分布式服务化架构关键技术

看Cloud Native云化应用架构及PaaS调度层

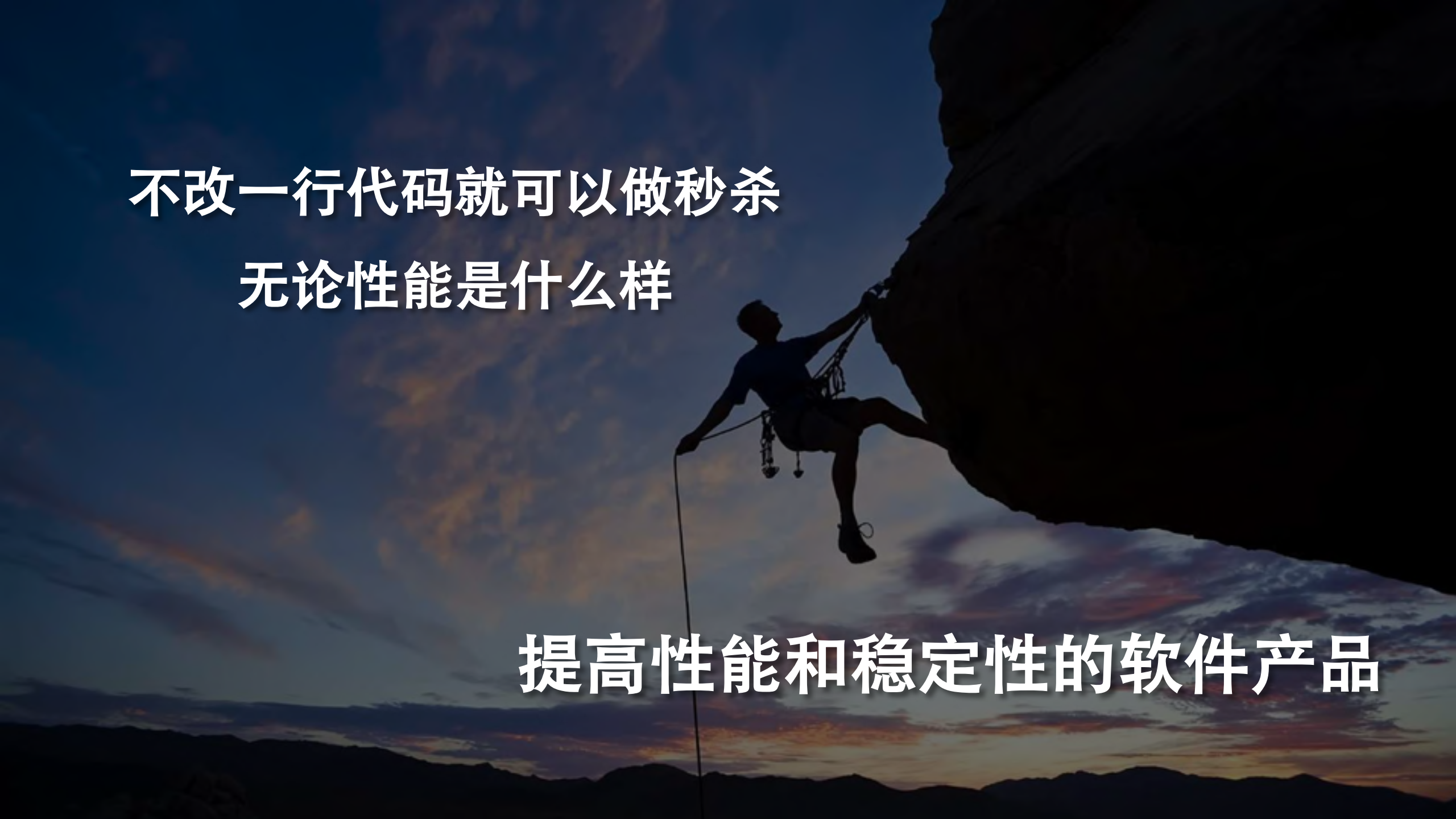
陈皓

MegaEase Inc.

个人简介

- **18年工作经历，超大型分布式系统基础架构研发和设计**
- **擅长领域：金融、电子商务、云计算、大数据**
- **职业背景**
  - 阿里巴巴资深架构师（阿里云、天猫、淘宝）
  - 亚马逊高级研发经理（AWS、全球购、商品需求预测）
  - 汤森路透资深架构师（实时金融数据处理基础架构）
- **目前在创业**
  - 2015-2016为40+公司提供过技术咨询服务
  - 2016-至今，致力于为企业提供高并发、高可用的IT服务保障

MegaEase


不改一行代码就可以做秒杀
无论性能是什么样
提高性能和稳定性的软件产品

大纲

- 如何提高性能和稳定性
- 分布式服务化架构的关键技术
- PaaS平台的核心
- “不改一行代码”案例分析
- MegaEase的产品介绍

如何提高性能和稳定性

提高性能

加缓存

缓存系统

缓存分区
缓存更新
缓存命中

负载均衡

网关系统

负载均衡
服务路由
服务发现

异步调用

异步系统

消息队列
消息持久
异步事务

数据镜像

数据镜像

数据同步
读写分离
数据一致性

数据分区

数据分区

分区策略
数据访问层
数据一致性

提高稳定性

服务拆分

服务治理

服务调用
服务依赖
服务隔离

服务冗余

服务调度

弹性伸缩
故障迁移
服务发现

限流降级

限流降级

异步队列
降级控制
服务熔断

高可用架构

高可用架构

多租户系统
灾备多活
高可用服务

高可用运维

运维系统

全栈监控
DevOps
自动化运维

MegaEase

我们要做多少事?

