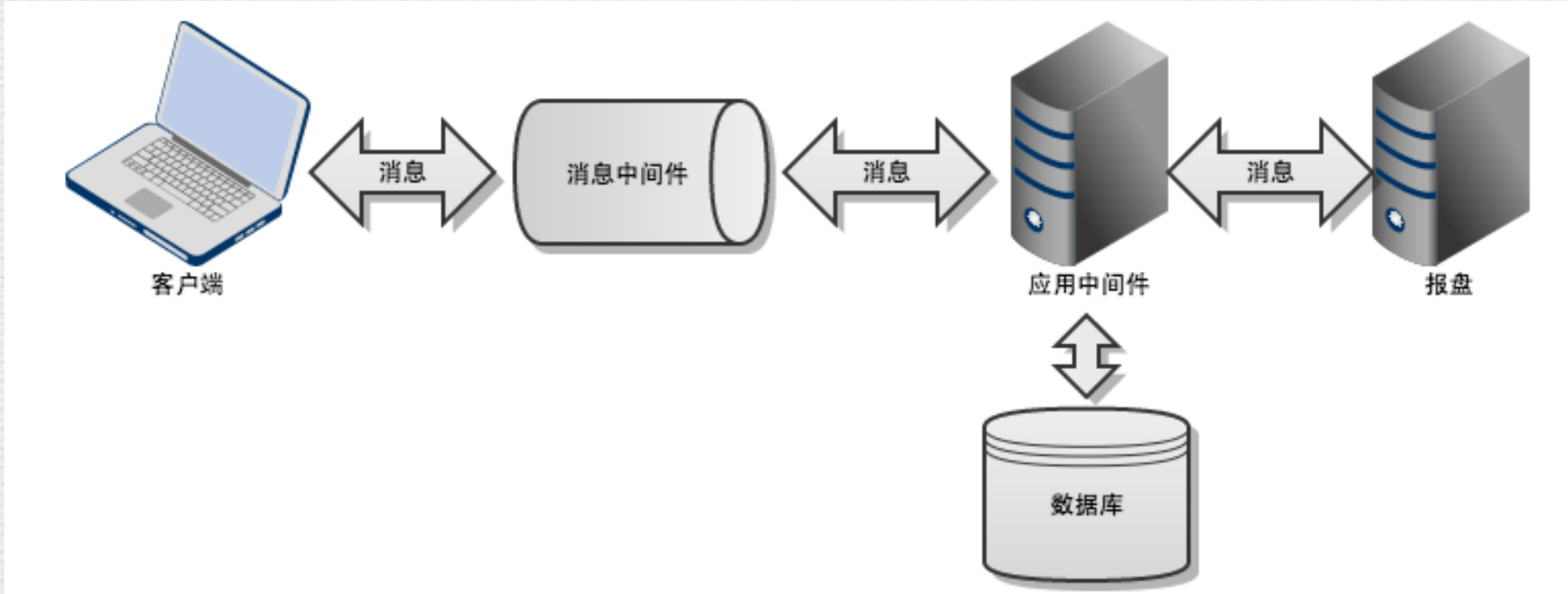
实时高可用金融交易系统架构与设计

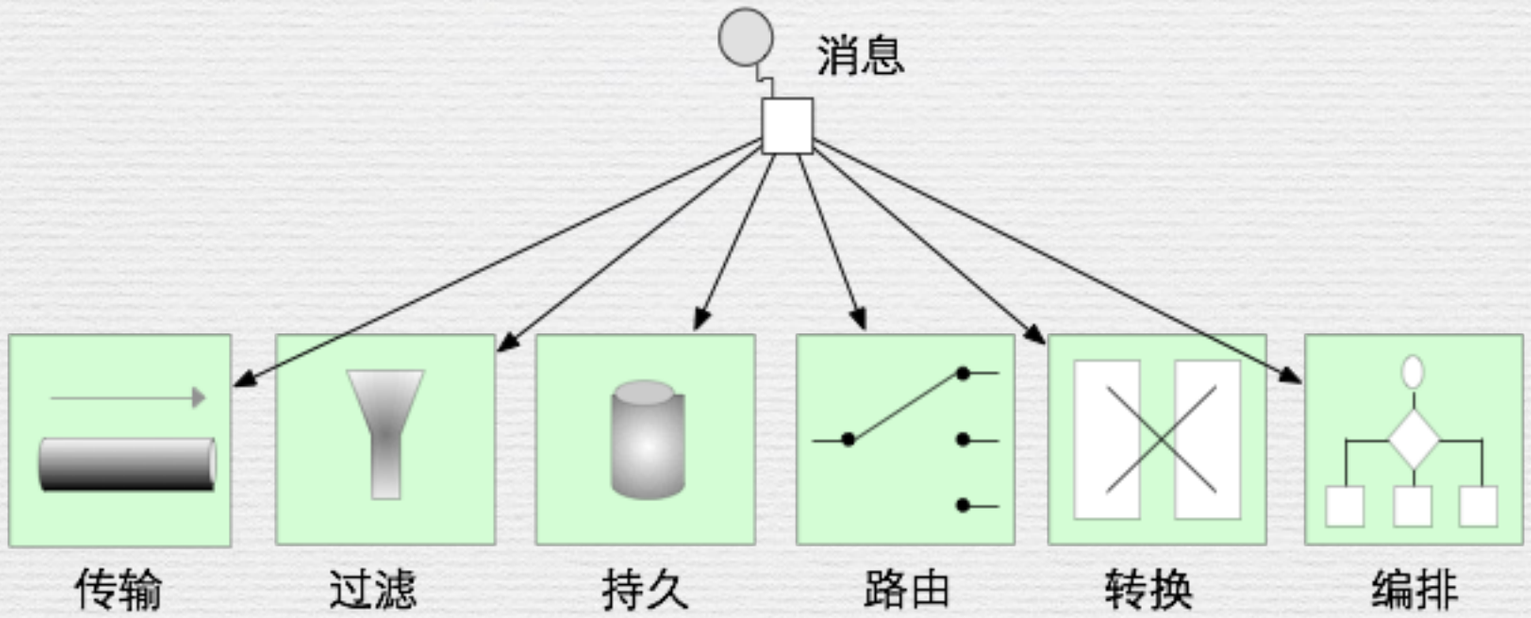
广发证券的去IOE实践

传统交易系统



- 系统最关键：数据库、消息中间件（IOE）

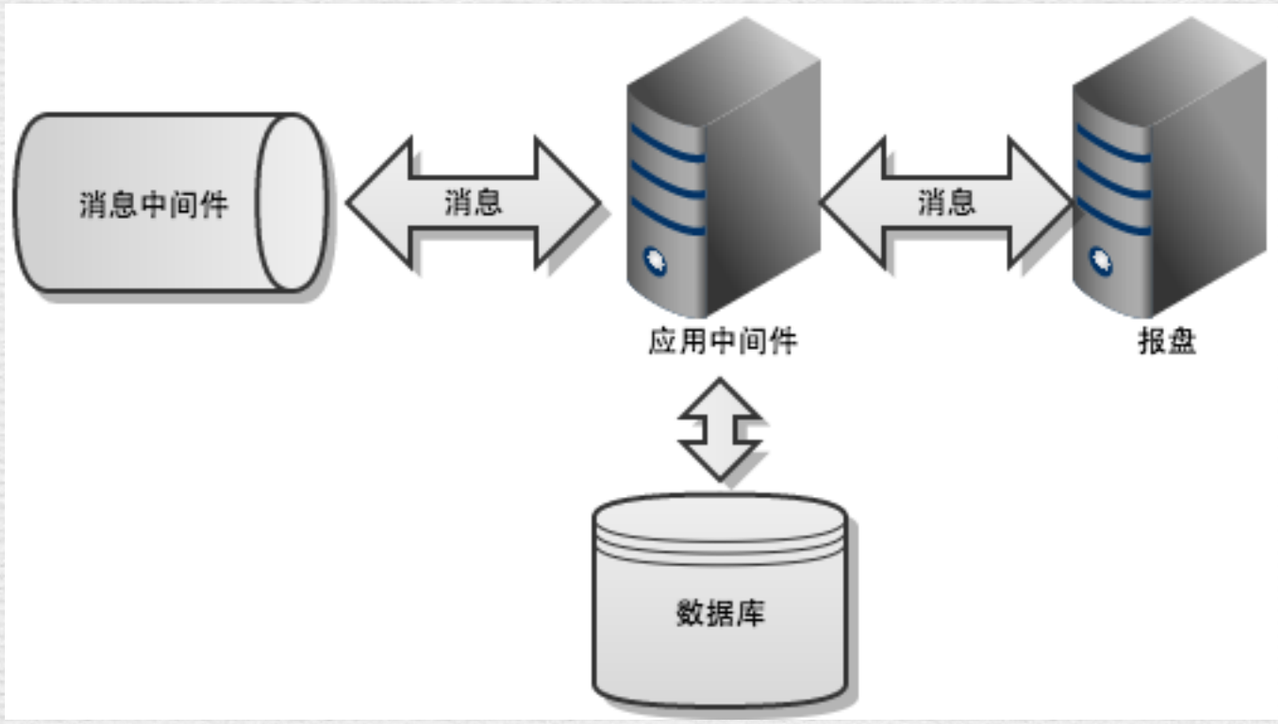
消息中间件



- 消息的发送者与接收者解耦、灵活部署

应用中间件

- 处理请求、保持状态、报盘、反馈结果
- 持久到中心数据库
- 并发通过锁同步或者数据库同步
- 冷备



需要解决的问题

- 去中心化
- 实时
- 高可用
- 敏捷

创新交易系统

