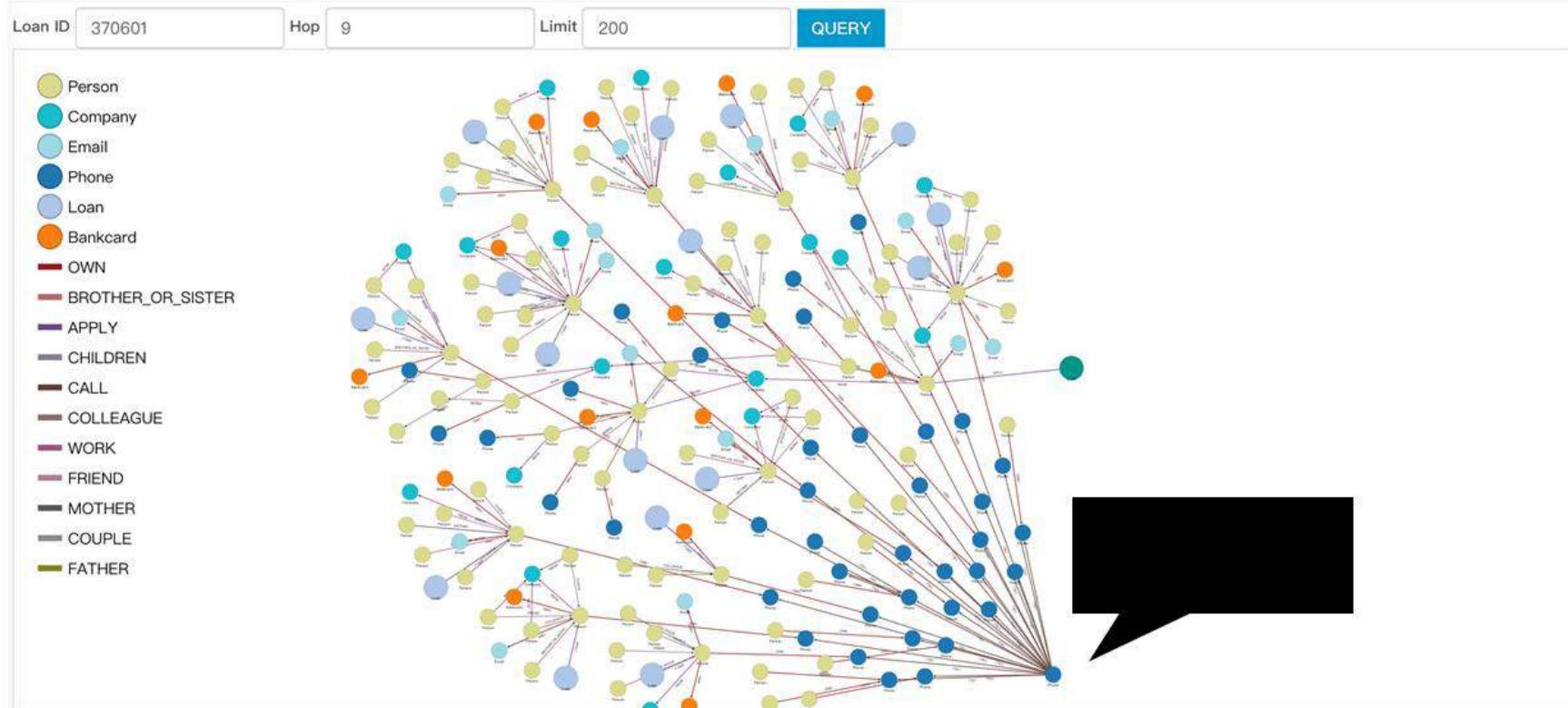


Loan Business – Knowledge graph

Loan ID 280229 Hop 5 Limit 20 **QUERY**



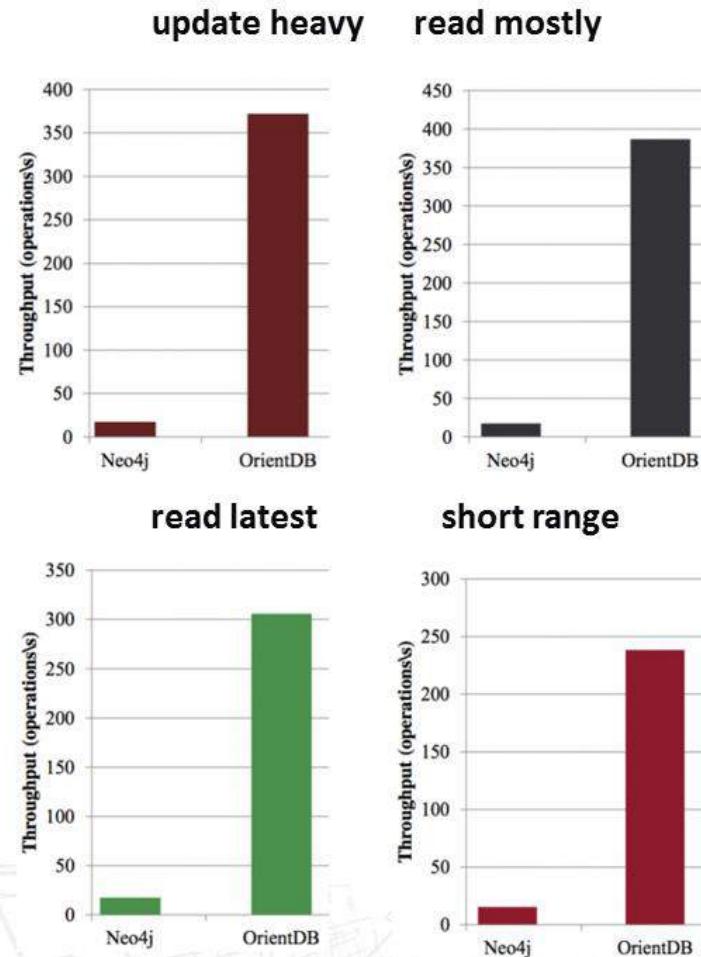
Loan Business – Knowledge graph



Loan Business – Knowledge graph

Features & Capabilities	OrientDB Community	Neo4j Community
Graph Database	✓	✓
ACID	✓	✓
TinkerPop Compliance	✓	✓
Java Hooks	✓	✓
Sharding	✓	
Multi-Master replication	✓	
SQL*	✓	
...

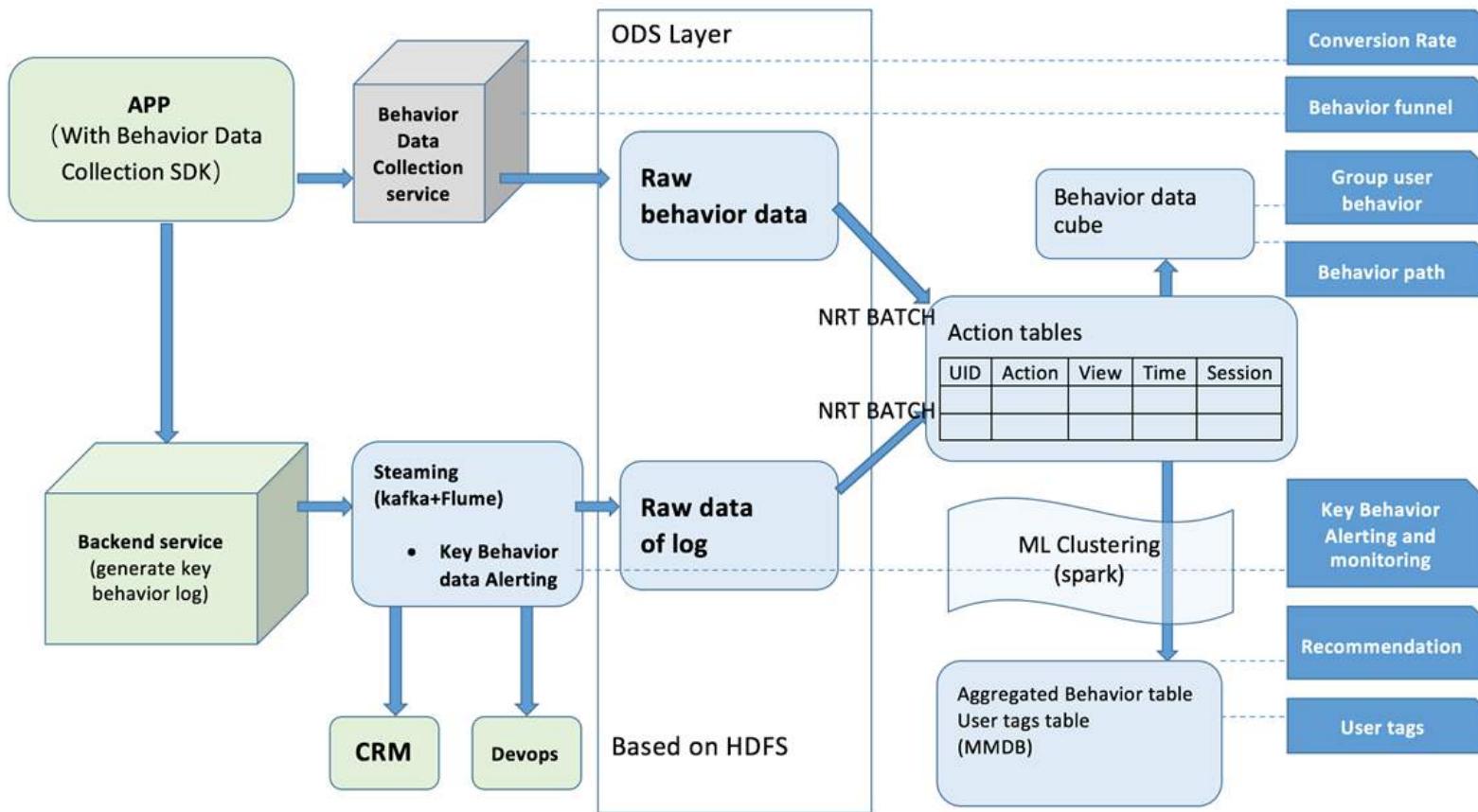
Other options: Titan, GraphDB



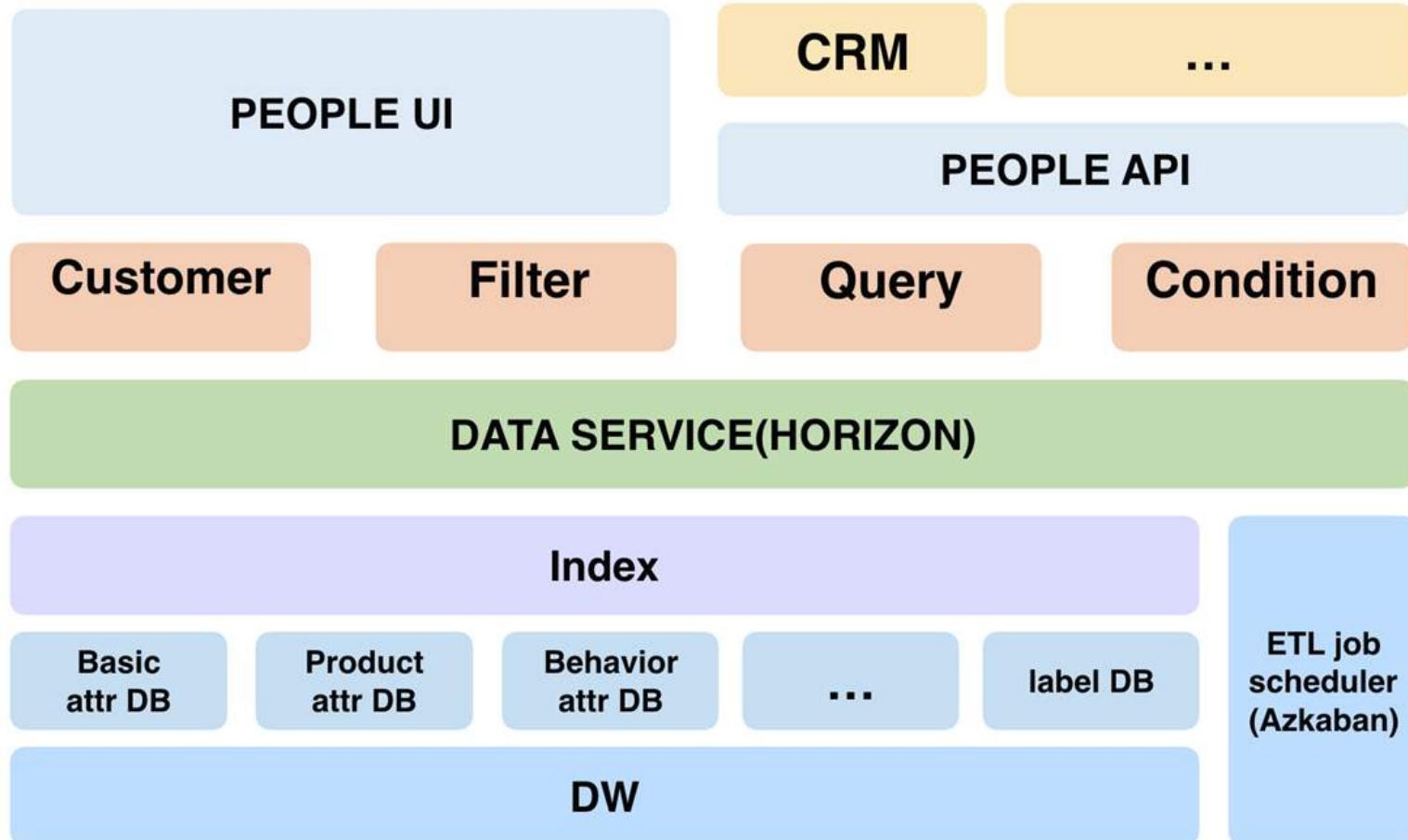
Contents

- Fintech Overview
- Loan business
- **Lender management**
- Intelligent business management
- Big data solution architecture overview

Customer Management – Behavior Data



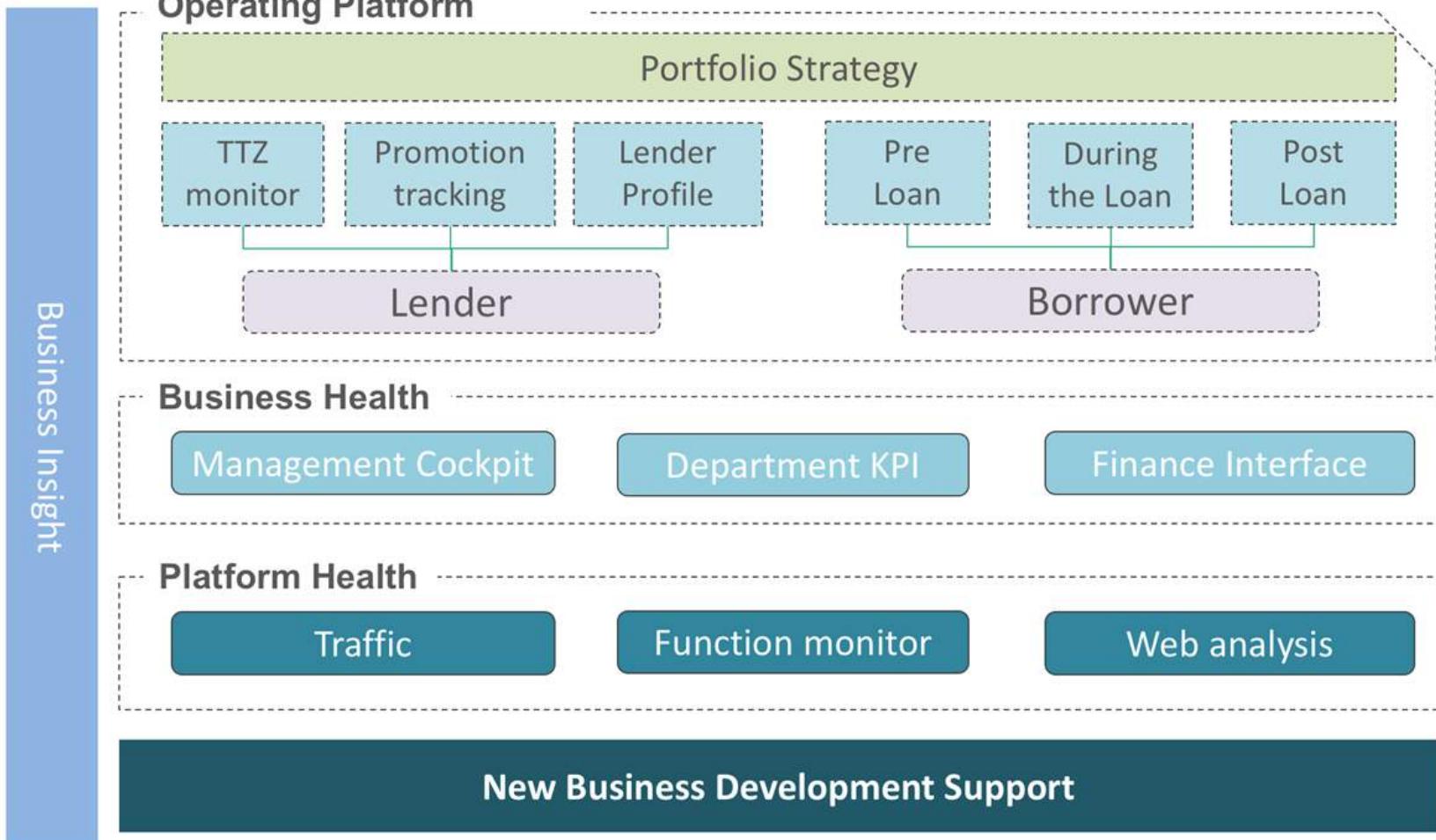
Customer Management – Profile Data



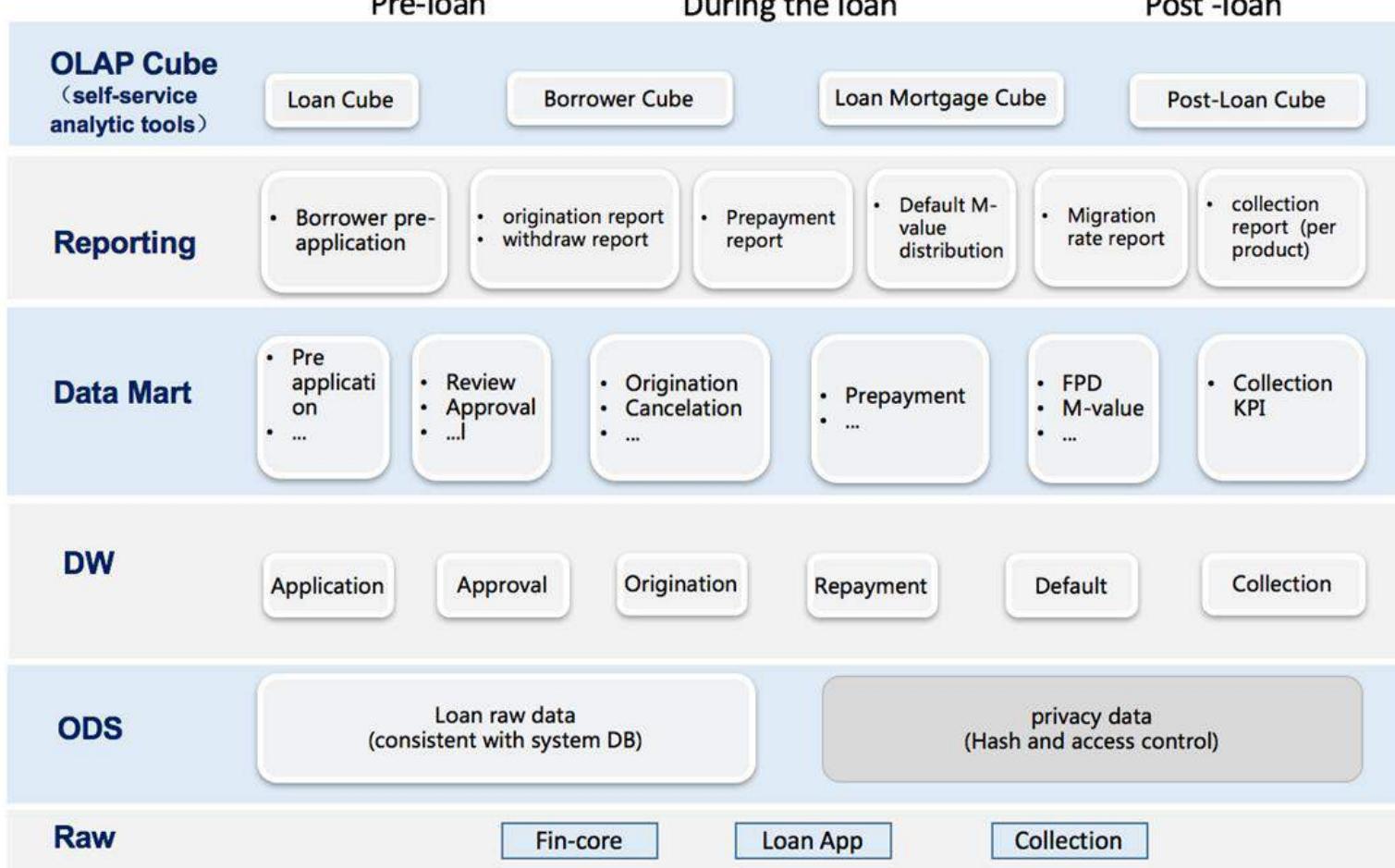
Contents

- Fintech Overview
- Loan business
- Lender management
- **Intelligent business management**
- Big data solution architecture overview

Intelligent business management



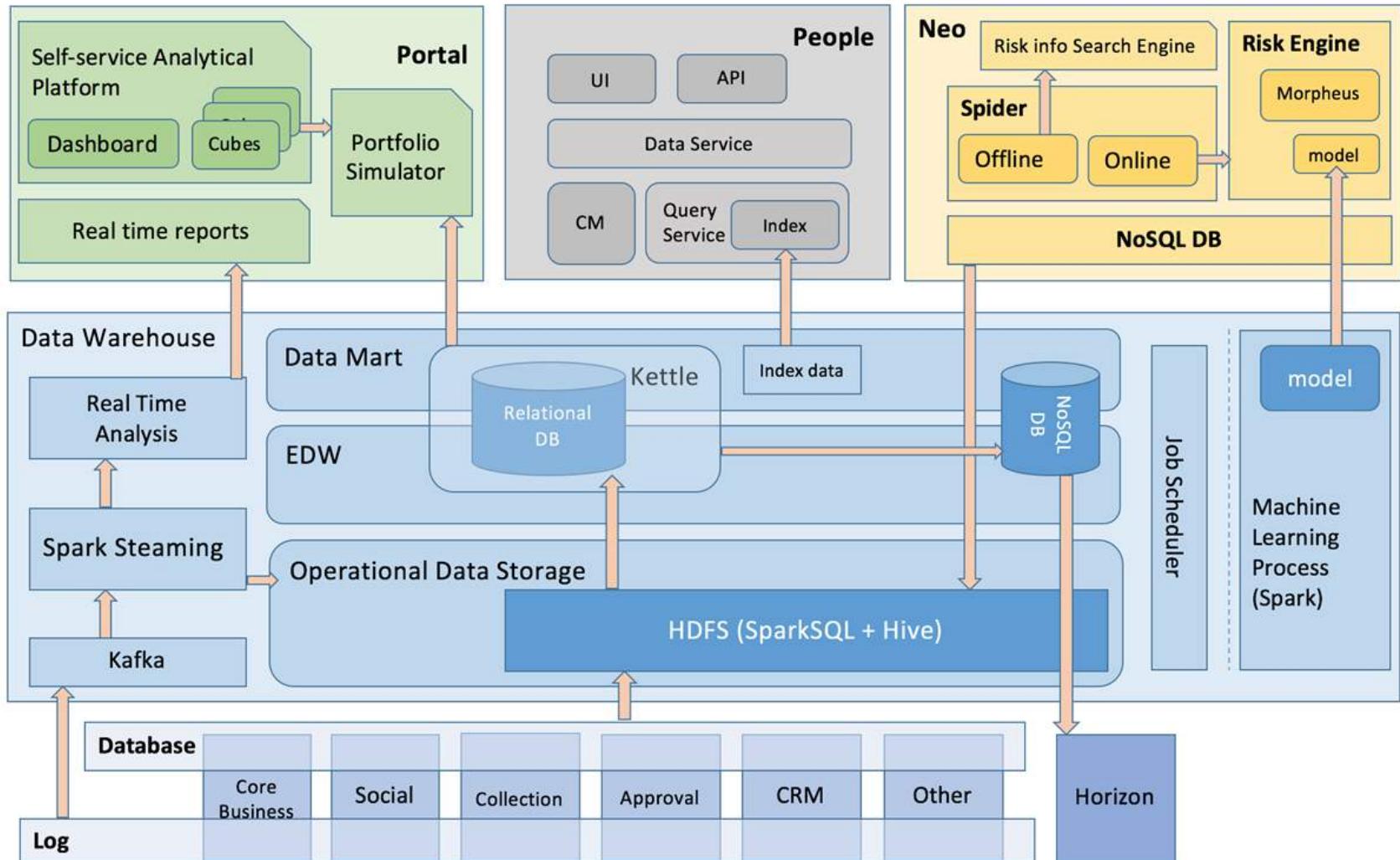
Intelligent business management



Contents

- Fintech Overview
- Loan business
- Lender management
- Intelligent business management
- **Big data solution architecture overview**

Architecture Overview



Thanks!

2017 Software Architecture Summit

微服务实践与反思

黄亮

ThoughtWorks



| 演讲大纲

- 实践微服务，你够个儿吗？
- 微服务实践全景图
- 微服务之演化式架构师
- 微服务之建模
- 微服务之集成
- 重构到微服务
- 微服务技术框架

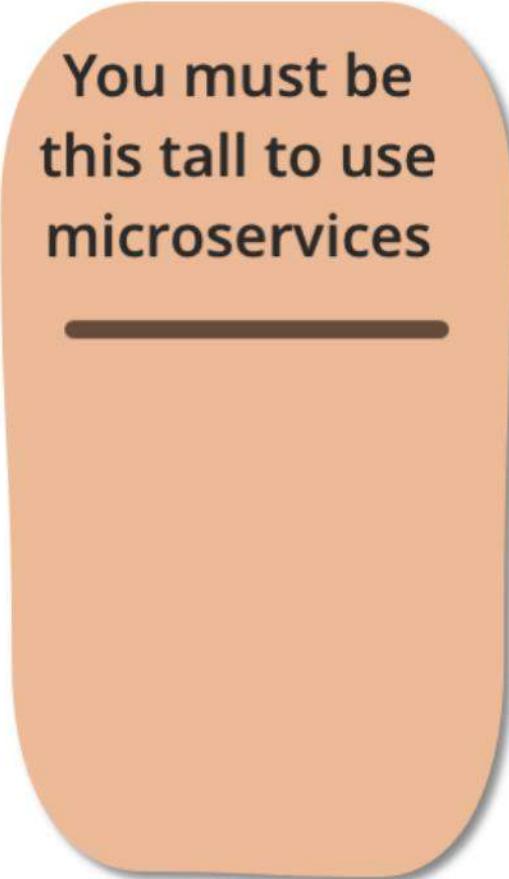
2017 Software Architecture Summit

持续集成证书测试



| 实践微服务，你够个儿吗？

You must be
this tall to use
microservices



微服务架构

微服务架构是一种**架构模式**，它提倡将单一应用程序划分成**一组小的服务**，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其**独立的进程中**，服务与服务间采用**轻量级的通信机制**互相沟通（通常是基于HTTP协议的RESTful API）。每个服务都**围绕着具体业务**进行构建，并且能够被**独立的部署**到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。



领域驱动
设计

持续交付

按需
虚拟化

基础设施
自动化

小型自治
团队

大规模
集群系统

| 微服务与SOA

Based on
SOA principles

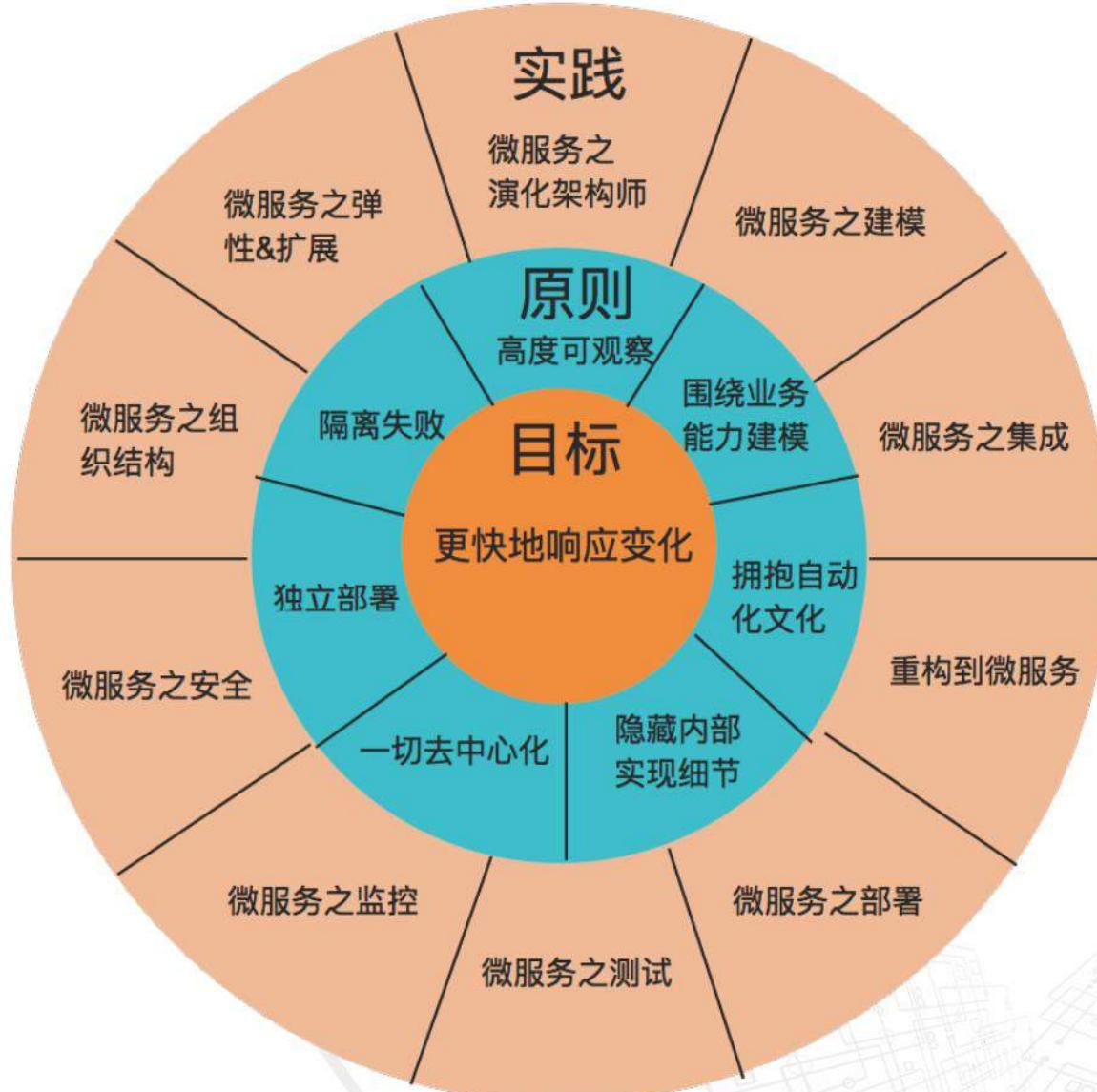
- Separation of concerns
- Encapsulation
- Loose coupling

