Pivotal

现代化数据微服务架构实践

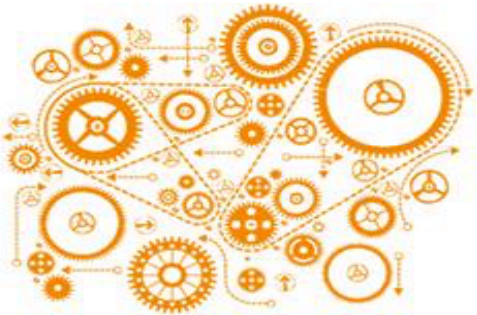
Hogon Zhuang 庄怀轩
Solution Architect
Oct 2018

架构的演变轨迹

1990s and earlier | **2000s** | **2010s**

Coupling

Pre-SOA (monolithic)
Tight coupling



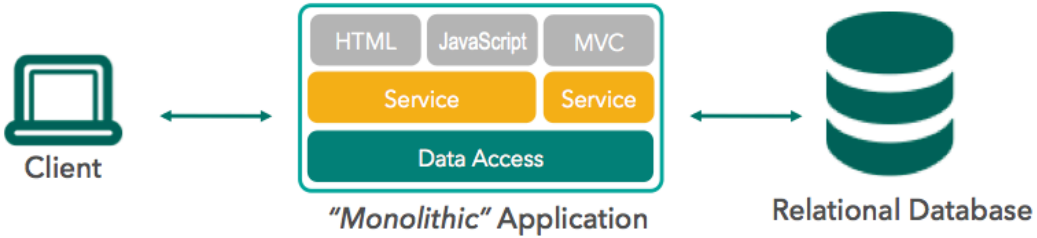
Traditional SOA
Looser coupling

Microservices
Decoupled

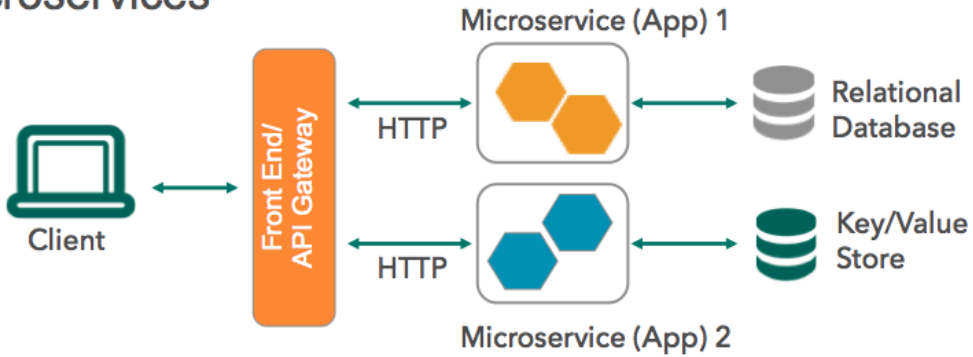
架构对数据服务的影响

微服务架构下数据服务也必须变革

- Classic *three-tier* application

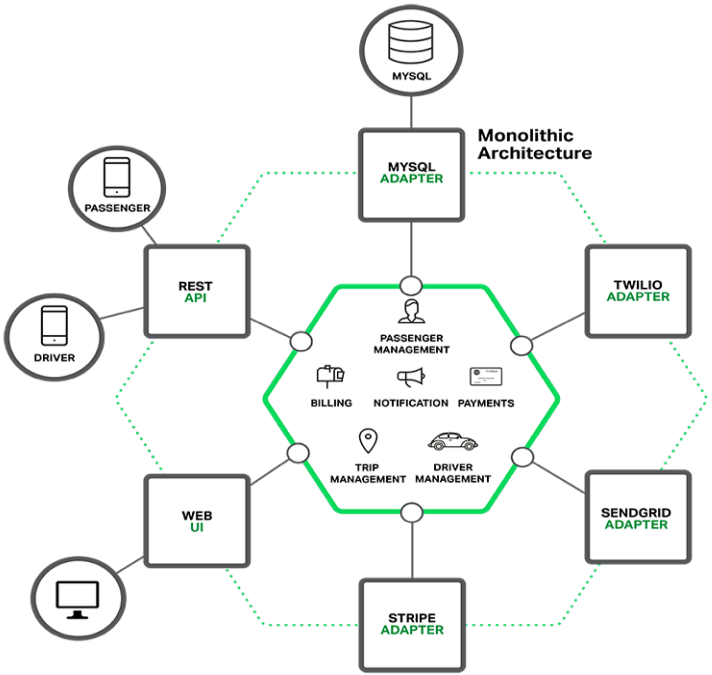


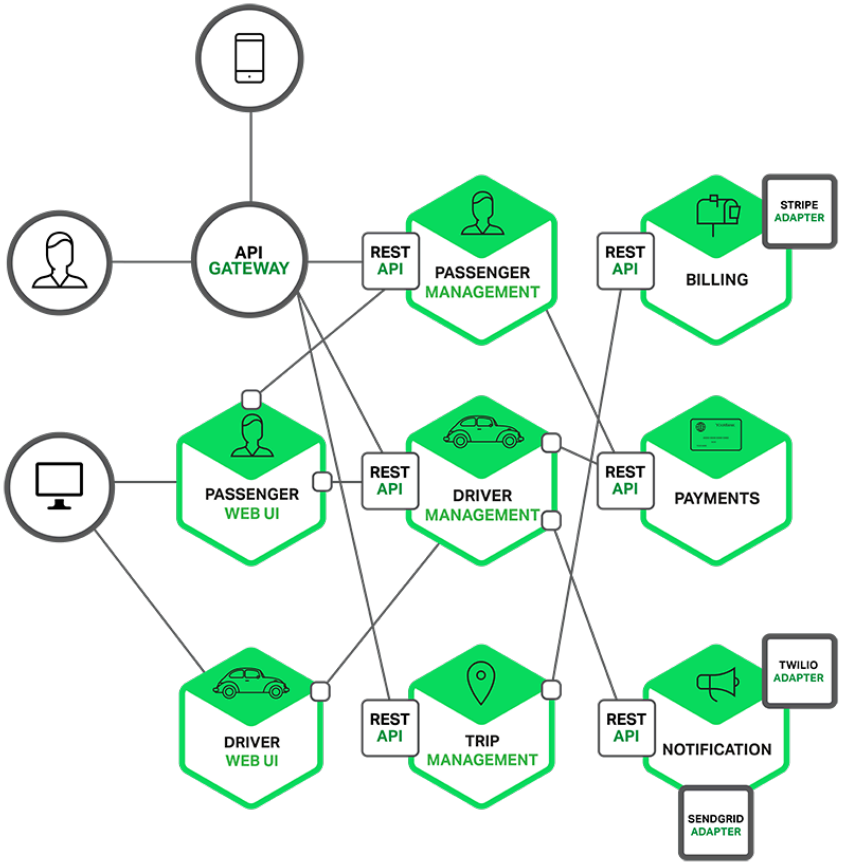
- Microservices



Pivotal

改造实例


MYSQL
Monolithic Architecture
MYSQL ADAPTER
PASSENGER
DRIVER
REST API
TWILIO ADAPTER
PASSENGER MANAGEMENT
BILLING
NOTIFICATION
PAYMENTS
TRIP MANAGEMENT
DRIVER MANAGEMENT
WEB UI
SENDGRID ADAPTER
STRIPE ADAPTER


API GATEWAY
REST API
PASSENGER MANAGEMENT
BILLING
STRIPE ADAPTER
PASSENGER WEB UI
DRIVER MANAGEMENT