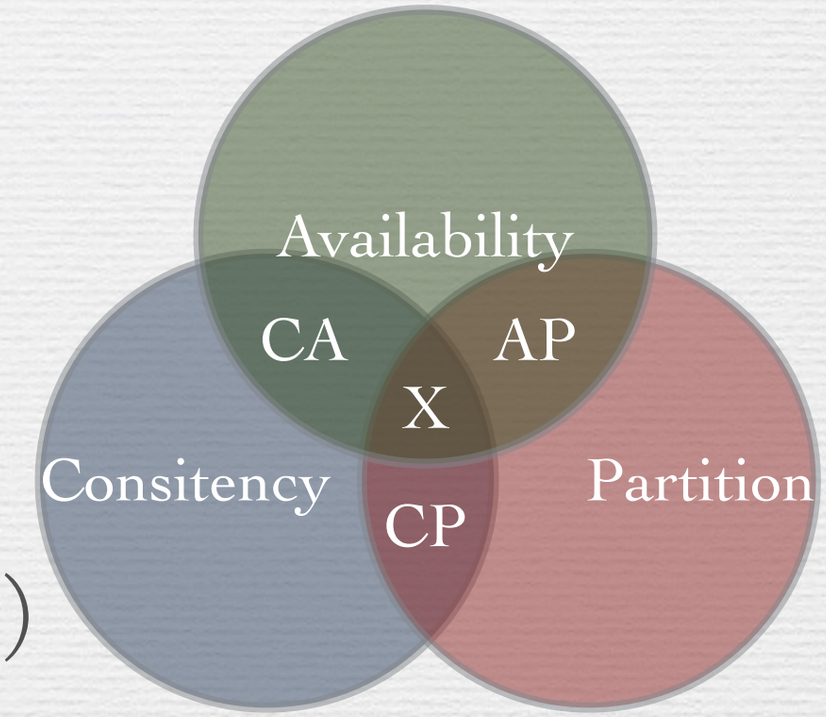
- 采用互联网开源技术与理念
- 多播消息总线、无锁线程间高速通信、EDA
- Event Sourcing架构，CQRS模式
- 基于Java的应用框架、模块化、自动化构建、测试、部署

互联网技术


BDD
Docker
Git
Jenkins
LMAX
Disruptor
Maven
NodeJS
OpenPGM
QuickFix
Reactive
RxJava
Spring
WAL
ZeroMQ

互联网技术的冲击

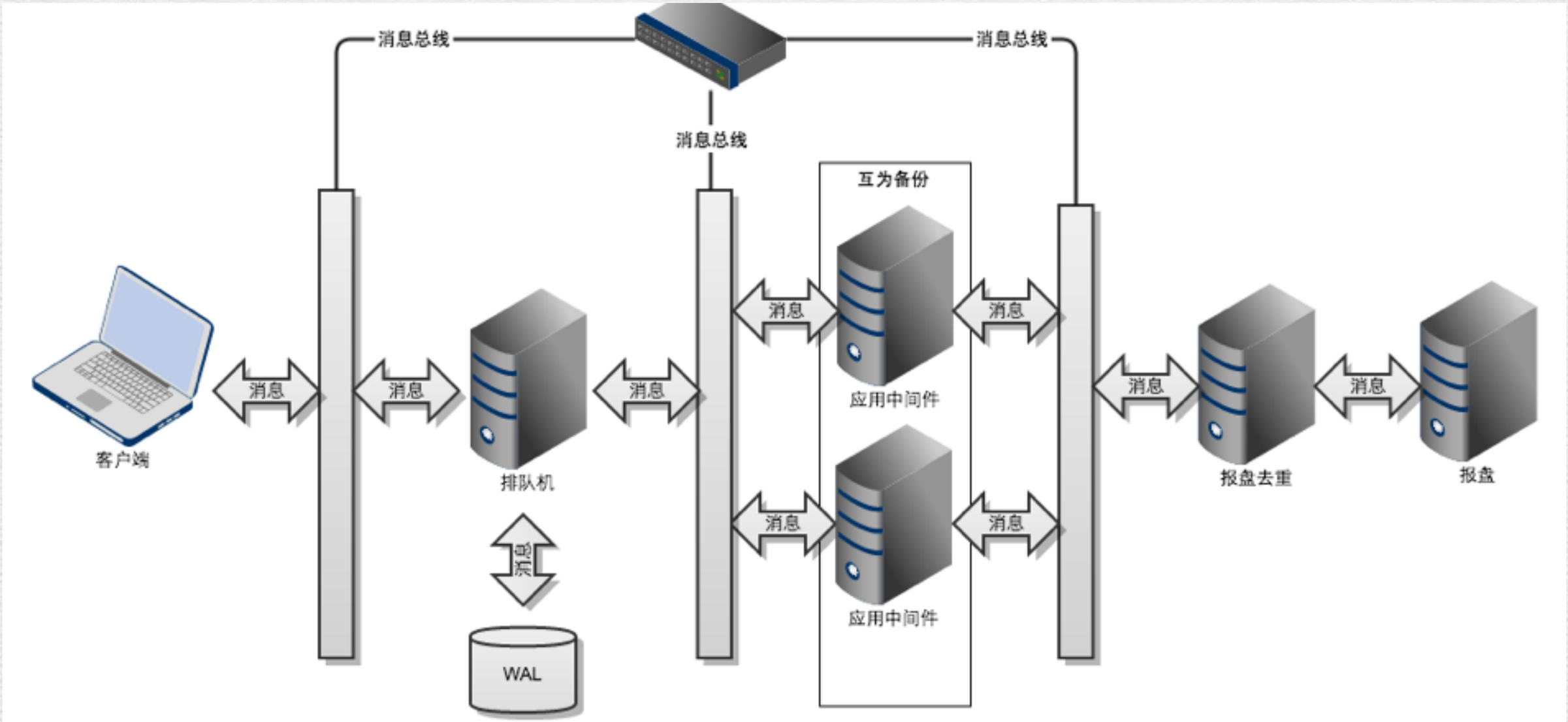
- 云计算、云存储
- 弱一致性，高可用性（AP）
- 强一致性，分布式事务，同步复制（CP）