- **高性能处理**
  - 缓存、弹性伸缩、异步处理、数据复制……
- **关键业务保护**
  - 高可用、故障隔离、业务降级……
- **流量控制**
  - 负载均衡、服务路由、熔断、降级……
- **整体架构监控**
  - 三层系统监控（应用层、中间件层、基础层）
- **DevOps**
  - 环境构建、持续集成、持续部署
- **架构管理**
  - 架构版本、生命周期管理，服务管理……
- **自动化运维**
  - 自动伸缩、故障迁移、配置管理，状态管理……
- **基础资源调度管理**
  - 计算、存储、网络资源调度和管理

然而还没完……

这些东西都不是功能性需求

- 不是功能性需求，容易被忽略
- 属于基础设施，不容易被理解

技术含量高，各种技术坑

- 解决了一个问题，新增多个问题
- 需要投入的时间和人力成本极高

好的技术人员越来越难招

- 能做好这些软件的人并不多
- 好的技术人员，对工作极其挑剔

分布式系统的问题

	传统单体架构	分布式服务化架构
新功能开发	新功能开发需要时间	容易开发和实现
部署	不经常且容易部署	经常发布，部署复杂
隔离性	故障影响范围大	故障影响范围小
系统性能	响应时间快，吞吐量小	响应时间慢，吞量大
系统运维	运维简单	运维复杂
新人上手	学习曲线大（应用逻辑）	学习曲线大（架构逻辑）
技术	技术单一且封闭	技术多样且开放
测试	简单	复杂
系统扩展性	扩展性很差	扩展性很好
系统管理	重点在于开发成本	重点在于服务治理和调度

如何面对如此纷乱的技术

怎么面对呢?

- **张开一个一个的网眼?**
  - 如果你是一个一个的去做，你就是在使蛮力
  - 一个一个的做，你会发现连不起来。
- **张开一个大网，需要找到“纲”**
  - 什么才是这个渔网的“纲”？
  - 如何才能做到“纲举目张”？

分布式系统的本质

应用整体监控

- **基础层监控**
  OS、主机、网络…
- **中间件层监控**
  消息队列、缓存、数据库、应用容器、网关、RPC框架、JVM…
- **应用层监控**
  API请求、吞吐量、响应时间 、错误码、SQL语句、调用链路、函数调用栈、业务指标…

资源/服务调度

- **计算资源调度**
  CPU，内存、磁盘、网络…
- **服务调度**
  服务编排、服务复本、服务容量伸缩、故障服务迁移、服务生命周期管理…
- **架构调度**
  多租户、架构版本管理、架构部署、运行、更新、销毁管理、多租户管理、灰度发布…