PAYMENTS
DRIVER WEB UI
TRIP MANAGEMENT
NOTIFICATION
TWILIO ADAPTER
SENDGRID ADAPTER

两种架构的对比

巨石架构

优点:

- 使用单一的war/jar包装所有的组件，包括业务逻辑，数据链接等 （all-in-one）
- 易于部署
- 易于水平复制

需要注意:

- 高复杂度降低了敏捷性:

--维护困难

--持续部署困难

--发布周期很长

- 非云原生架构:

--不满足12-要素 要求

--没有数据伸缩性

--没有功能伸缩性

微服务架构

优点：

- 易于部署
- 投产时间短
- 使用单一三维快速伸缩（**akf模型**）
- 简单复用
- 云原生架构

需要注意：

- 过分原子化
- 高压延迟
- 完整测试复杂

新的挑战需要新的特性

挑战：

- 无状态应用的离散化数据存储
- 不同的微服务之间实现隔离（一服一库）
- 规模化分布式数据访问
- 服务间通信分布式事务
- 持续集成快速迭代

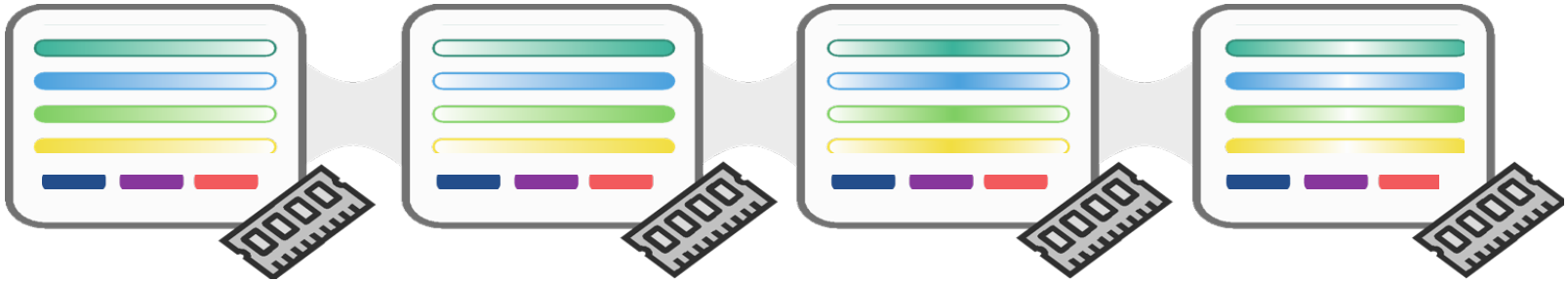
特性：

- 去中心数据治理（Decentralized）
- 故障隔离与数据一致性（集群内与集群间）
- 通过提升个体微服务实例的性能提升整体性能
- 可以随着实例数目变化快速伸缩
- 事件驱动机制下的消息流
- 混合持久化多模数据库
- 支持动态模式

内存数据网格 Pivotal GemFire
In-Memory Data Grid Powered by Apache Geode

- Key-value 内存快速存取
- 原生特有序列化支持多态多接口
- 弹性可伸缩架构
  - 在集群内多个服务器间实现数据分区
  - 在服务实例增减时可以动态重新均衡分布
  - MapReduce风格处理: 将代码发送到分区中以实现并行计算
  - 通过数据复制实现灾备和高速读取