互联网技术的冲击

- 保证C的前提下，尽量提高A
- 超高速通信
  - 网络（PGM多播）
  - 线程间（LMAX Disruptor）
- 本地计算，增量计算（EDA）

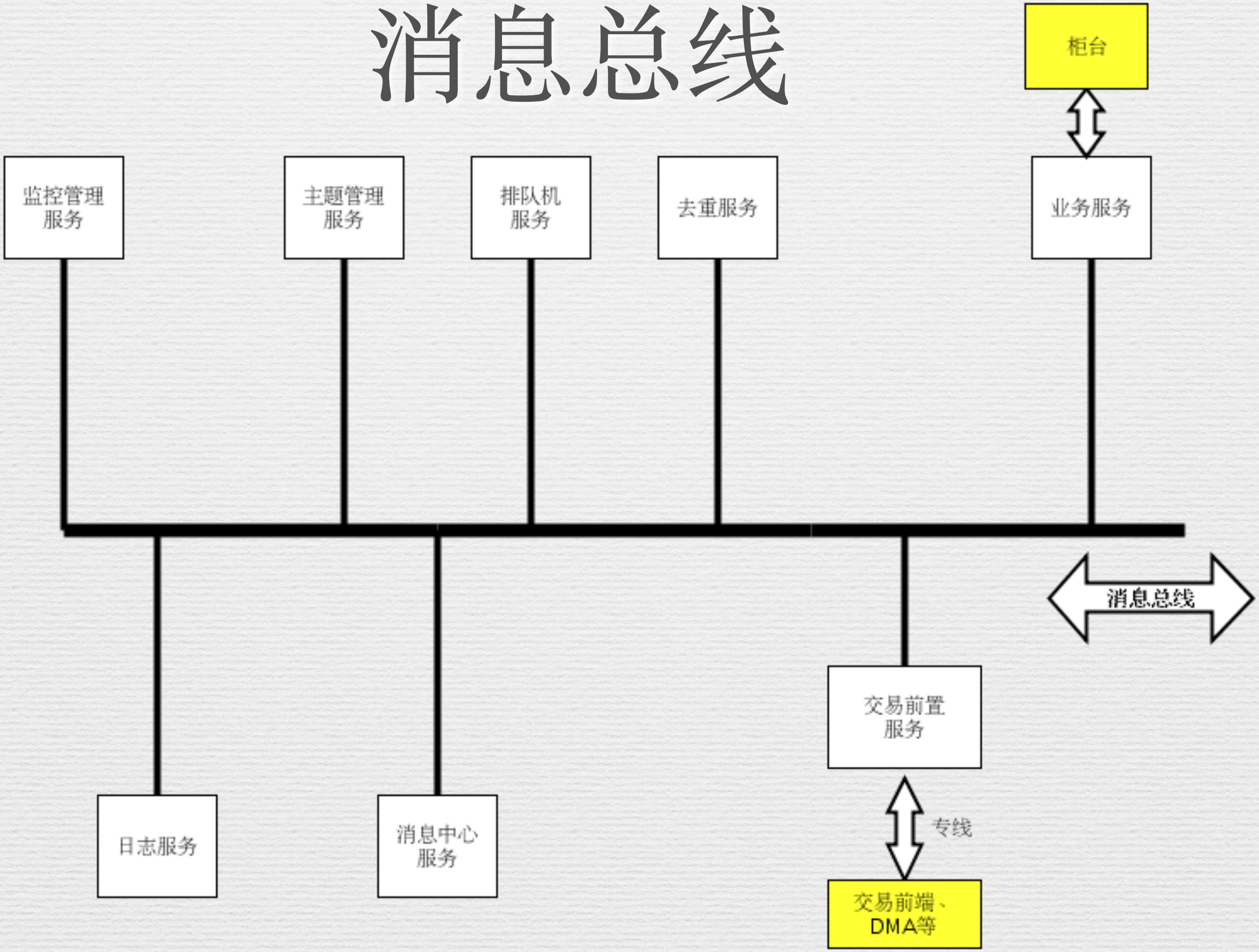
创新交易系统


消息总线
消息总线
消息总线
互为备份
消息
消息
消息
消息
消息
消息
消息
消息
消息
消息
客户端
排队机
应用中间件
应用中间件
报盘去重
报盘
WAL

应用可靠性

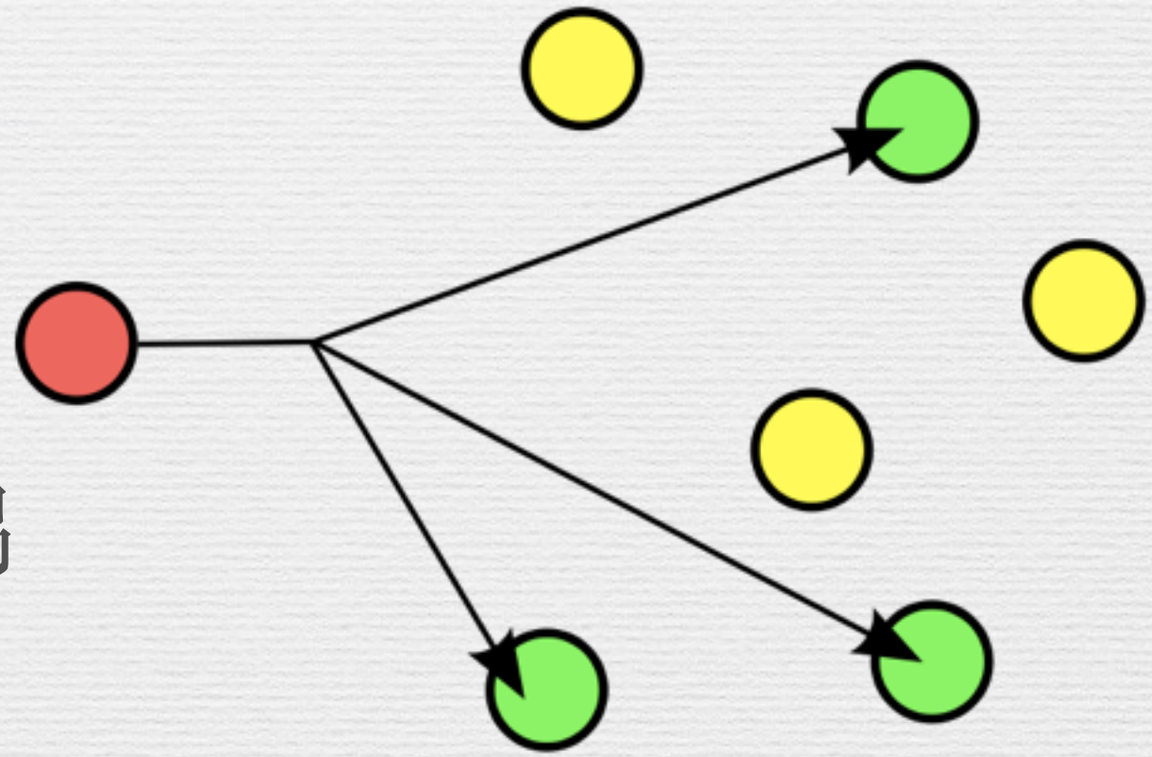
- 取决于排队机
- 持久化（Persistence）：Event Sourcing，WAL
- 复制（Replication）：排队机同步复制
- 冗余（Failover）：应用中间件多活