Added
microservices
constraints

- Independent
- Single responsibility
- Owns its data



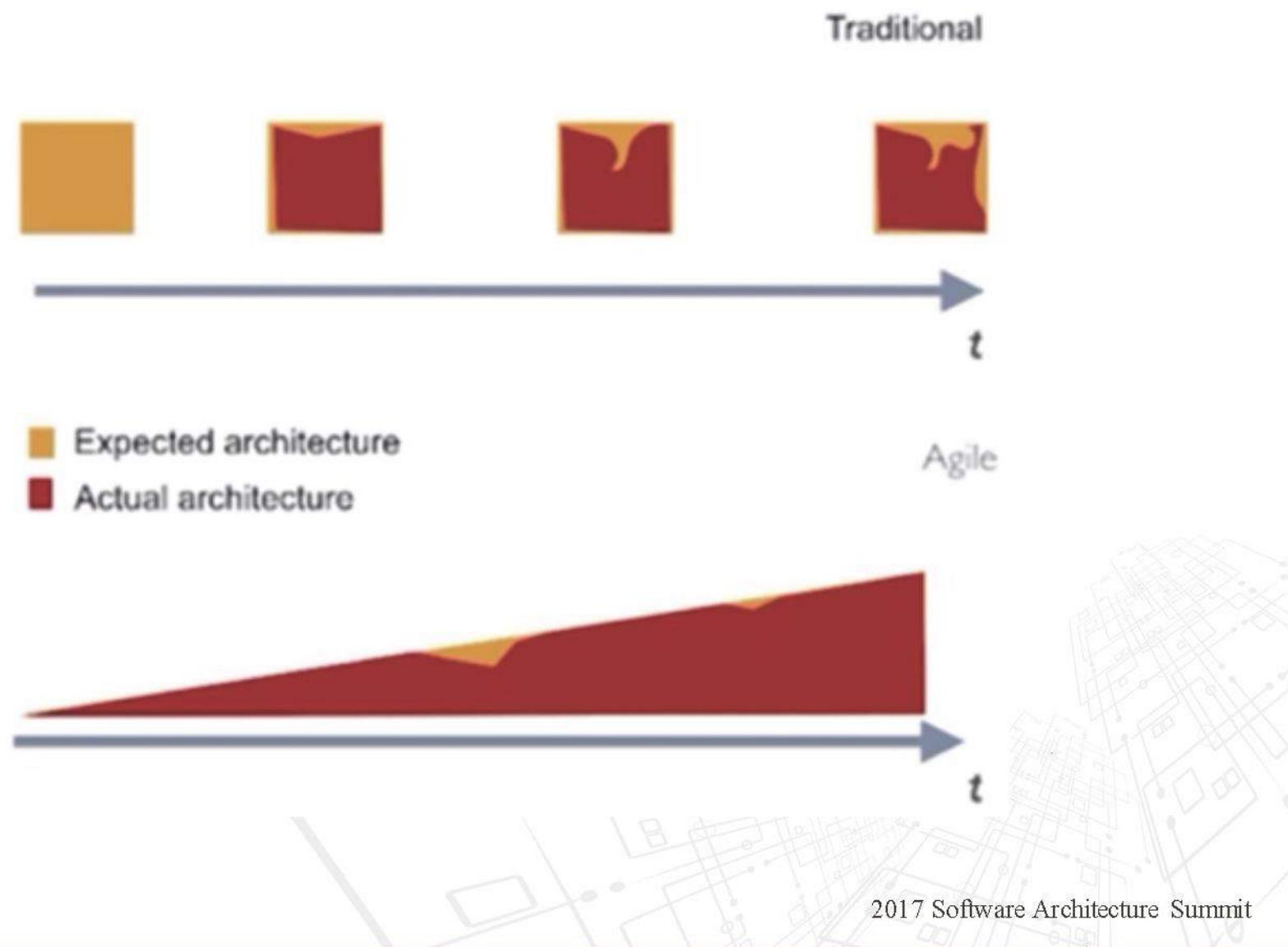
微服务全景图



| 微服务之演化式架构师

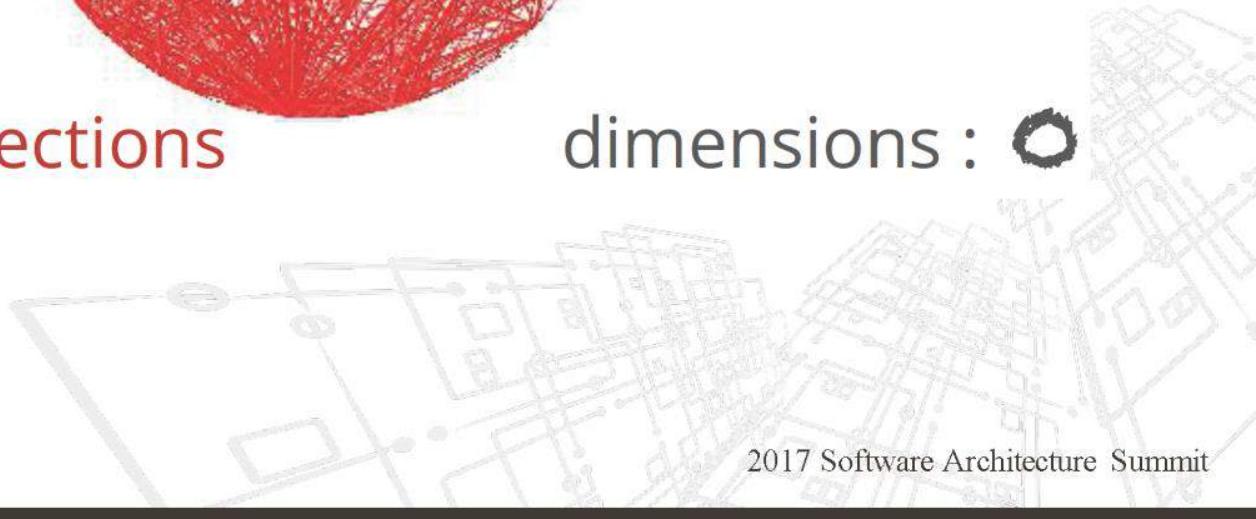
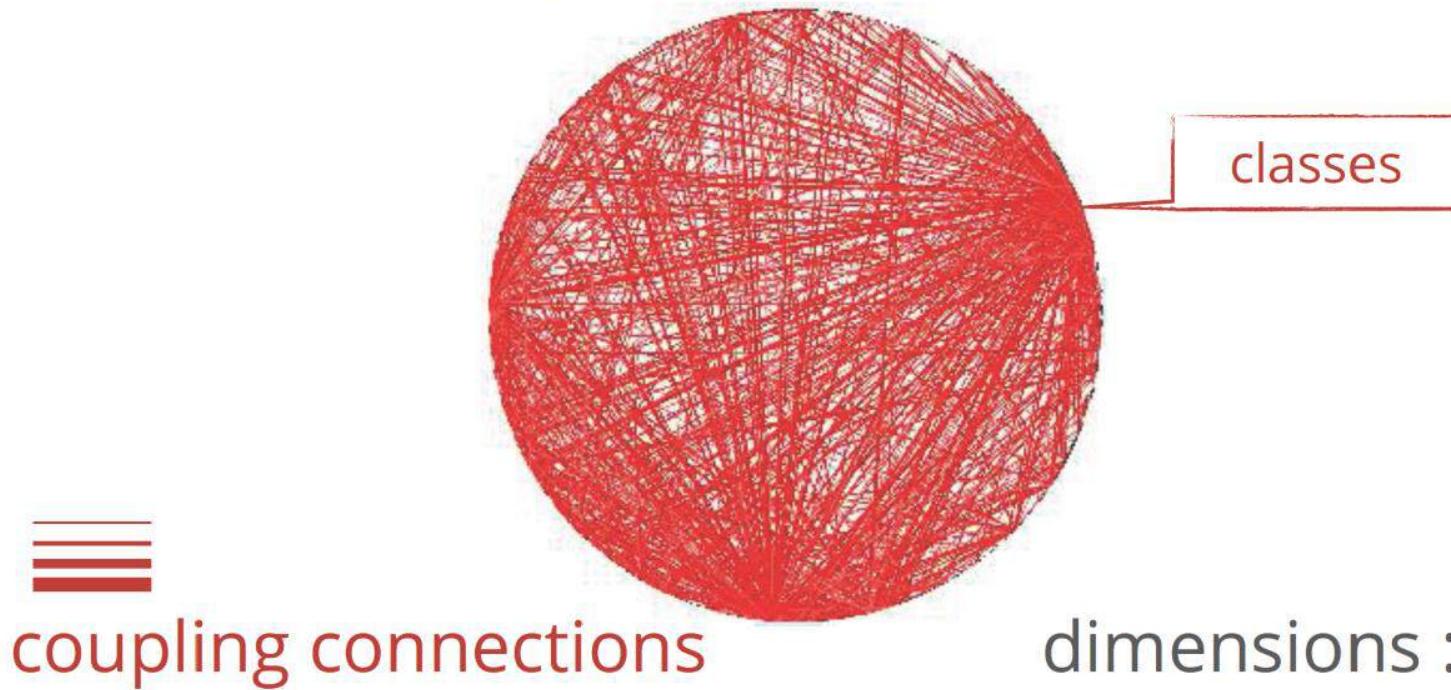