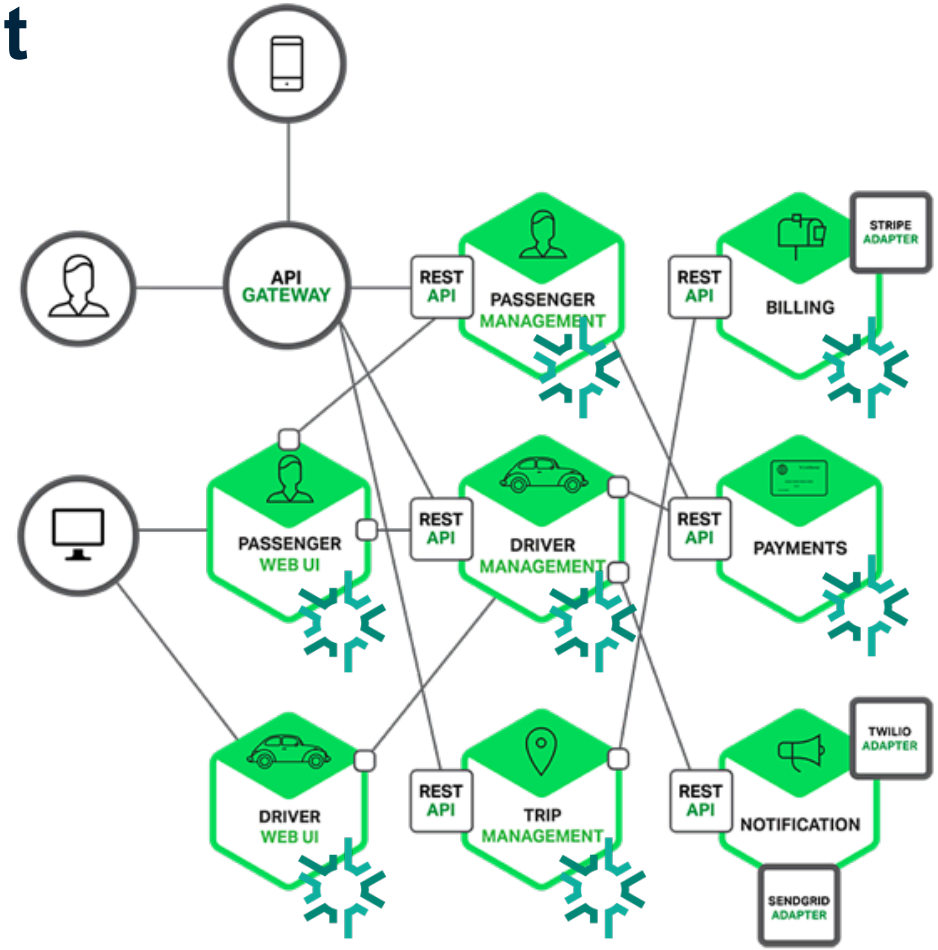
Key feature



- Key/Value 对象存储
- 水平热伸缩和弹性
- 基于内存
- 多活数据中心部署

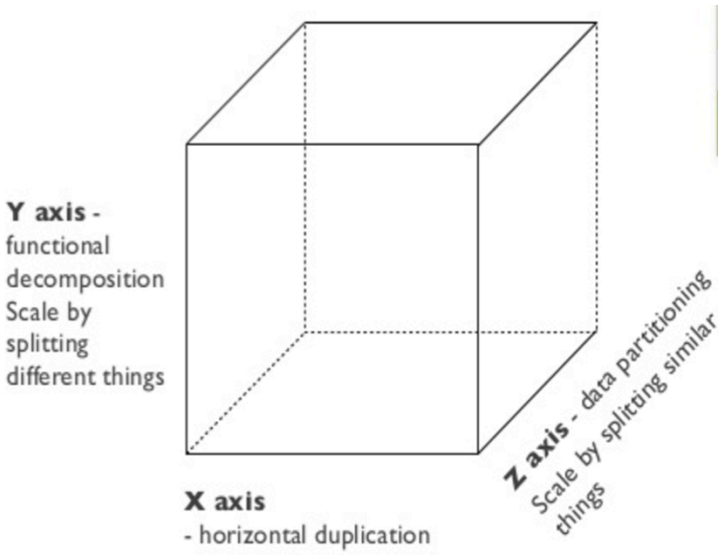
Pivotal

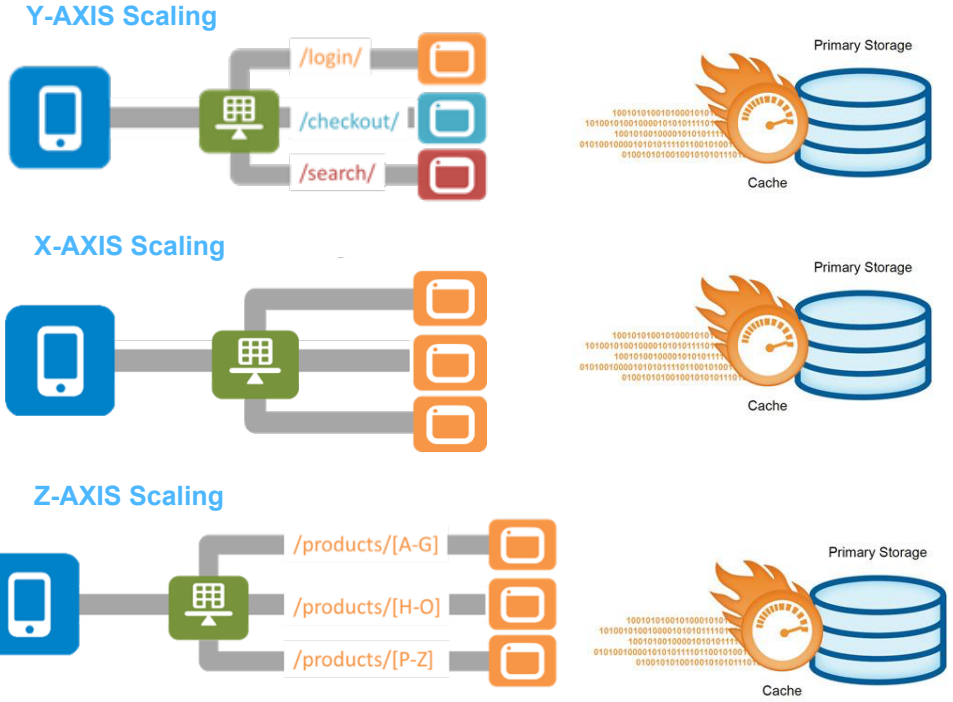
Where to use it



Pivotal

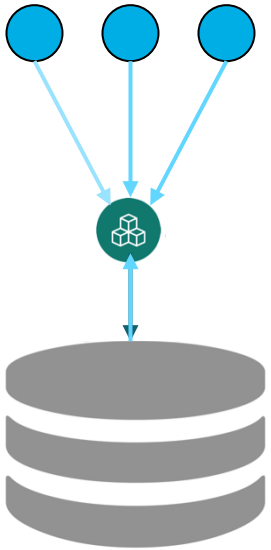
AKF模型（架构即未来，the art of scalability)