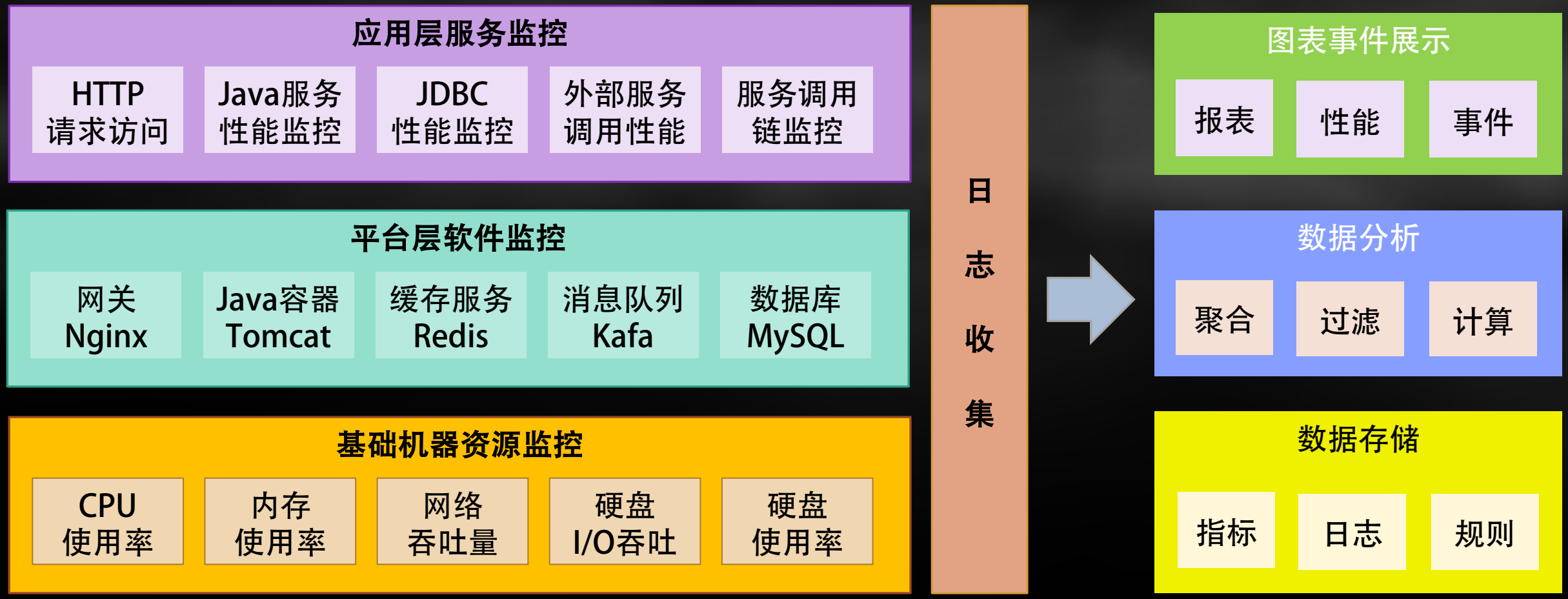
状态/数据调度

- **数据可用性**
  多复本保存
- **数据一致性**
  读写一致性策略
- **数据分布式**
  数据索引、分片

流量调度

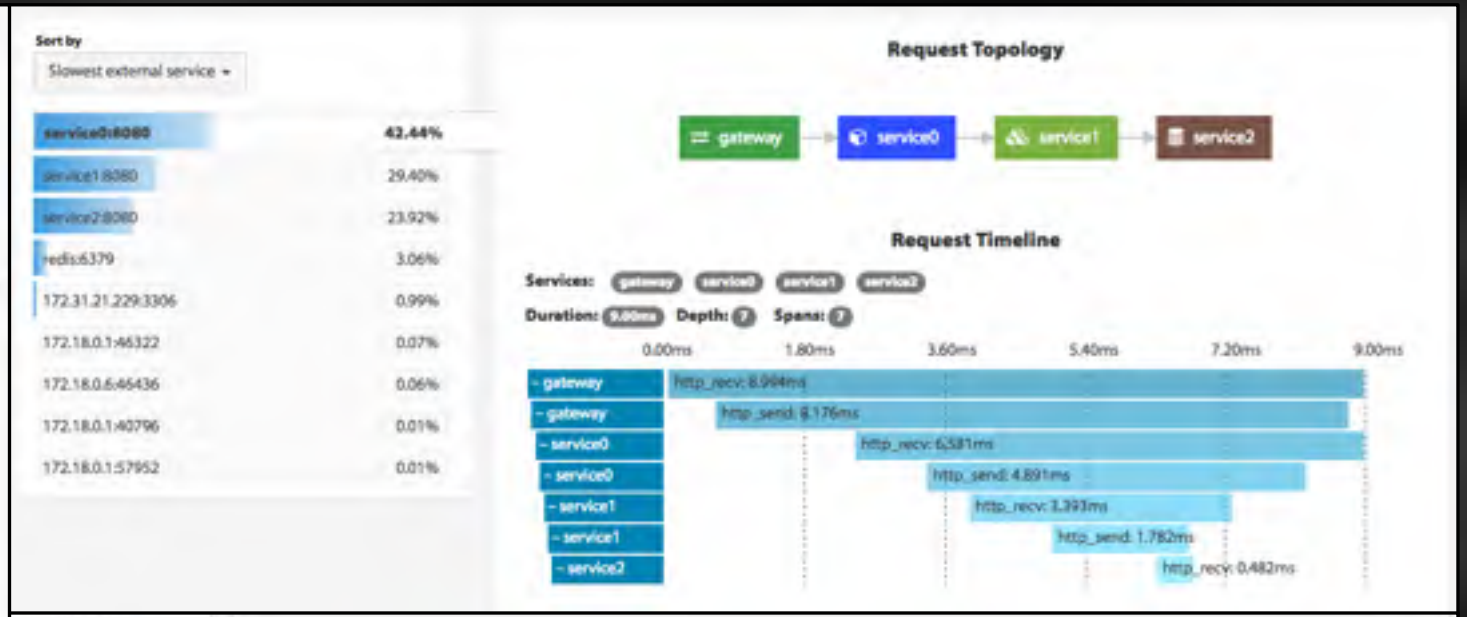
- **服务治理**
  服务发现、服务路由、服务降级、服务熔断、服务保护…
- **流量控制**
  负载均衡、流量分配、流量控制、异地灾备 …
- **流量管理**
  协议转换、请求校验、数据缓存、数据计算…