| 微服务之演化式架构师

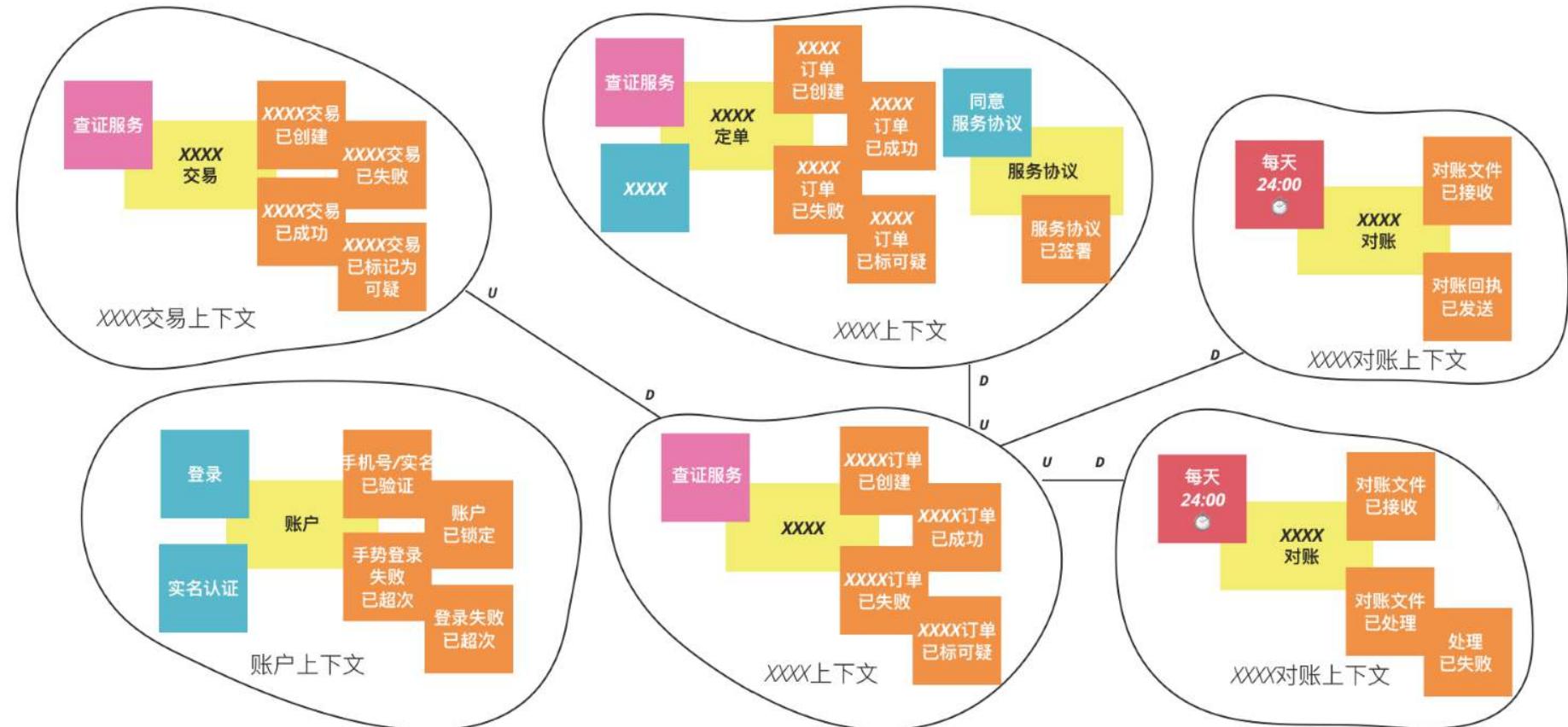


| 微服务之建模

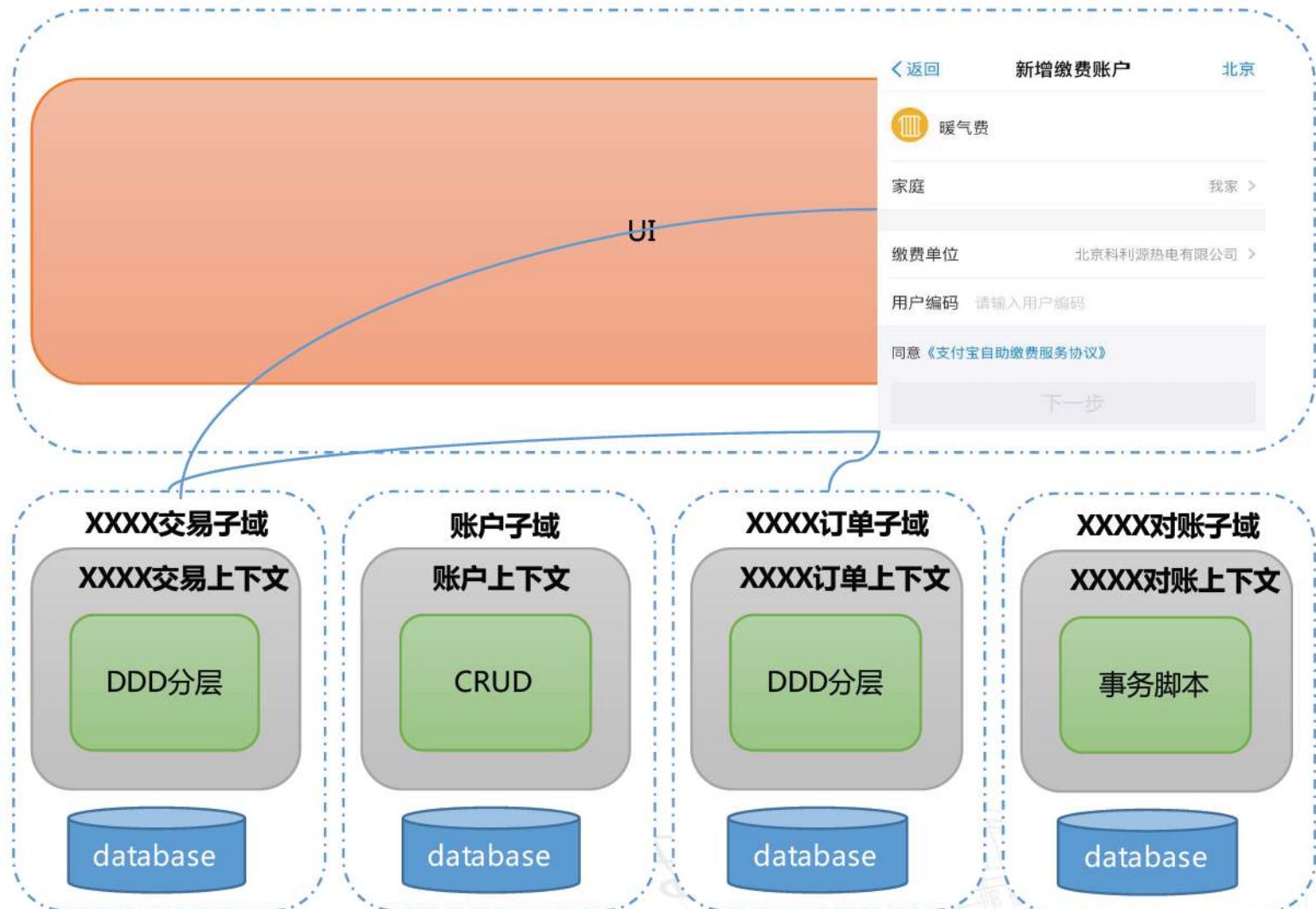
Big Ball of Mud



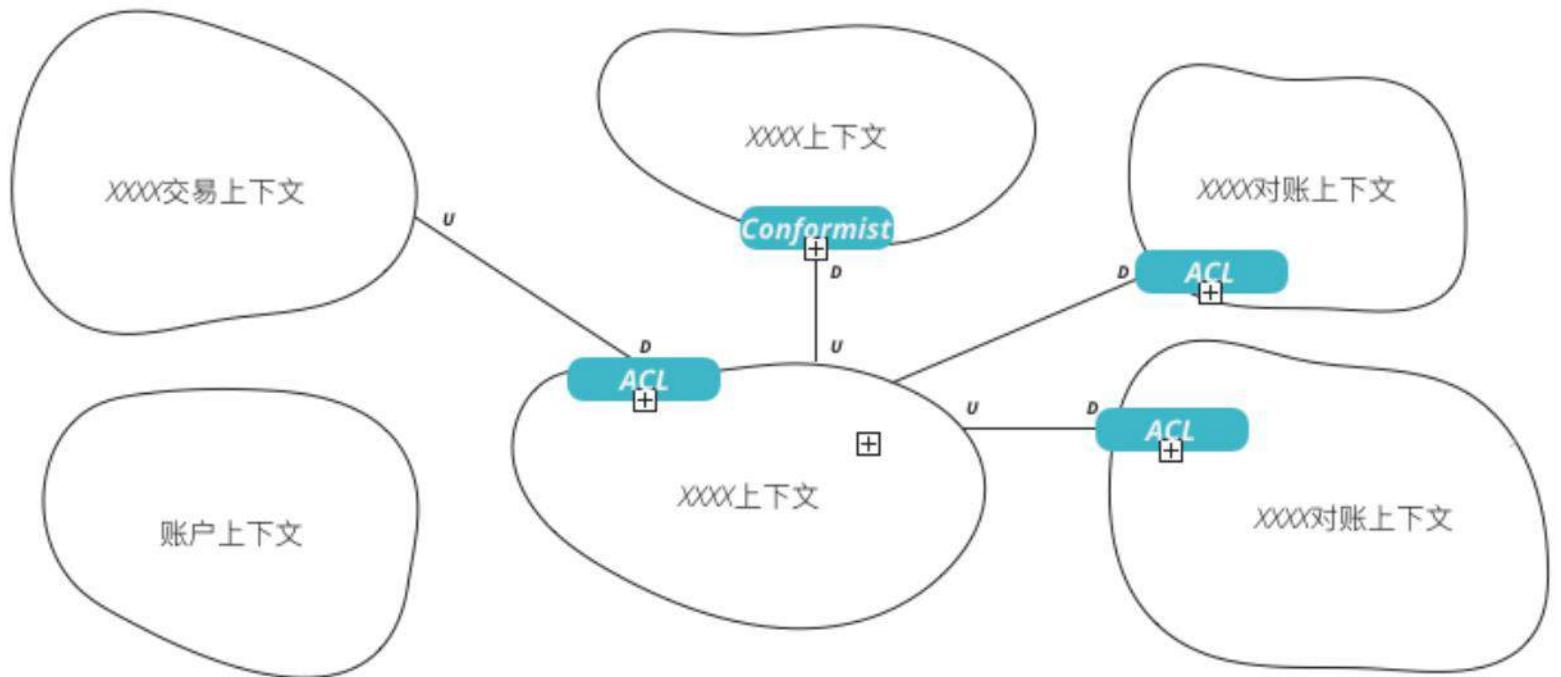
微服务之建模实践



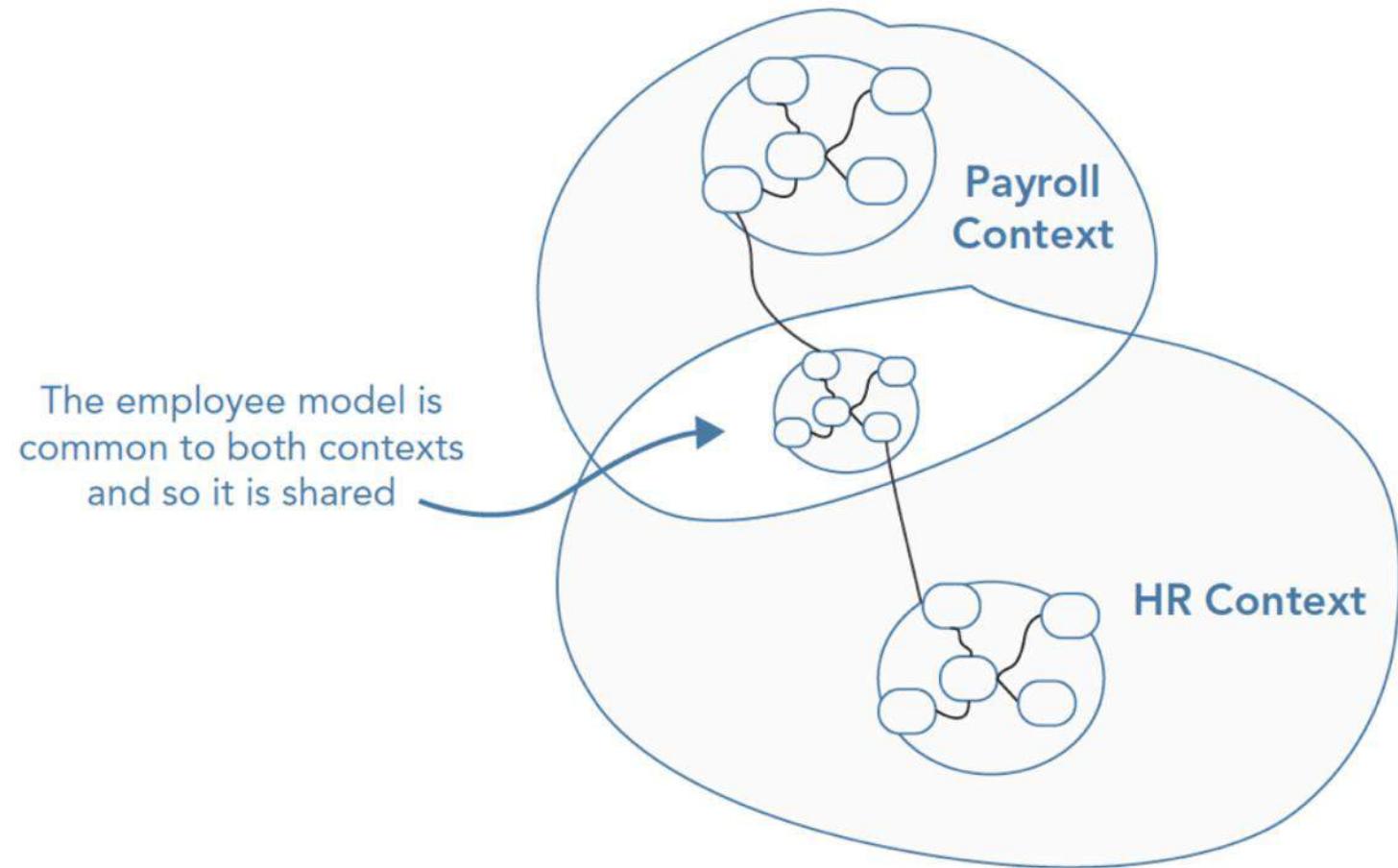
• 微服务之建模实践



• 微服务之建模实践

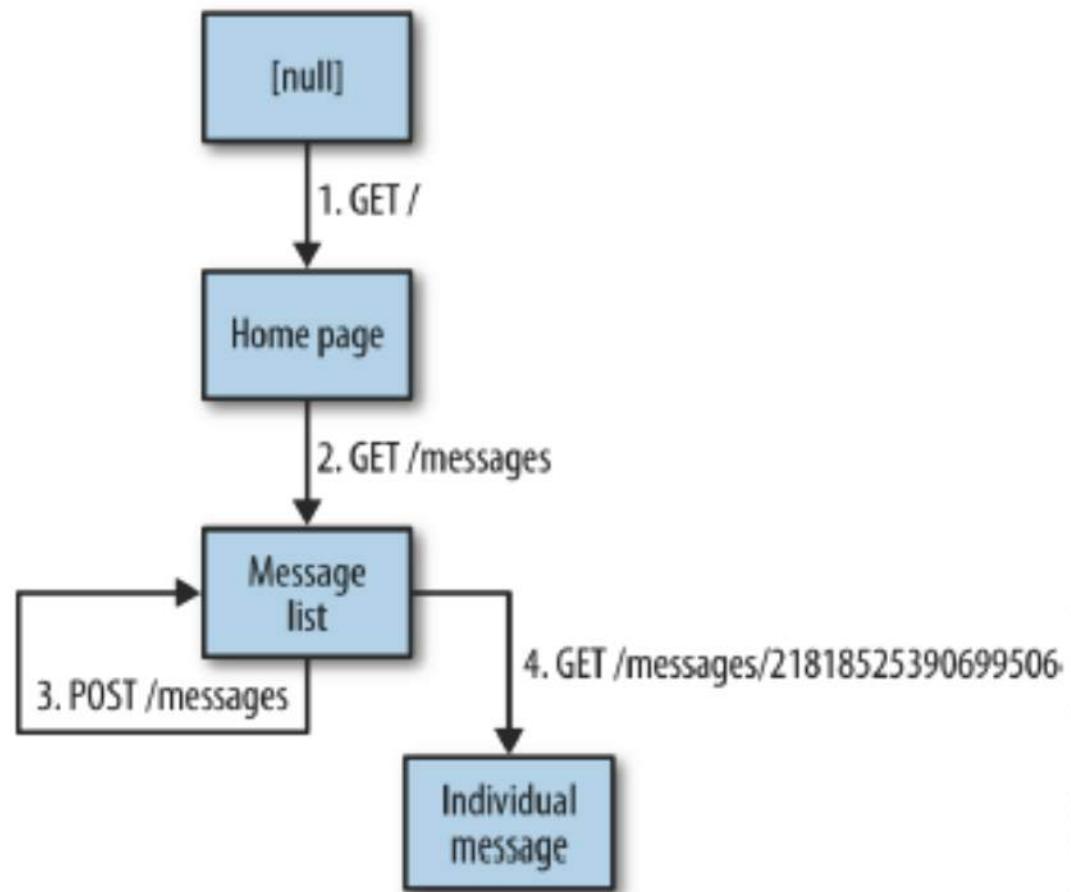


- 微服务之建模反思-加强隔离性，避免紧耦合



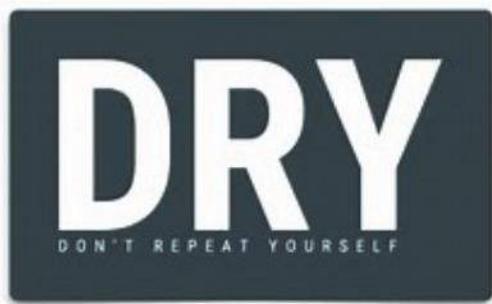
• 微服务之集成实践-RESTful WEB API

- 罗列语义描述符
- 画状态图
- 调整命名
- 选择一种媒体类型
- 编写Profile
- 实现
- 发布



- 微服务之集成-REST API实践反思

- 微服务间维护冗余的参数实体，是否违背了DRY
- 跨语言的RPC会不会是一个好的选择

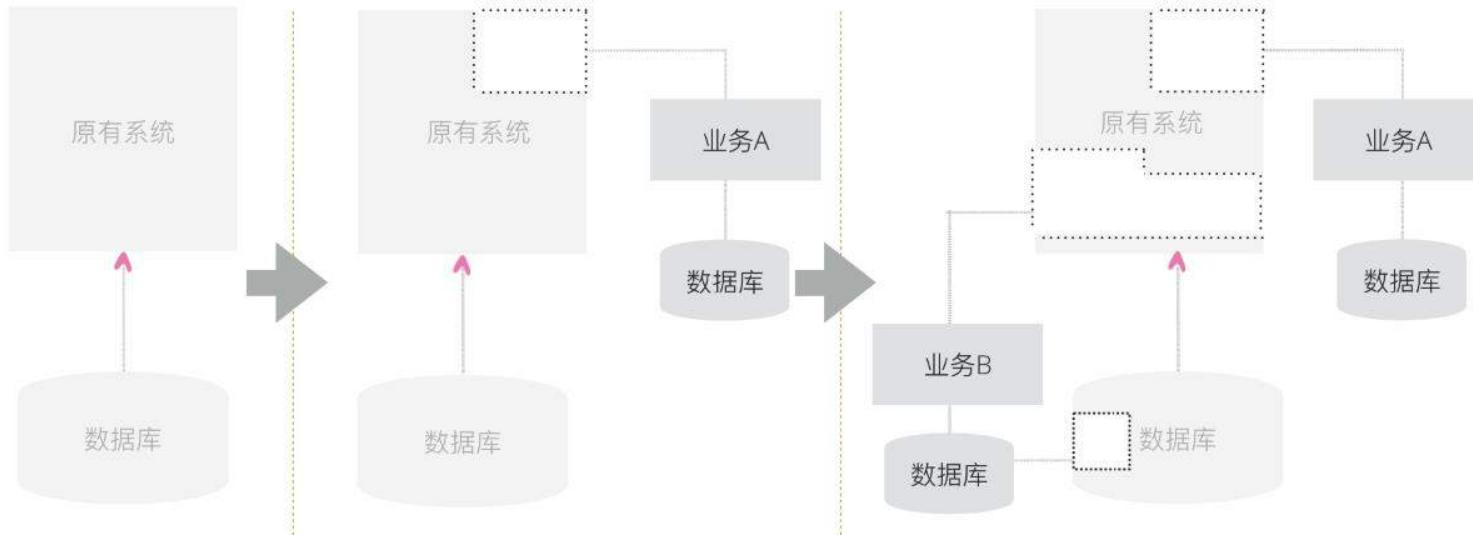


RESTful
API

RPC

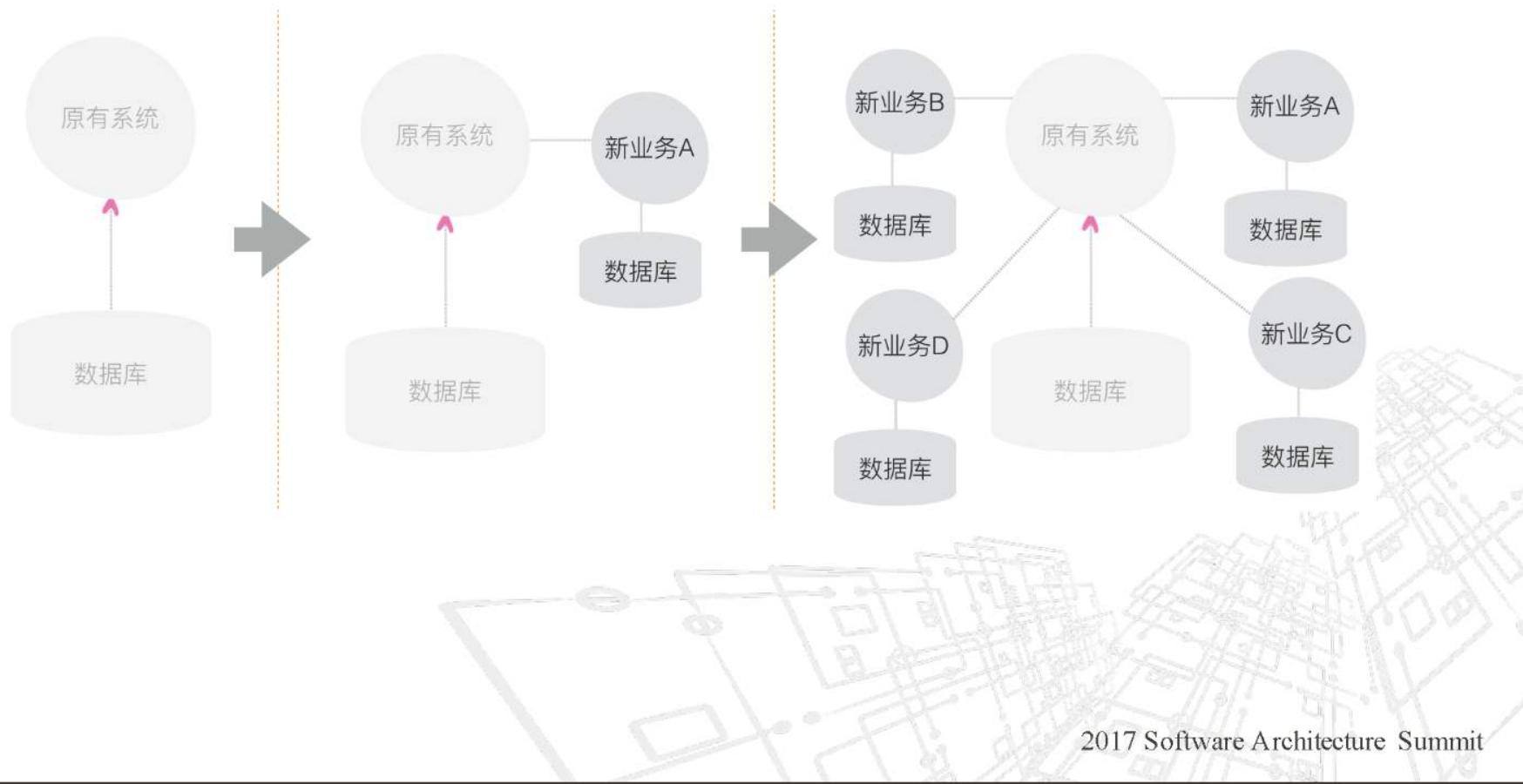
| 重构到微服务-修缮模式

- 通过剥离新业务和功能，逐步“释放”现有系统的耦合度
- 适合需求变更频度不高的存量系统



| 重构到微服务-绞杀模式

- 在既有的系统资产的基础上实现IT创新
- 通过在新的应用上实现新特性，保持和现有系统的松耦合，仅在必要时将功能从原系统中剥离，以此逐步的替换原有系统



| 遗留系统重构的项目实践

2 Microservices



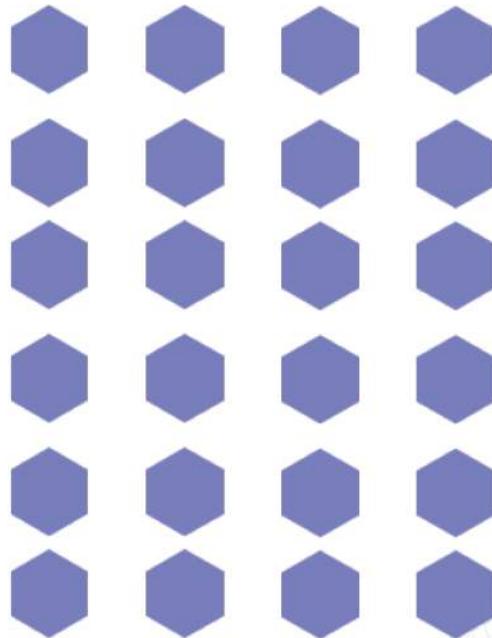
3个月

10 Microservices



12个月

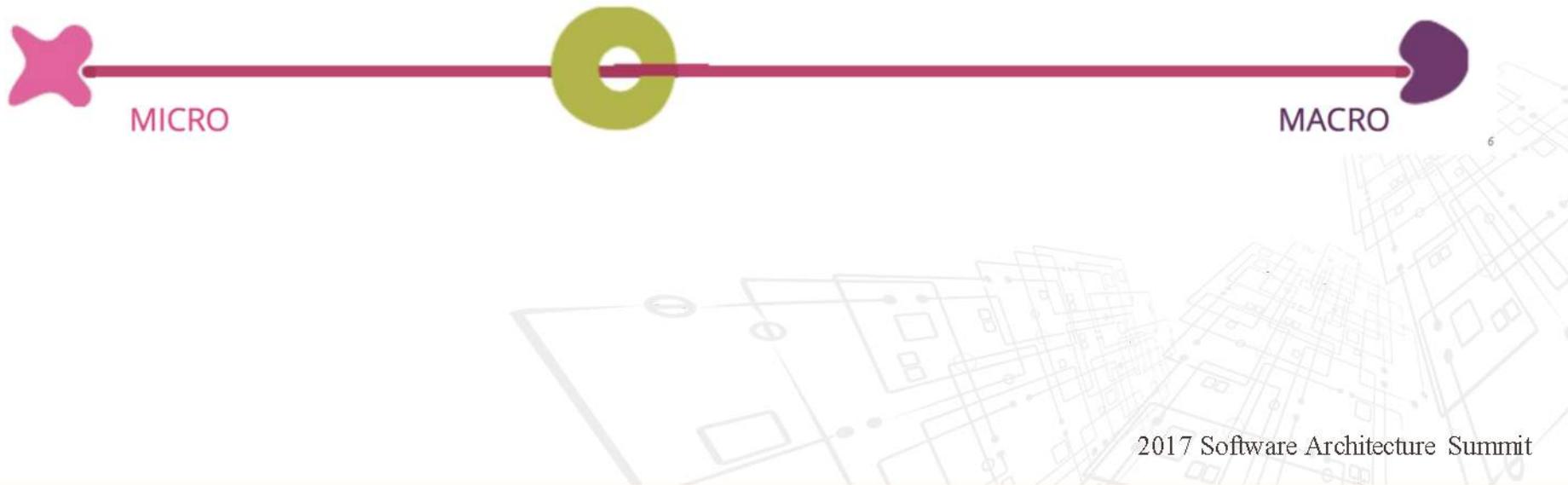
60 Microservices



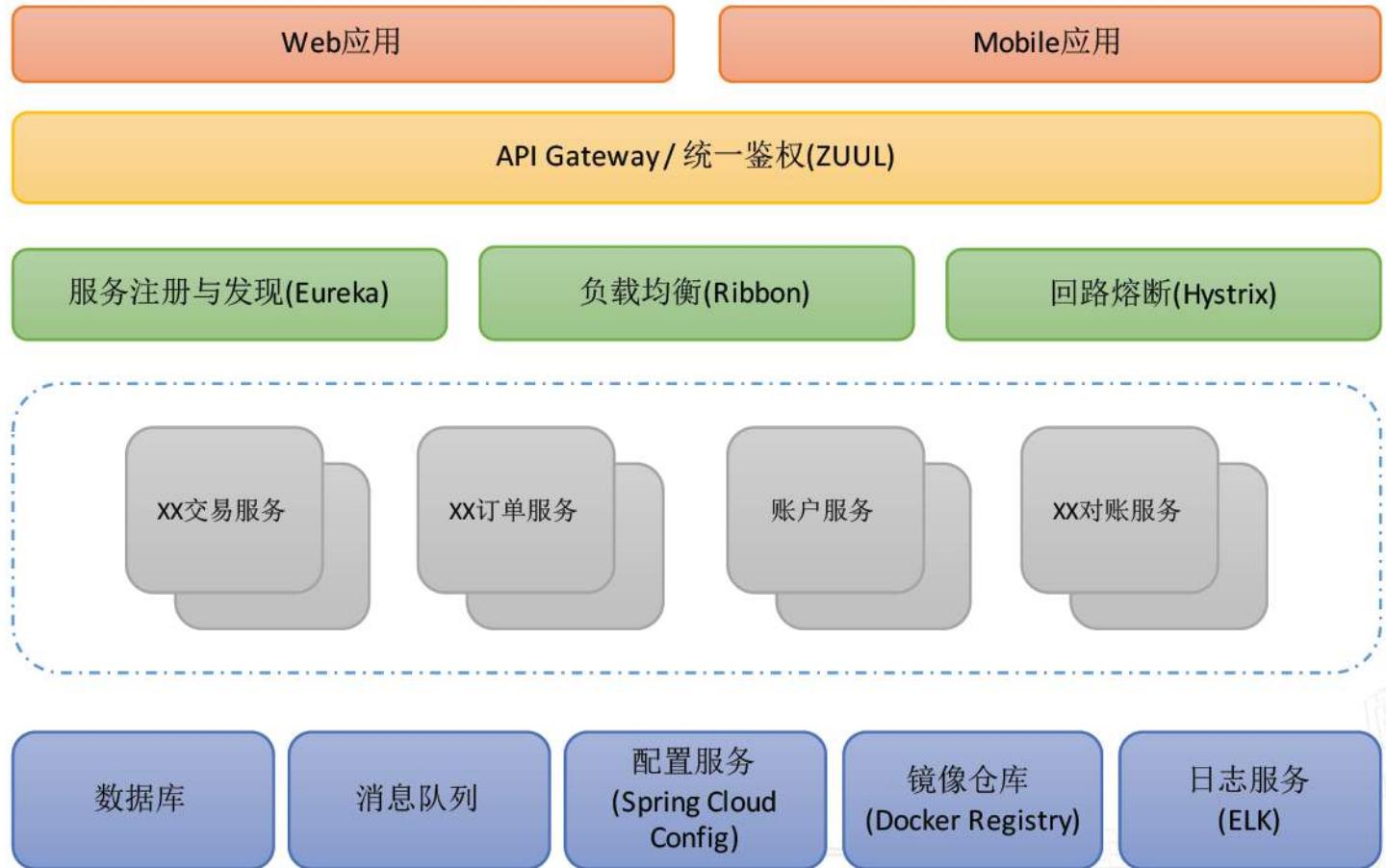
18个月

微服务实践反思

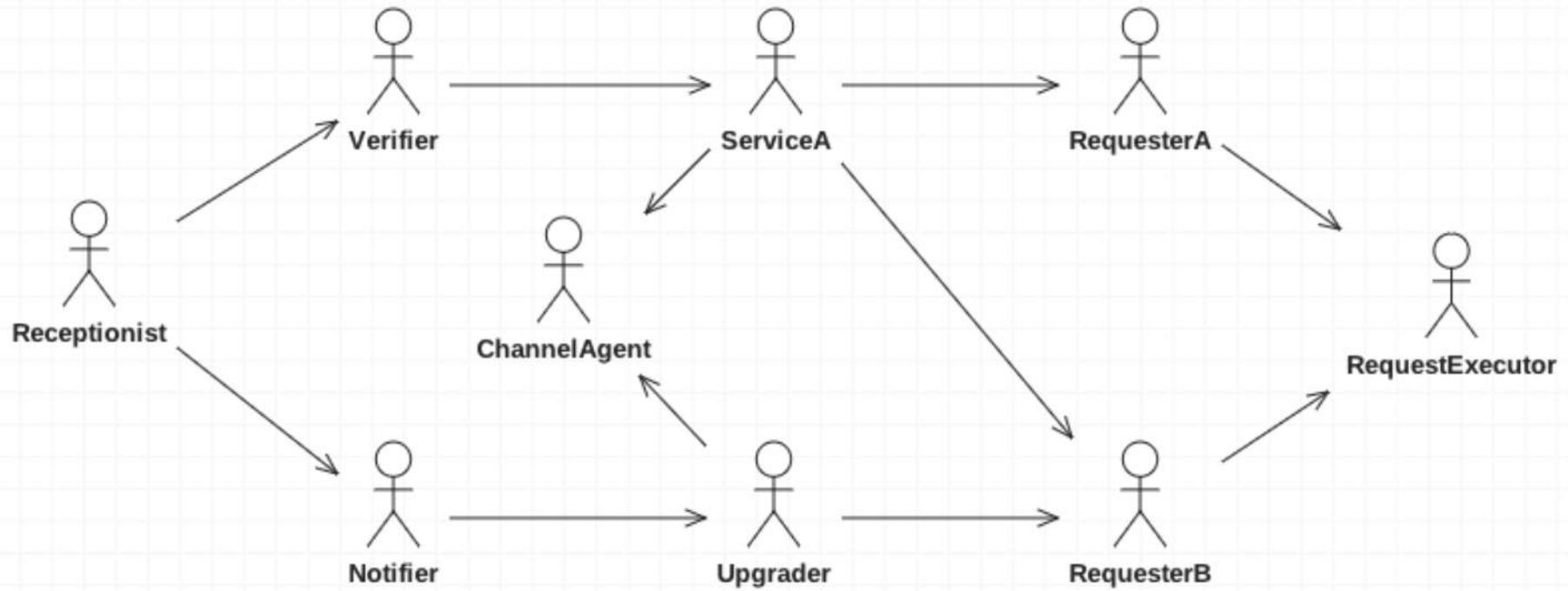
- ▶ 敏捷交付
 - ▶ 自治
 - ▶ 可组合性
 - ▶ 技术异构
 - ▶ 弹性扩展
- ▶ 维护成本
 - ▶ 沟通成本
 - ▶ 治理成本
 - ▶ 运维成本
 - ▶ 容错和故障管理成本



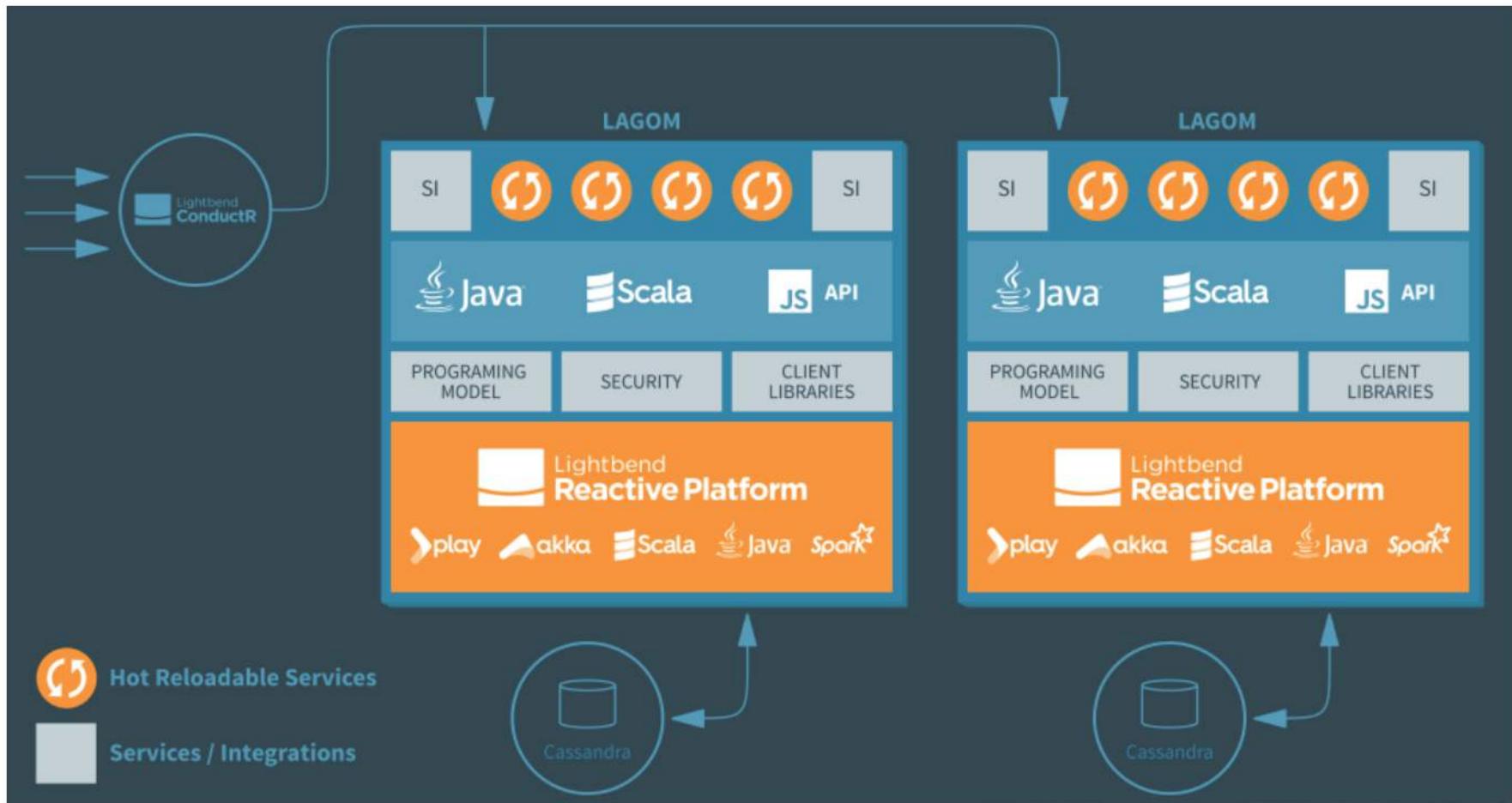
| 基于SPRING CLOUD的微服务项目实践



| 响应式微服务实践- AKKA



响应式微服务架构- LAGOM



2017 Software Architecture Summit

Q&A

黄亮



When TiDB Meets Spark

=> TiSpark

Who am I

- 马晓宇@PingCAP
- Architect@TiDB team
- Working on OLAP related products and features
- Previously lead of Big Data infra team@Netease
- Mainly working on SQL on Hadoop and BigData related stuff

Agenda

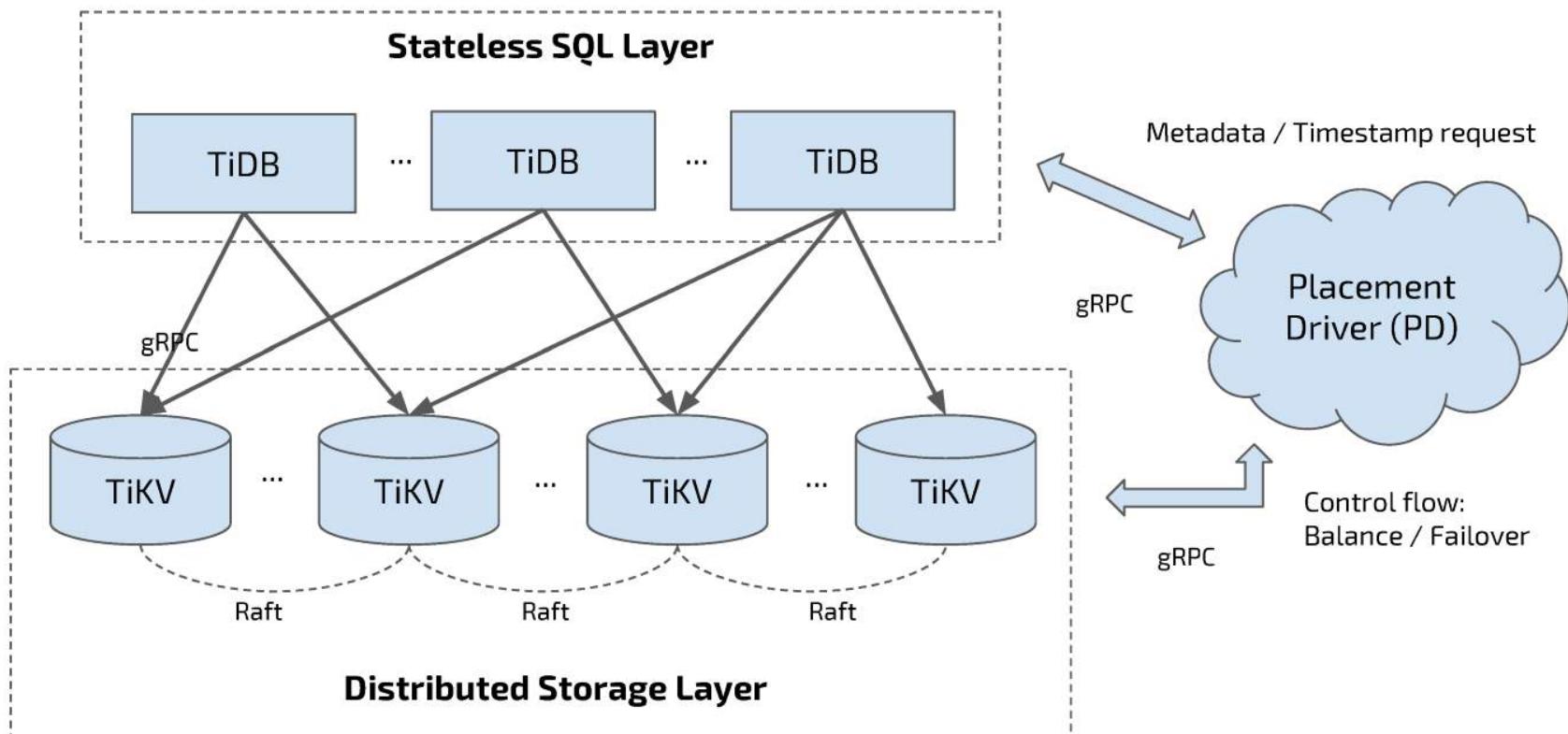
- A little bit about TiDB / TiKV
- What is TiSpark
- Architecture
- Features beyond raw Spark
- Use case
- Current Status

What's TiDB

- Horizontal Scalability
- ACID Transaction
- High Availability
- Auto-Failover
- SQL at scale
- TiKV is its storage engine

What's Really New with NewSQL?

A little bit about TiDB and TiKV



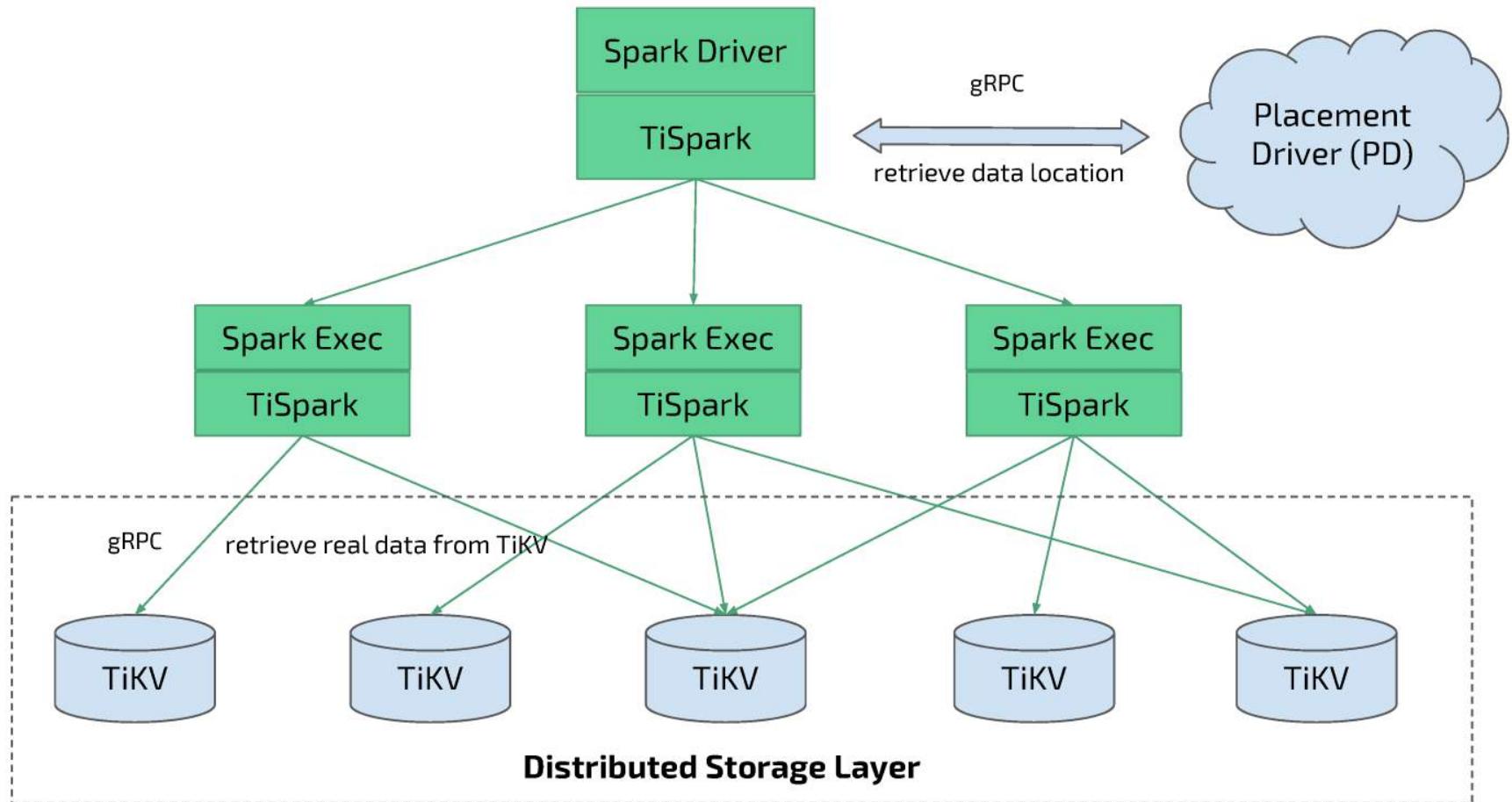
What is TiSpark

- TiSpark = Spark SQL on TiKV
 - Spark SQL directly on top of a distributed Database Storage
- Hybrid Transactional/Analytical Processing(HTAP) rocks
 - Provide strong OLAP capacity together with TiDB

What is TiSpark

- Complex Calculation Pushdown
- Key Range pruning
- Index support
- Cost Based Optimization
 - Pick up right Access Path (WIP)
- Batch write (Q4 2017)
- Index Lookup join and merge join (Q4 2017)

Architecture



Architecture

- On Spark Driver
 - Translate metadata from TiDB into Spark meta info
 - Hijack Spark SQL logical plan, pick up elements to be leverage by storage (TiKV) and rewrite plan
 - Do extra optimization based on extra information (basically stats, maybe order)
 - Locate Data based on Region info from Placement Driver and split partitions;
- On Spark Executor
 - Encode Spark SQL plan into TiKV's coprocessor request
 - Decode TiKV / Coprocessor result and transform result into Spark SQL Rows

How everything made possible

```
public class ExperimentalMethods  
extends Object
```

Holder for experimental methods for the bravest. We make NO guarantee about the stability regarding binary compatibility and source compatibility of methods here.

```
spark.experimental.extraStrategies += ...
```

Since:

1.3.0

Method Summary

Methods

Modifier and Type	Method and Description
scala.collection.Seq<org.apache.spark.sql.catalyst.rules.Rule<org.apache.spark.sql.catalyst.plans.logical.LogicalPlan>>	<code>extraOptimizations()</code>
scala.collection.Seq<org.apache.spark.sql.execution.SparkStrategy>	<code>extraStrategies()</code> Allows extra strategies to be injected into the query planner at runtime.

- Two extension points for Spark SQL Internal
- Extra Optimizer Rules allows us to do logical plan transform like Join Reorder
- Extra Strategies allow us to inject our own physical executor and that's what we leveraged for phase 1 in TiSpark
- Kept Spark Internal untouched to avoid compatibility issue

How everything made possible

- A fat project, tikv-client-lib-java, paying the price of bypassing TiDB
 - <https://github.com/pingcap/tikv-client-lib-java>
 - Parsing Schema, Type system, encoding / decoding, coprocessor
 - A full featured TiKV client (well, without write-support for now)
 - Predicates / Index - Key Range related logic
 - Aggregates pushdown related
 - Limit, Order, Stats related (WIP)
- A very thin layer inside Spark SQL
 - <https://github.com/pingcap/tispark>
 - TiStrategy for plan hijacking
 - And other utilities for mapping things from Spark SQL to TiKV client library
 - Thin enough for not bothering much of compatibility with Spark SQL

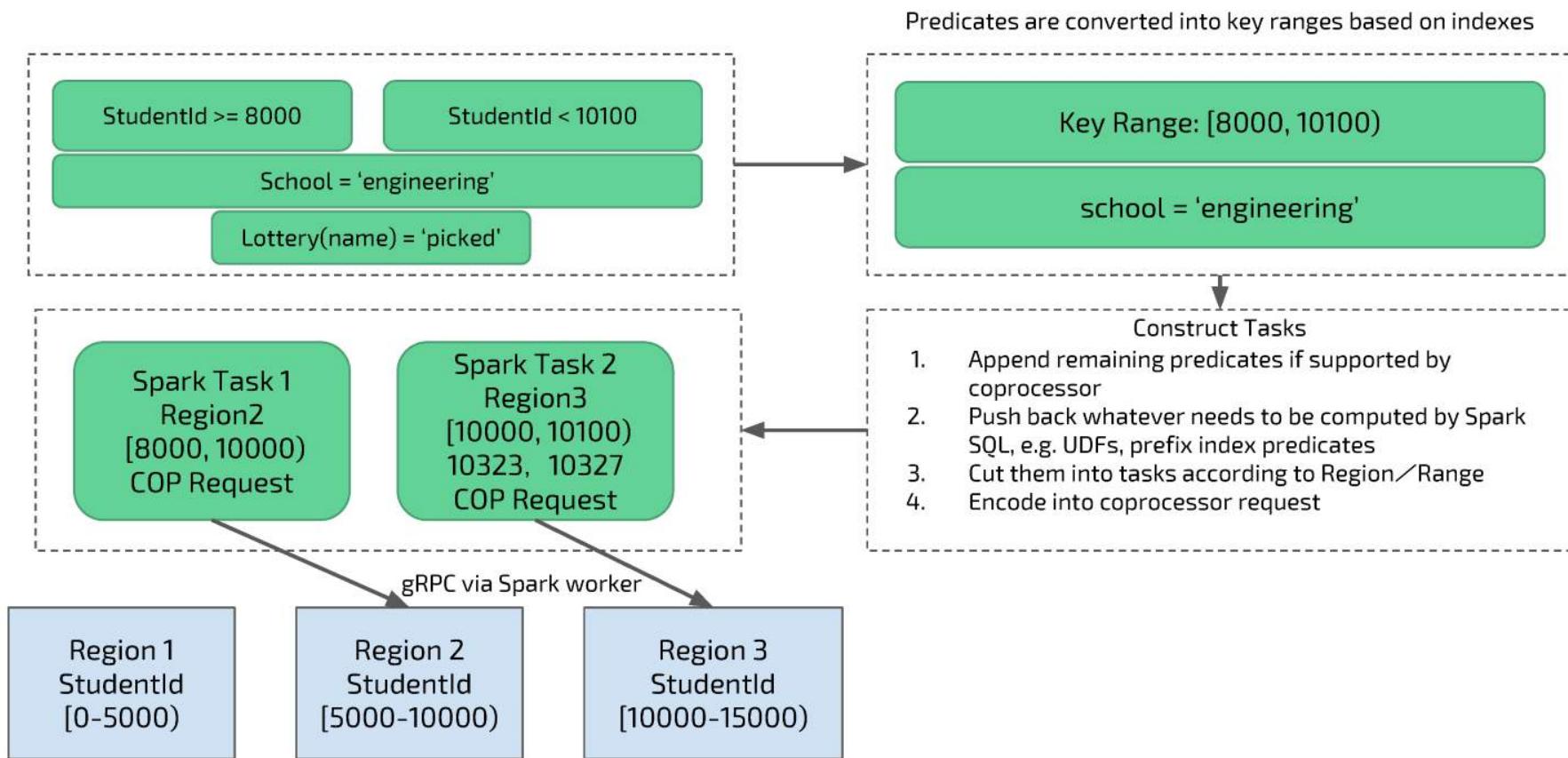
Too Abstract? Let's get concrete

```
select class, avg(score) from student  
WHERE school = 'engineering' and lottery(name) = 'picked'  
and studentId >= 8000 and studentId < 10100  
group by class ;
```

- Above is a table on TiDB named student
- Clustered index on StudentId and a secondary index on School column
- Lottery is an Spark SQL UDF which pick up a name and output 'picked' if RNG decided so

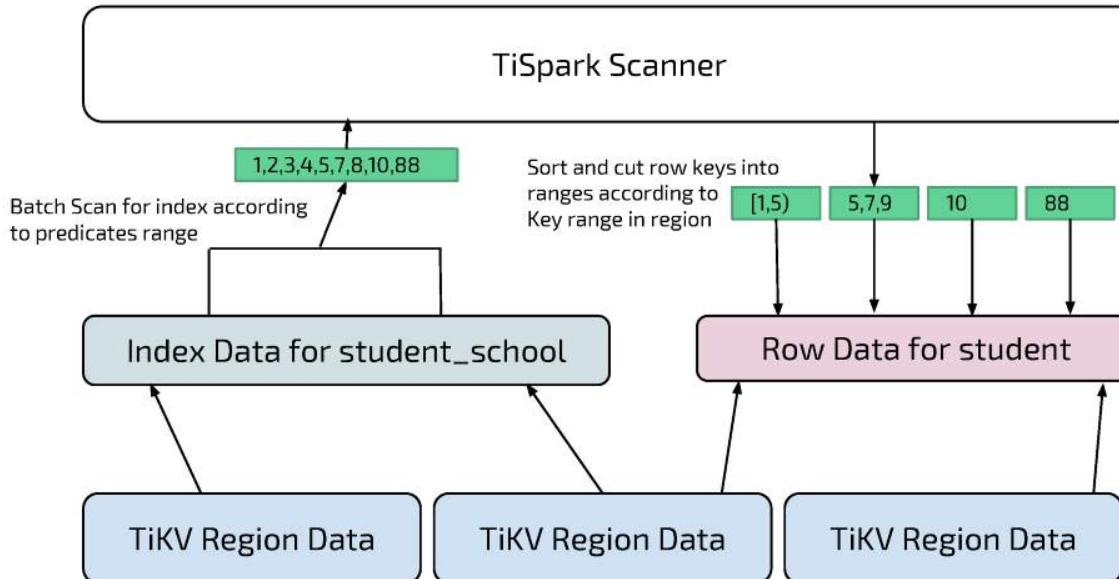
Predicates Processing

WHERE school = 'engineering' and lottery(name) = 'picked'
and **studentId >= 8000 and studentId < 10100**



Index Scan (WIP)

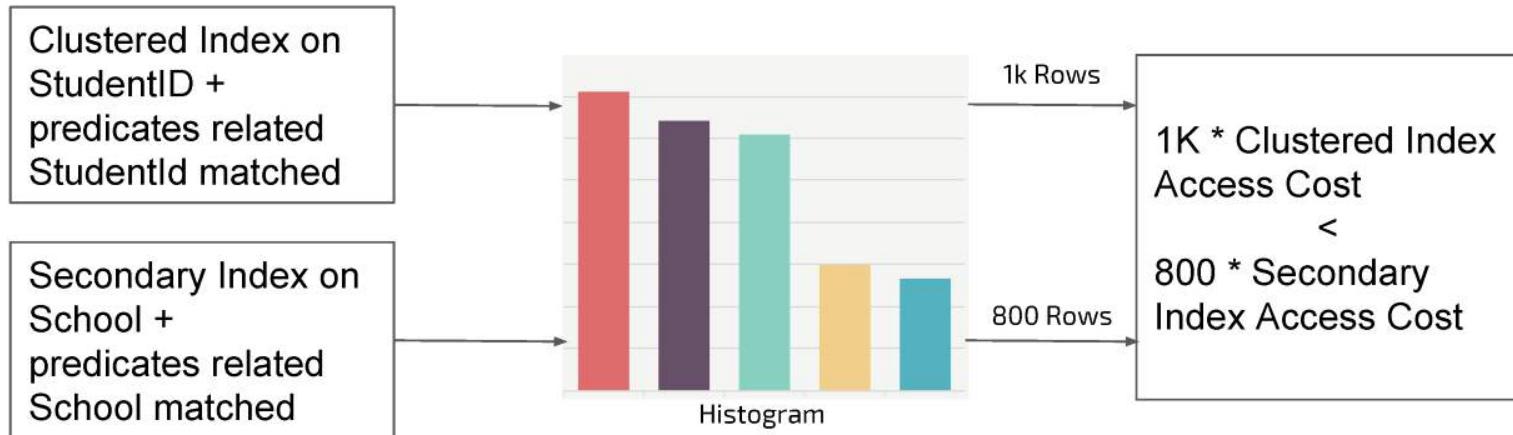
WHERE school = 'engineering' and lottery(name) = 'picked'
and (studentId >= 8000 and studentId < 10100)



- Secondary Index is encode as key-value pair
 - Key is comparable bytes format of all index keys in defined order
 - Value is the row ID pointing to table row data
- Reading data via Secondary Index usually requires a double read.
 - First, read secondary index in range just like reading primary keys in previous slide.
 - Sort all row IDs retrieved and combine them into ranges if possible
 - Encoding row IDs into row keys for the table
 - Send those mini requests in batch concurrently
- Possible to optimize away second read operation
 - If all required column covered by index itself already

Index Selection (WIP)

WHERE school = 'engineering' and lottery(name) = 'picked'
 and (studentId >= 8000 and studentId < 10100) or studentId in
 (10323, 10327)



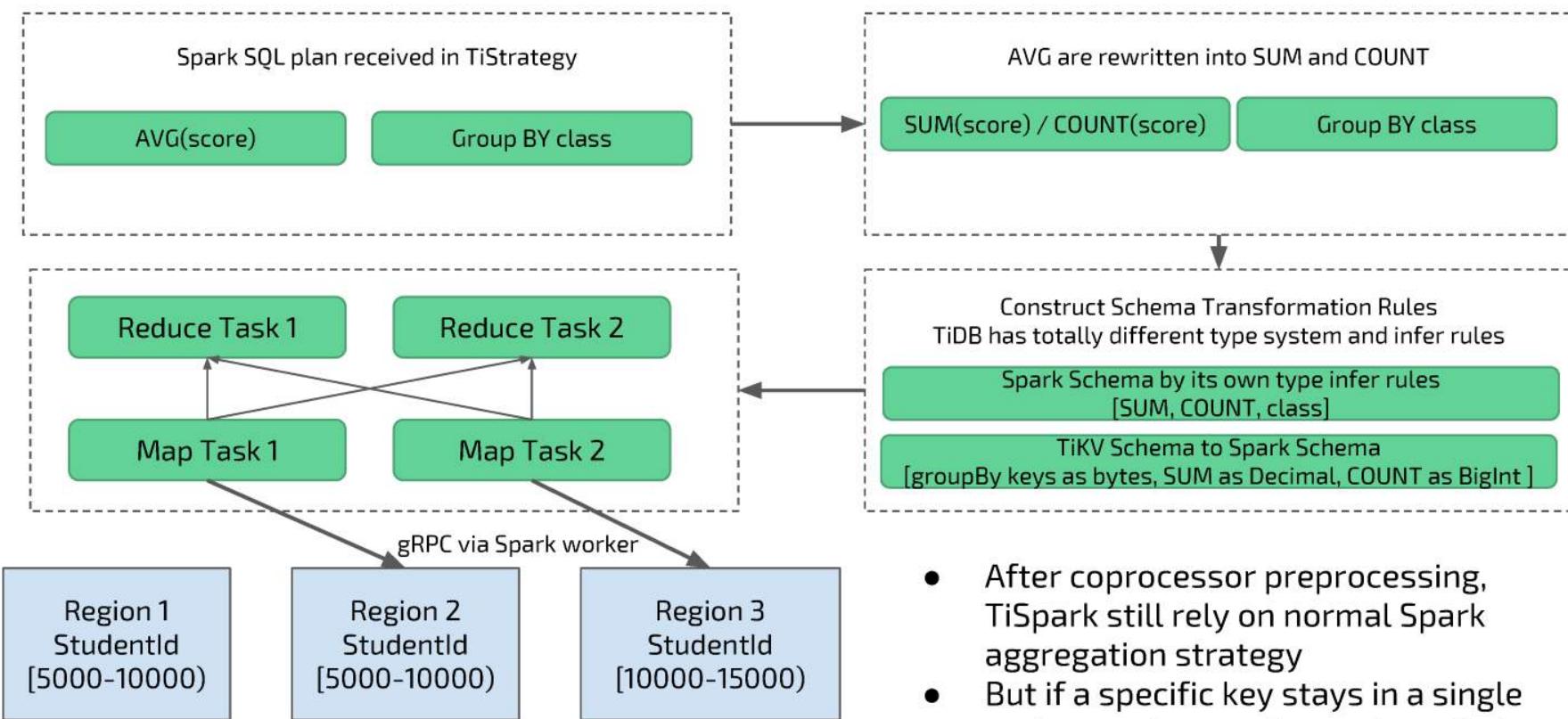
- If we the columns referred are all covered by index, then instead of retrieving actual rows, we apply index only query and cost function will be different
- Index Selection based on Histogram is working in progress. For now TiSpark using pseudo selection logic.

Aggregates Processing

`select class, avg(score) from student`

.....

`group by class ;`



- After coprocessor preprocessing, TiSpark still rely on normal Spark aggregation strategy
- But if a specific key stays in a single region, we have a chance to optimize aggregates away (planned)

TiSpark specific storage features

- Different Transaction Isolation Level
 - For OLTP: Snapshot Isolation
 - For OLAP: Read Committed to avoid lock
- Different resource schedule priority
 - Larger query tend to be assigned lower priority hence less resource when other queries running

Features Beyond Raw Spark SQL

- What we have more than traditional SQL-On-Hadoop systems:
 - Point query
 - Clustered Index and Secondary Index
 - Global Order
 - Update / Delete and ACID (via TiDB)

Features Beyond Raw Spark SQL

- What we have more than traditional SQL-On-Hadoop systems (WIP or planned):
 - Index and Point query -> Index look-up join
 - Global Order -> Parallel Merge Join (without shuffle)
 - Update and Delete -> real-time sync with other database
 - And a lot other techniques apply only to database (since it IS a database underneath)

Use Case

- Analytical / Transactional support all on one platform
 - No need for ETL
 - Real-time query with Spark
 - Simplify your platform and reduce maintenance cost
- Embrace Spark eco-system
 - Support of complex transformation and analytics with Spark eco system
 - Machine Learning Libraries
 - Spark Streaming

Will be release soon...

- Open sourced already and currently in beta
- For now most of the stuff presented here is done and we are making both improvements and new features like CBO related
- Yeah, and bug fixing and code cleanup as well
- Plan for GA release before the end of 2017

Some frequently asked questions

- Why not build on top of TiDB instead of TiKV
 - Full control of details like Range, Data locations and stats which are not exposed via SQL interface
 - no extra DB layer slowing things down
 - We need a Java client for TiKV as well for other use cases
- Why Spark SQL?
 - Spark SQL is not just SQL, it's the core of a whole eco-system
- Is UDF supported? Sure
- What Spark version supported?
 - For now it's built upon Spark 2.1 but will not be too hard to port to other versions since Spark layer is very relatively thin

Thanks!