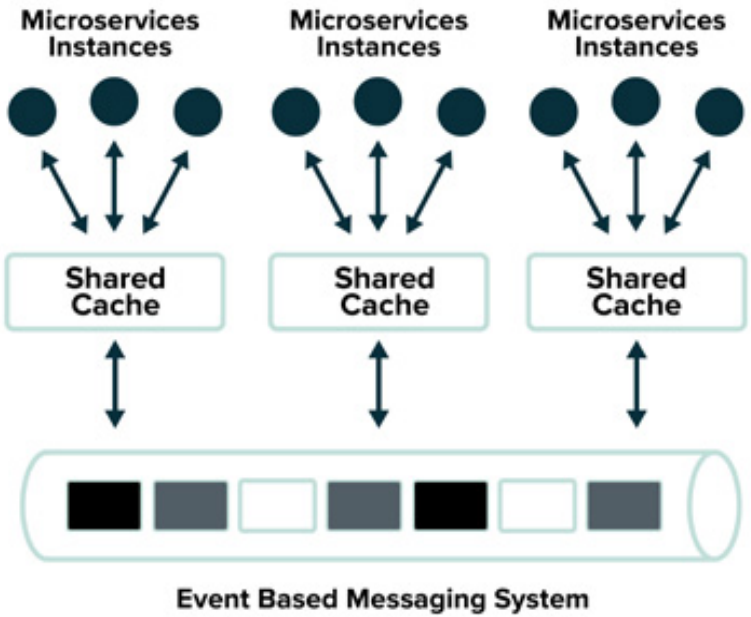
通过微服务克隆提升性能和扩展性

- 通过增加实例扩展个体微服务
- 同时提供高可用性
- 通过一个包括多个相同镜像的单体集群利于实践操作
  - 遵从不同微服务之间的隔离需求
  - 降低了扩展后台存储的压力
  - 在一个集群存储所有的用户数据，提供给所有的实例使用
  - 无须在具体存储设备之间或者特定用户工作负载之间复制数据