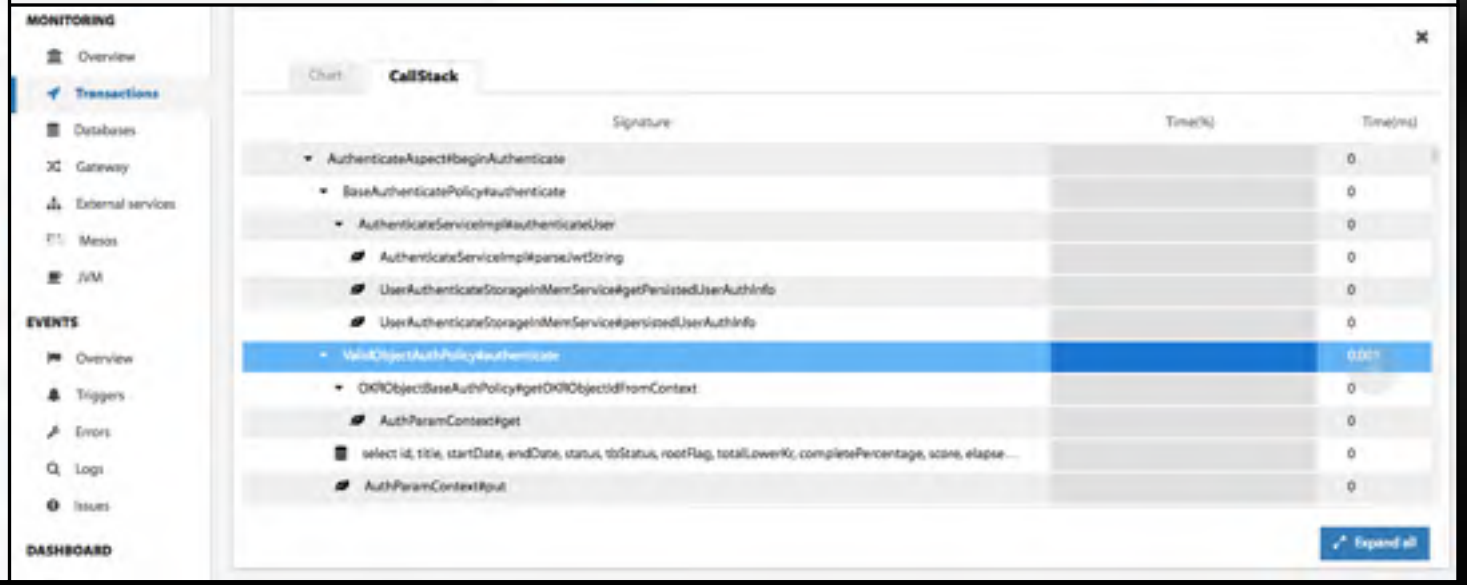
全栈监控技术说明



MegaEase


System Health Score
System Key Index Overview
31
2,028
173.97
Throughput(RPM)
System Topology
Top Transactions
HTTP Status Code Distribution
Servers
Request Error EWMA (m1 Rate)


Sort by
Slowest external service
service0:8080 42.44%
service1:8080 29.40%
service2:8080 23.92%
redis:6379 3.06%
172.31.21.229:3306 0.99%
172.18.0.1:46322 0.07%
172.18.0.6:46436 0.06%
172.18.0.1:40796 0.01%
172.18.0.1:57952 0.01%
Request Topology
gateway
service0
service1
service2
Request Timeline
Services:
Duration: 9.00ms
Depth: 7
Spans: 7
0.00ms
1.80ms
3.60ms
5.40ms
7.20ms
9.00ms
- gateway
- gateway
- service0
- service0
- service1
- service1
- service2
http_recv: 8.994ms
http_send: 8.176ms
http_recv: 6.581ms
http_send: 4.891ms
http_recv: 3.393ms
http_send: 1.782ms
http_recv: 0.482ms


MONITORING
Overview
Transactions
Databases
Gateway
External services
Mesos
JVM
EVENTS
Overview
Triggers
Errors
Logs
Issues
DASHBOARD
Chart
CallStack
Signature
Time(%)
Time(ms)
AuthenticateAspect#beginAuthenticate
BaseAuthenticatePolicy#authenticate
AuthenticateServiceImpl#authenticateUser
AuthenticateServiceImpl#parseJwtString
UserAuthenticateStorageInMemService#getPersistedUserAuthInfo
UserAuthenticateStorageInMemService#persistedUserAuthInfo
ValidObjectAuthPolicy#authenticate
OKRObjectBaseAuthPolicy#getOKRObjectIdFromContext
AuthParamContext#get
select id, title, startDate, endDate, status, toStatus, rootFlag, totalLowerKr, completePercentage, score, elapse...
AuthParamContext#put
Expand all