2017 Software Architecture Summit

饿了么大数据平台演进实践

岑玉海

高级架构师



目录

- 平台概况
- 服务治理
- 提速增效
- 数据化运营



公司和团队介绍

2015年5月成立

2年

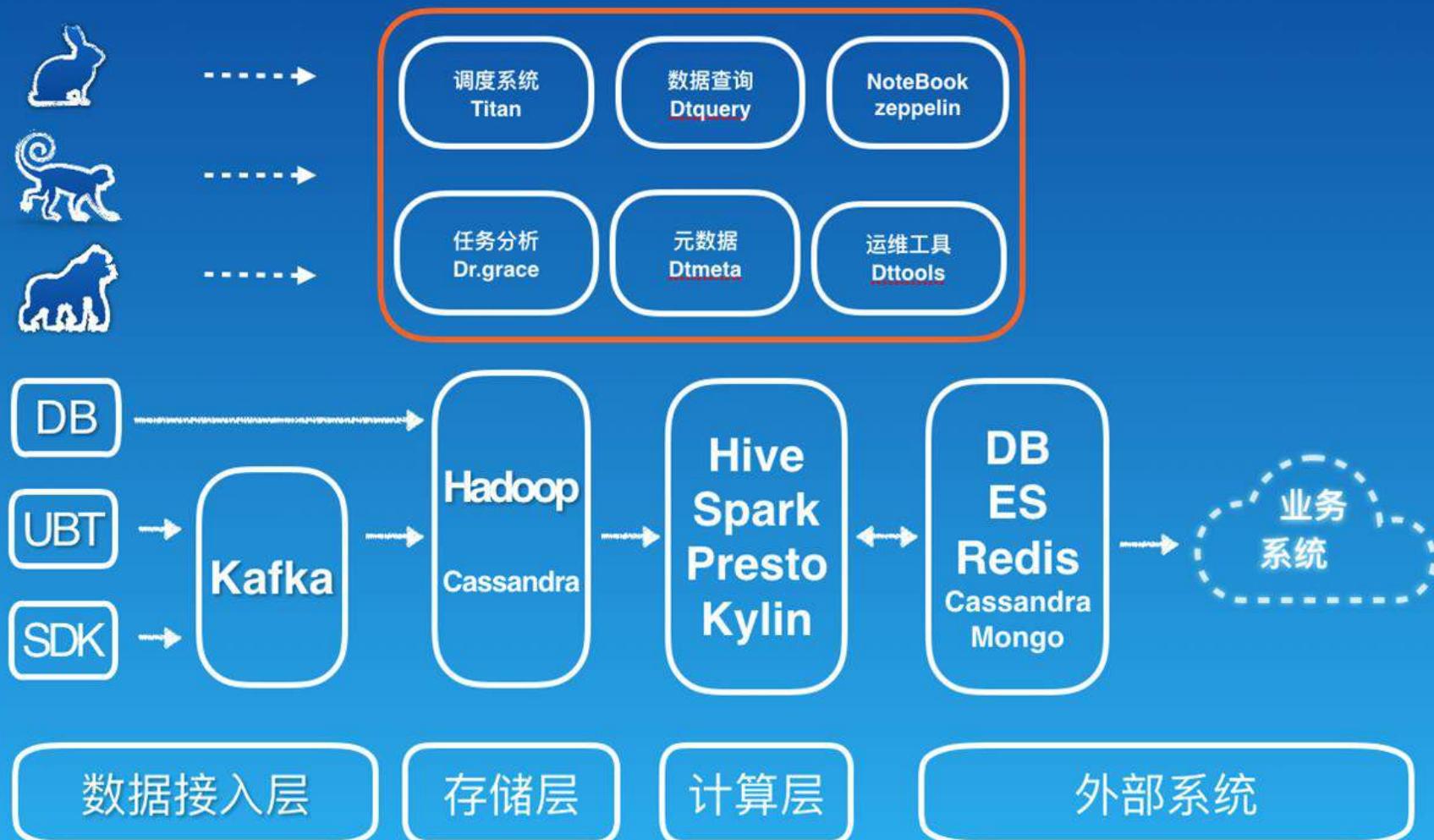


离线平台规模

- 数据增量80TB/天
- Hadoop集群规模1000-1500节点
- 调度任务2万+
- 作业数10W+
- 计算吞吐量3PB/天



离线平台架构



目录

- 平台概况
- 服务治理
- 提速增效
- 数据化运营



平台存储挑战

- Namenode

- 数据量和文件数飞速增长，存储告急
- 文件数增多导致频繁Young GC和RPC出现瓶颈
- 如何实现升级不停服务

- 元数据治理

- 建表没有规范
- 只有数据没有表
- 数据存放目录随意
- 临时数据不清理



Namenode治理

- 合并小文件
 - Hive/Spark集成自动合并小文件
 - 多种文件快速合并工具
- 不停服务升级
 - Service RPC和Client RPC端口分离
 - 动态刷新datanode超时时间
- 锁优化 (DOING)
 - 锁开销Metrics
 - 非公平锁策略：适合读多写少场景
 - 锁拆分

Spark多种文件快速合并：

<https://github.com/eleme-datainfra/spark/tree/ESPARK-116>

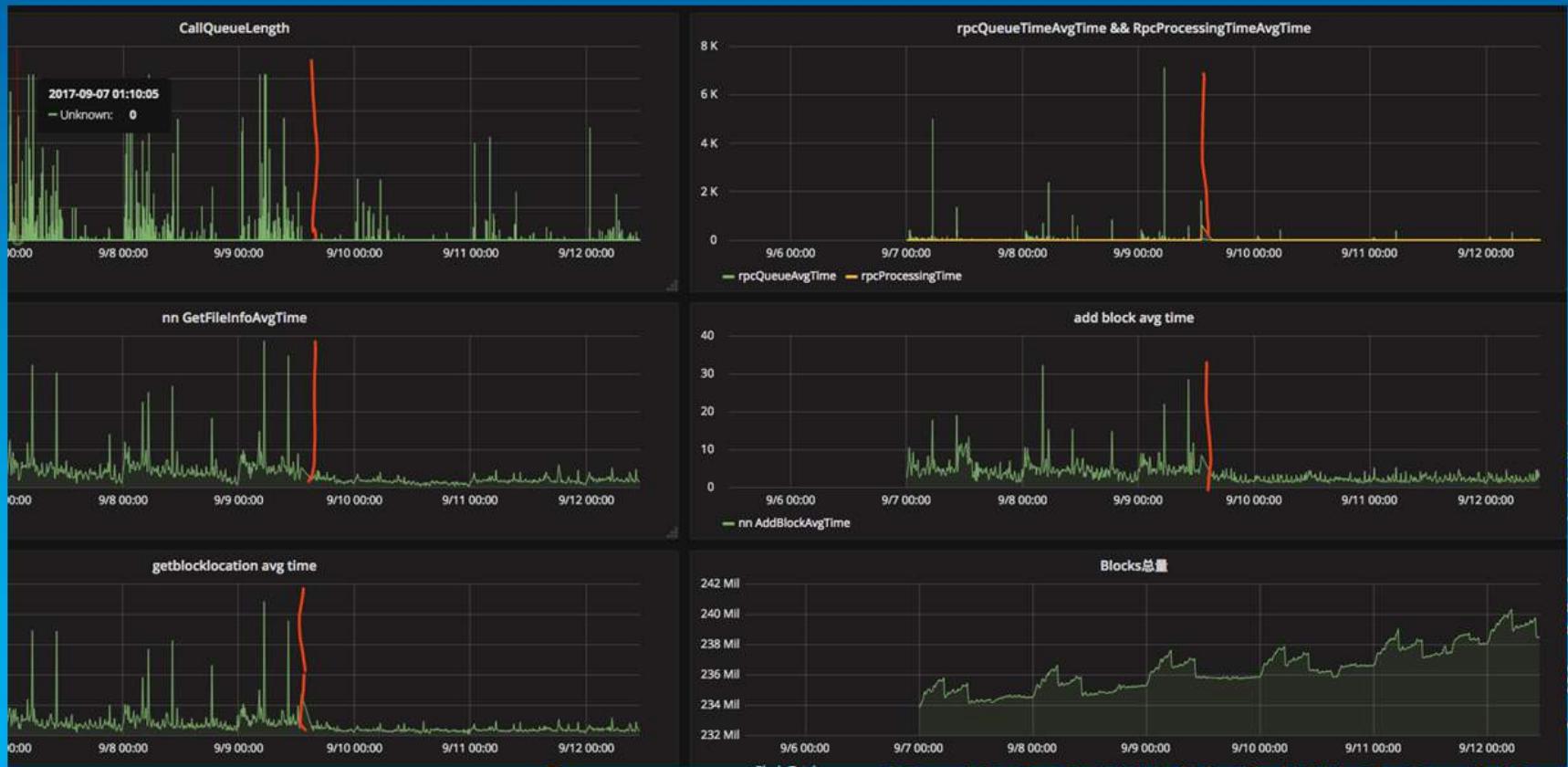
Namenode JVM调优

- CMS调优关键点
 - Young GC的频率和时间
 - CMS GC的Initial mark和Remark时间
 - 防止Concurrent mode failure和Promotion failure
- 对应措施
 - -Xmn30g -XX:MaxTenuringThreshold=3
 - 升级JDK1.8 -XX:+CMSScavengeBeforeRemark
 - -Xmx180g -XX:CMSInitiatingOccupancyFraction=85



Namenode治理效果

高峰期间YoungGC频率降低到17s一次
吞吐率提升到 99.59%



存储容量治理

- 数据生命周期
- 数据热度反向推动清理
- ORC/Parquet文件格式推广



元数据治理

- 建表收口，只能在元数据系统建表
- 临时表只能建在temp库下
- 统一数据表存放位置
- 所有数据都要有表



目录

- 平台概况
- 服务治理
- 提速增效
- 数据化运营

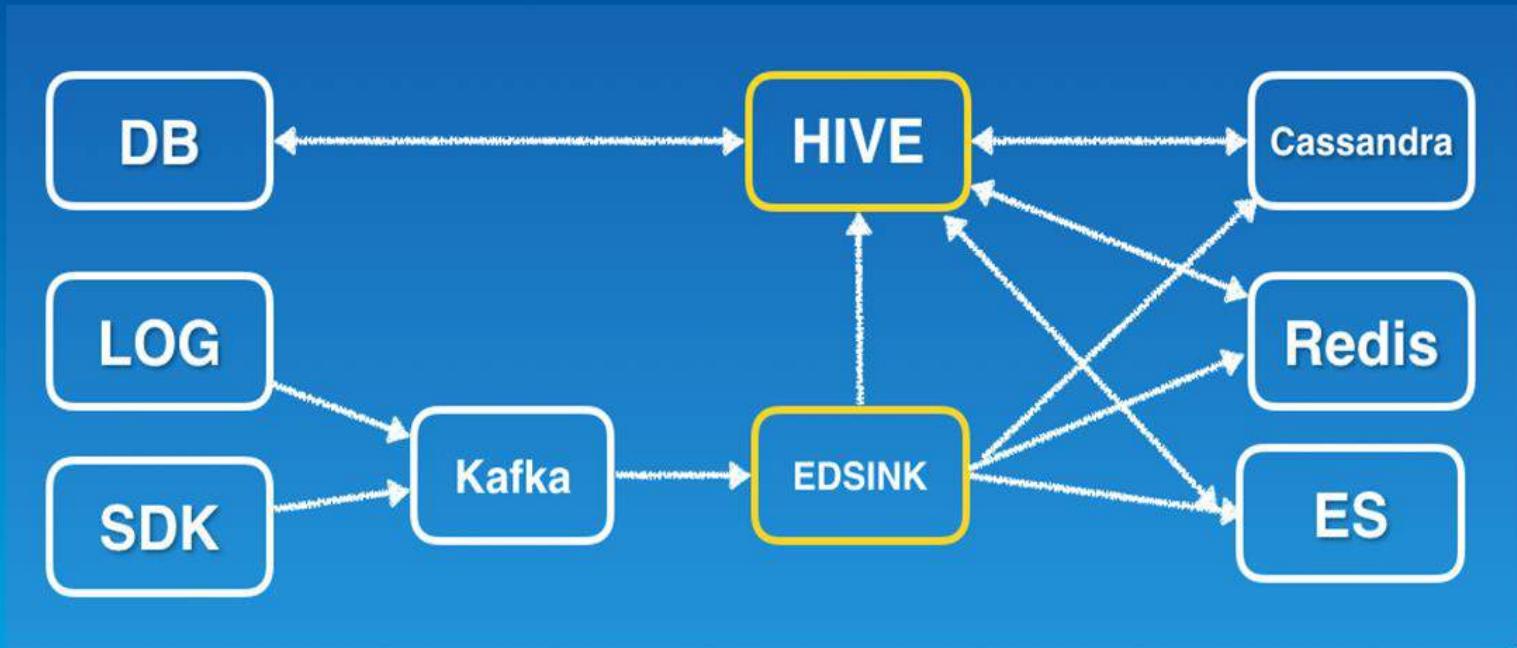


提效增速

- 日益增长的计算需求与落后的生产计算工具之间的矛盾
- 数据流通效率
- 计算效率
- 开发效率



数据高速公路



Spark推广

- Hive ETL切换SparkSQL
- Adhoc接入Spark ThriftServer
- Carbondata
- Streaming SQL
- Zeppelin + Livy



SparkSQL切换

- HiveSQL成功率50% -> 92.5%
- 测试平均加速6倍
- MapJoin深入优化
 - 支持分区表
 - 根据表原始大小来选择
- 自动化数据质量校验

更多详细信息 : <http://dwz.cn/elemespark>

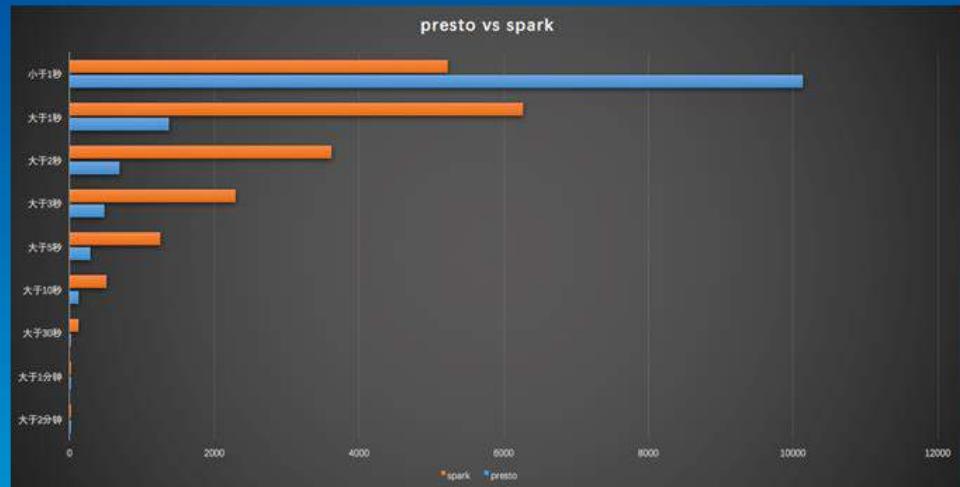
Spark ThriftServer增强

- 动态Executor模式无法正常回收Executor
- Task级别多用户数据权限功能
- FileSystem Cache bug引发OOM
- SQL作业无法取消
- SQL作业进度打印



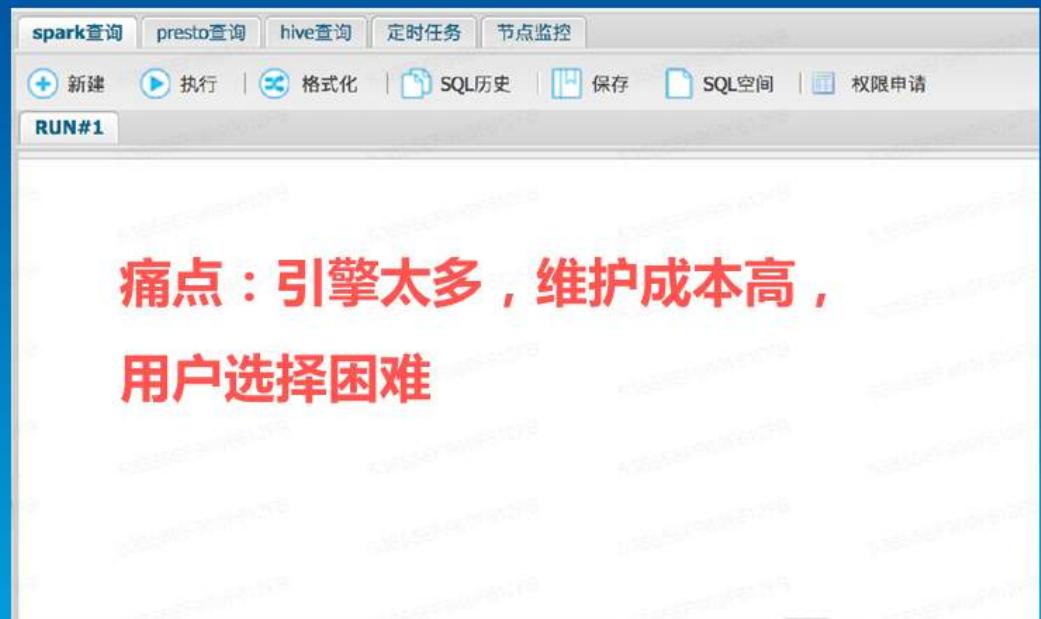
Presto

- 优势
 - 内存MPP引擎
 - Pipeline DAG执行计划
 - 调度延迟低
- 劣势
 - 大查询容易失败
 - 方言与HiveSQL有差别
- 应用业务：餐厅报表
- 每日查询量10W+

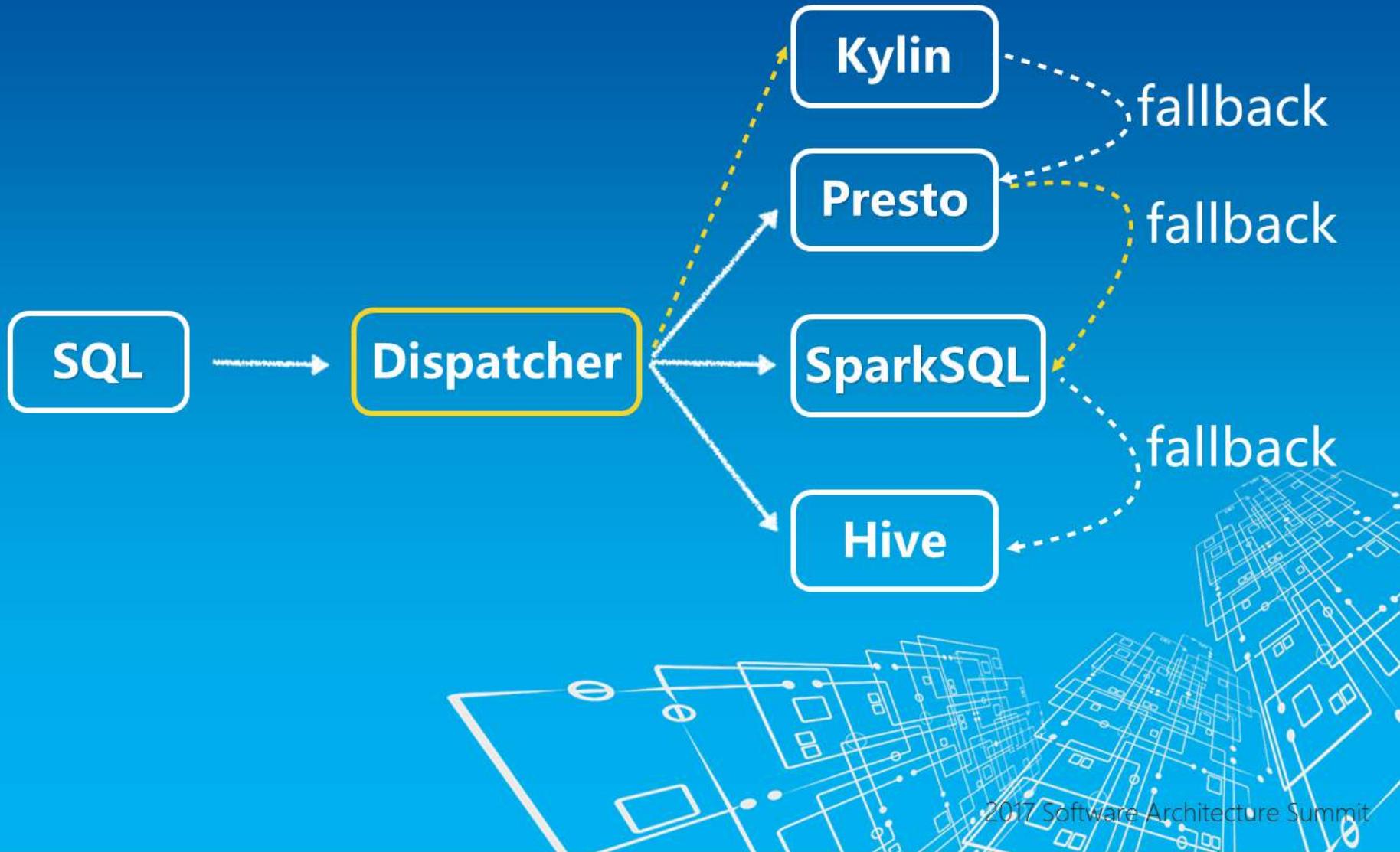


多种引擎的烦恼

- Adhoc Query查询量
 - SparkSQL 4500+
 - Hive 1700+
 - Presto 1000+
- 使用场景
 - Presto : 中小型查询
 - SparkSQL : 大中型查询
 - Hive : 呕底
- 还有刚上线的Kylin...



统一路由引擎



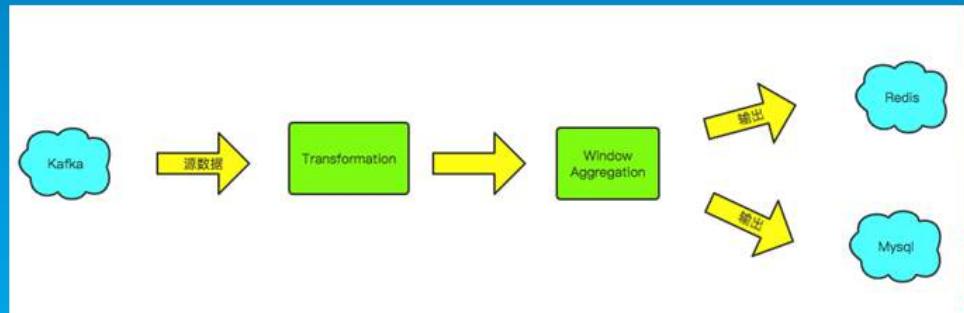
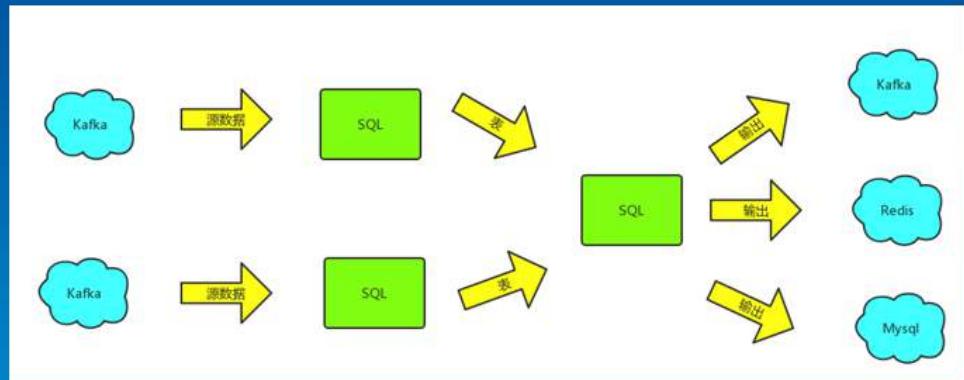
Carbondata

- 读取数据量是影响查询速度关键
- Hive integration(doing)
- 测试数据：3列索引(总共126列 42亿行数据)
- 数据导入时间是parquet的3倍
- 存储大小是parquet的1.5倍
- 命中索引的批量查询比parquet快一倍以上
- 详单查询2秒



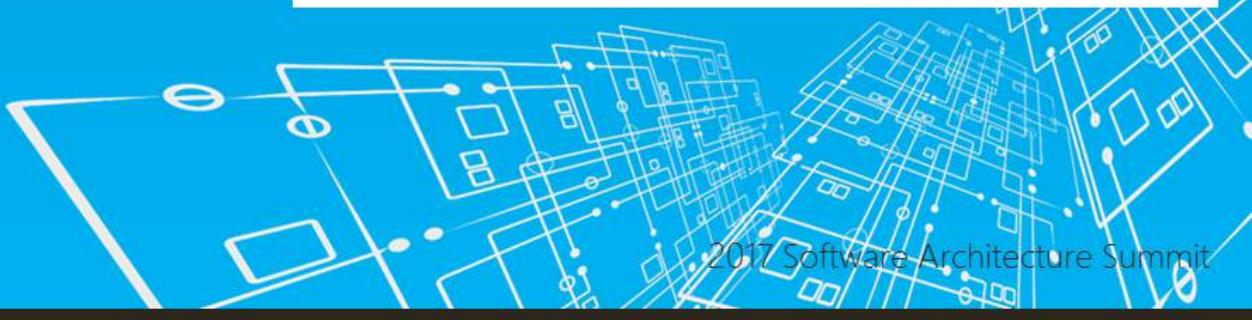
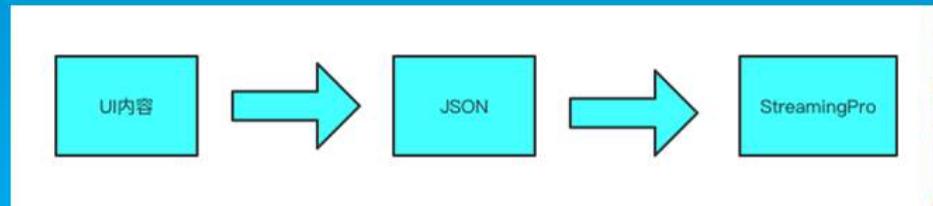
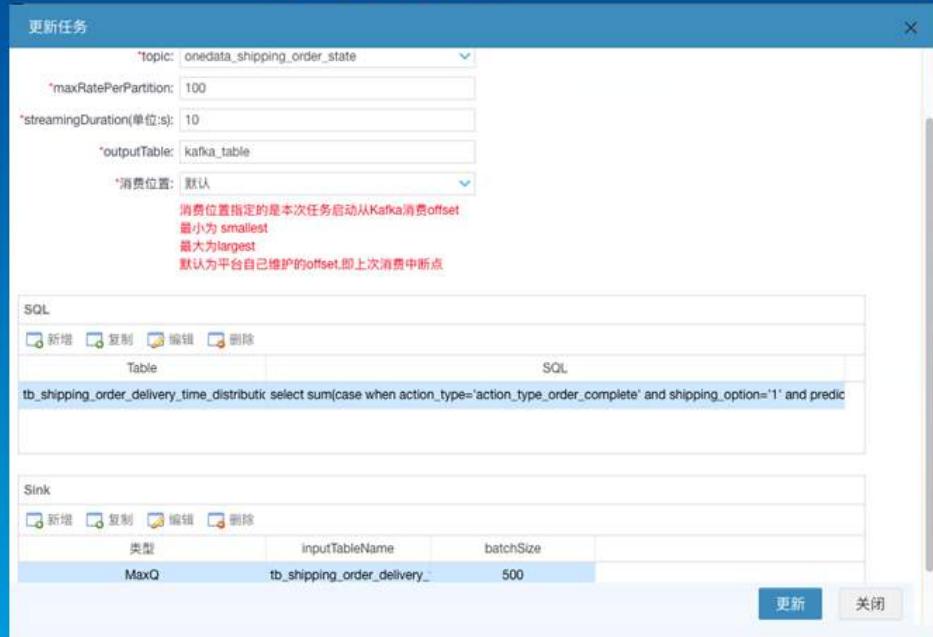
Streaming SQL

- 支持两种场景
- 无状态ETL场景
- Structured Streaming



无状态ETL场景

- Topic管理
- Metrics接入influxdb
- 异常Task日志接入ES
- 自动注册UDF
- 支持多种SINK

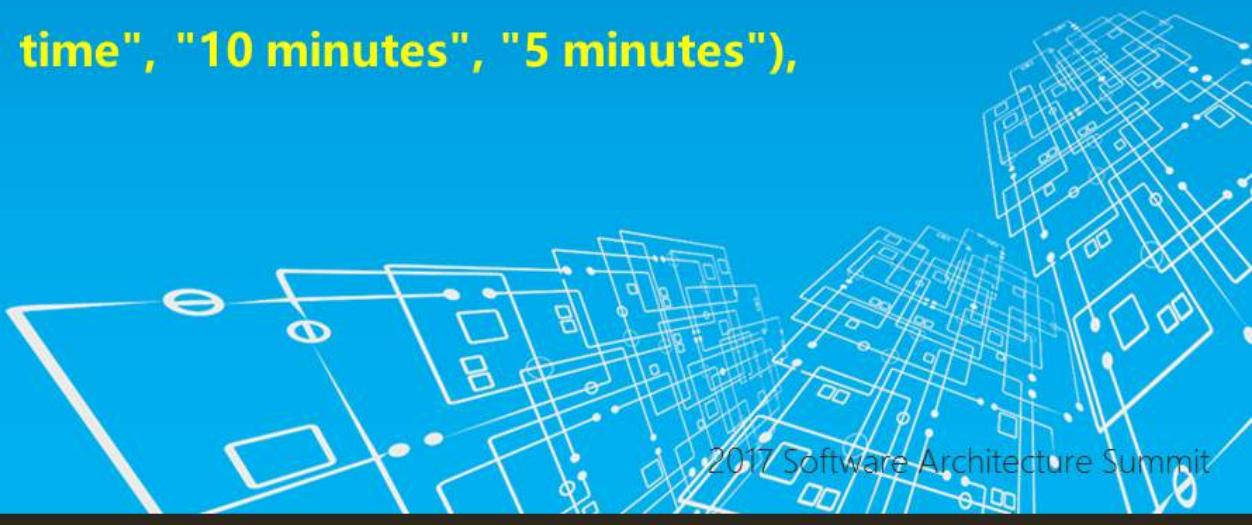


Structured Streaming SQL

```
SELECT action, WINDOW(time, "10 minutes"), COUNT(*)  
FROM events  
GROUP BY action, WINDOW(time, "10 minutes")
```



```
val windowedCounts = actions  
.withWatermark("time", "10 minutes")  
.groupBy(  
    $"action"  
    window($" time", "10 minutes", "5 minutes"),  
)  
.count()
```



目录

- 平台概况
- 服务治理
- 提速增效
- 数据化运营



离线平台运维挑战

如何在分布式环境里快速发现和定位问题？

如何把握平台大盘趋势？

如何做到用户自助错误/性能分析？



如何把握平台大盘趋势？

磁盘IO竞争是“万恶之源”



如何实现用户自助性能/错误分析？



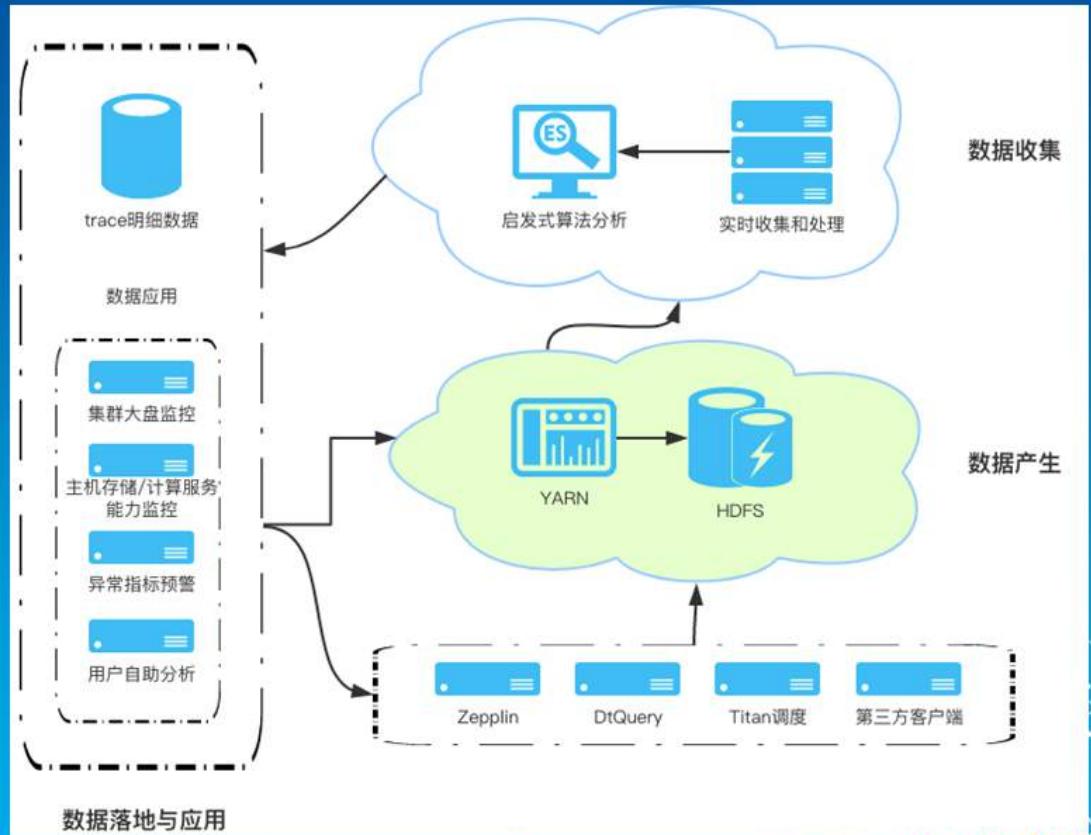
Solution : 数据化运营

数据从何而来 ?