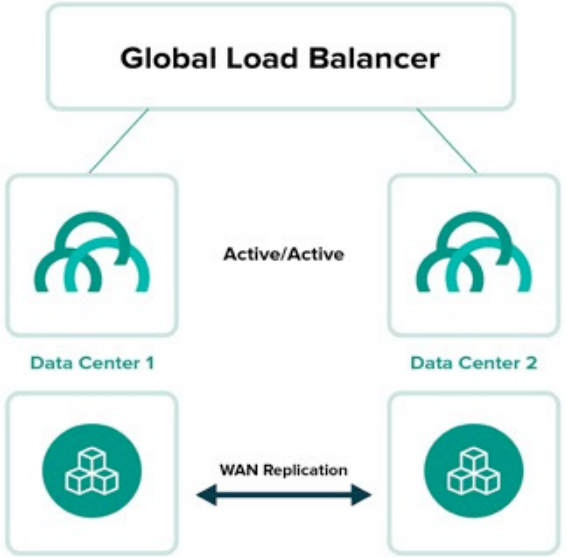
在同一个微服务实例间共享事件

- 可以在镜像间使用产品原生的事件功能
  - 基于内容的订阅分发,或者基于主键的特定查询
  - 客户端可编程的监听器/处理器



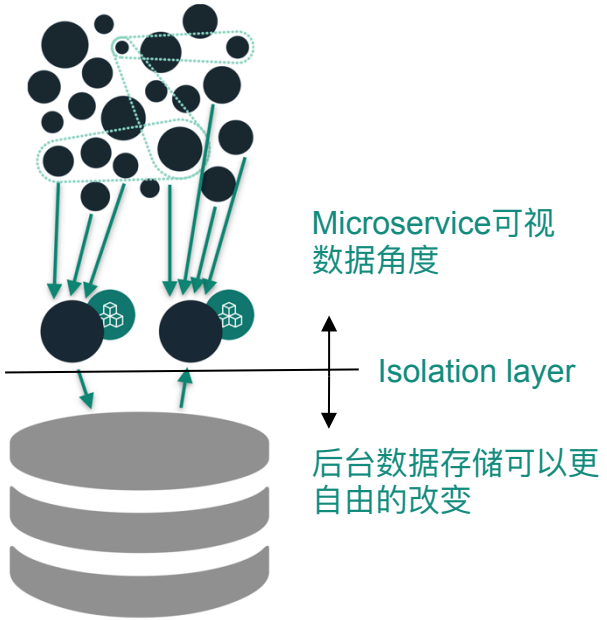
针对高可用性而设计

- 站点/ 区域级故障的灾难恢复
- 集群内强一致性
- 集群间最终一致性
- 可以依托Pivotal Cloud Foundry云原生平台（突破CAP）



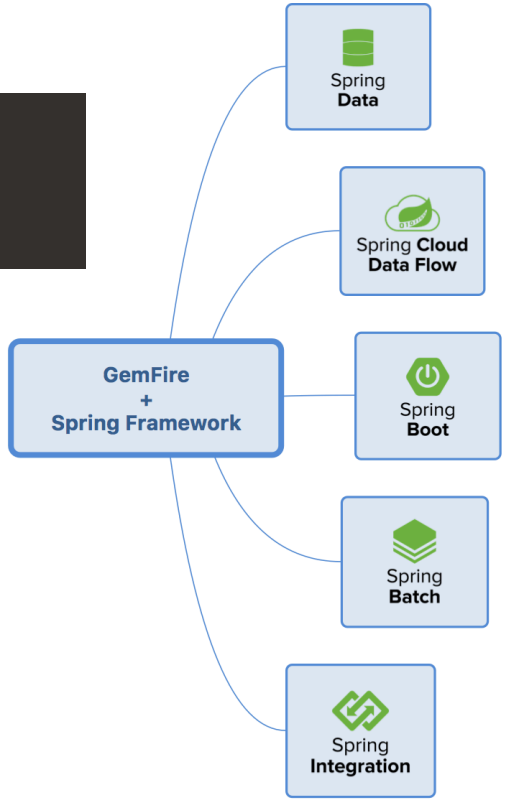
数据隔离保护层

- 微服务和传统后台存储之间的隔离层
- 避免后台数据的改变对于前端微服务的影响
  - 批量改变
  - 修改Schema
  - 受益点
    - 业务持续性
    - 对微服务投入的保护
    - 针对后台存储的灵活性



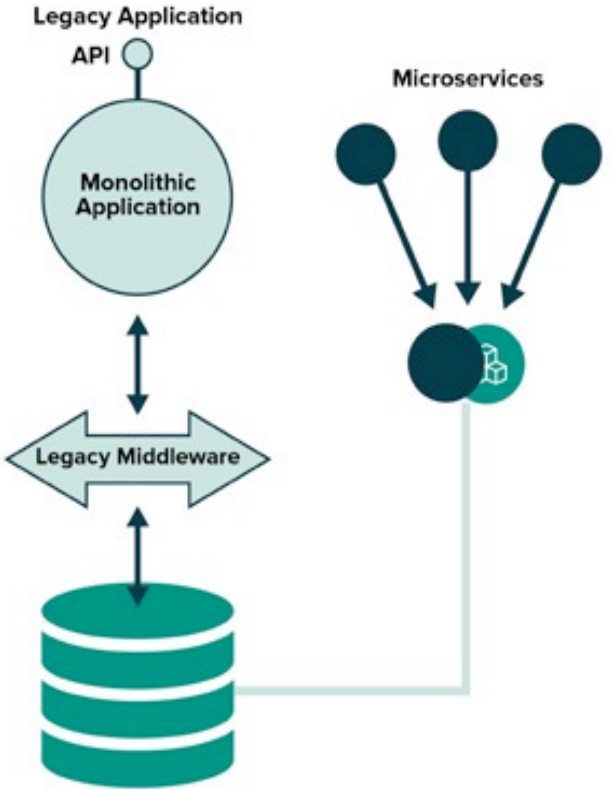
与Spring框架集成的快速开发

- Spring Framework 是最优秀的开源微服务框架（ https://spring.io/ ）
- Spring Data for Pivotal GemFire（ https://projects.spring.io/spring-data-gemfire）
  - GemFire APIs 和数据功能被集成到 Spring框架中
  - Annotation可以完成大量工作
  - Spring Data Repository 可以对应，持久化和查询实体
  - 实现持久化存储中变化的保护
  - 模板化读取GemFire中的数据
  - 大量减少代码编写