MegaEase

一栈式调度技术说明

Stack 定义

- 描述定义
- 相关组件
- 环境配置
- 集群描述
- 依赖关系
- 健康检查
- 部署方式
- 结点存储

Stack 运行实例

基础物理层

- 服务部署
- 服务发现
- 服务路由（负载均衡）
- 服务健康检查
- 服务调度（伸缩/故障迁移）
- 物理层资源管理
- 服务监控

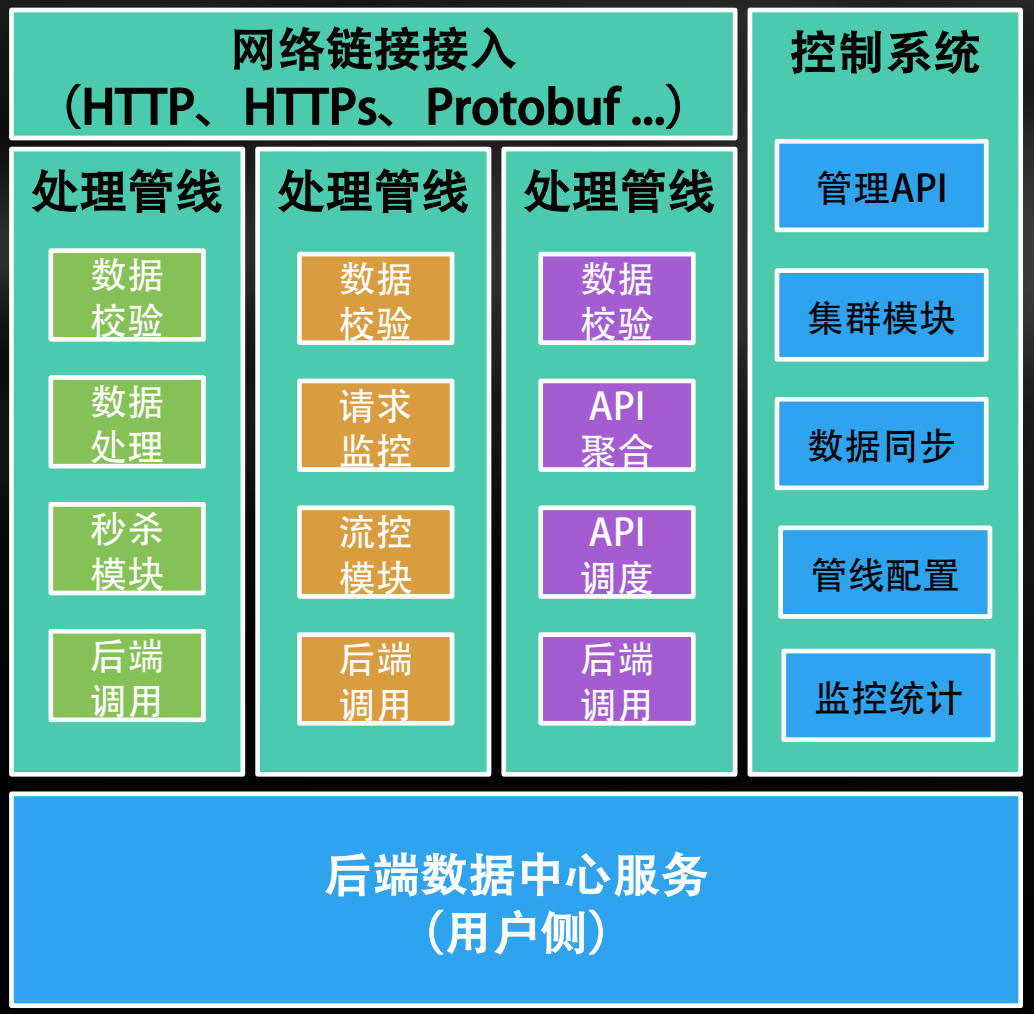
MegaEase

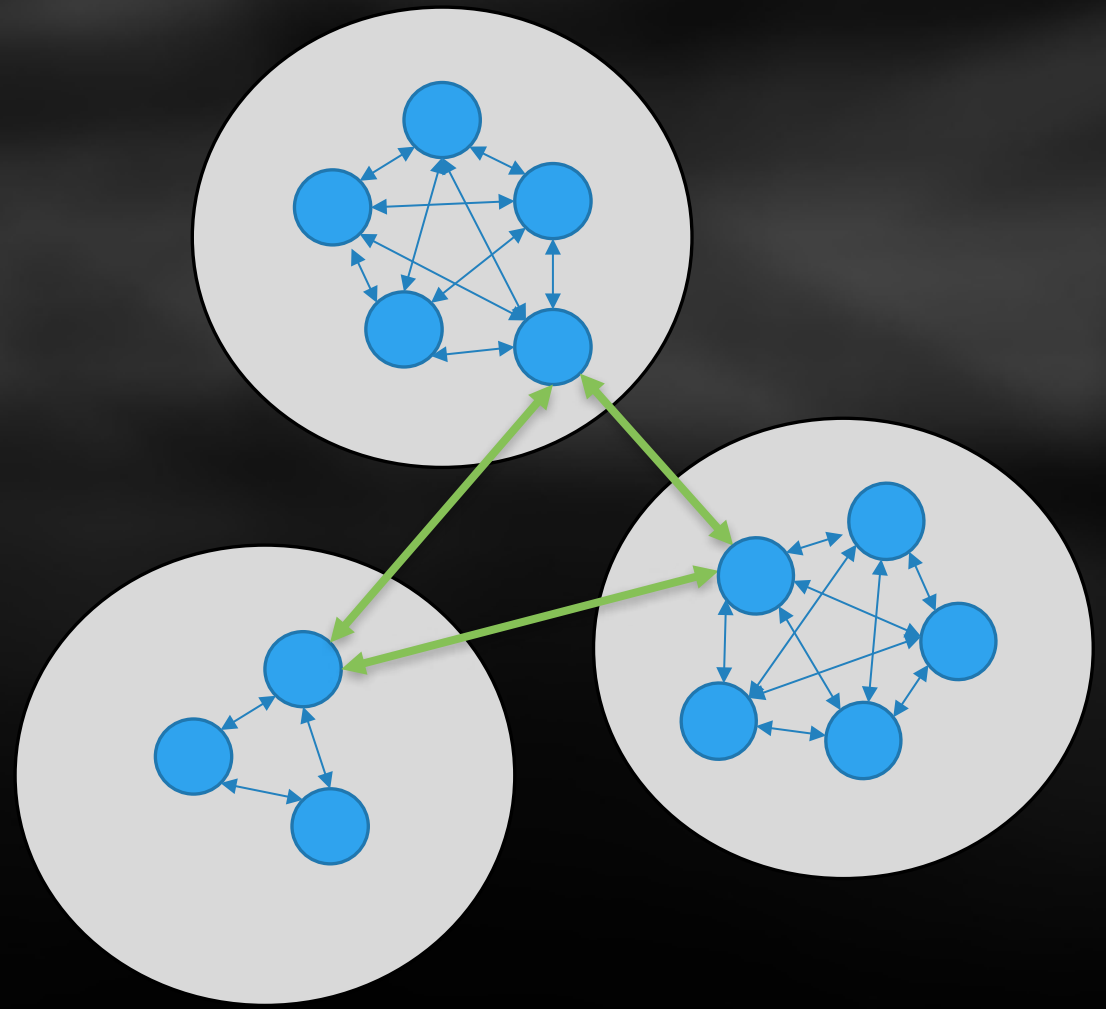
面向服务的Docker镜像

- **为什么要做服务化的Docker镜像**
  - 想想现有的中间件 – nginx 和 zookeeper