数据监控体系dr.grace

- **数据源**
 - 主机基础指标数据
 - 应用的history log
 - YARN/HDFS Metrics
 - Hive/HDFS Audit log
- **近实时架构**
 - Zabbix
 - Flume+Spark Streaming
- **作业性能分析**
 - 基于Dr-Elephant



获取哪些指标数据？



1. 应用级别监控指标

- 应用运行基本指标
- 应用HDFS读写情况
 - 汇总Container的HDFS读写数据量
- 应用资源消耗情况
 - 汇总Container持续时间 * 资源消耗量
- 应用调度延迟
 - 应用启动与应用提交时间之差
 - 最后一个Mapper和第一个Mapper启动时间之差



2. 集群级别监控指标

- 集群基础指标
- Namenode监控
 - CallQueueLength
 - AvgProcessingTime
 - Operation Count
 - GC time/Count
- ResourceManager监控
 - 自定义Metrics : 平均Container分配时间
- YARN队列快照监控
- HDFS读写吞吐量监控
- HDFS容量监控



3. 主机级别监控指标

- 主机基础指标
- 主机存储服务能力*
 - 主机磁盘Average/Max IOUtil
 - Datanode BlockedThread持续时间和数量
- 主机计算服务能力*
 - 执行计算任务的失败率/KILL率
 - 失败率下钻：区分为系统错误和业务错误
- 系统指标快照*
 - iotop/iftop/top/process stack list



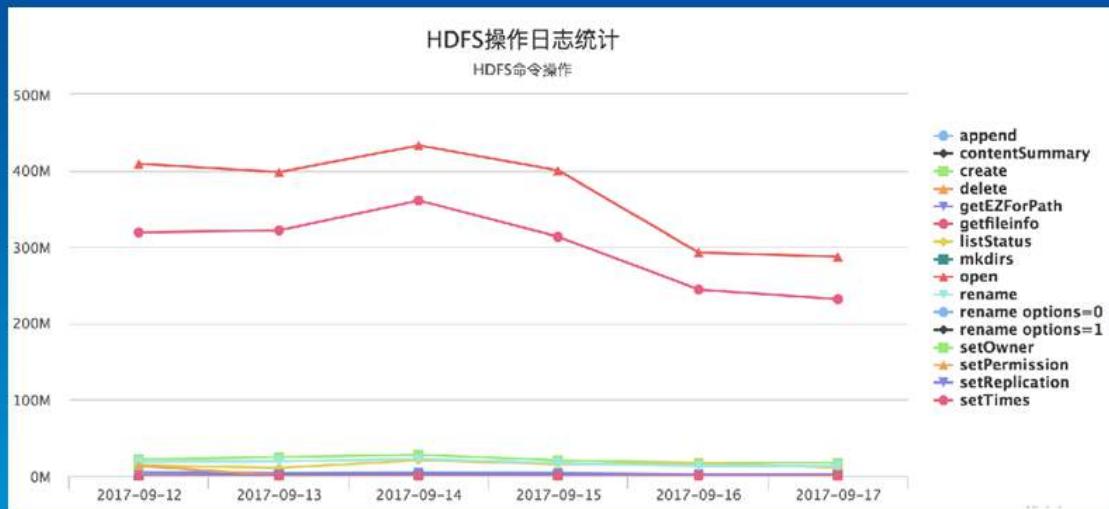
数据用来干什么？



HDFS操作监控

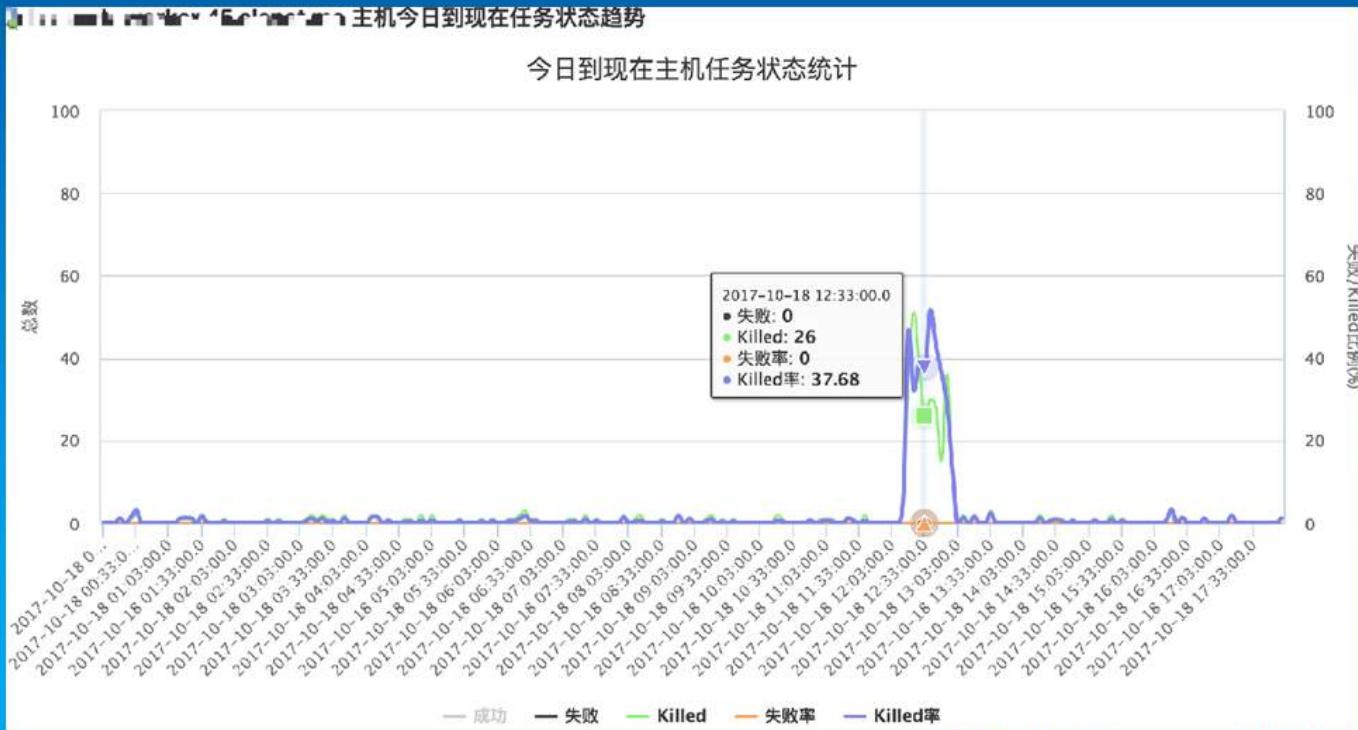
Hive Jar优化

HDFS log优化



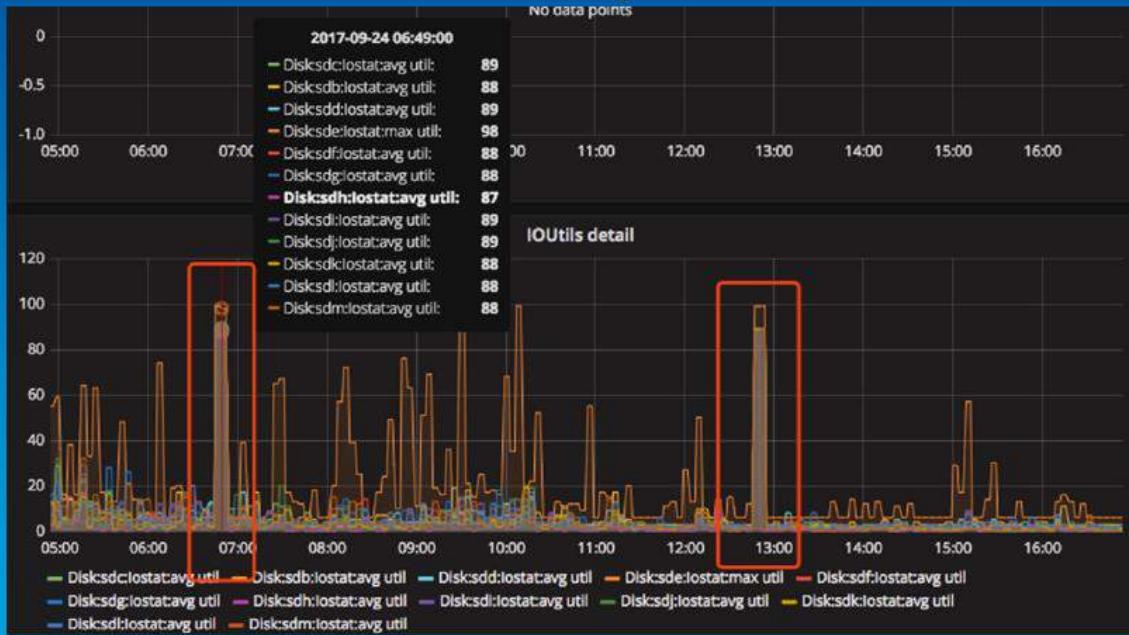
主机计算服务能力应用

上线后累计帮助我们快速发现了20+个主机异常问题



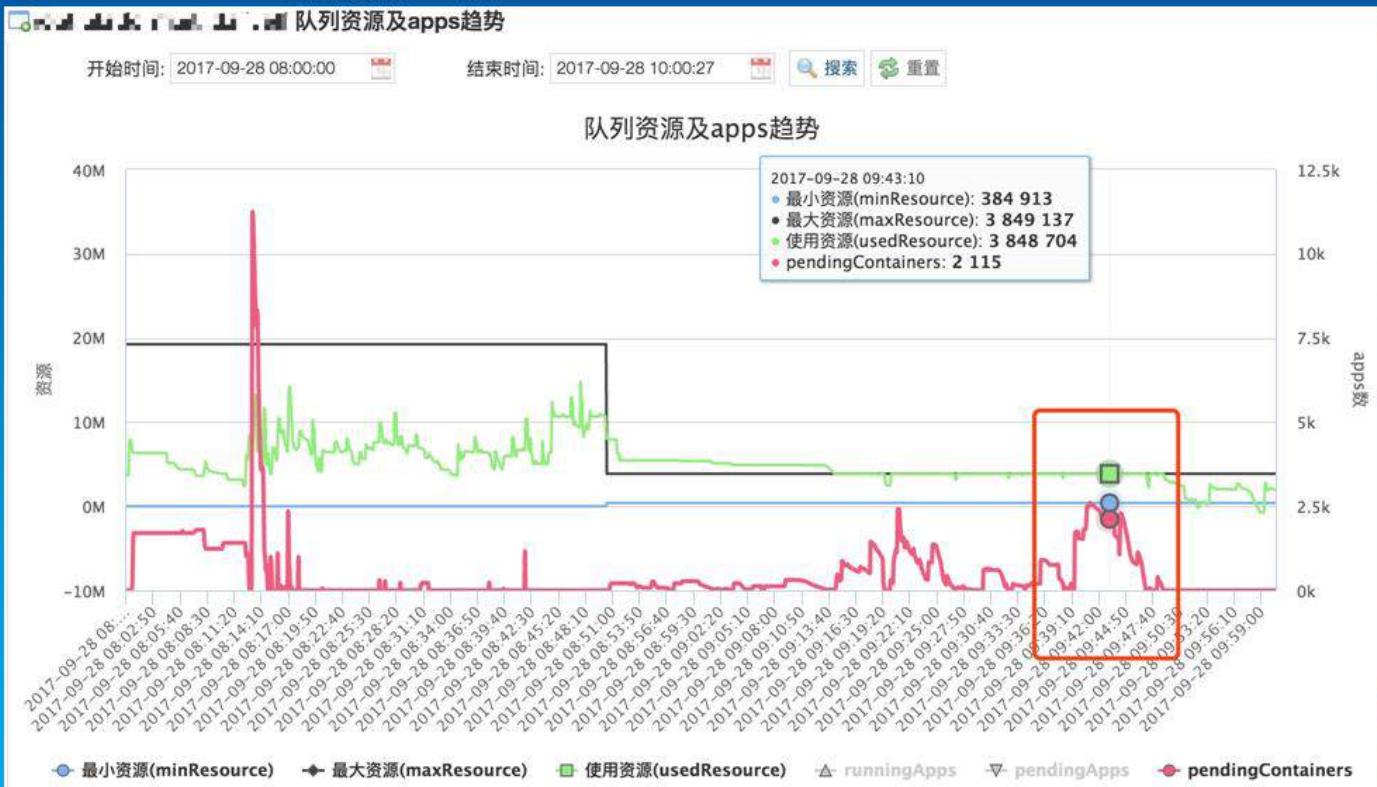
主机存储服务能力应用

Full block report问题



Total DISK READ : 15.41 M/s Total DISK WRITE : 47.95 M/s							
Actual DISK READ: 15.59 M/s Actual DISK WRITE: 448.18 K/s							
TID	PRIO	USER	DISK READ	DISK WRITE	SWAPIN	IO-	COMMAND
8452	be/4	master	596.48 K/s	0.00 B/s	0.00 %	99.99 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
16188	be/4	master	557.49 K/s	0.00 B/s	0.00 %	99.99 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
26682	be/4	master	528.34 K/s	0.00 B/s	0.00 %	99.68 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
8459	be/4	master	641.29 K/s	0.00 B/s	0.00 %	99.66 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
6929	be/4	master	582.83 K/s	0.00 B/s	0.00 %	99.43 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
26689	be/4	master	488.97 K/s	0.00 B/s	0.00 %	99.31 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
14774	be/4	master	634.01 K/s	0.00 B/s	0.00 %	99.26 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
26688	be/4	master	601.21 K/s	0.00 B/s	0.00 %	99.28 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
8458	be/4	master	426.32 K/s	0.00 B/s	0.00 %	99.18 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
6930	be/4	master	623.08 K/s	0.00 B/s	0.00 %	99.05 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
8454	be/4	master	488.26 K/s	0.00 B/s	0.00 %	98.82 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
14787	be/4	master	466.48 K/s	0.00 B/s	0.00 %	98.78 %	java -Dproc_datanode -Xmx2048m -Djava.net.preferIPv4Stack=true -Dlogger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNode
9118	be/3	root	0.00 B/s	47.39 M/s	0.00 %	4.28 %	[jbd2/sqld-8]
9113	be/3	root	0.00 B/s	10.93 K/s	0.00 %	3.37 %	[jbd2/sdhd-8]

队列快照监控应用



应用错误分析

知识库中已经包含**20+**个错误和对应方法

11. Java heap space

错误信息：导致OOM报错

解决方案：1) 配置mapjoin导致，限制关闭mapjoin时将 set hive.a.to.convert.or=false; 2) 增大map或reduce队列数 3) 直接开启有无常数拆分

12. Connection refused

java.net.ConnectException: Connection refused. Attempt 0 attemp_1483804030C21_623288_m_003456_0 Timed out after 600 sec

解决方案：插入数据过多，冲突作业时间长，减少map线程数量或增加线程

13. StringIndexOutOfBoundsException

java.lang.StringIndexOutOfBoundsException: String index out of range:

解决方案：数据溢出，检查调用的udf参数类型或数据值是否正确

14. Communications link failure

com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure. The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.

解决方案：禁用数据库，telnet到port端口测试dts是否连接成功或直接重启

15. Data truncation

com.mysql.jdbc.MySQLDataTruncation: Data truncation: Data too long for column

解决方案：增加长度限制，联系owner调整为cbo导出或直接对db进行修改

16. Duplicate entry

com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException: Duplicate entry '20161220000000000' for key 'PRIMARY'

解决方案：主键重复，联系owner检查数据

17. Can't parse input data

java.lang.RuntimeException: Can't parse input data: '2015-10-28 00:16:24'

解决方案：1. 数据是字段中含有非法字符，导致数据读取错误，字段类型不匹配。联系owner检查数据 2. 如果是相关myisam 看看是不是某个列带有注释

解决方案

on: Can't export data, please check failed map task logs at
xtExportMapper.map(TextExportMapper.java:112) at
xtExportMapper.map(TextExportMapper.java:39) at
Mapper.run(Mapper.java:145) at
toProgressMapper.run(AutoProgressMapper.java:64) at
Task.runNewMapper(MapTask.java:787) at
Task.run(MapTask.java:341) at
Child\$2.run(YarnChild.java:164) at
>Privileged(Native Method) at
(Subject.java:415) at
'GroupInformation.doAs(UserGroupInformation.java:1693) at
Child.main(YarnChild.java:158) Caused by: java.io.IOException:
4.MySQLIntegrityConstraintViolationException: Duplicate entry '2017-09-
'log_date' at
syncSqlRecordWriter.write(AsyncSqlRecordWriter.java:233) at
syncSqlRecordWriter.write(AsyncSqlRecordWriter.java:48) at
Task\$NewDirectOutputCollector.write(MapTask.java:658) at
ask.TaskInputOutputContextImpl.write(TaskInputOutputContextImpl.java:89)
e.lib.map.WrappedMapper\$Context.write(WrappedMapper.java:112) at
xtExportMapper.map(TextExportMapper.java:84) ... 10 more Caused by:
4.MySQLIntegrityConstraintViolationException: Duplicate entry '2017-09-
'log_date' at
cessorImpl.newInstance0(Native Method) at
cessorImpl.newInstance(NativeConstructorAccessorImpl.java:57) at
>AccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
ewInstance(Constructor.java:526) at
stance(Util.java:408) at com.mysql.jdbc.Util.getInstance(Util.java:383) at
com.mysql.jdbc.Connection.createNewIO(Connection.java:173)

Hive应用性能分析

应用等待时间

数据处理速度

数据倾斜

Reducer处理数据量

Reducer GC

Dr.Grace 首页 今日分析 搜索 快速搜索 错误查询 帮助 解决方案

搜索

集群: ETL集群

应用ID (精确匹配): job_1507632609305_737874

用户名 (精确匹配): User

队列名:

action_sid (调度任务id): actionSid

query_id (dtquery查询id): queryId

Job类型: 请选择任务类型

严重性:

错误项:

等待启动时间: 0.023 GB Hours (20.57%)

Mapper数据倾斜: 0.023 GB Hours (20.57%)

Mapper溢出: 0.023 GB Hours (20.57%)

Mapper内存: 0.023 GB Hours (20.57%)

MapperRecordSkew: 0.023 GB Hours (20.57%)

MapperRecordSkew: 0.023 GB Hours (20.57%)

MapperGC: 0.023 GB Hours (20.57%)

Mapper时间: 0.023 GB Hours (20.57%)

Mapper内存: 0.023 GB Hours (20.57%)

ShuffleSort: 0.023 GB Hours (20.57%)

2017-10-18 00:38:41

INSERT OVERWRITE DIRECTORY('2017-10-17', -30)(Stage-2)

Jobtracker: http://bigdata-rsm.elent.me:20020/jobhistory/job/job_1507632609305_737874

等待启动时间 Severity: success

Start Time: 1508258308000

Submit Time: 1508258304000

Wait Time: 4 sec

Mapper数据倾斜 Severity: danger [Explain]

Group 1: 1 tasks @ 12 MB avg

Group 2: 2 tasks @ 275 MB avg

Number of tasks: 3

MapperRecordSkew Severity: danger [Explain]

Group 1: 1 tasks @ 38 K records avg

Group 2: 2 tasks @ 792 K records avg

Number of tasks: 3

Spark应用性能分析

Spark配置

SparkExecutor内存使用

Spark数据倾斜

Dr.Grace 首页 今日分析 搜索 快速搜索 错误查询 帮助 解决方案

搜索

集群 ETL集群

应用ID (精确匹配) job_1507632609305_738693

用户名 (精确匹配) User

队列名

action_sid (调度任务id) actionSid

query_id (dtquery查询id) queryId

Job类型 请选择任务类型

严重性

错误项

搜索

[root.prepare.low][master] [Spark] job_1507632609305_738693
54475_25389385:dm_mkt_act_order_subject_54475.sd:s1q1:10.0.146.84:27757:create
2017-10-18 00:43:57

Jobtracker: http://bigdata-rsm.elenet.me:8088/proxy/application_1507632609305_738693
Job History
Flow History

Spark配置最佳实践 SparkExecutor内存使用 Spark数据倾斜 SparkJob运行时 Spark执行负载均衡 SparkEventLog限制

0.163 GB Hours 27.55 % Failed 41:42:43 0.00 % [Explain]

Spark配置最佳实践
Severity: success [Explain]
spark.driver.memory 5g
spark.executor.cores 4
spark.serializer org.apache.spark.serializer.KryoSerializer
spark.shuffle.manager Not presented. Using default

SparkExecutor内存使用
Severity: success
Memory utilization rate 0.001
Total driver memory allocated 5 GB
Total executor memory allocated 0.3 (6 GB x 0)
Total memory allocated for storage 3.31 GB
Total memory used at peak 3.34 MB

Spark数据倾斜
Severity: danger

应用同比性能分析

- 调度时间
- 执行时间
- 消耗资源
- 性能分析
- 应用下钻



总结

- 集群运行状况可控
- 快速发现定位问题
- 用户自助解决问题，解放人力



数据化监控与治理规划





Join us !
Thank you



A dark blue Airstream trailer is parked on a beach at sunset. The word "AIRSTREAM" is printed in white on the side of the trailer. The sky is filled with warm orange and yellow hues from the setting sun. The ocean waves are visible in the background.

HAO WANG

Airbnb Data Platform & Streaming Processing



About Me

- University of Southern Calif
- IBM Watson
- Airbnb Data Platform





3 Million

TOTAL HOMES ON AIRBNB

65K

CITIES

191+

COUNTRIES

Agenda

- Data Platform at Airbnb
 - High-level overview
 - ReAir: Incremental Data Replication
 - Airflow: Scheduling
 - Analytics: Superset
- AirStream: Unified Streaming and Batch Processing



AIRBNB DATA PLATFORM

Scale of Data Infrastructure at Airbnb

>13B

#Events Collected

>35PB

Warehouse Size

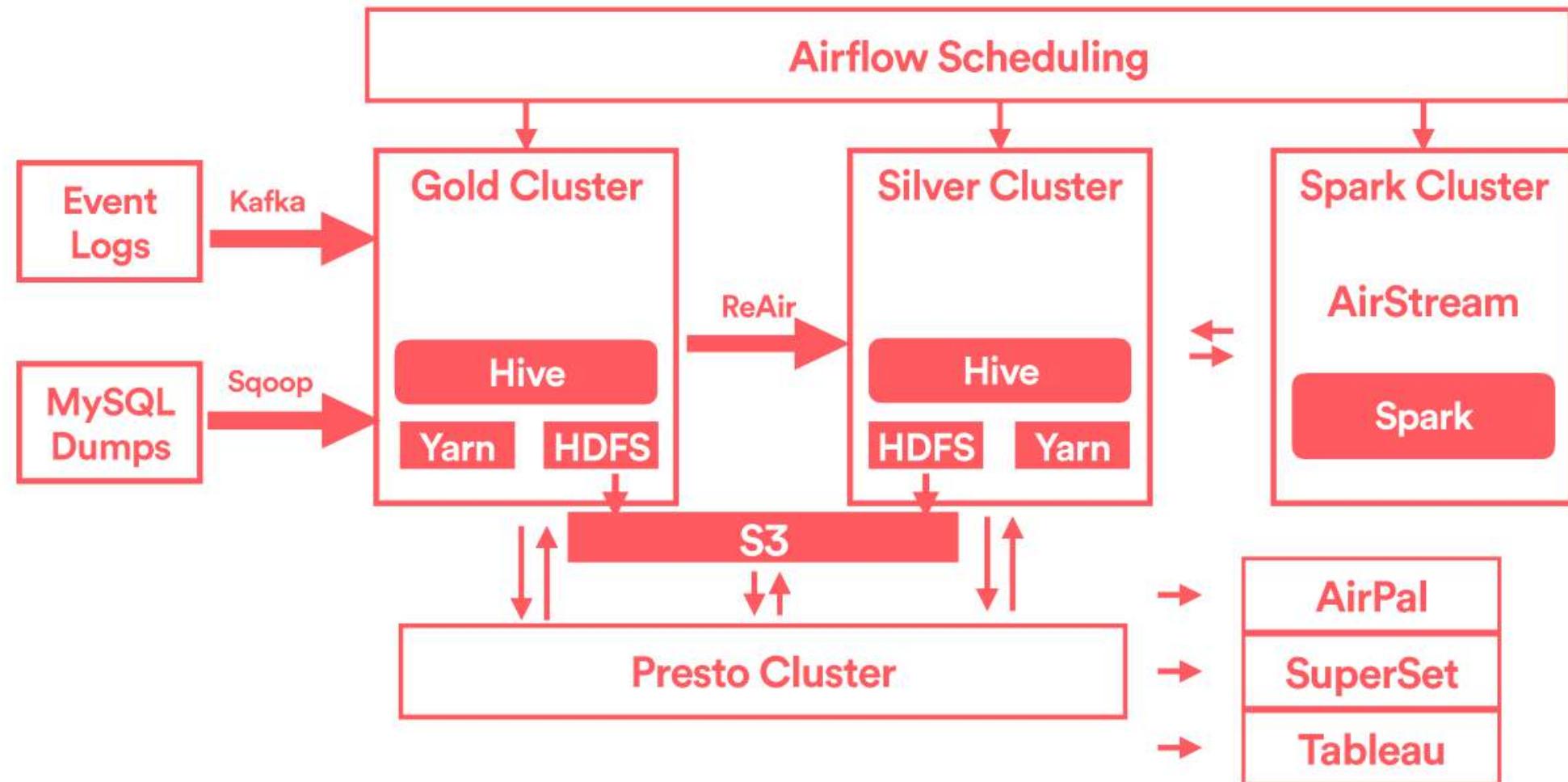
1400+

Machines
Hadoop + Presto +
Spark

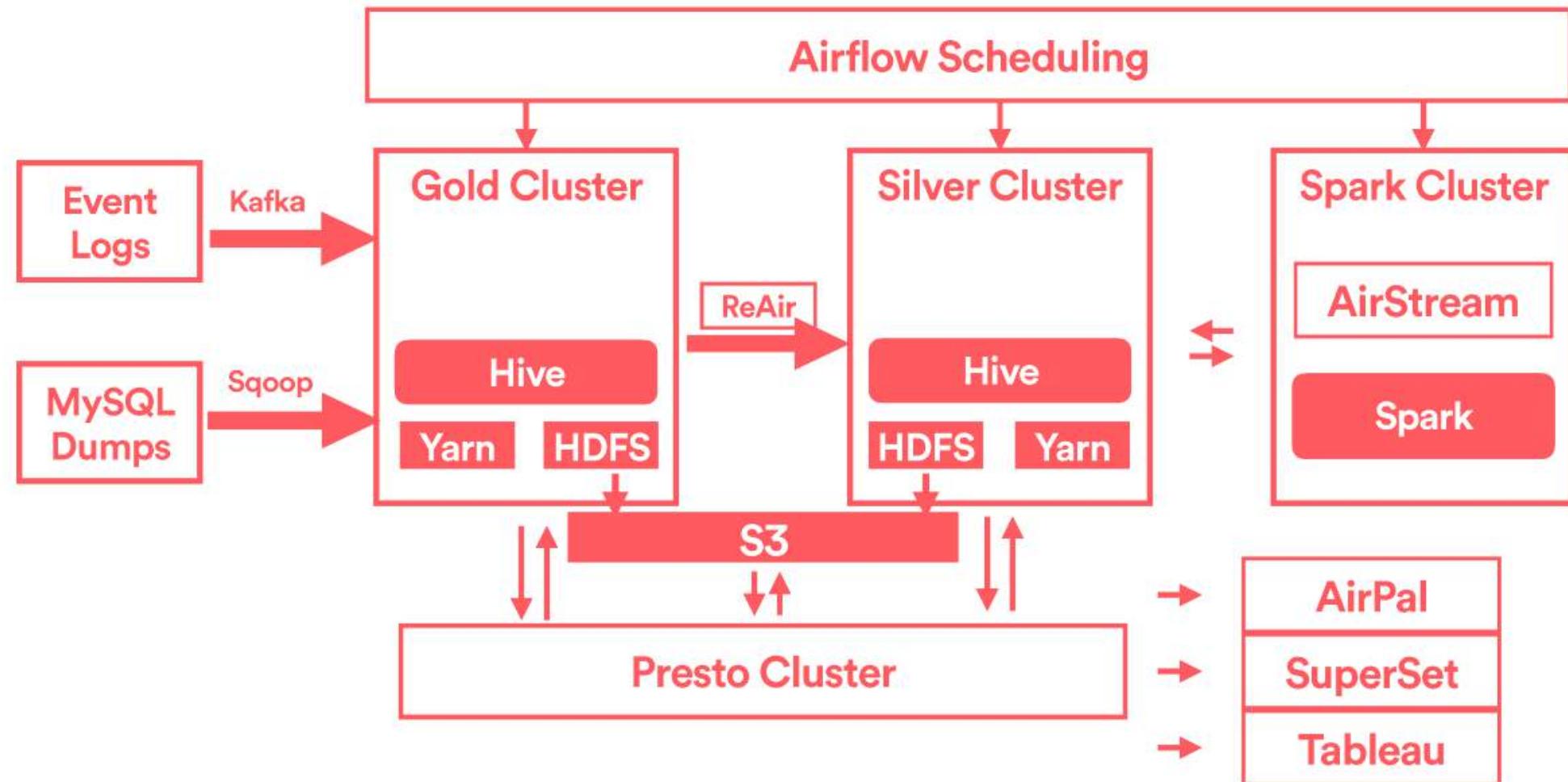
5x

YoY Data Growth

Data Platform

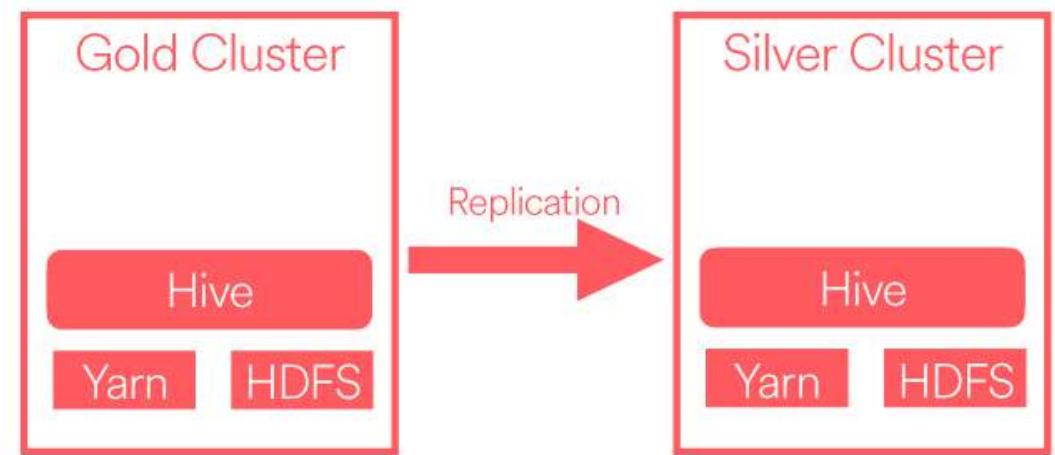


Data Platform



Two Clusters

- Two independent HDFS, MR, Hive metastores
- d2.8xlarge w/ 48TB local
- ~250 instances in final setup
- Replication of common / critical data - Silver is super of Gold
- For disaster recovery, separate AZ's



Multi-Cluster Trade-Offs

Advantages	Disadvantages
<ul style="list-style-type: none">• Failure isolation with user jobs• Easy capacity planning• Guarantee SLA's• Able to test new versions• Disaster Recovery	<ul style="list-style-type: none">• Data synchronization• User confusion• Operational overhead

Multi-Cluster Trade-Offs

Advantages	Disadvantages
<ul style="list-style-type: none">• Failure isolation with user jobs• Easy capacity planning• Guarantee SLA's• Able to test new versions• Disaster Recovery	<ul style="list-style-type: none">• Data synchronization• User confusion• Operational overhead

REAIR: INCREMENTAL DATA REPLICATION

Warehouse Replication Approaches

Batch	Incremental
<ul style="list-style-type: none">• Scan HDFS, metastore• Copy relevant entries• Simple, no state• High latency	<ul style="list-style-type: none">• Record changes in source• Copy/re-run operations on destination• More complex, more state• Low latency (seconds)

Incremental Replication

- Record Changes on Source
- Convert Changes to Replication Primitives
- Run Primitives on the Destination

Record Changes On Source

- Hive provides hooks API to fire at specific points
 - Pre-execute
 - Post-execute
 - Failure
- Use post-execute to log objects that are created into an audit log
- In critical path for queries

Convert Changes to Primitive Operations

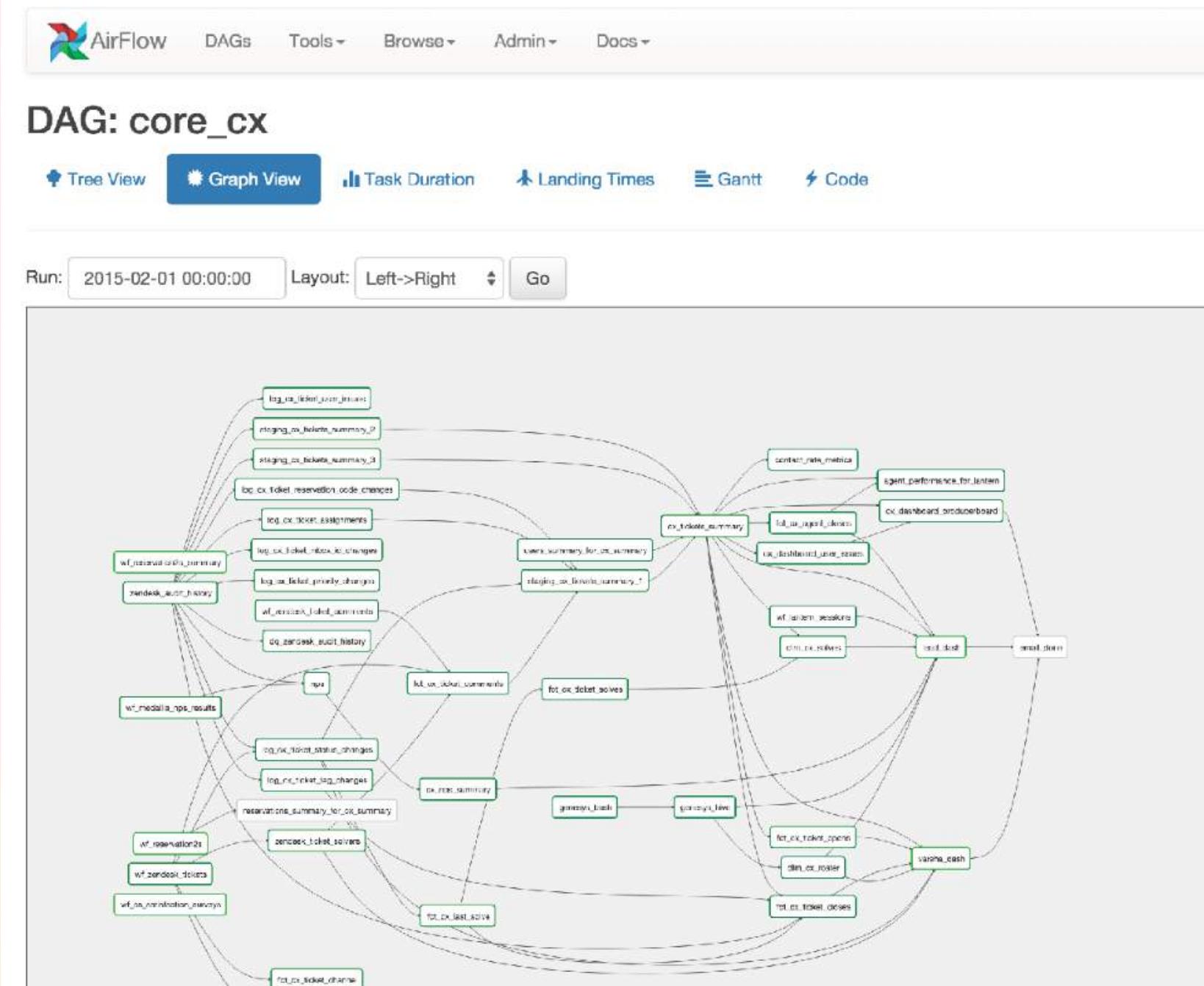
- 3 types of objects - DB, table, partition
- 3 types of operations - Copy, rename, drop
- 9 different primitive operations
- Idempotent

WORKFLOW & SCHEDULING

Airflow

A PLATFORM TO PROGRAMMATICALLY AUTHOR, SCHEDULE AND MONITOR WORKFLOWS

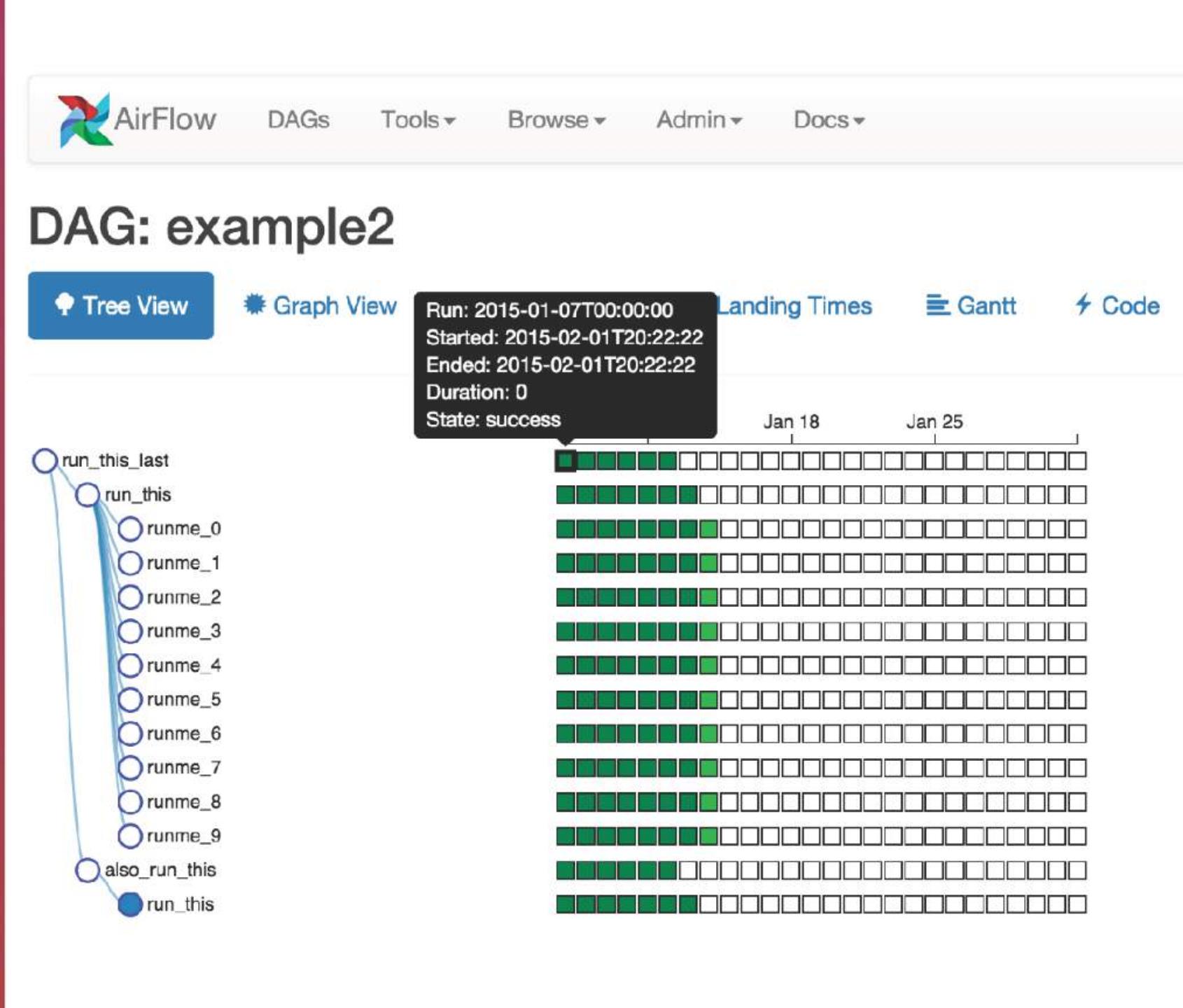
[HTTP://AIRFLOW.APACHE.ORG](http://airflow.apache.org)



Airflow

A PLATFORM TO
PROGRAMMATICALLY AUTHOR,
SCHEDULE AND MONITOR
WORKFLOWS

[HTTP://AIRFLOW.APACHE.ORG](http://AIRFLOW.APACHE.ORG)



ANALYTICS

Superset

- QUERY AND VISUALIZE DATA
- CREATE AND SHARE DASHBOARDS
- ENTERPRISE-READY AUTHENTICATION
- INTEGRATION WITH DRUID

[HTTP://SUPerset.apache.org](http://SUPerset.apache.org)

The screenshot shows the Apache Superset interface with the 'SQL Lab' tab selected. At the top, there's a navigation bar with links for Security, Manage, Sources, Slices, Dashboards, and SQL Lab. Below the navigation is a section for 'Untitled Query 2' with dropdown menus for Select a database (1), Select a schema (0), and Add a table (0). To the right of these dropdowns is a large text input field containing the text '1 SELECT ...'. Below the input field are two tabs: 'Results' (which is selected) and 'Query History'. A light blue banner at the bottom of the screen says 'Run a query to display results here'.