消息总线


柜台
监控管理服务
主题管理服务
排队机服务
去重服务
业务服务
消息总线
日志服务
消息中心服务
交易前置服务
专线
交易前端、DMA等

多播（Multicast）

- 出口带宽O(1)
- 发送端与接收端解耦



总线协议

- 每一个可靠多播通道具有一个唯一的Subject
- 每一个通道的消息具有一个序号，接收端根据序号顺序处理消息
- 多个服务可以产生同一个Subject的消息
- 相同序号的消息必须一样

可靠传输

- 为了保证可靠传输，需要接收端确认消息（ACK）
  - 基于Positive ACK（TCP）
  - 基于Negative ACK（PGM）
- 多播采用Negative ACK的原因
  - 接收端的ACK可能占满发送端的入口带宽
  - 发送端耦合接收端，不好Scale

拥塞控制

- 发送端的发送速度不能超过接收端的接收速度
  - 基于ACK的拥塞控制（TCP、PGMCC、TFMCC、ORMCC）
  - 基于NACK的拥塞控制
- 局域网的多播拥塞控制基于NACK
  - 定位最慢接收端算法复杂
  - 局域网丢包少，不会出现由于NACK导致发送速度逼近0的问题

应用框架

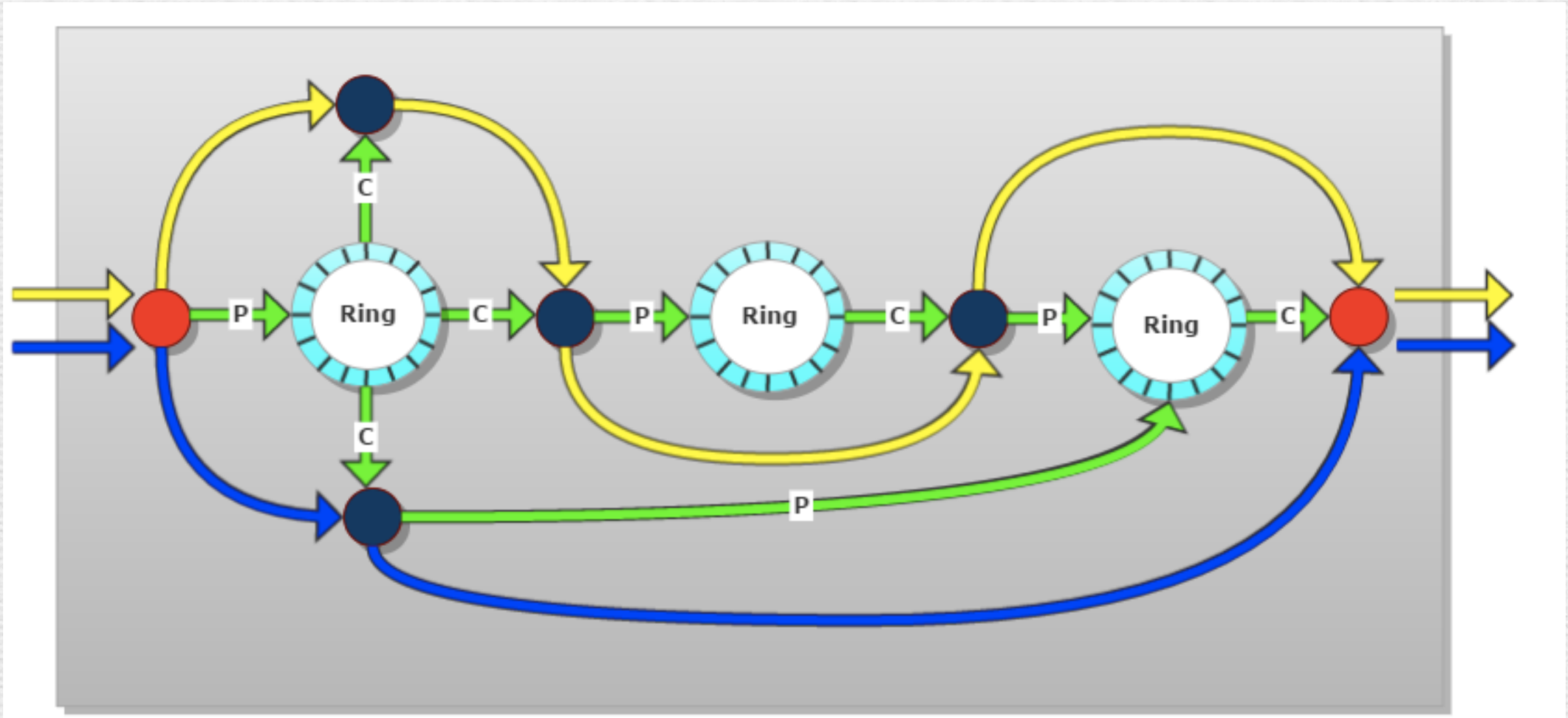
- 模块化
- 事件驱动、事件流
- 并发模型
- 应用协议

模块化

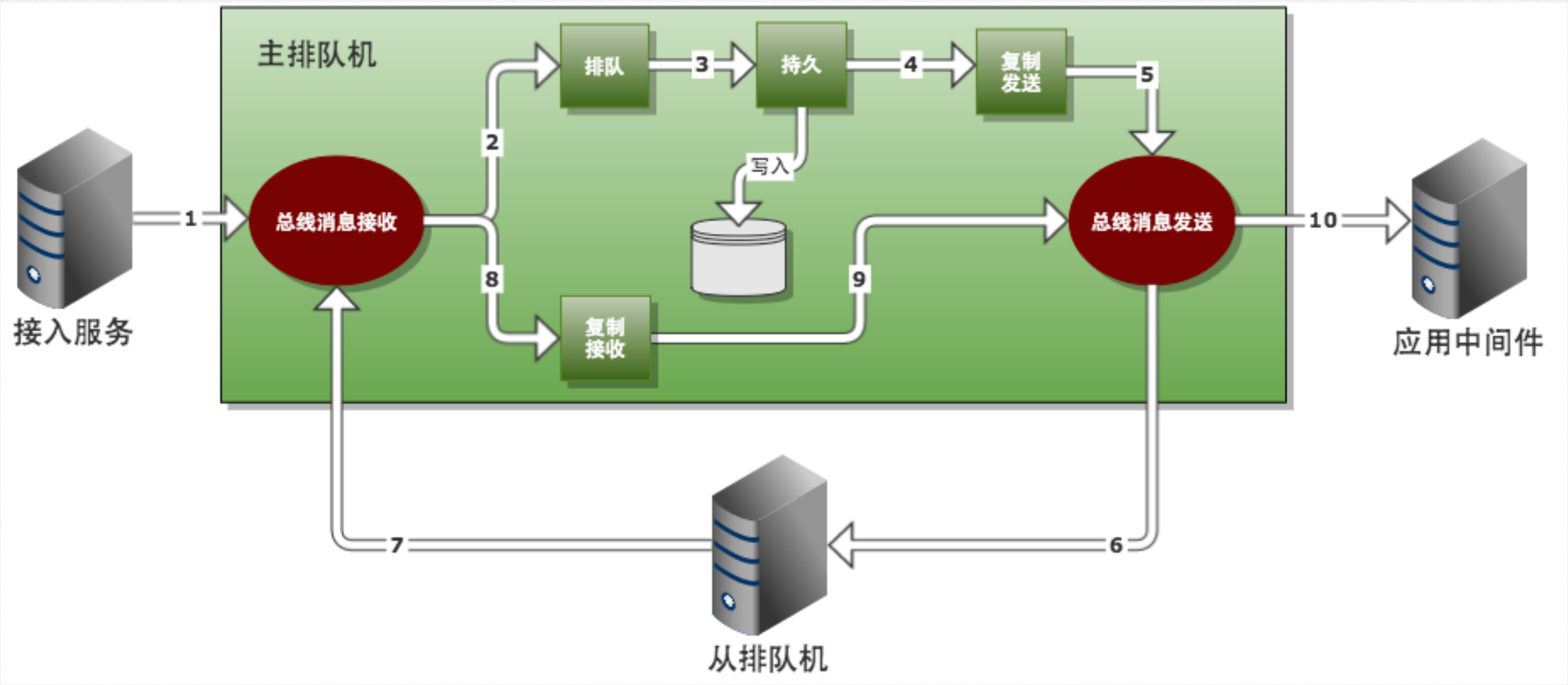
- 库一级的模块化
- 组件一级的模块化