  - 都不是服务化的，任何静态配置或动态配置的改变都需要通过改conf文件
- **面对不完美世界的Workaround的方案**
  - 我们开发了一个服务化的Docker EntryPoint框架，提供如下最基本回调脚本
  - Start， Stop， ApplyConfig，HealthChecking

MegaEase

网关流量调度技术



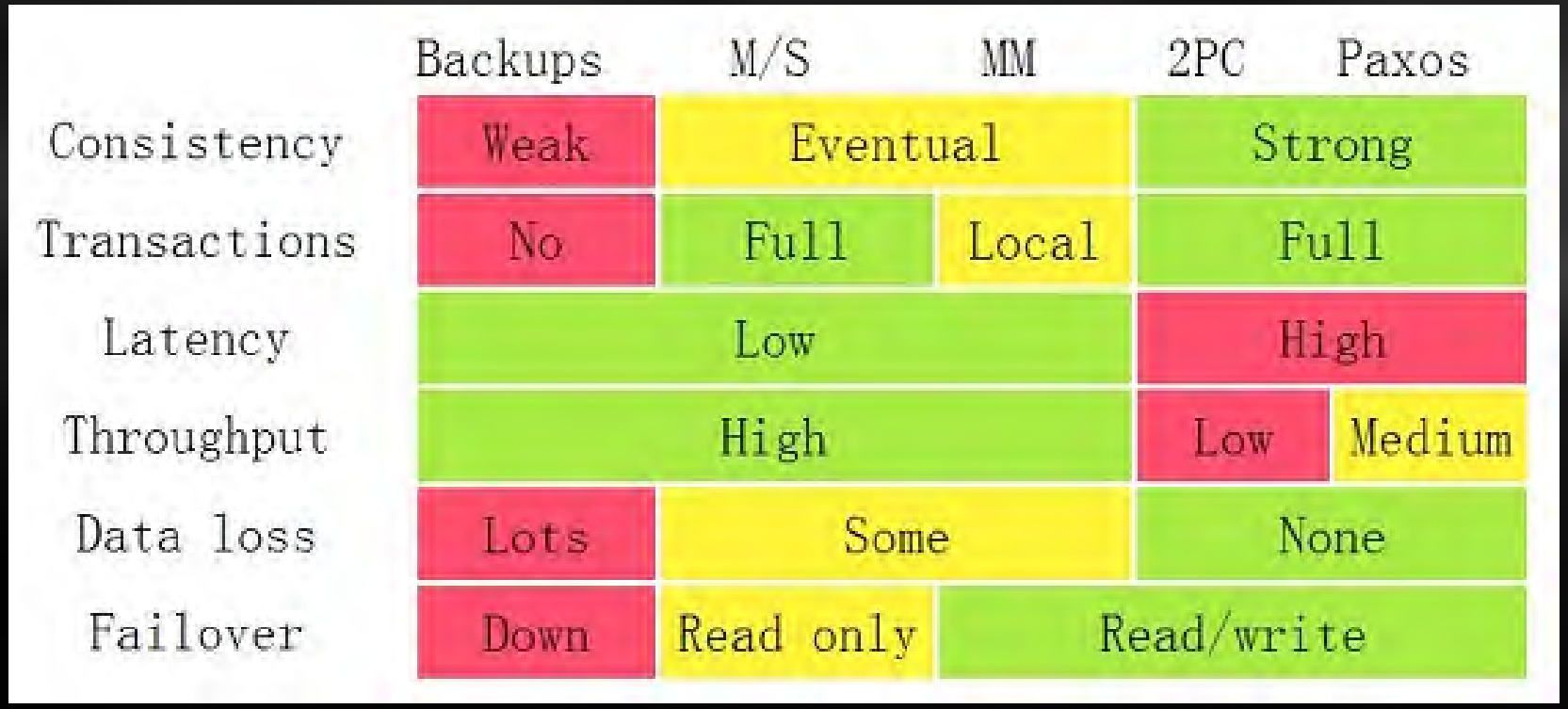


如何设计一个好的流量调度网关

- **现在的所有的Gateway在设计理念上都达不我们想要的高度。**
  - Nginx和OpenResty明显就是给运维人员用的。
- **一个好的Gateway需要有如下的特性**