旧系统不应被排除在改造之外

- 绞杀式改造：基于微服务的心系统构建在旧系统的周围边缘上
- 数据互通：数据缓冲层活动在新旧系统之间完成

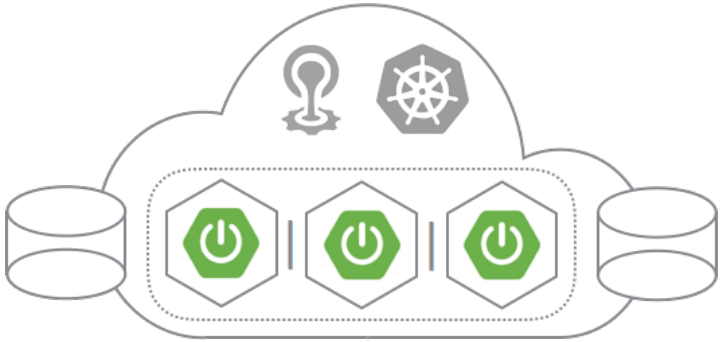


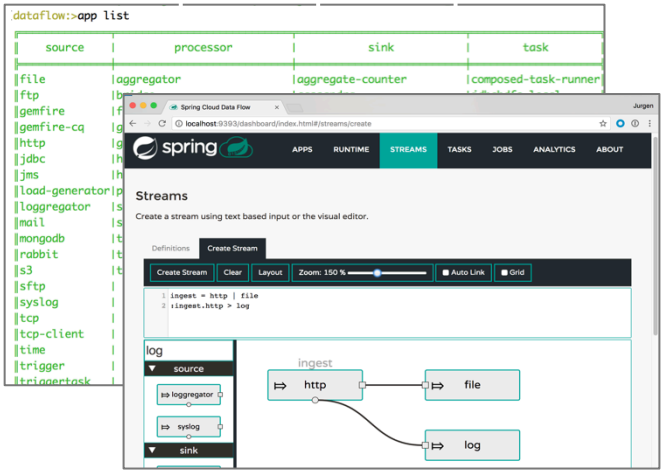
Spring Cloud Data Flow

Spring Cloud Data Flow是一个微服务工具包，用于构建数据集成和实时数据处理管道。

管道由Spring Boot应用程序组成，使用Spring Cloud Stream进行事件处理，或使用Spring Cloud Task进行批处理。

SCDF提供接口，以便将管道组合和部署到Cloud Foundry等平台上。





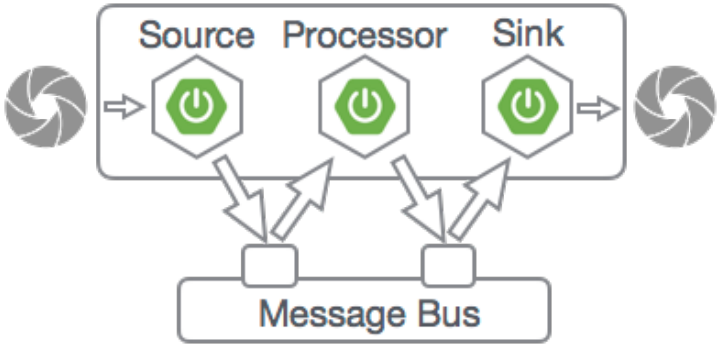
两种构建方式

Spring Cloud Task：

任务是独立的Spring Boot微服务，连接到数据/存储上。系统跟踪调用，退出状态。 用于ETL，例如在DBMS和像Hadoop这样的分析集群之间

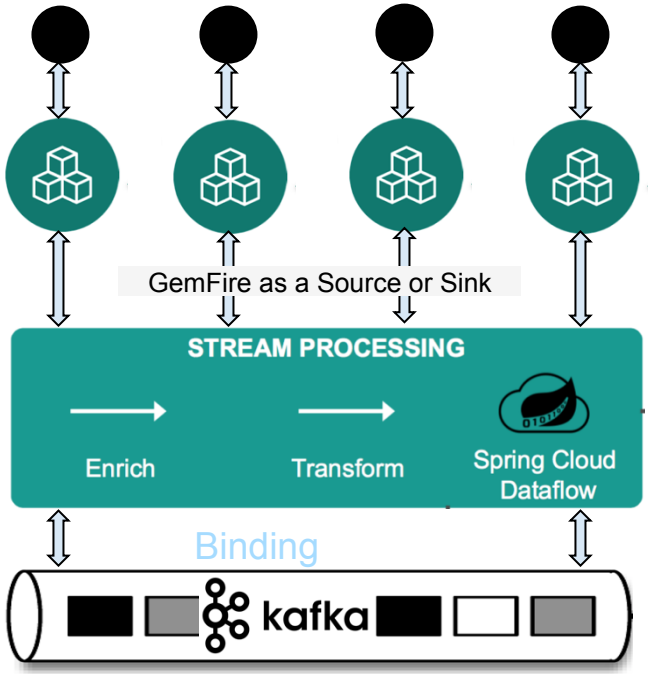
Spring Cloud Stream：

- 基于消息的微服务
- 通过pub / sub主题松散耦合
- 可插拔消息总线Kafka或RabbitMQ



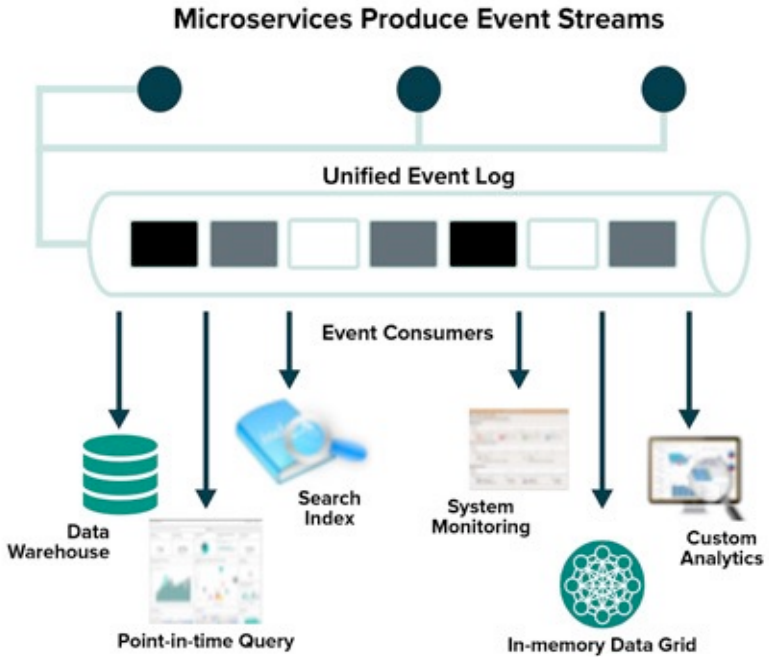
在不同的微服务之间共享事件

- 事件流 (pipelines)需要:
  - 一个事件source去启动事件流 (GemFire)
  - 事件流的协同,包括管道调度管理服务 (Spring Cloud Data Flow)
  - 一个可靠有序的事件存储 (Apache Kafka/rabbitmq)
  - Sink，事件流的终点目标 (GemFire)