- 服务一级的模块化

事件流

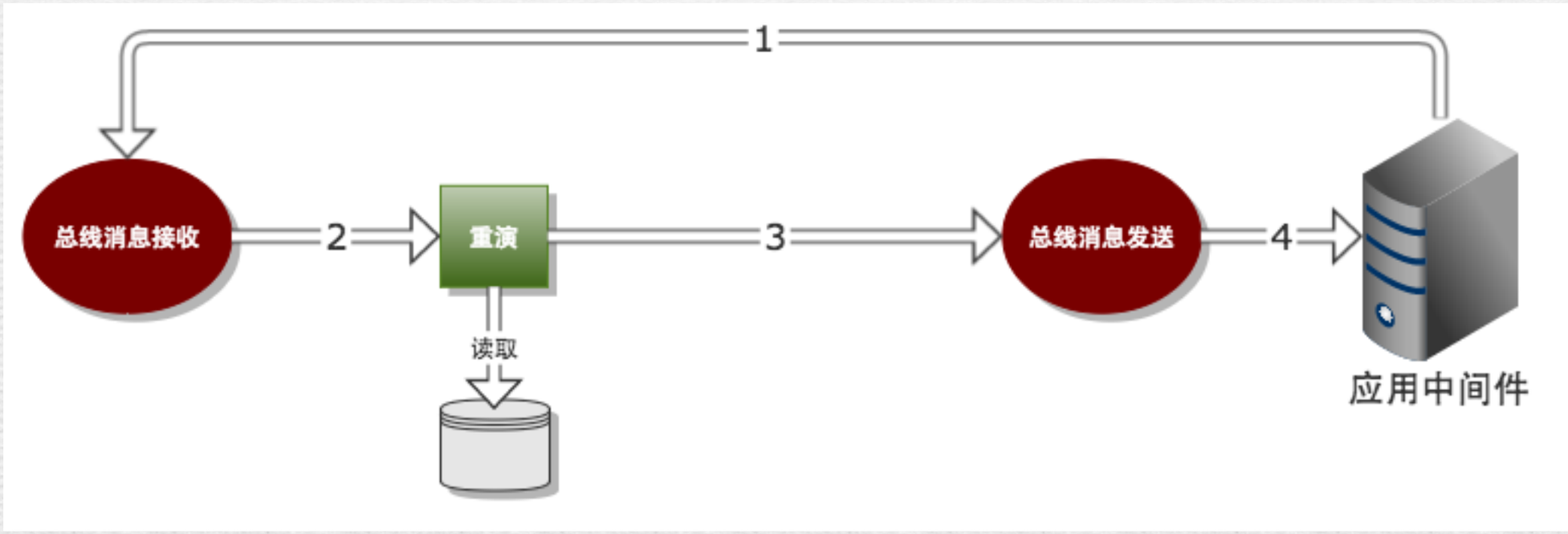
- 高度可配置
- 处理器与RingBuffer多对多的关系



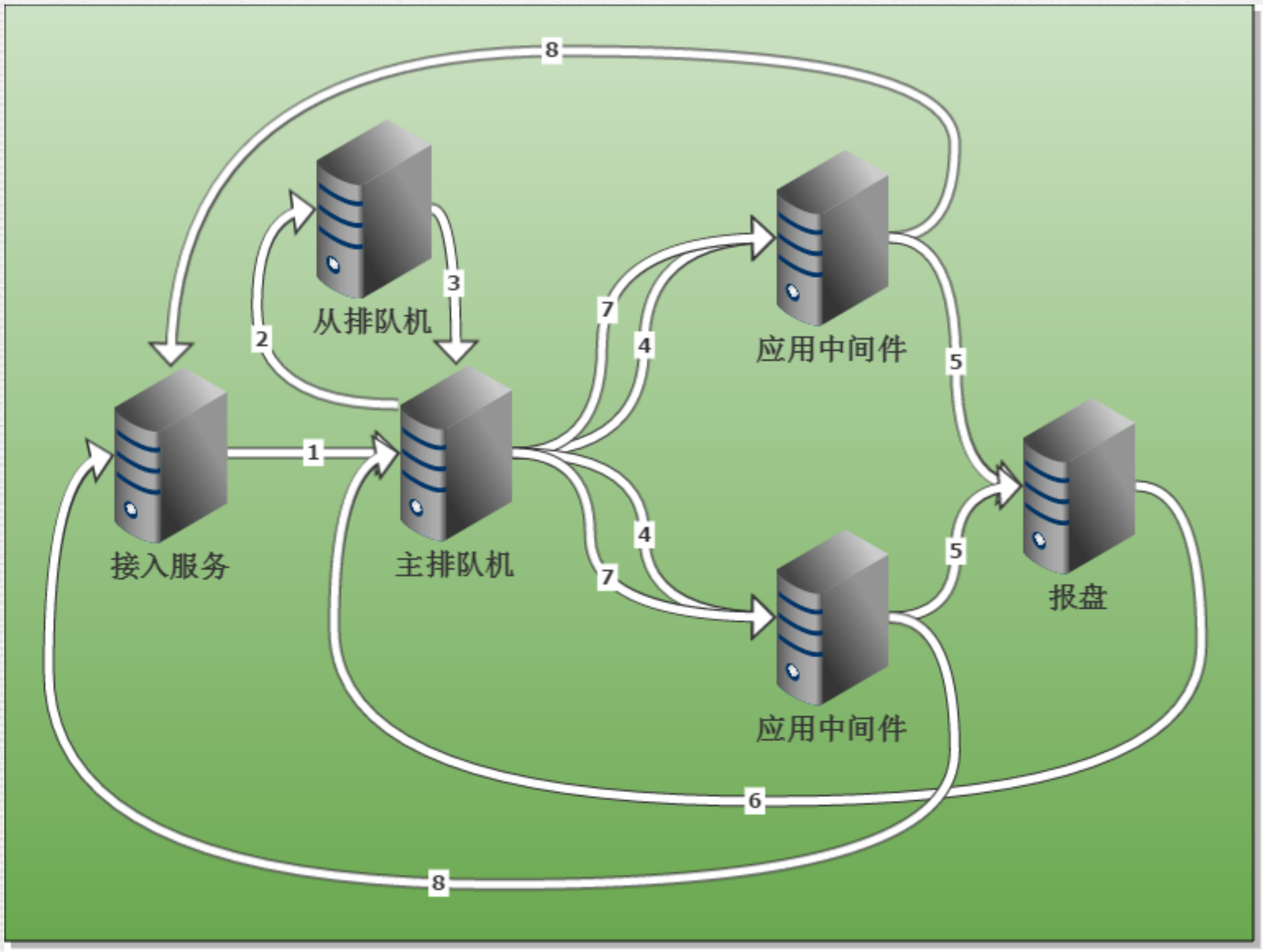
示例：主排队机事件流


主排队机
排队
持久
复制
发送
写入
总线消息接收
总线消息发送
复制
接收
1
2
3
4
5
6
7
8
9
10
接入服务
应用中间件
从排队机

重演消息流



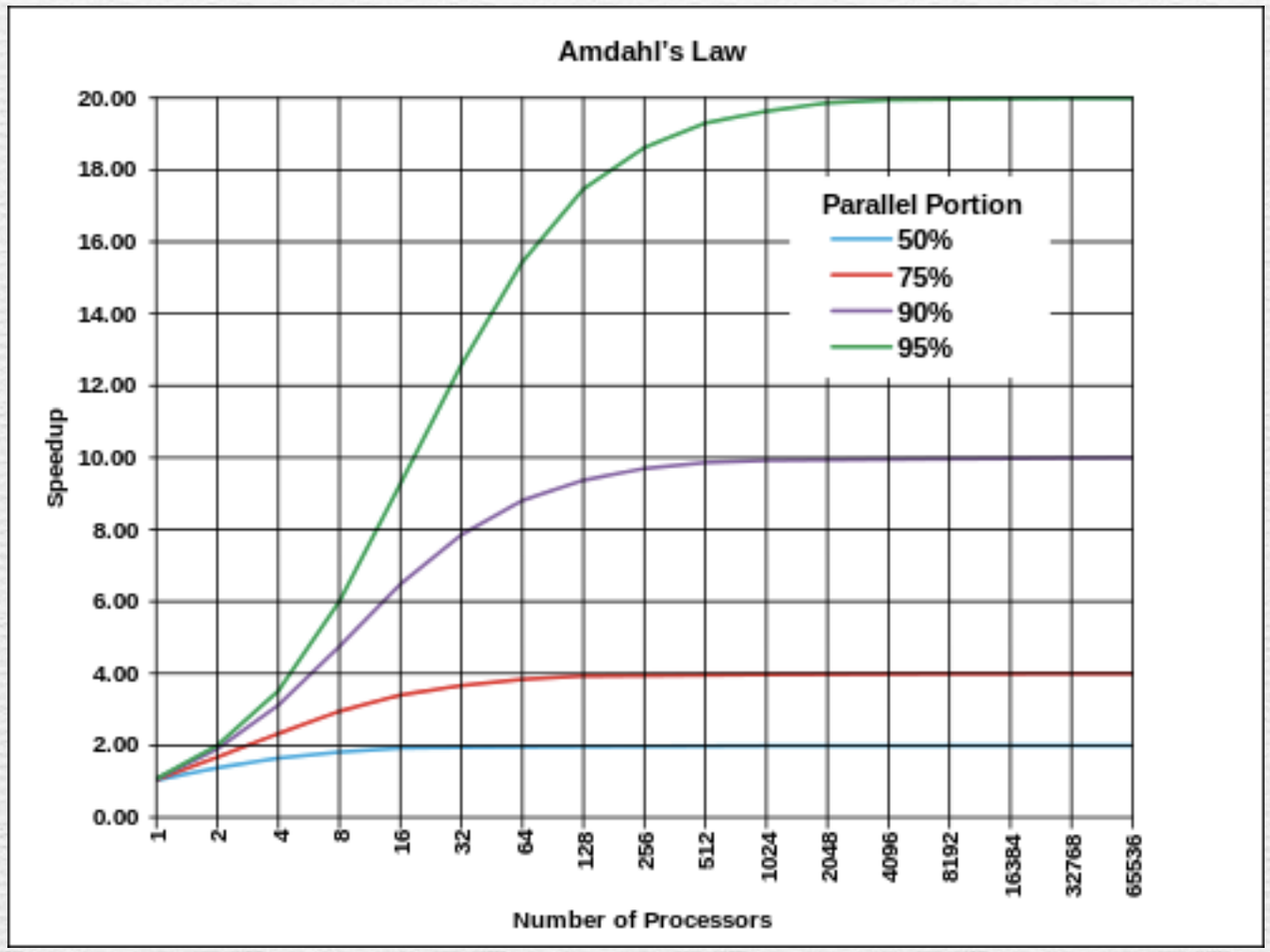
示例：消息流



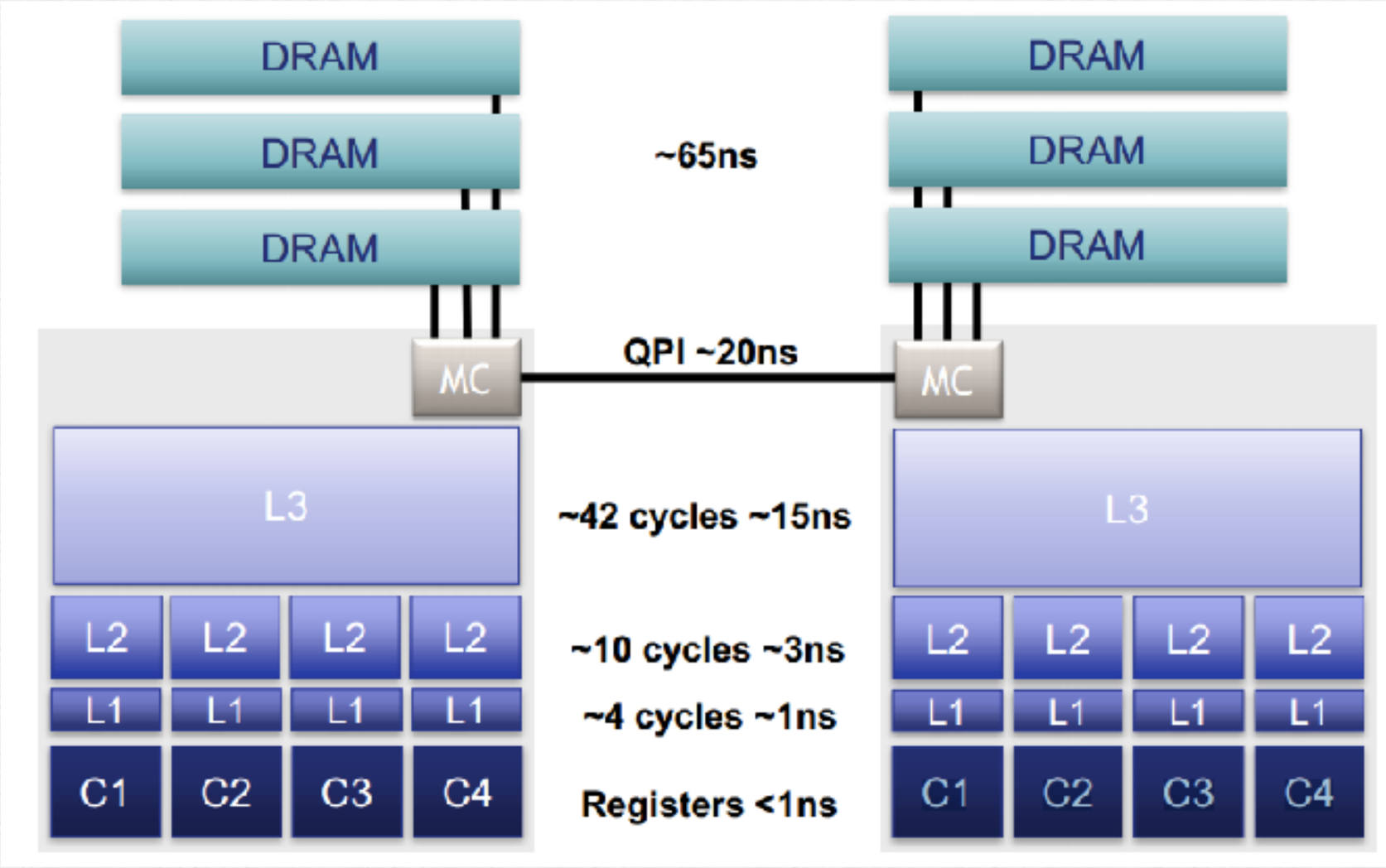