Service	Cluster	Lambda
• 首先一定要是一个Service，作为service的一个标志是：通过API去改配置，而不是通过文件。	• 其次，还要是一个Cluster，作为Cluster的一个标志是：能够互相复制数据，可以集群内分组，我们使用NRW模型+Gossip/Raft协议。	• 最后，还可入嵌入用户代码，既 Serverless 或 AWS Lambda。

状态/数据调度



Google App Engine的co-founder Ryan Barrett
2009年的google i/o上的演讲《Transaction Across DataCenter》
（视频： http://www.youtube.com/watch?v=srOgpXECblk）

MegaEase

状态/数据调度 - 分布式存储系统

Data Scheme

关系型数据库、NoSQL
时序数据库、搜索数据库
OLAP/OLTP、漏洞数据……

数据格式太多，很难做出放之四海皆准的

Data Storage

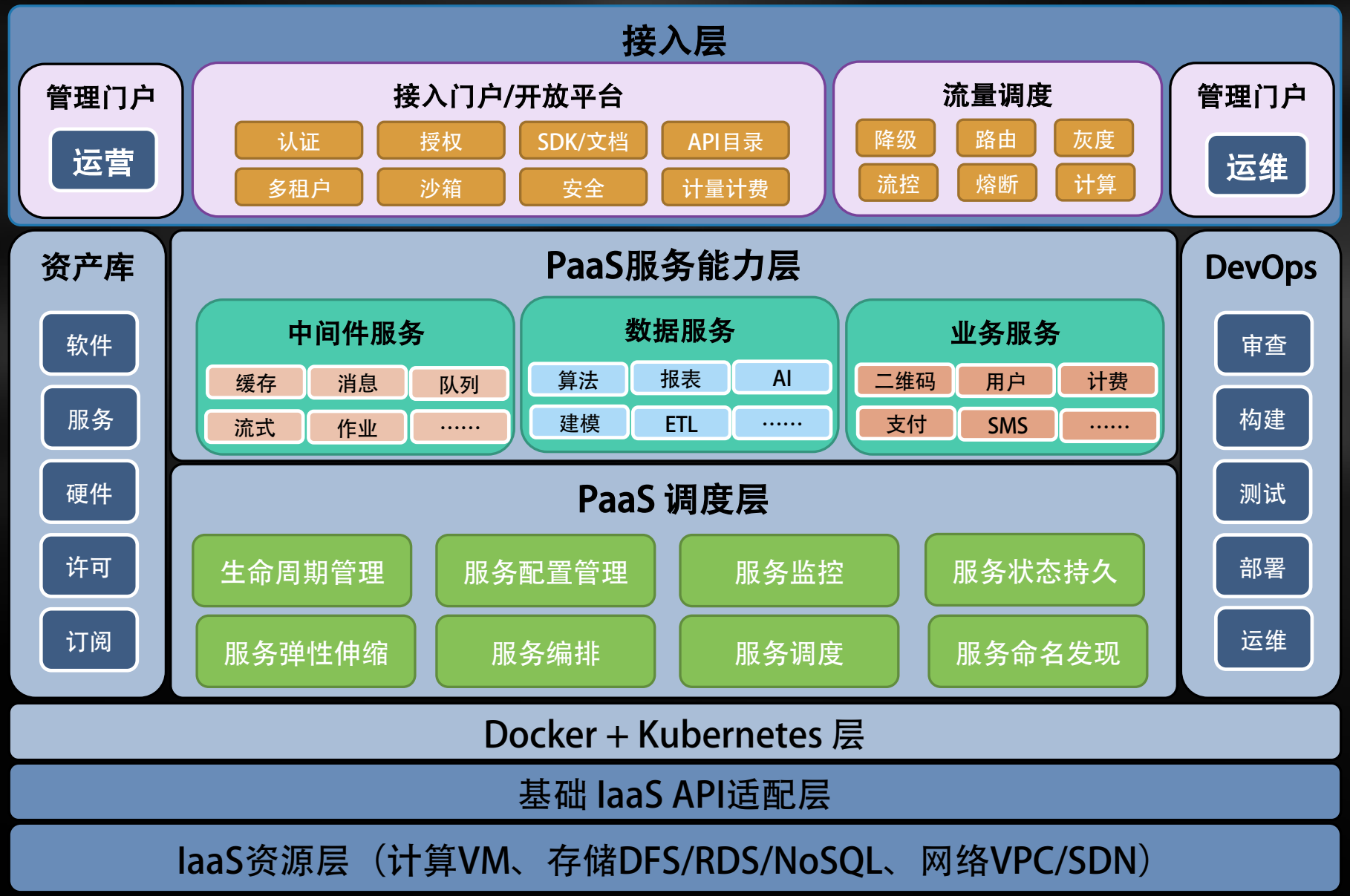
廉价的：开源方案NFS/Ceph/TiDB/Cockroach…
中间价：云产商方案 S3/Dynamo/Aurora…
昂贵的：EMC、 Nutanix…