Superset

- QUERY AND VISUALIZE DATA
- CREATE AND SHARE DASHBOARDS
- ENTERPRISE-READY AUTHENTICATION
- INTEGRATION WITH DRUID

[HTTP://SUPerset.apache.org](http://SUPerset.apache.org)

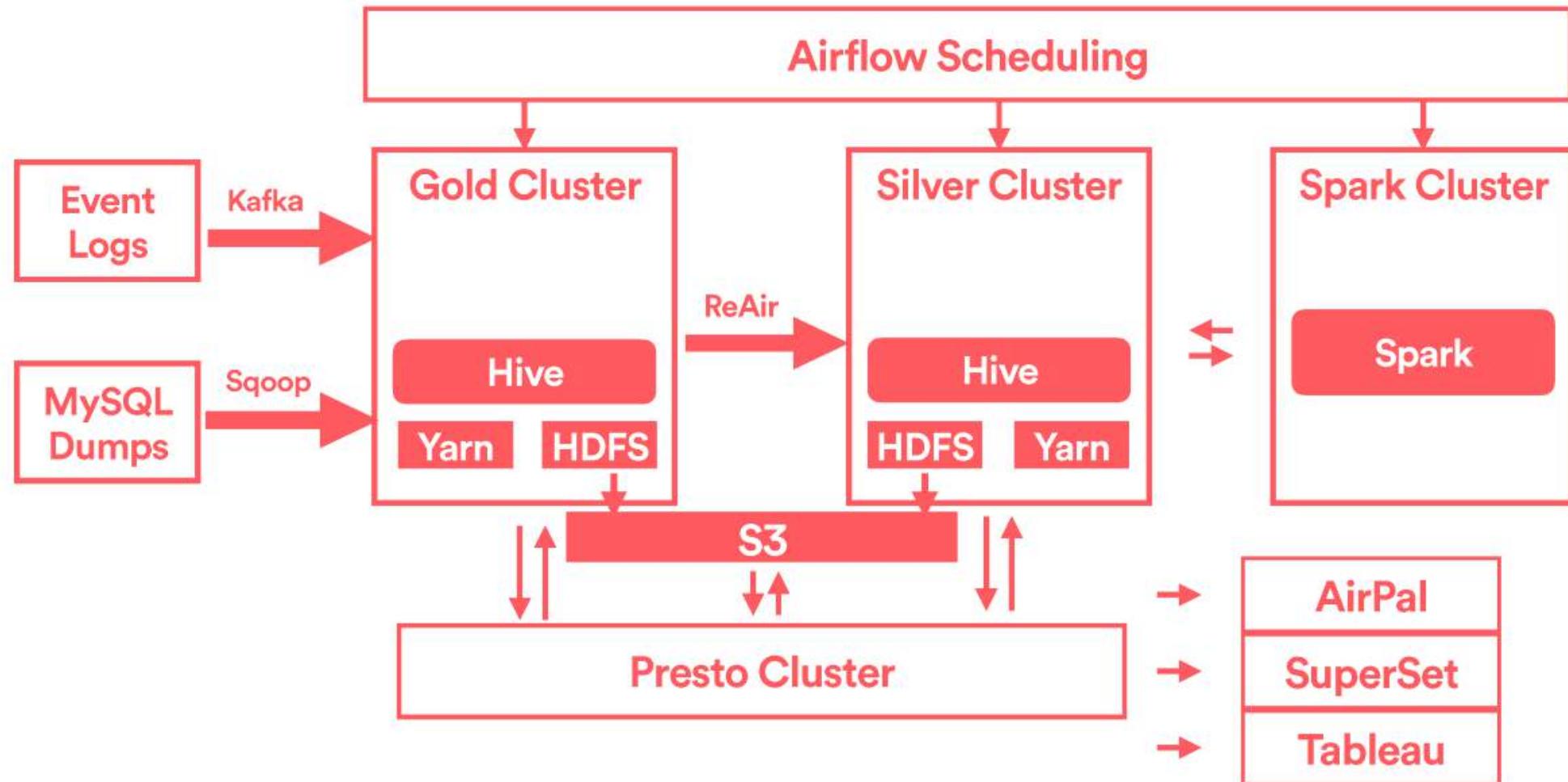
The screenshot shows the Apache Superset interface. At the top, there is a navigation bar with links for Security, Manage, Sources, Slices, Dashboards, and SQL Lab. On the far right of the header are icons for refresh, print, and user profile. Below the header, the word "Dashboards" is displayed in a large, bold font. To the right of this, there is a search bar with a magnifying glass icon. The main content area is titled "Dashboards" and contains a table listing three items:

Dashboard	Creator	Modified
Misc Charts		a minute ago
Births		a minute ago
World's Bank Data		a minute ago

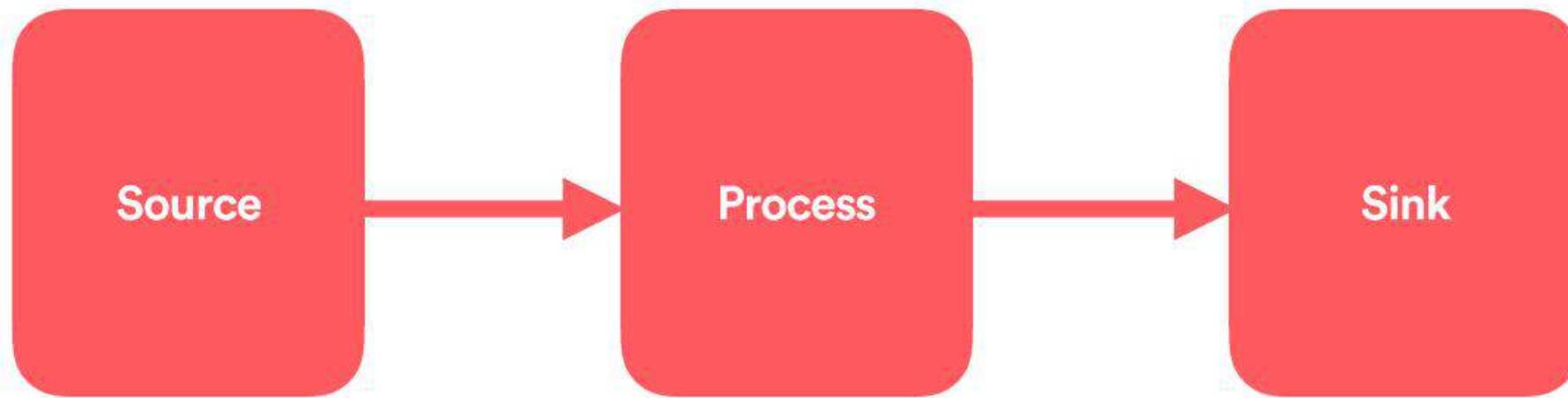
At the bottom right of the dashboard list, there are navigation buttons labeled "Previous", "1", and "Next".

AIRSTREAM: UNIFIED STREAMING AND BATCH PROCESSING

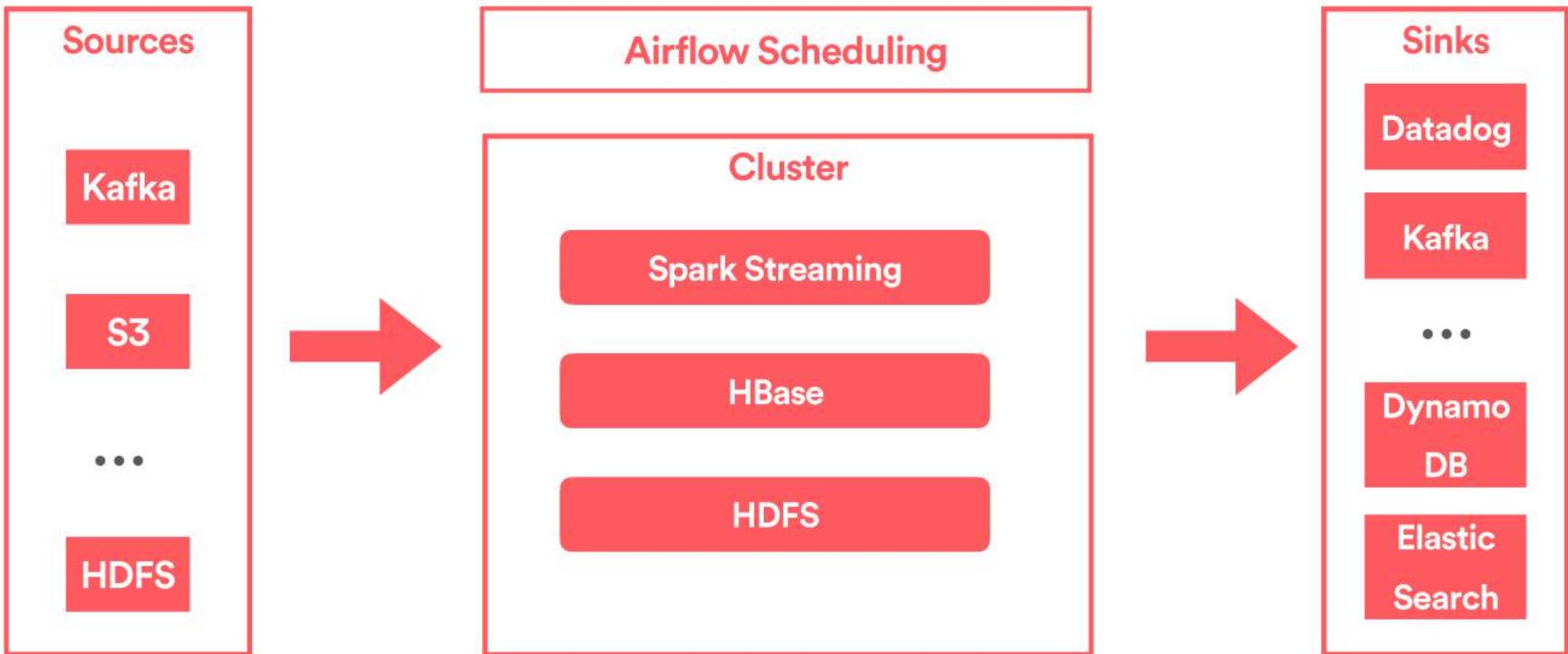
Batch Infrastructure



AirStream

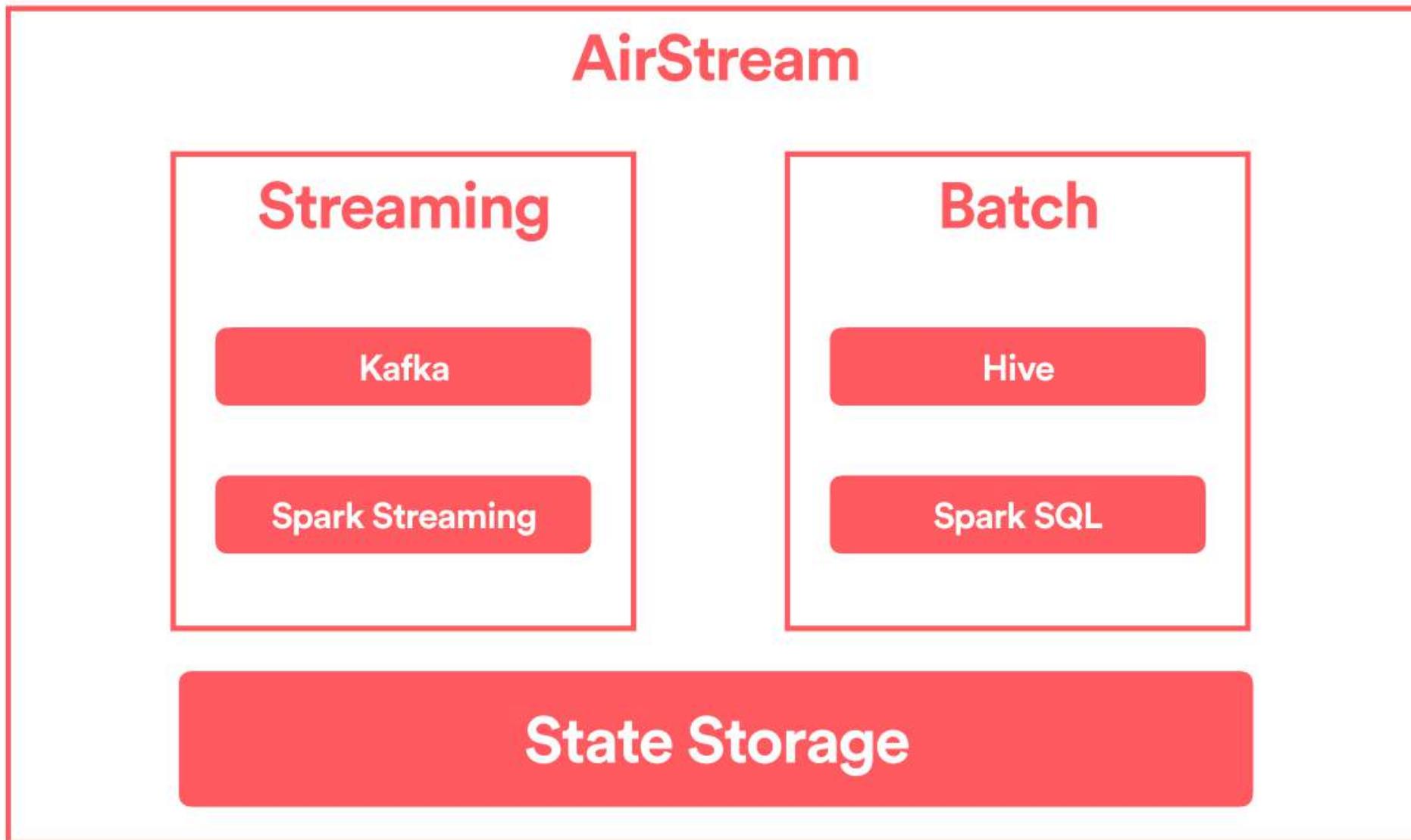


Streaming at Airbnb - AirStream



LAMBDA ARCHITECTURE

Lambda Architecture



Sources

Streaming

```
source: [  
  {  
    name: source_example,  
    type: kafka,  
    config: {  
      topic: "example_topic",  
    }  
  }  
]
```

Batch

```
source: [  
  {  
    name: source_example,  
    type: hive,  
    sql: {  
      select * from db.table  
      where ds='2017-06-05';  
    }  
  }  
]
```

Computation

Streaming/Batch

```
process: [{  
    name = process_example,  
    type = sql,  
    sql = """  
        SELECT listing_id, checkin_date, context.source as source  
        FROM source_example  
        WHERE user_id IS NOT NULL    """  
}]
```

Sinks

Streaming

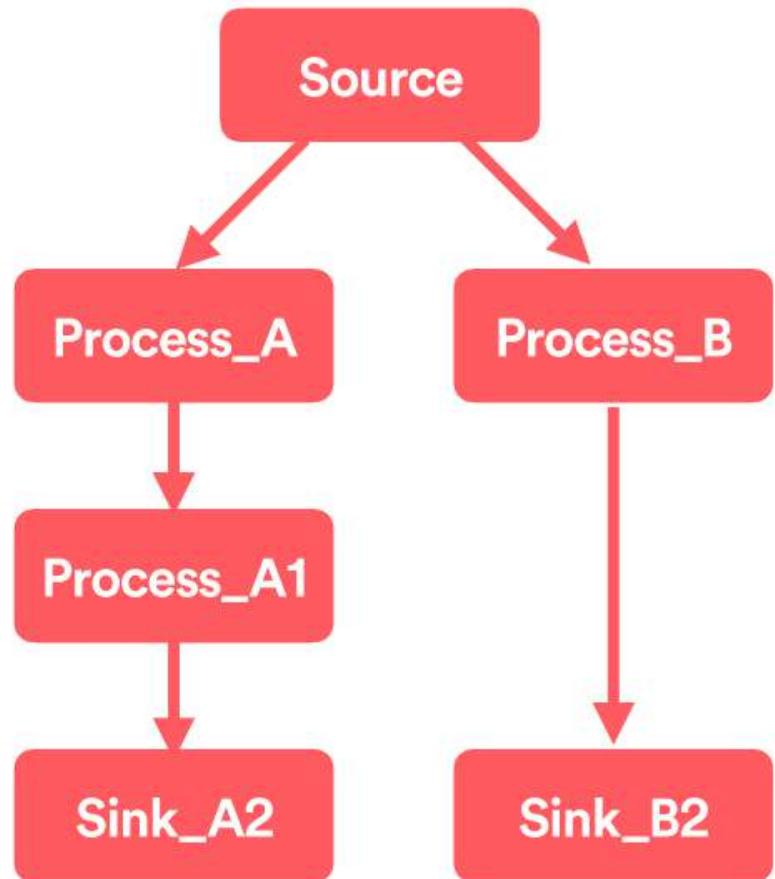
```
sink: [  
  {  
    name = sink_example  
    input = process_example  
    type = hbase_update  
    hbase_table_name = test_table  
    bulk_upload = false  
  }  
]
```

Batch

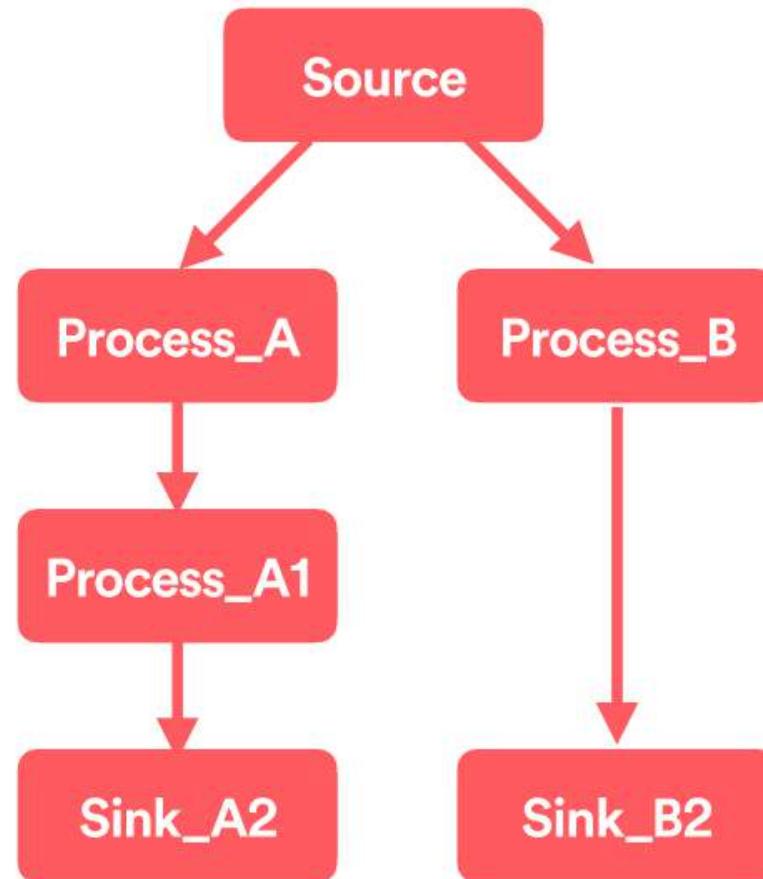
```
sink: [  
  {  
    name = sink_example  
    input = process_example  
    type = hbase_update  
    hbase_table_name = test_table  
    bulk_upload = true  
  }  
]
```

Computation Flow

Streaming



Batch



Unified API through AirStream

- Declarative job configuration
- Streaming source vs static source
- Computation operator or sink can be shared by streaming and batch job.
- Computation flow is shared by streaming and batch
- Single driver executes in both streaming and batch mode job

SHARED STATE STORAGE

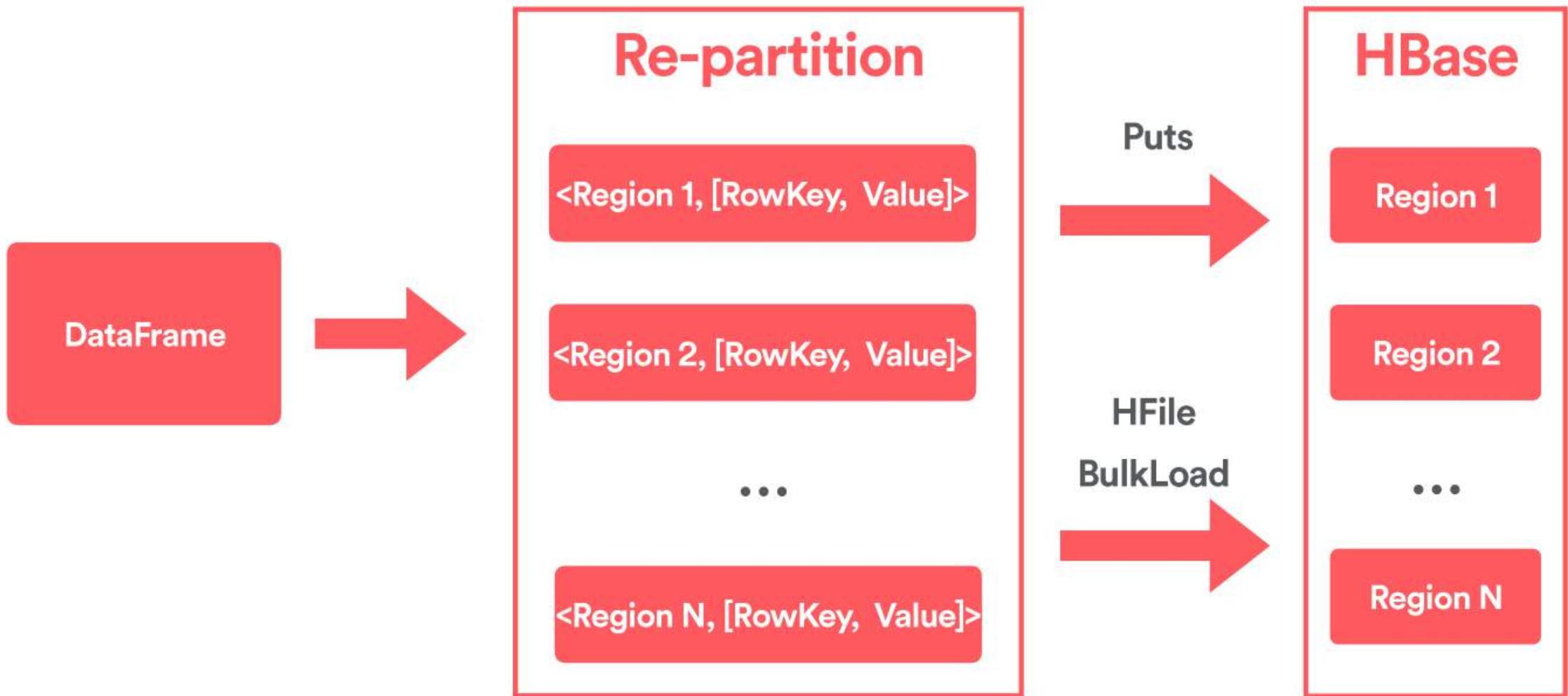
Shared Global State Store



Why HBase

- Well integrated with Hadoop eco system
- Efficient API for streaming writes and bulk uploads
- Rich API for sequential scan and point-lookups
- Merged view based on version

Unified Write API



Rich Read API

Spark Streaming/Batch Jobs

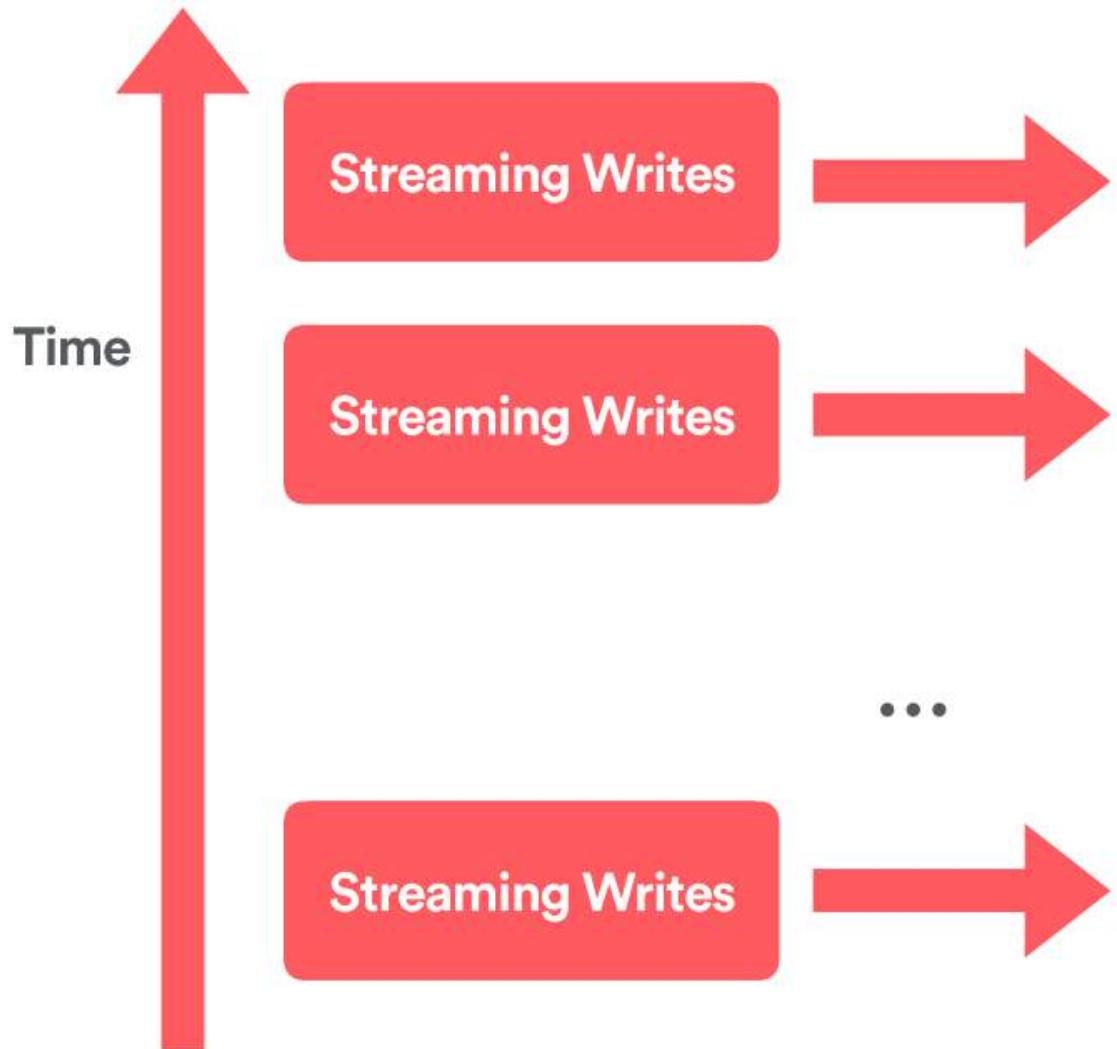
Multi-Gets

Prefix Scan

Time Range Scan

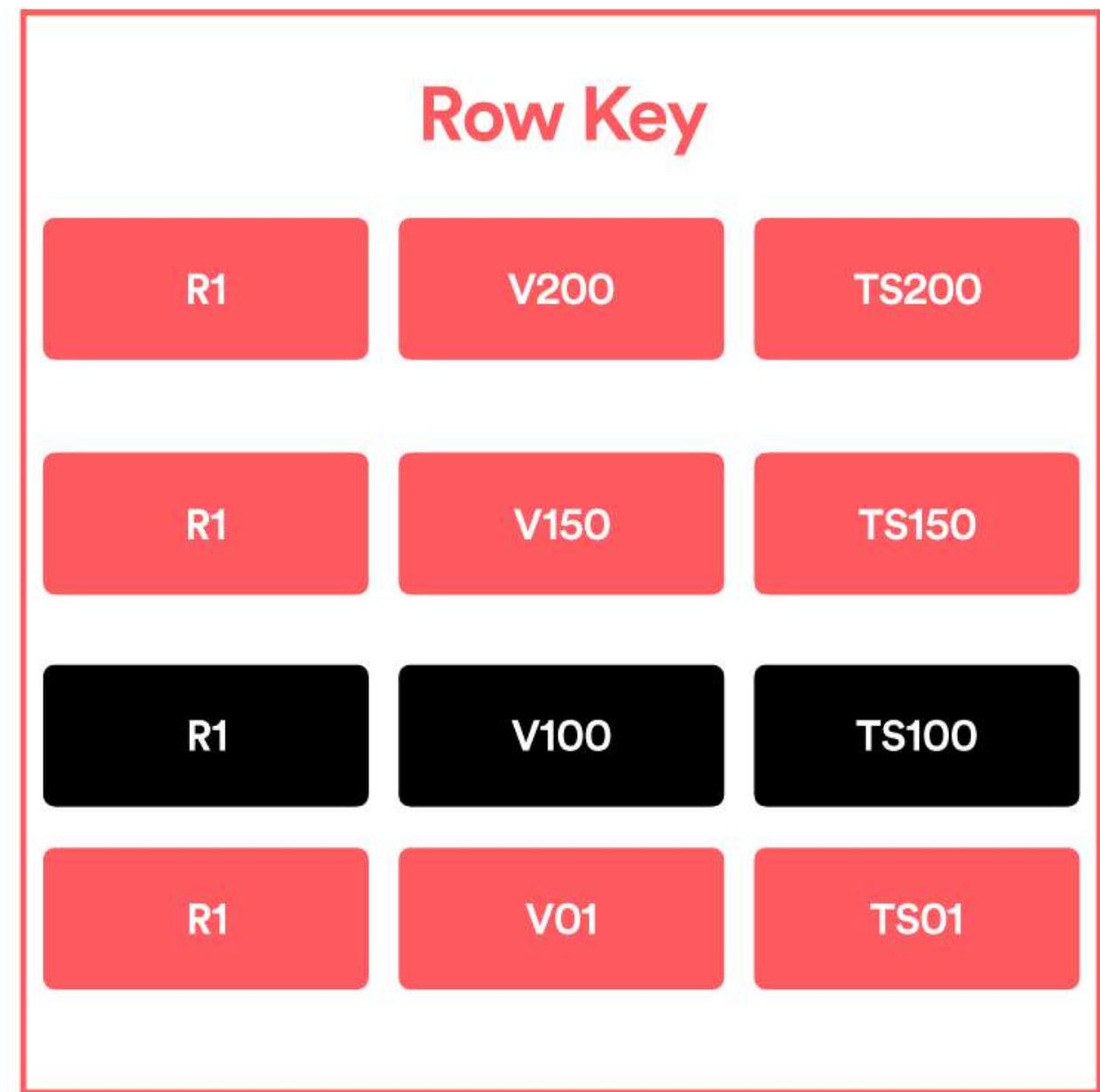
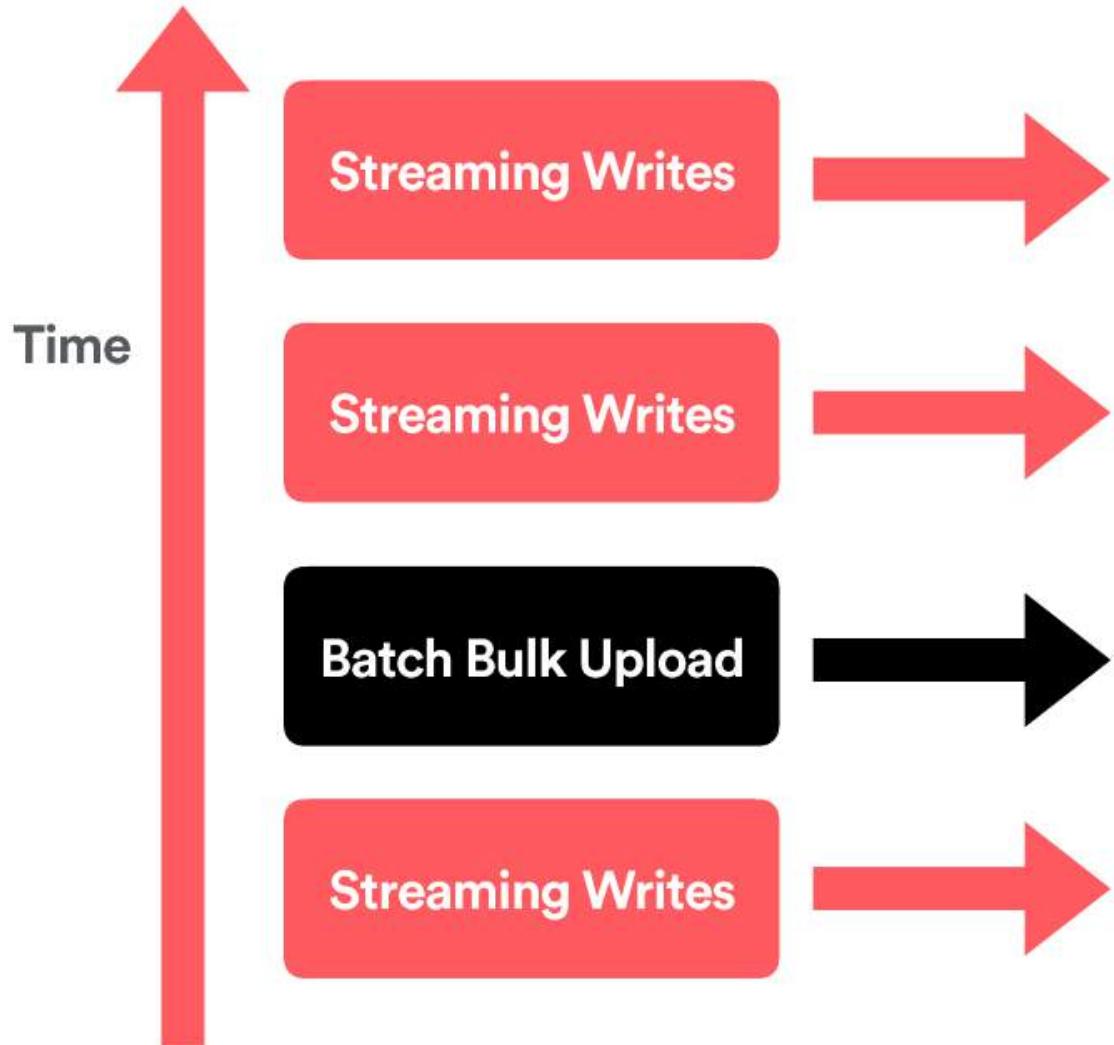
HBase Tables