并发模型

- EDA、无共享状态
- 无锁
- 确定性
- 无需事务

Amdahl's law


Amdahl's Law
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Speedup
Parallel Portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors

线程间通信



- 基于Disruptor的消息分派与流水处理

应用协议

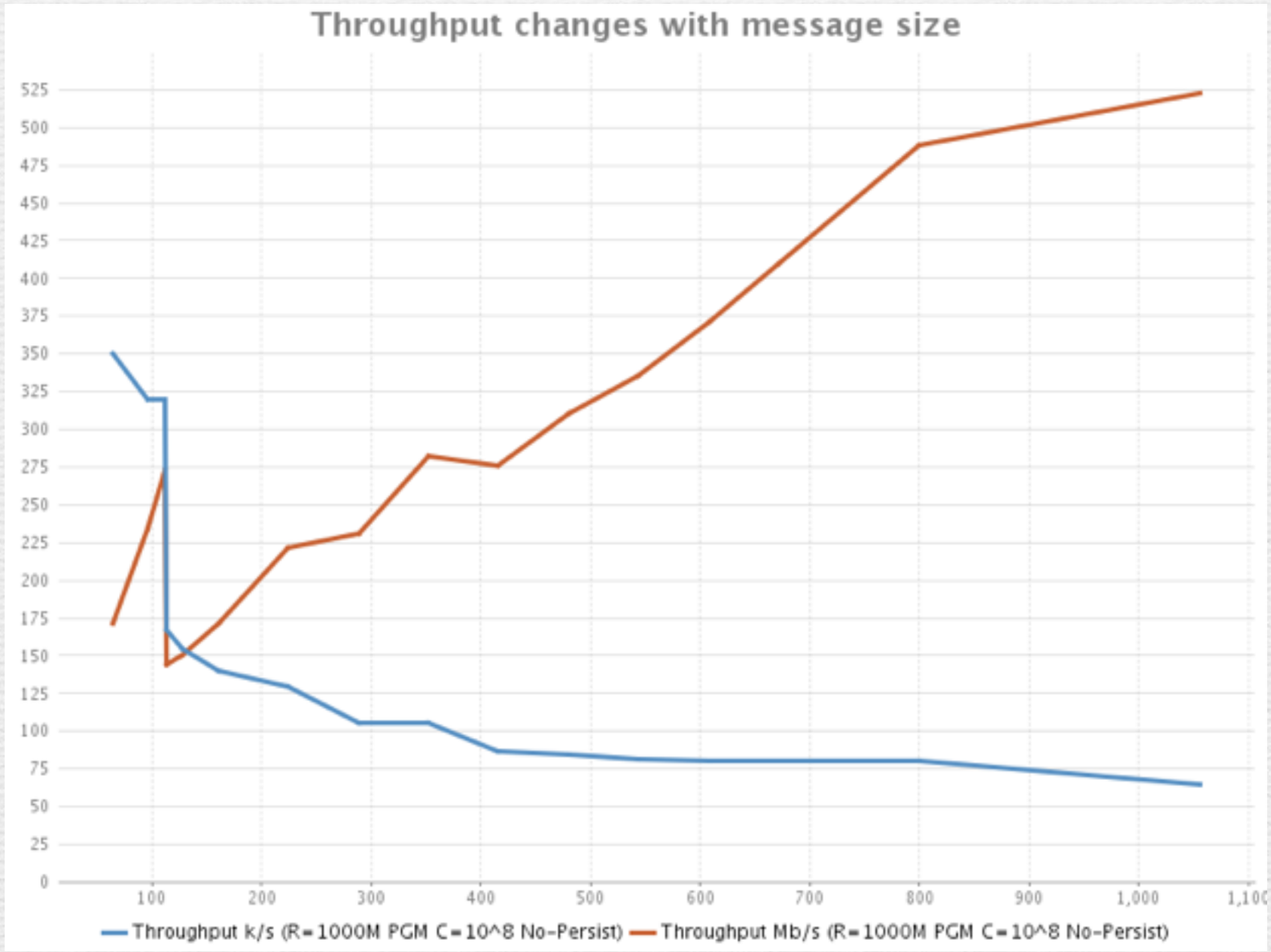
- 以流的方式操作数据
- 无需动态分配内存，也就不需要垃圾回收
- 不产生数据的拷贝（ZeroCopy）