不同的分布式存储方案有不同优缺点

MegaEase

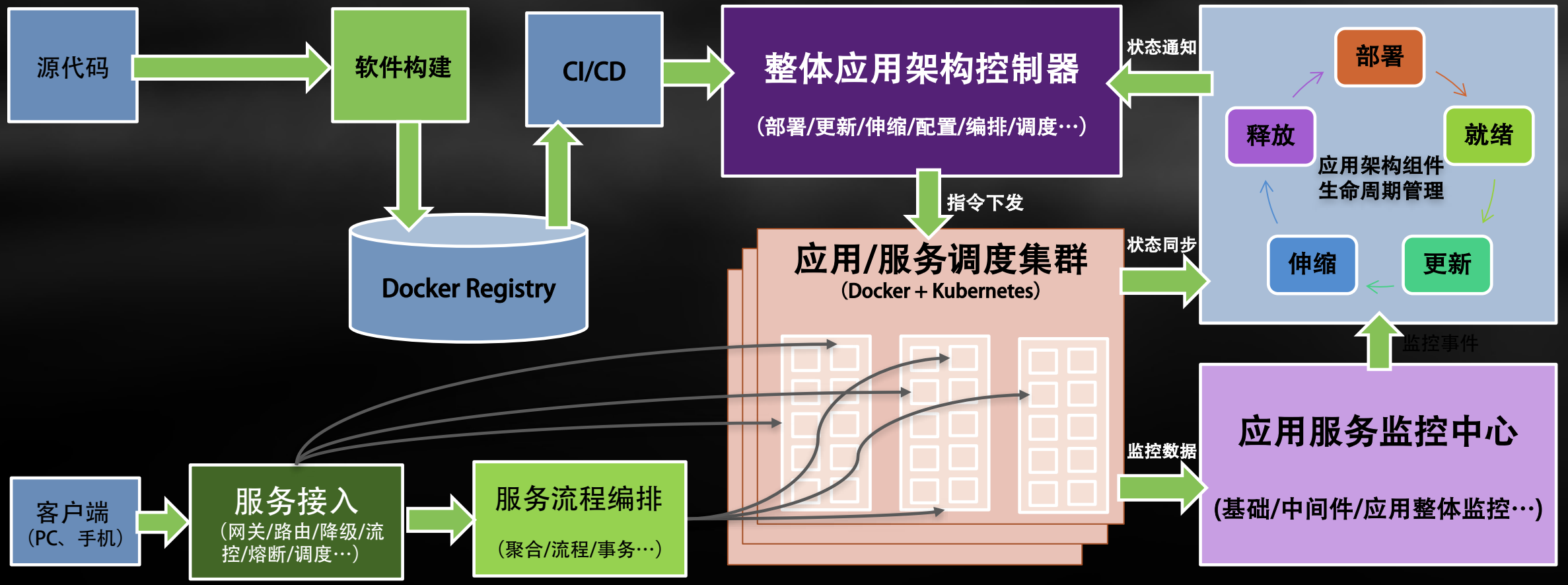
PaaS平台是什么样的

MegaEase


接入层
管理门户
运营
接入门户/开放平台
认证
授权
SDK/文档
API目录
多租户
沙箱
安全
计量计费
流量调度
降级
路由
灰度
流控
熔断
计算
管理门户
运维
资产库
软件
服务
硬件
许可
订阅
PaaS服务能力层
中间件服务
缓存
消息
队列
流式
作业
……
数据服务
算法
报表
AI
建模
ETL
……
业务服务
二维码
用户
计费
支付
SMS
……
PaaS 调度层
生命周期管理
服务配置管理
服务监控
服务状态持久
服务弹性伸缩
服务编排
服务调度
服务命名发现
DevOps
审查
构建
测试
部署
运维
Docker + Kubernetes 层
基础 IaaS API适配层
IaaS资源层（计算VM、存储DFS/RDS/NoSQL、网络VPC/SDN）

MegaEase

开发/运维示意图



MegaEase

如何做到不改一行代码

MegaEase

“不改一行代码”案例分析

性能再差也能做秒杀

十倍提高性能

提高系统稳定性

节省￥100万

MegaEase 的三个产品

Ease Monitor

基于ELK完全开放式的APM

- **基础层监控**
  OS、主机、网络…
- **中间件层监控**
  消息队列、缓存、数据库、应用容器、网关、RPC框架、JVM…
- **应用层监控**
  API请求、吞吐量、响应时间 、错误码、SQL语句、调用链路、函数调用栈、业务指标…

Ease Stack

基于 Kubernetes 的二级调度

- **一键全栈部署**
  一次架构定义、多次灵活部署
- **容量弹性伸缩**
  手动或自动的管理应用集群容量
- **动态配置更新**
  集中管理、自动下发
- **应用状态保持**
  持久应用状态、自动管理关联关系

Ease Gateway

自主研发的服务化集群式的api网关

- **集群化**
  Gossip/RAFT协议，NRW模型、集群分组
- **服务化**
  管理API、统计数据……
- **插件式**
  - 用户逻辑插入，Lambda

MegaEase


谢谢