Merged Views



Row Key		
R1	V200	TS200
R1	V150	TS150
...
R1	V01	TS01

Merged Views



Our Foundations

- Unify streaming and batch process
- Shared global state store

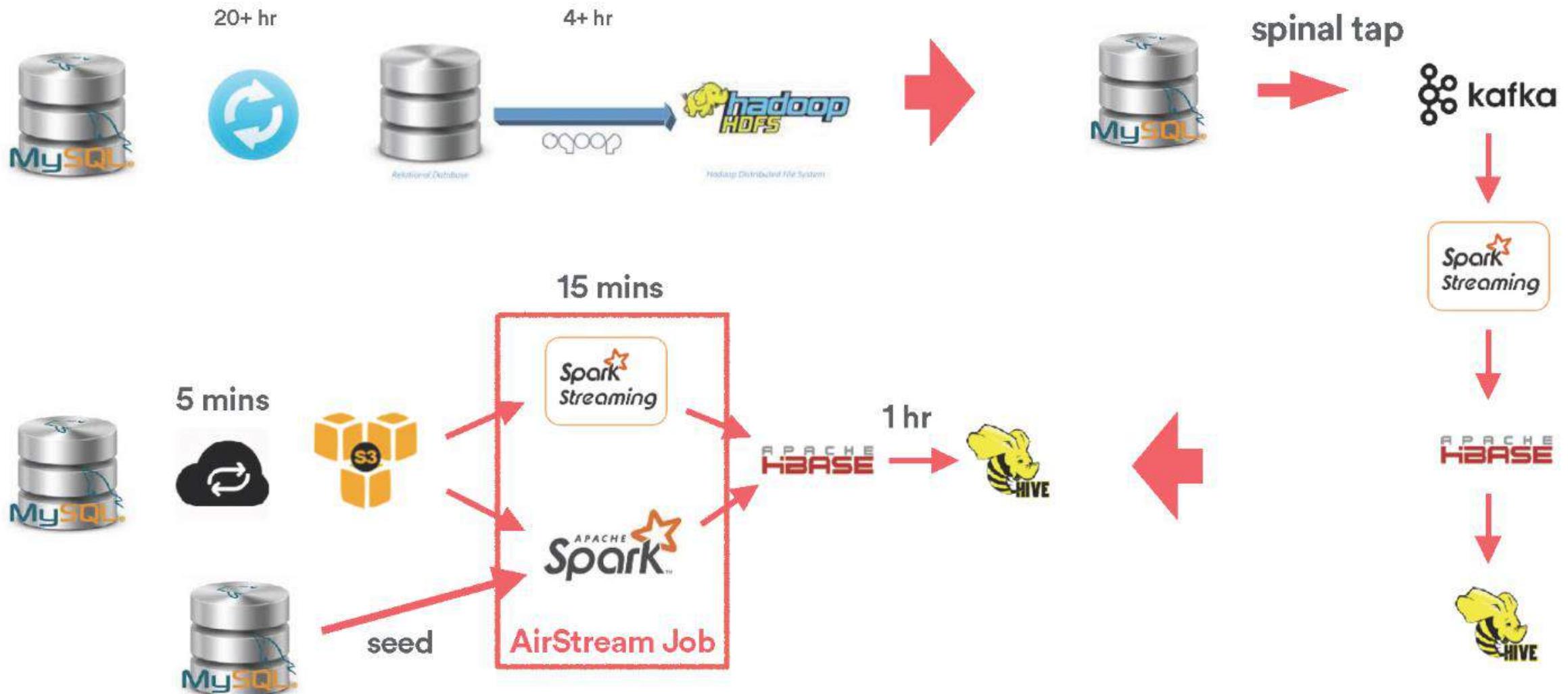
MYSQL DB SNAPSHOT USING BINLOG REPLAY

Move Elephant

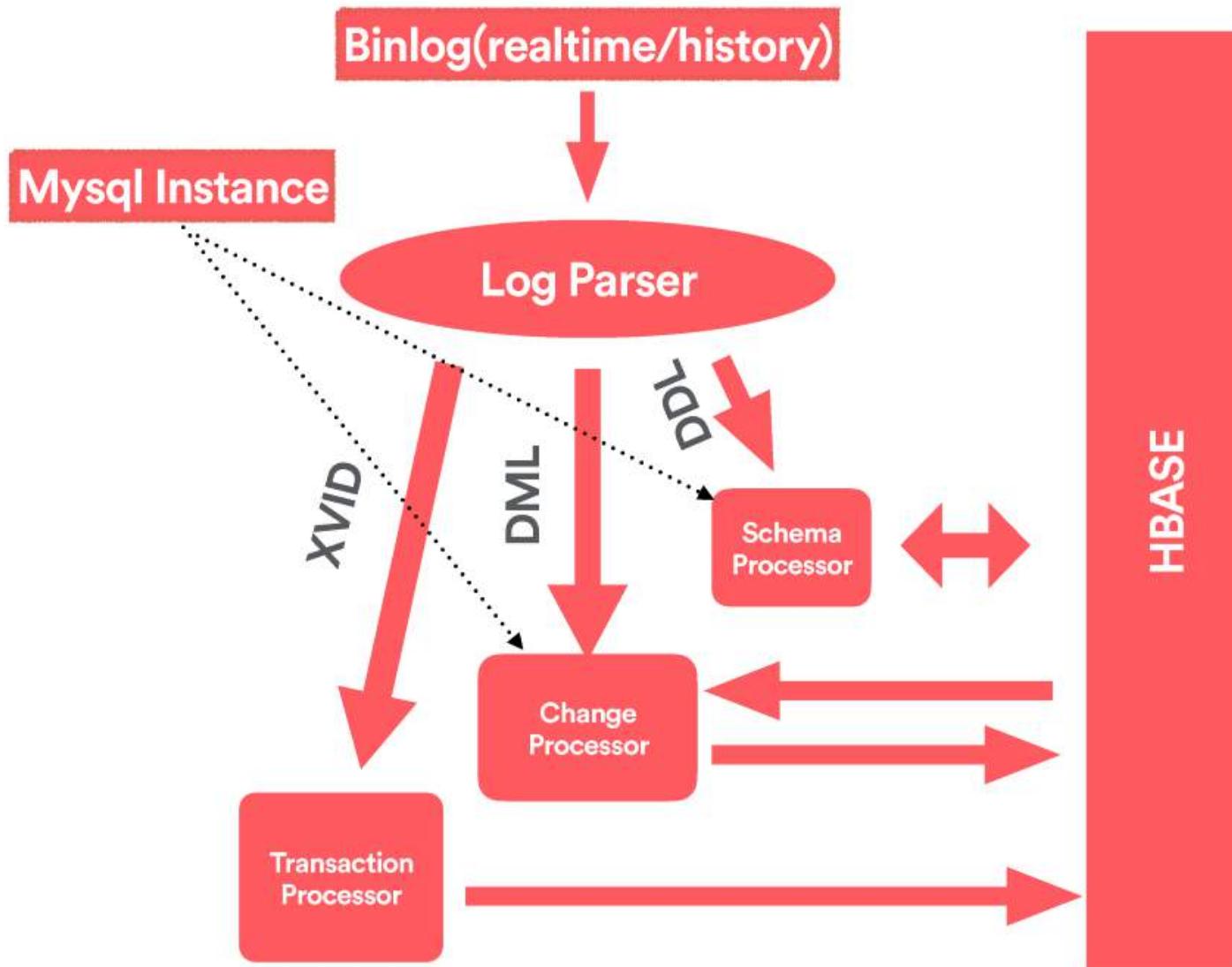
Database Snapshot

- **Large amount of data:** Multiple large mysql DBs
- **Realtime-ness:** minutes delay/ hours delay
- **Transaction :** Need to keep transaction across different tables
- **Schema change:** Table schema evolves

Binlog Replay on Spark



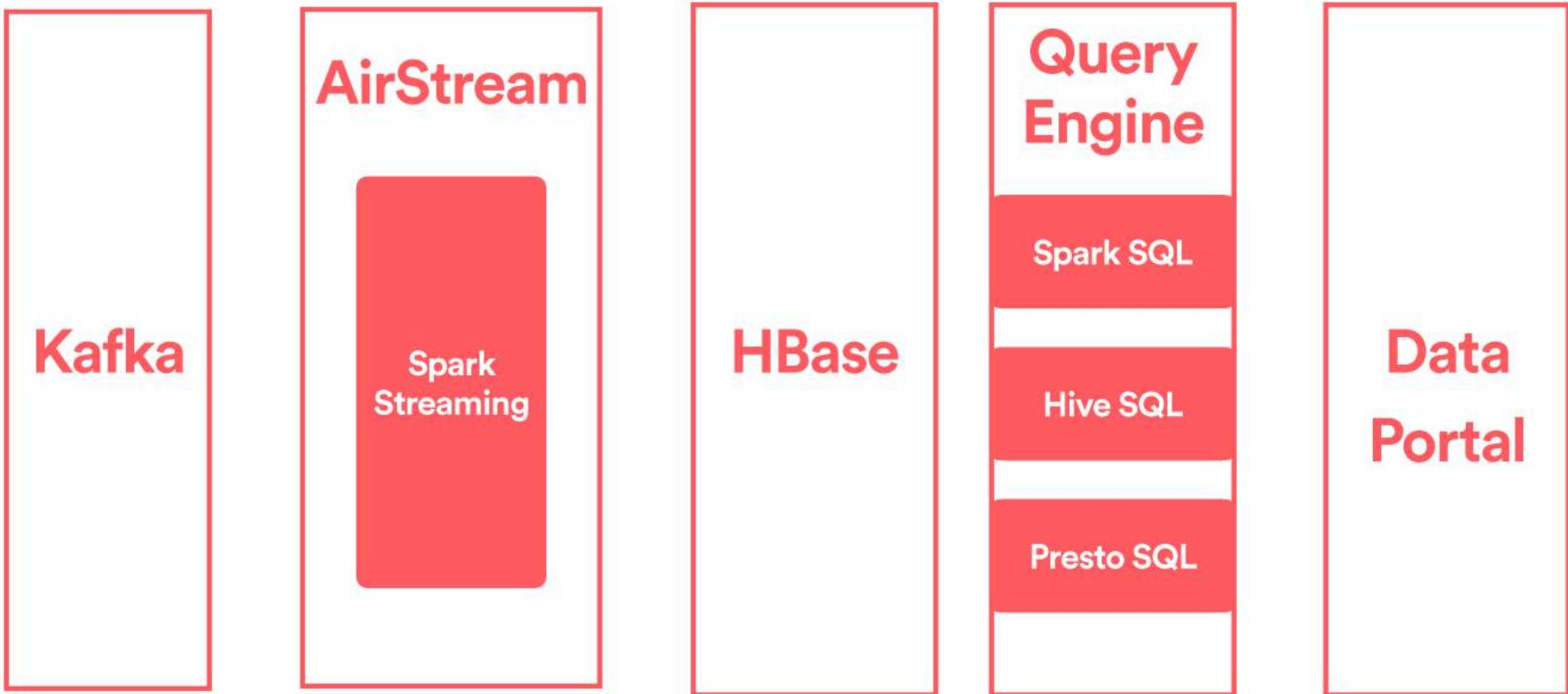
Lambda Architecture



- **Streaming and Batch shares Logic:** Binlog file reader, DDL processor, transaction processor, DML processor.
- **Merged by binlog position:** <filenum, offset>
- **Idempotent:** Log can be replayed multiple times.
- **Schema changes:** Full schema change history.

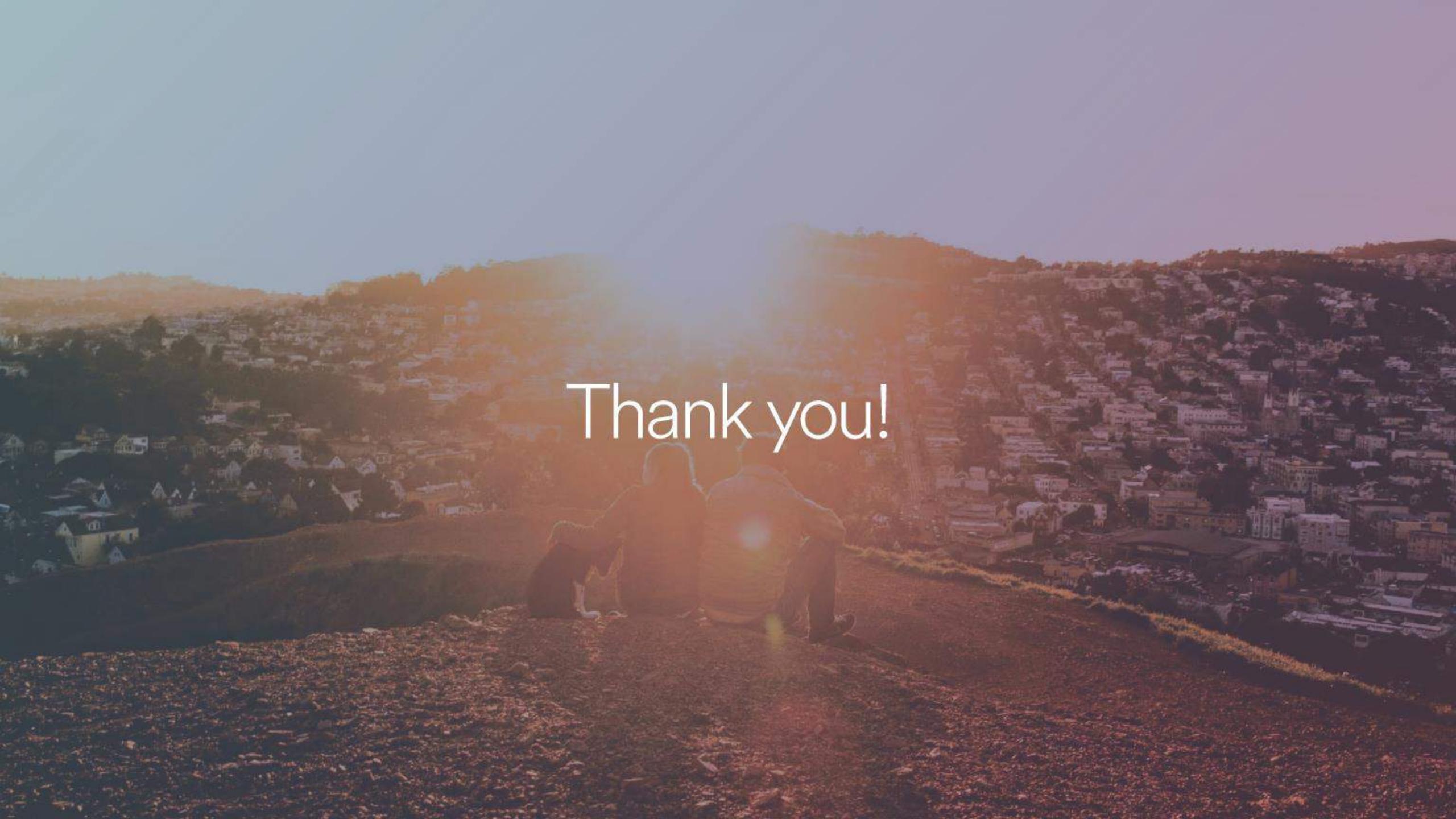
STREAMING INGESTION & REALTIME INTERACTIVE QUERY

Realtime Ingestion and Interactive Query



Interactive Query in SqlLab

The screenshot shows the Superset interface with the 'SQL Lab' tab selected. A new query titled 'Untitled Query 2' is open. On the left, there are dropdown menus for 'Select a database (1)', 'Select a schema (0)', and 'Add a table (0)'. The main area displays a single line of SQL: '1 SELECT ...'. Below the editor, there are two tabs: 'Results' (which is active) and 'Query History'. A teal banner at the bottom encourages users to 'Run a query to display results here'.

A photograph of a couple sitting on a grassy hillside, looking out over a city at sunset. The sun is low on the horizon, creating a bright lens flare and casting a warm glow over the scene. The city below is filled with buildings of various sizes and colors, nestled among hills. The couple is positioned in the foreground, facing away from the camera towards the city.

Thank you!



Apache Kylin v2

从OLAP引擎到实时数据仓库的演进

史少锋

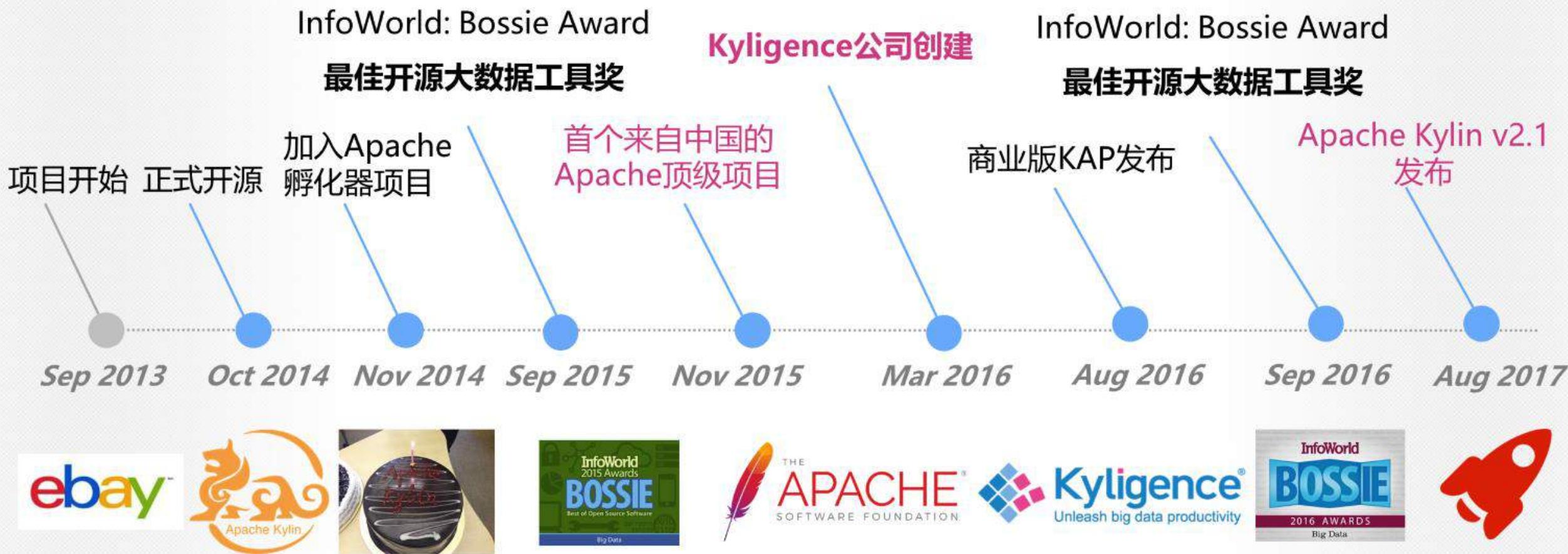
Apache Kylin committer & PMC

Kyligence 技术合伙人 & 高级架构师

Apache Kylin是什么？



Apache Kylin 历史

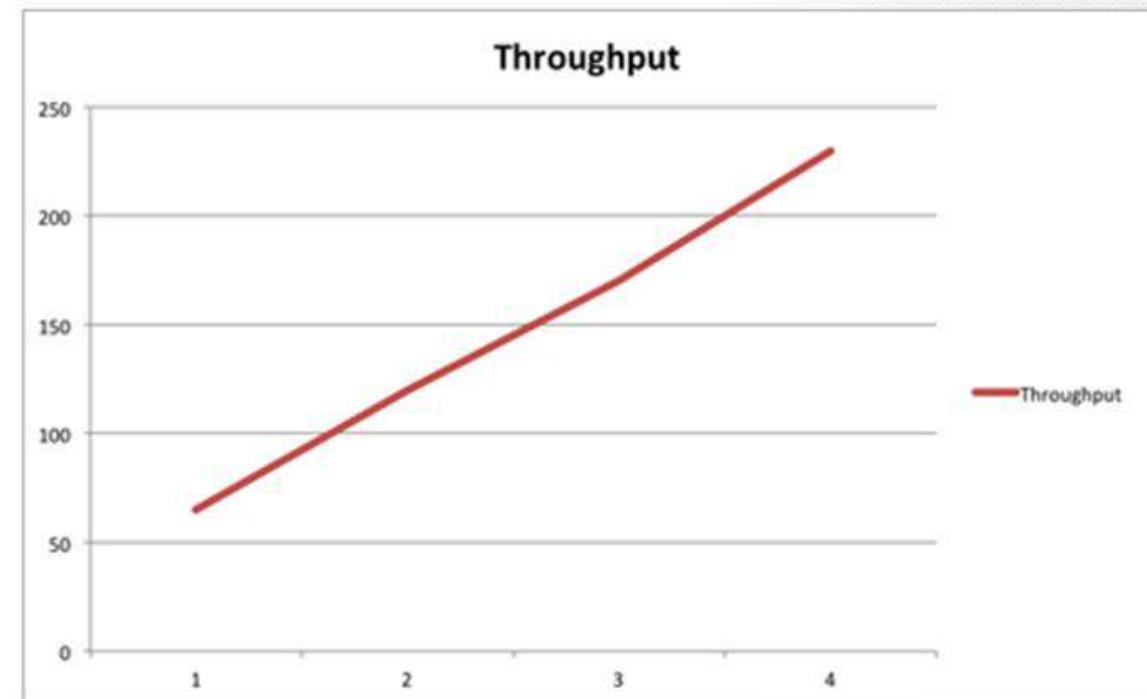
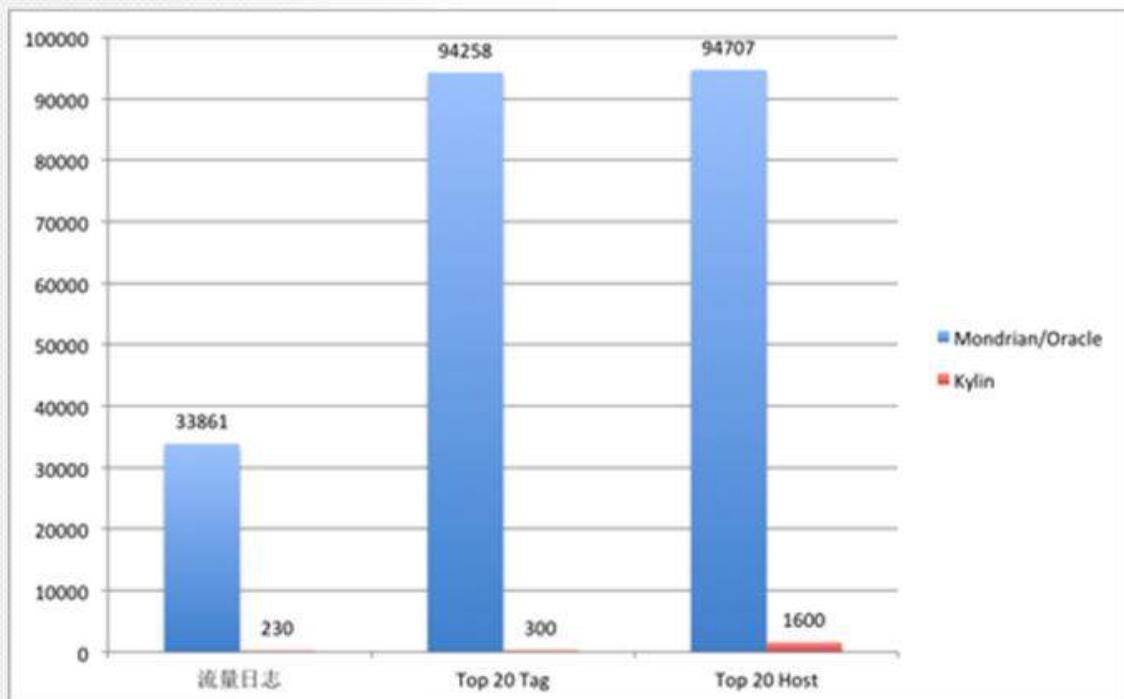


Apache Kylin全球案例



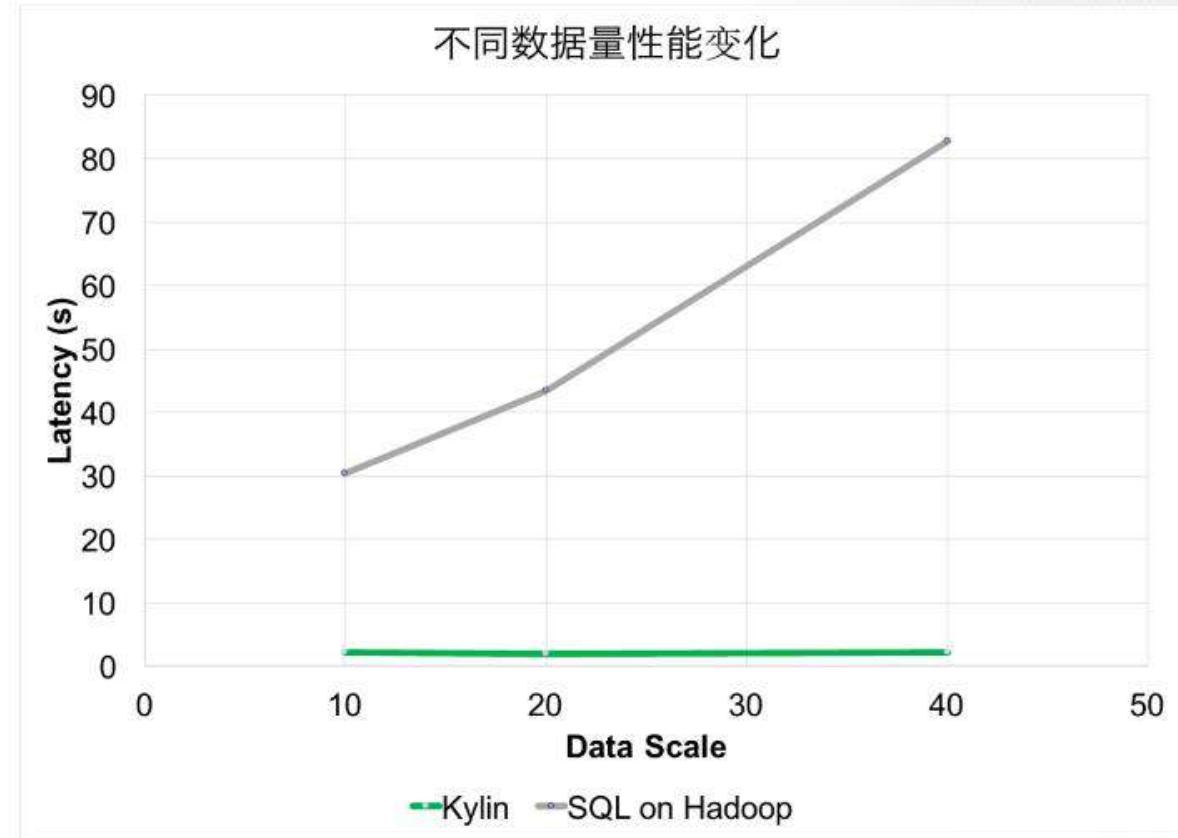
全球 500+ 用户部署到生产环境

Apache Kylin性能测试



By 网易:
<http://datalab.int-yt.com/archives/1708>

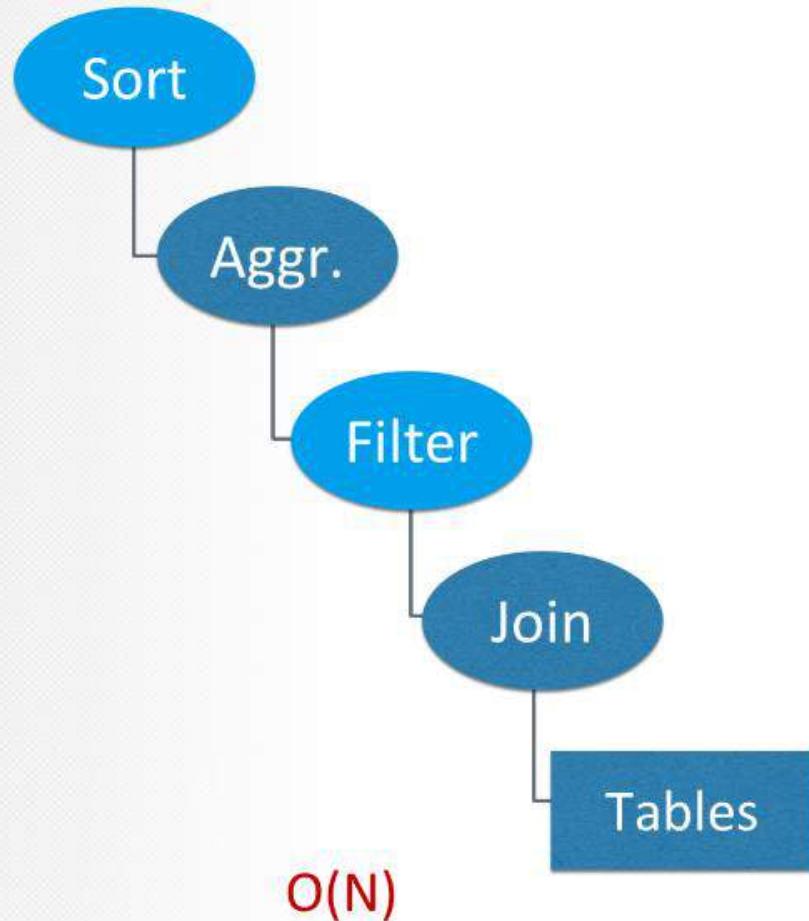
Apache Kylin vs. SQL-on-Hadoop



Star schema benchmark:

<http://www.cs.umb.edu/~poneil/StarSchemaB.PDF>

Apache Kylin 为什么快？

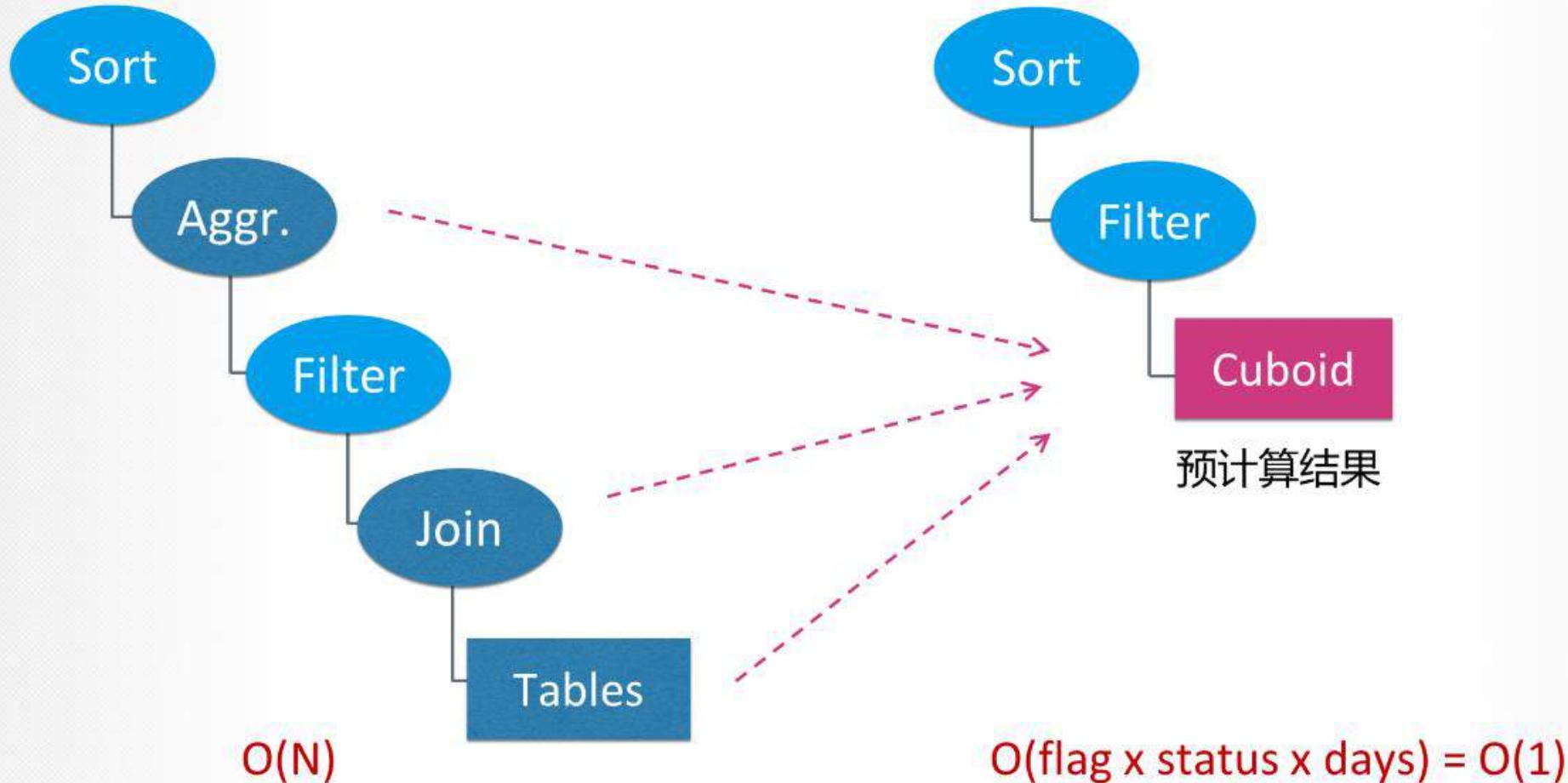


样例：

分析一段时间内，不同“returnflag”和“orderstatus”对应的销售情况

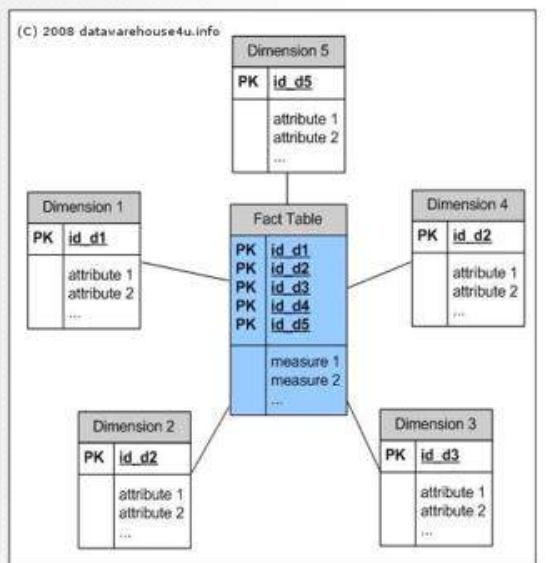
```
select  
    l_returnflag,  
    o_orderstatus,  
    sum(l_quantity) as sum_qty,  
    sum(l_extendedprice) as sum_base_price  
    ...  
from  
    v_lineitem  
    inner join v_orders on l_orderkey = o_orderkey  
where  
    l_shipdate <= '1998-09-16'  
group by  
    l_returnflag,  
    o_orderstatus  
order by  
    l_returnflag,  
    o_orderstatus;
```