事件流和统一的事件日志

- 所有事件的完整视图：
  - 安全违规检查
  - 欺诈预防
  - 微服务间搜索
  - 监控系统
  - 信息快照
  - 数据变化趋势



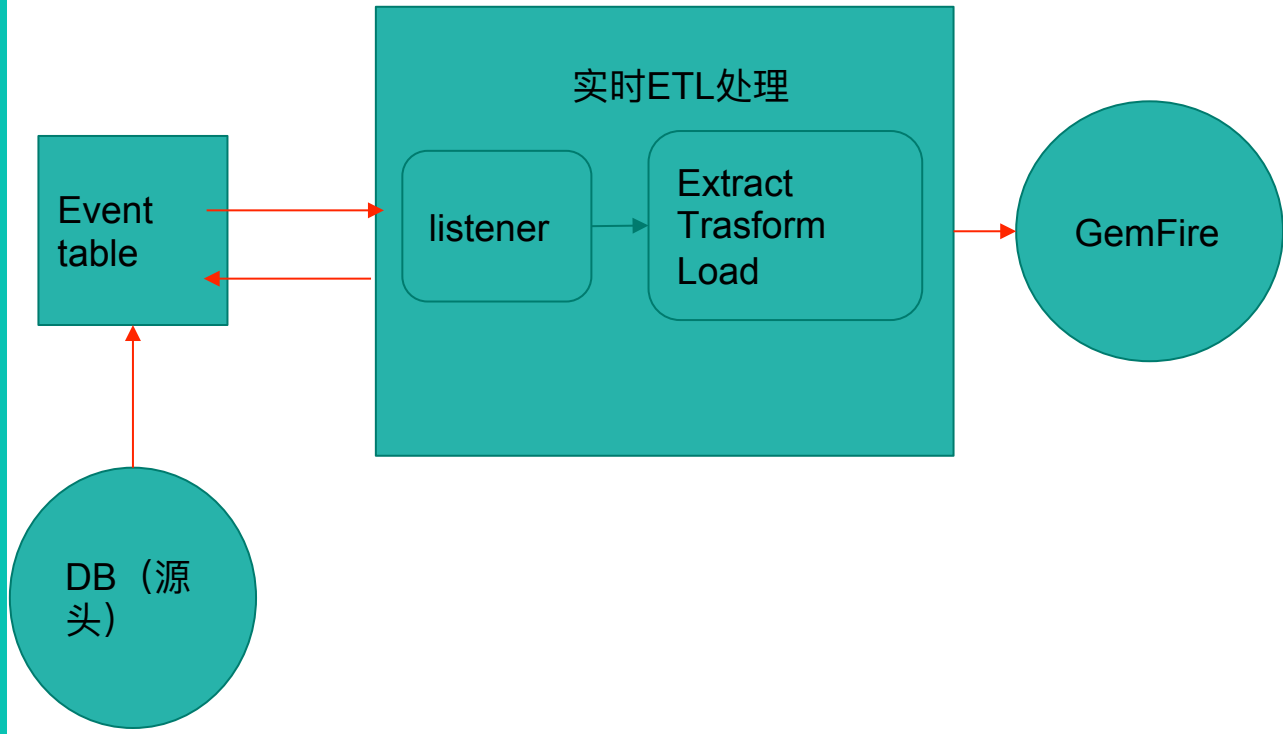
微服务架构数据服务改造实例

在不影响原有交易系统的前提下构建新的高性能高可用API查询系统。

微服务架构数据服务改造实例

1）事件表代替原有DB作为源头

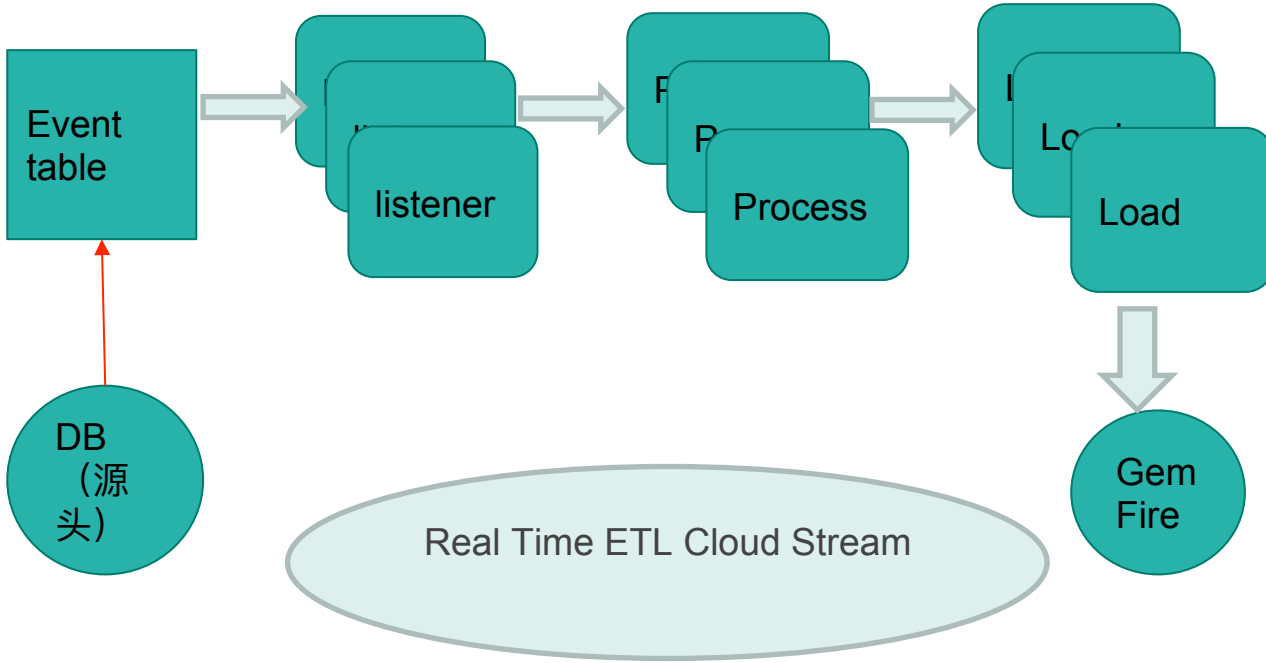
2）事件驱动的实时ETL代替文件批处理



微服务架构数据服务改造实例

1）用三组微服务协作代替原有巨石应用

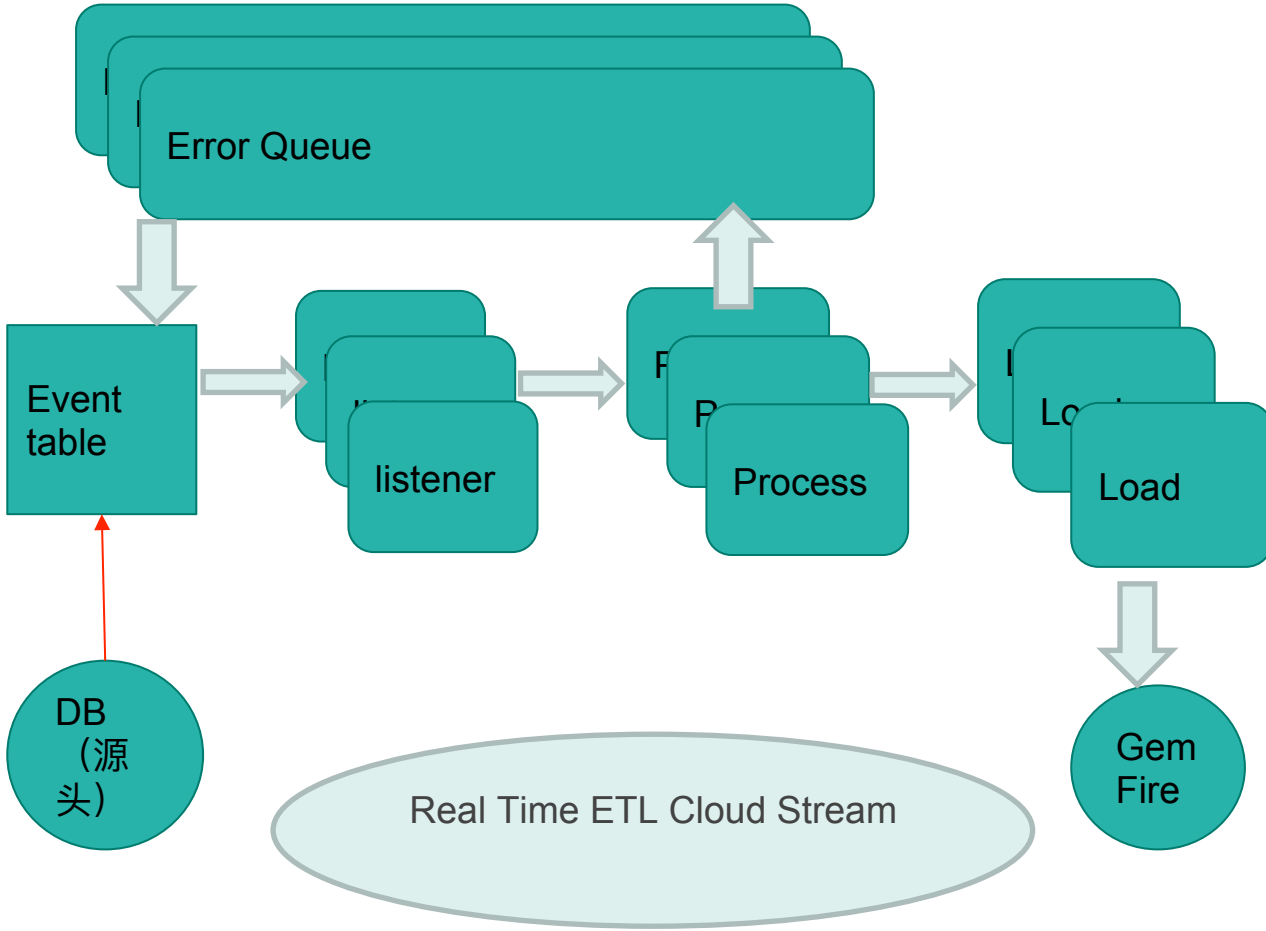
2）Spring Cloud Data Flow完成了从事件表到GemFire的数据流程



微服务架构数据服务改造实例

1）微服务架构下可以灵活增加新功能模块