Apache Kylin 为什么快？

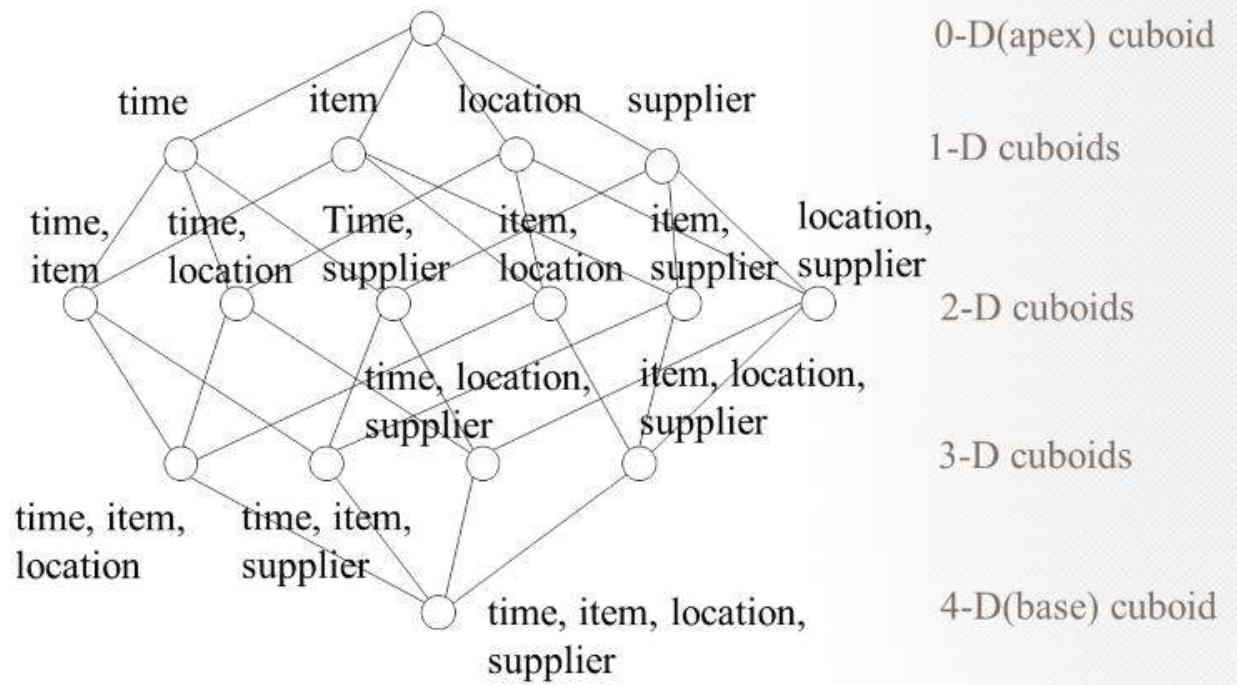


Apache Kylin 关键在于预计算

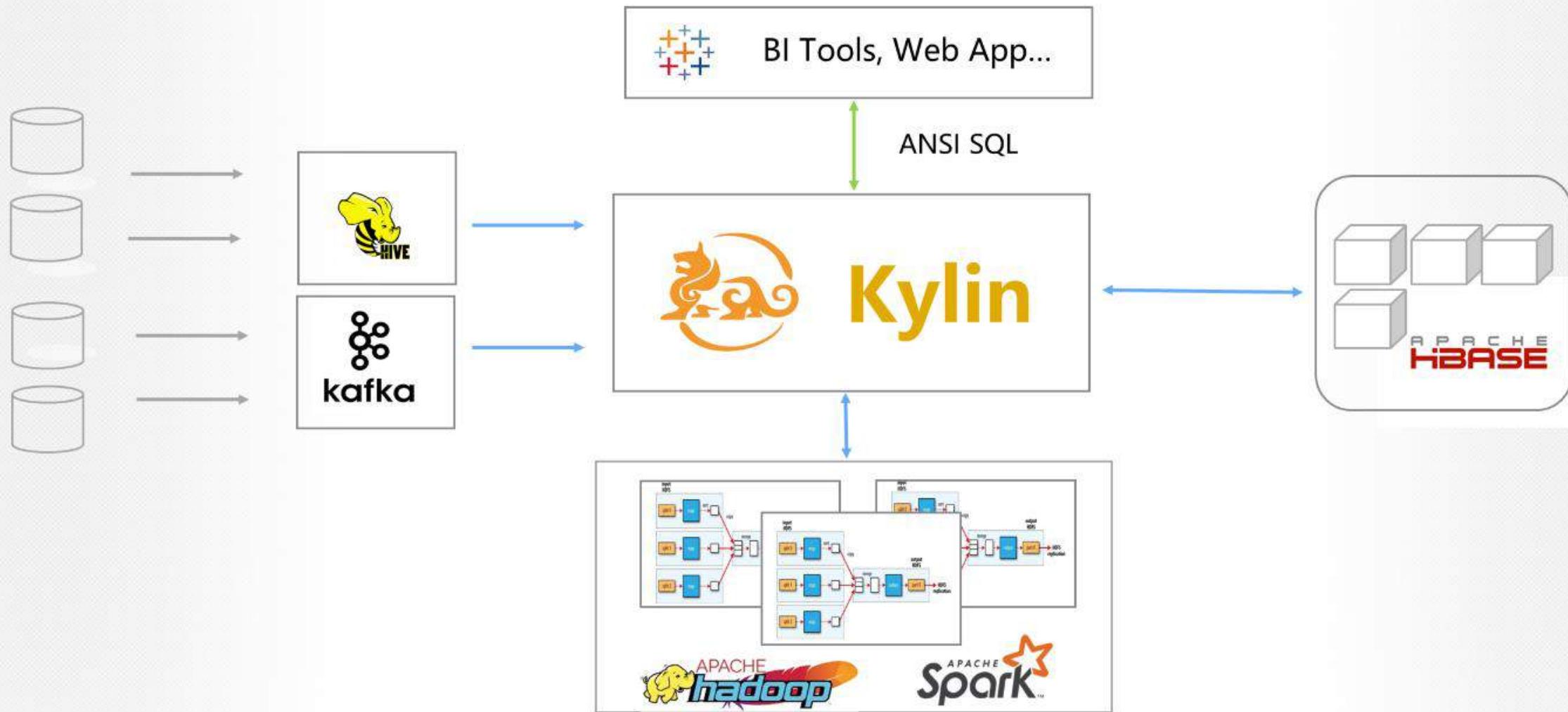
- OLAP Cube 理论基础
- Model 和 Cube 定义预计算范围
- Build Engine 执行预计算任务
- Query Engine 在预计算结果上完成查询



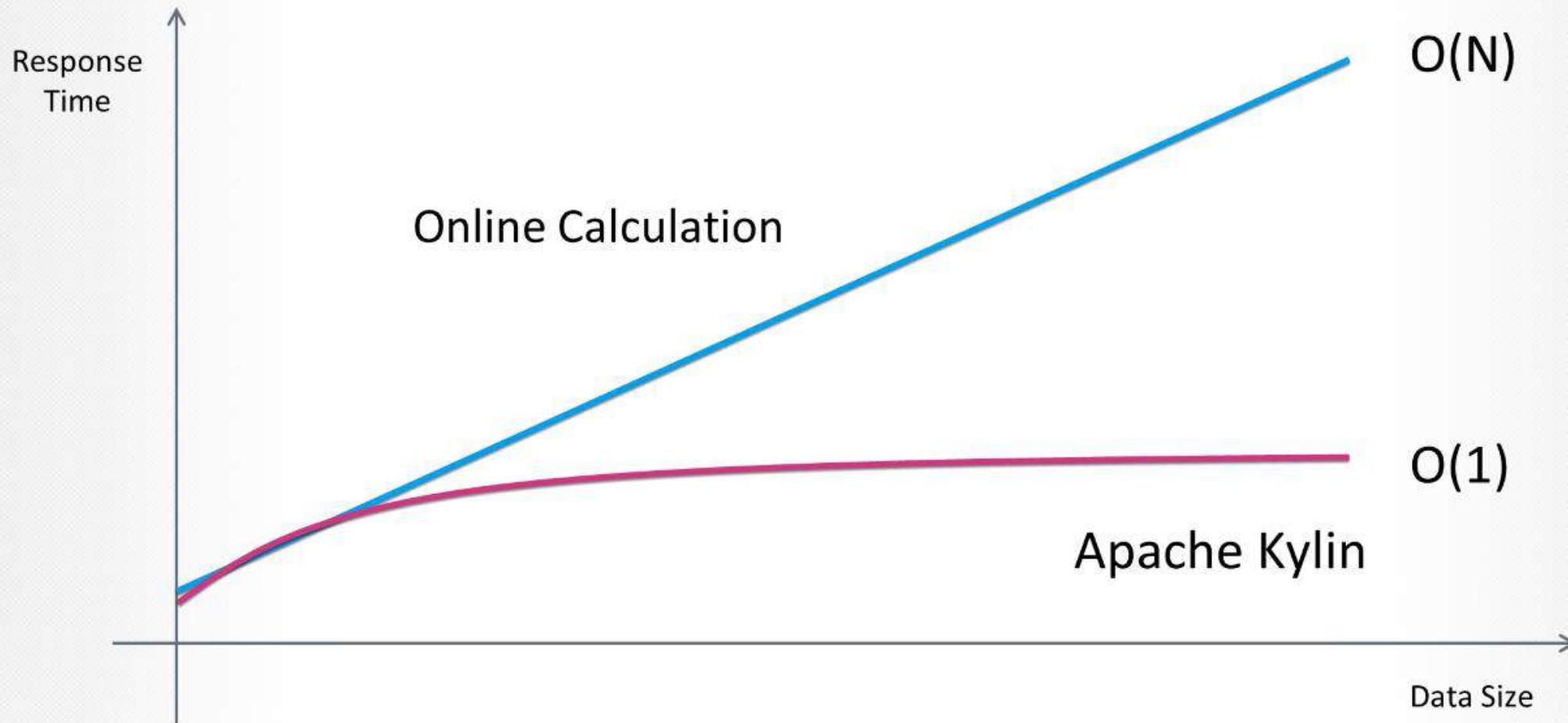
预计算



Apache Kylin 系统架构



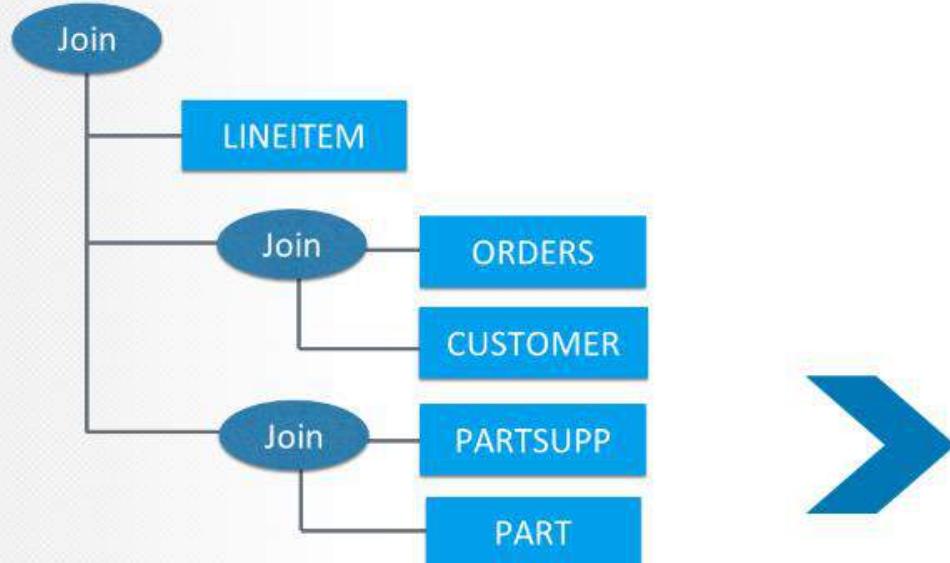
加速大数据OLAP分析



支持雪花模型

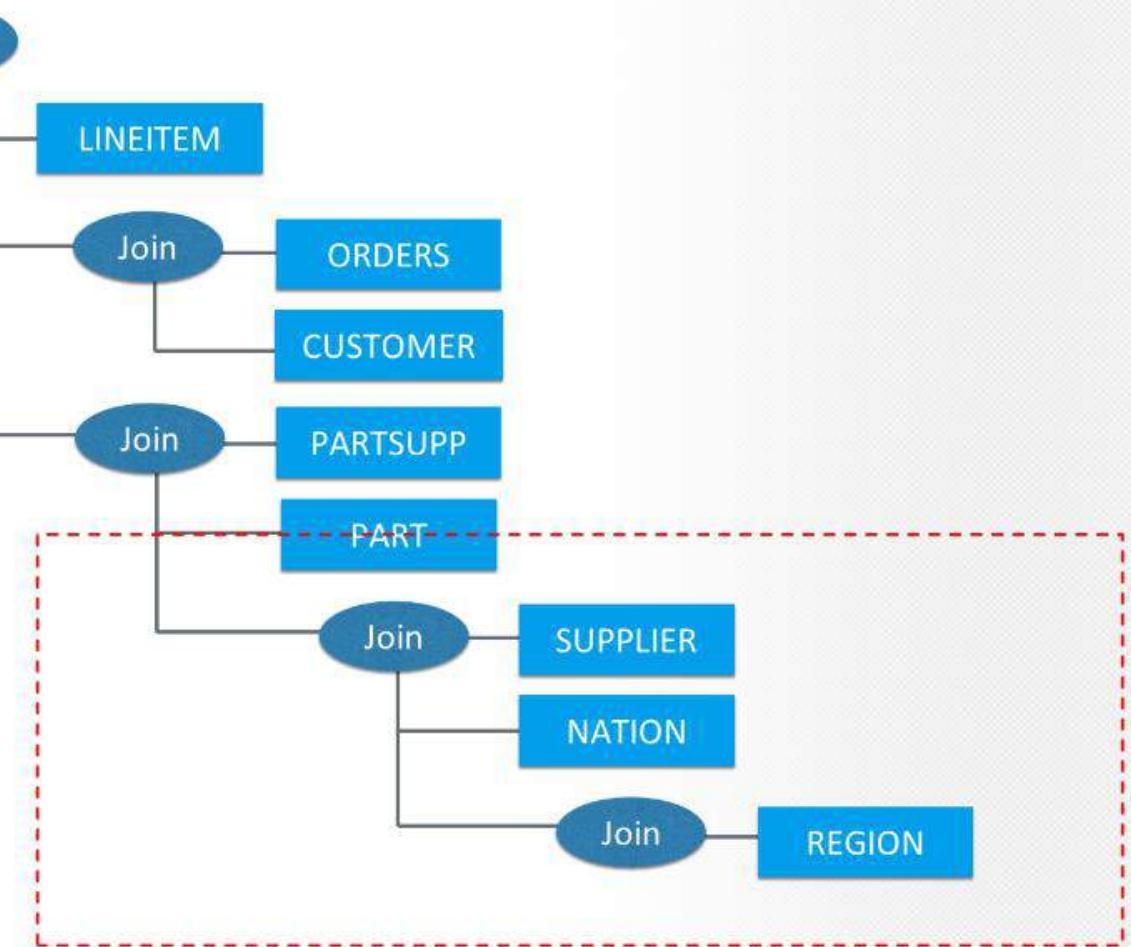
TPC-H Benchmarking

Kylin v2 支持雪花模型



解决了Kylin 1.0很多功能限制：

- 从星形模型到雪花模型
- 单表重复Join
-



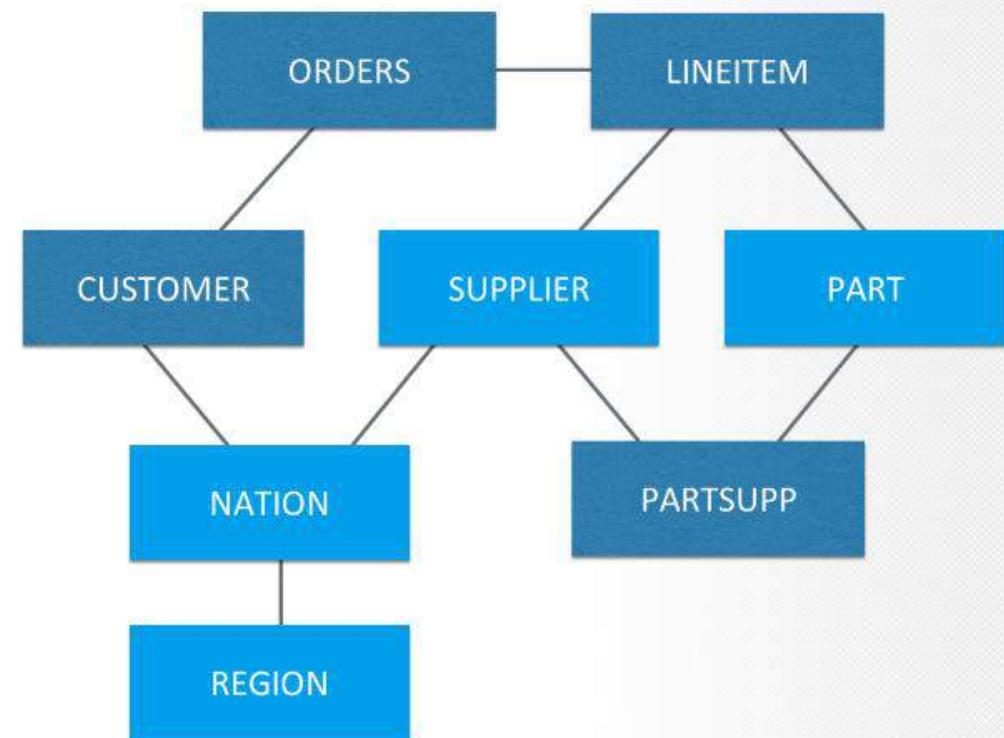
TPC-H Benchmarking

TPC-H is a benchmark for decision support system.

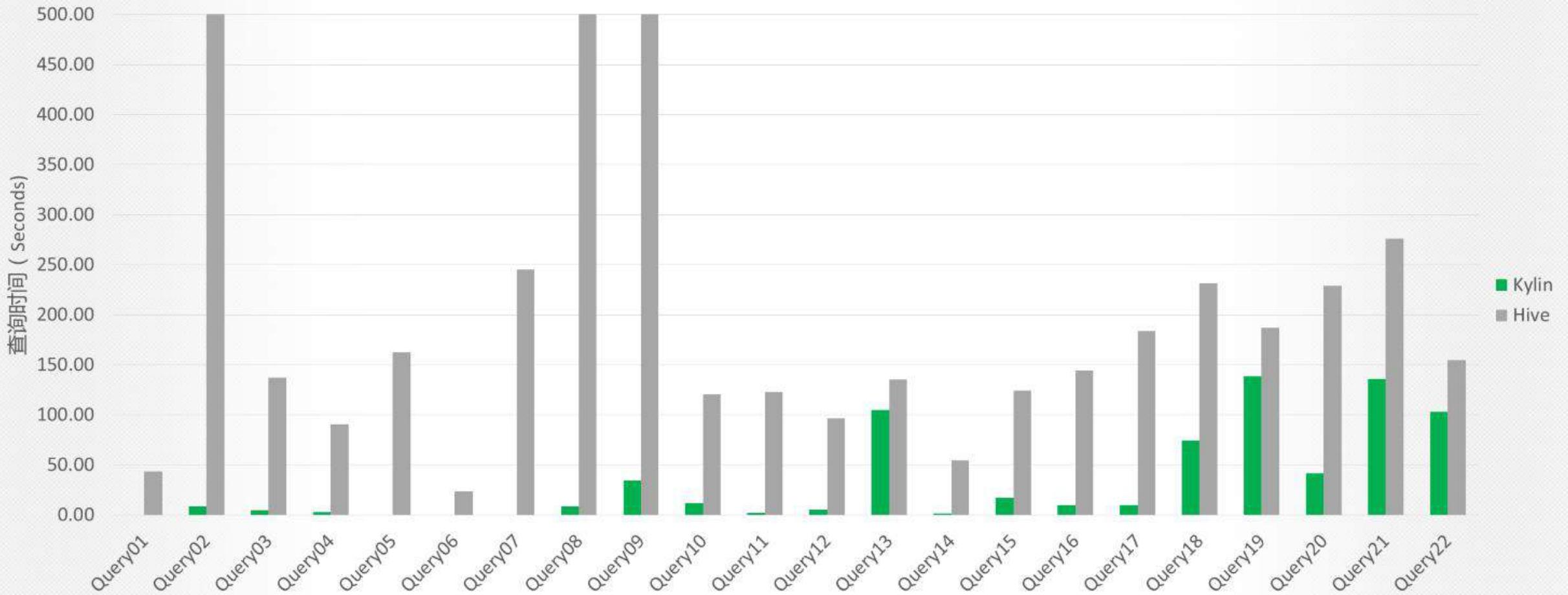
- Popular among commercial RDBMS & DW solutions
- Queries and data have broad industry-wide relevance
- Examine large volumes of data
- Execute queries with a high degree of complexity
- Give answers to critical business questions

Kylin 2.0 runs all the 22 TPC-H queries. ([KYLIN-2467](#))

- Pre-calculation can answer very complex queries
- Goal is functionality at this stage
- Try it: <https://github.com/Kyligence/kylin-tpch>

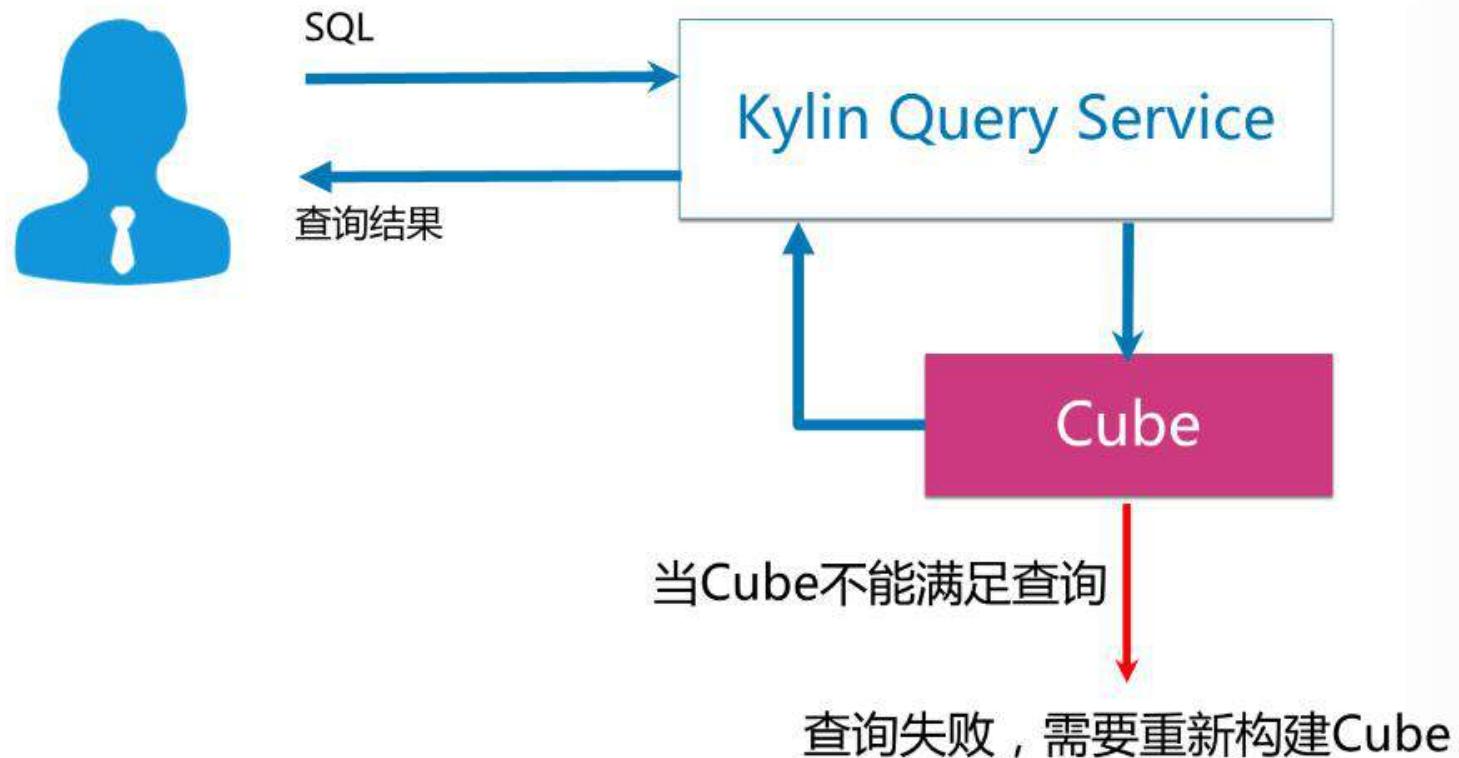


Kylin vs Hive

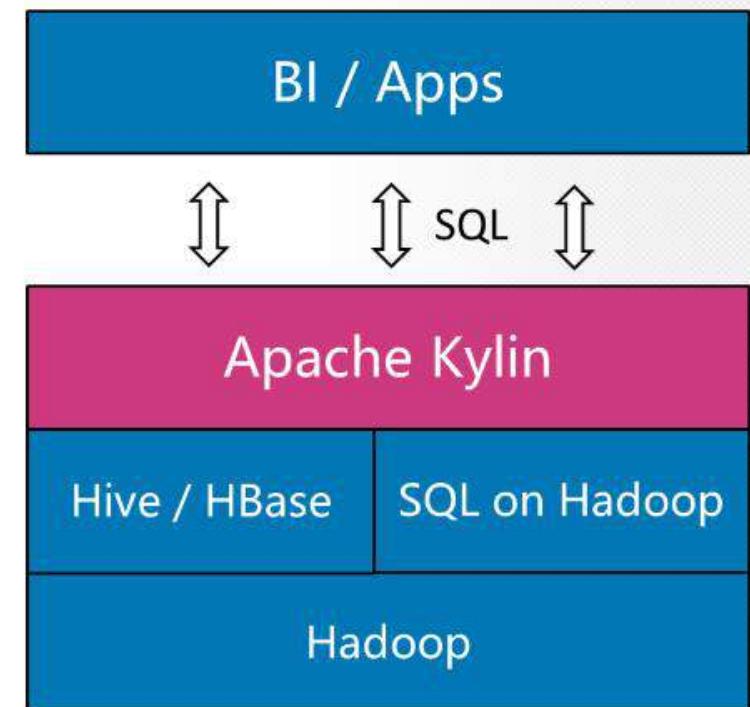
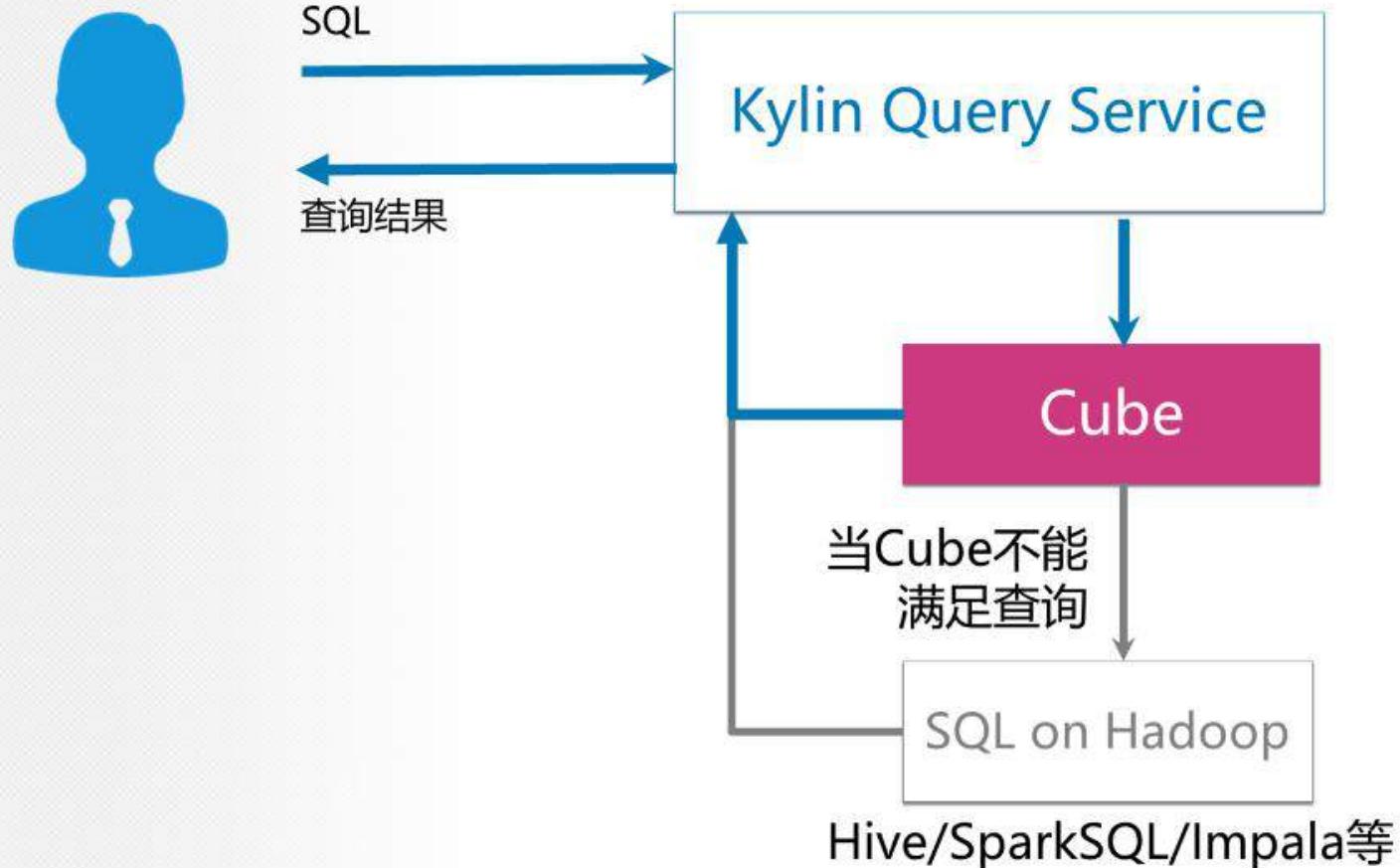


SQL Pushdown

v1.0 - 当Cube不能满足查询



v2.1 - 当Cube不能满足查询



Can be cooler ?

云端一键部署



Azure HDInsight



AWS EMR

Kyligence Cloud !

KAP on HDInsight

Search by app name

Offer details
Kyligence Analytics Platform 2.3
Kyligence
Terms of use | privacy policy

Terms of use

StreamSets Data Collector
STREAMSETS

Unavailable applications

Kyligence Cloud Beta

AWS GLOBAL 中文

Dashboard / cluster

Name	Status	Create Time	Action
testkap24	CREATING	2017-08-29 13:54:13	Start
ssb	RUNNING	2017-09-21 18:35:10	Start

Basic Info

Cluster Name: ssb Status: RUNNING

Cluster Type: EMR Create Time: 2017-09-21 18:35:10

Topology: SINGLE

Cluster START STOP RESIZE

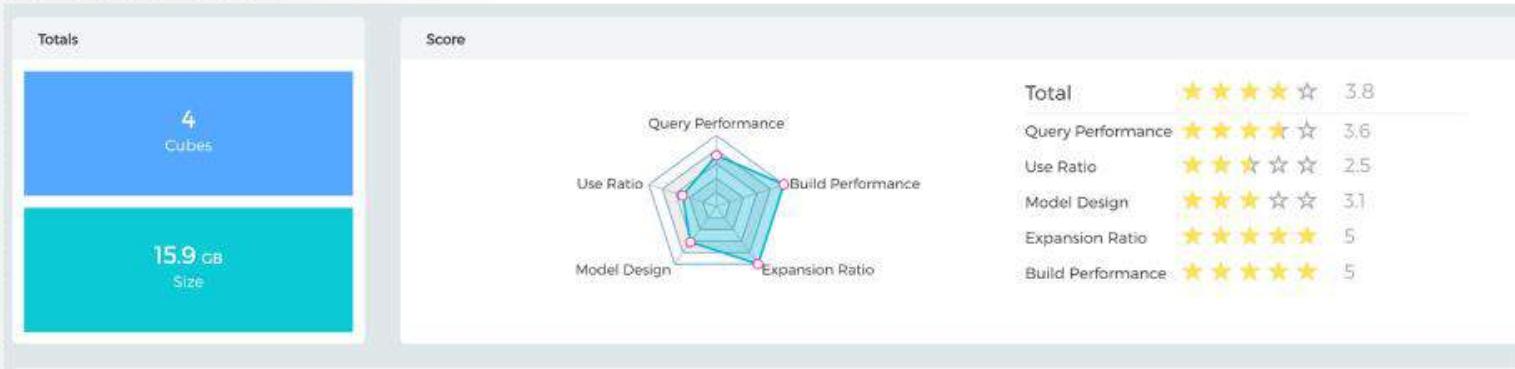
Cluster Type: Status: RUNNING

Work Node Count: 1 Create Time: 2017-09-21 18:35:10

KAP

Total 2 / 10 Go to 1

自助式 Cube 调优



- 统计评分
- 优化建议



Kyligence Robot !

<https://kybot.io>

Rowkey

Suggestion	Origin	Appear	Absent	difference with parent cuboid
Basic Information				
Rowkey	Cardinality	Column Type	Encoding	
CATEGORY	1007	varchar(256)	dict	
COUNTRY_CODE	9	integer	integer4	
PROVINCE_CODE	8	integer	integer4	
CITY	6	varchar(256)	fixed_length:100 dict	
POSTCODE	6	varchar(256)	fixed_length:120 dict	
AREA	5	varchar(256)	fixed_length:100 dict	
GENDER	2	varchar(256)	fixed_length:40 dict	
IS_VALID	2	varchar(256)	fixed_length:40 dict	
IS_ONLINE	2	varchar(256)	dict	
IS_ACTIVITY	2	varchar(256)	dict	

自助式查询优化



关于Kyligence



Apache Kylin v2.2 is coming!