2）error queue微服务完成错误事件的追踪处理



Pivotal Data Suite

Greenplum

Analyze, learn, predict
Build and refine the model

GemFire

Operationalize the model
Make real-time decisions

Spring Cloud Data Flow

Ingest, process, and transform
Agile data movement and enrichment

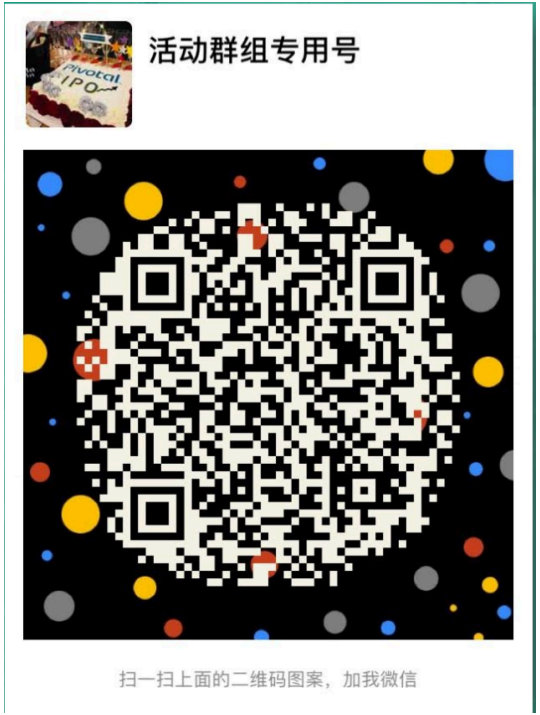
PIVOTAL PRESENTATION THEME

Q & A

技术研讨会PPT分享

Pivotal中国研发中心官方微信号

加入我们的技术讨论!



Transforming How The World Builds Software