敏捷

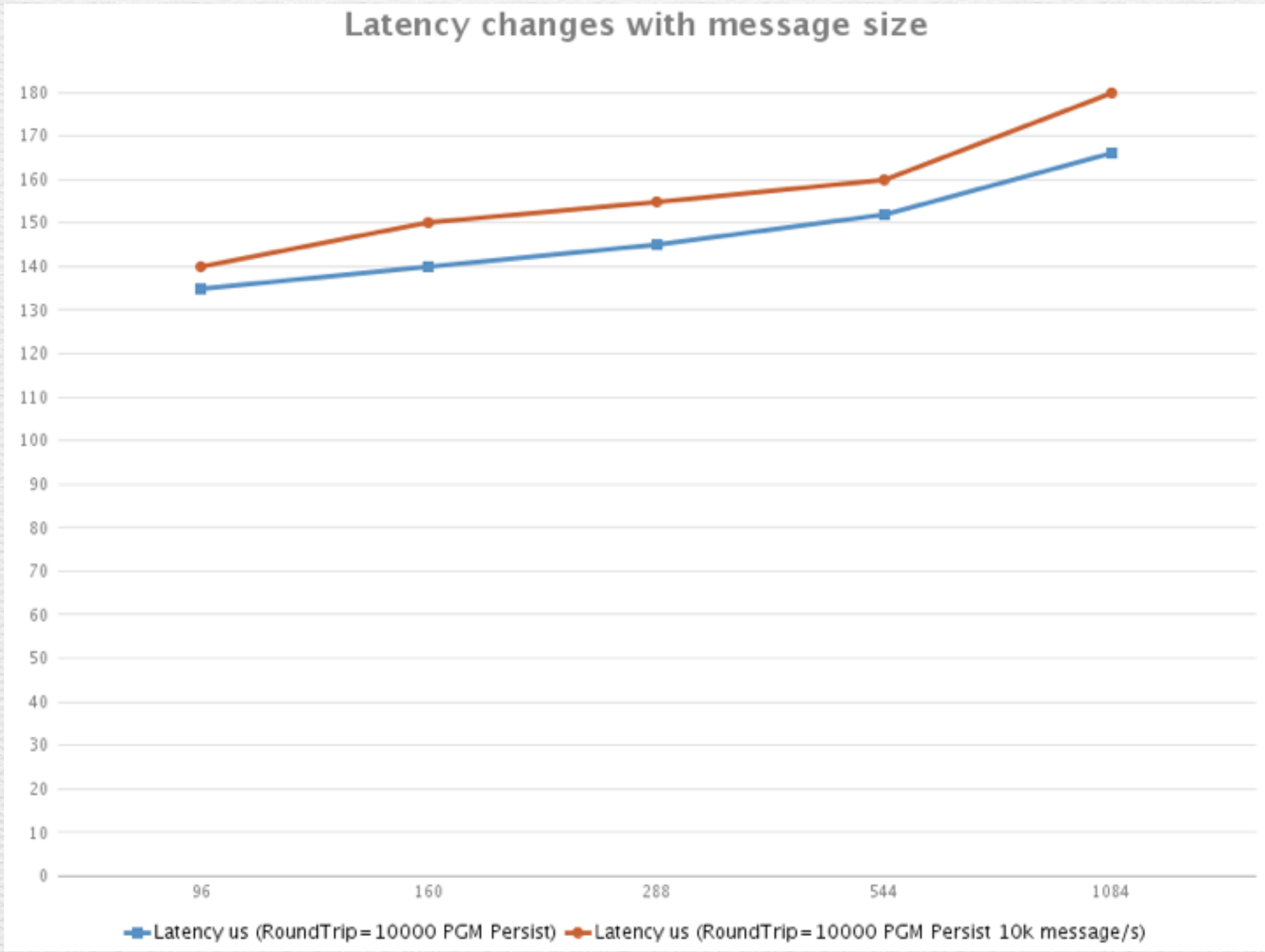
- 开发：模块化、重用
- 自动构建（Maven）
- 自动化测试（BDD）
- 自动化部署与监控（Docker、Puppet、JMX）
- 运营

性能——吞吐

- 轻松达到300kmps

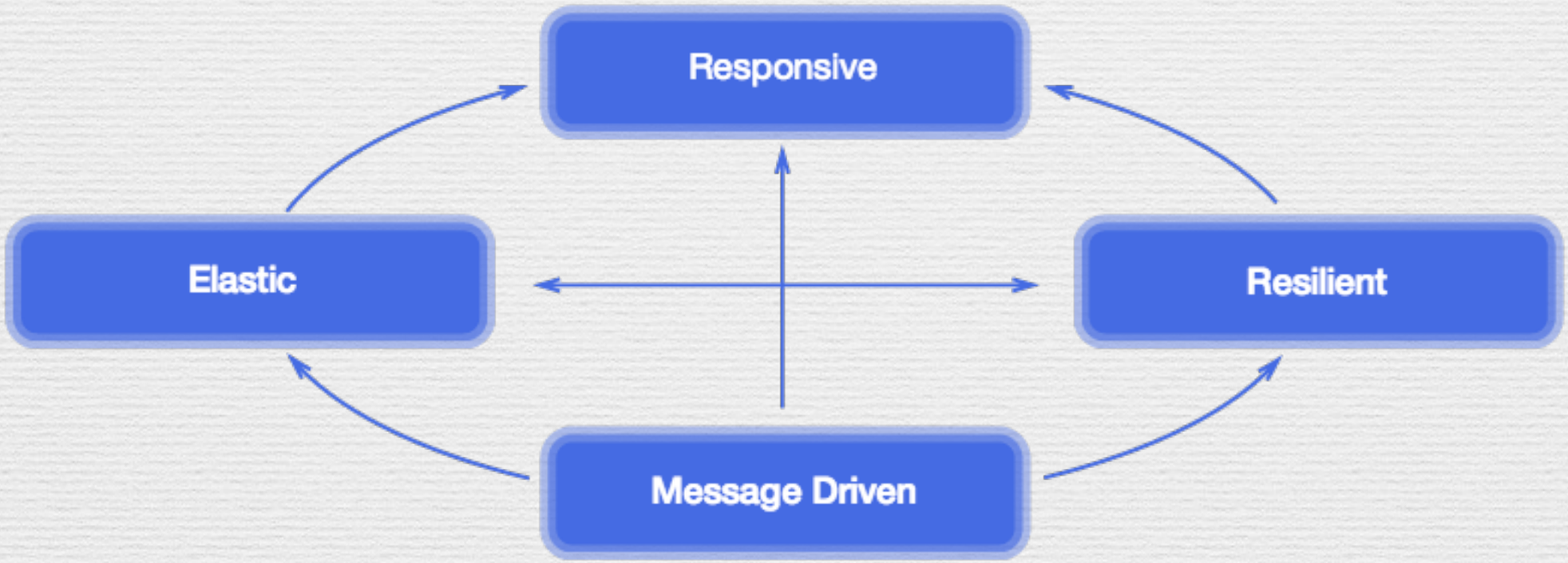


性能——延时


Latency changes with message size
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
96
160
288
544
1084
Latency us (RoundTrip=10000 PGM Persist)
Latency us (RoundTrip=10000 PGM Persist 10k message/s)

设计理念

- The Reactive Manifesto



Reactive to Events

- 异步
- 事件一等公民
- 对事件建模、对事件处理器建模、对事件流程建模

Reactive to Scale

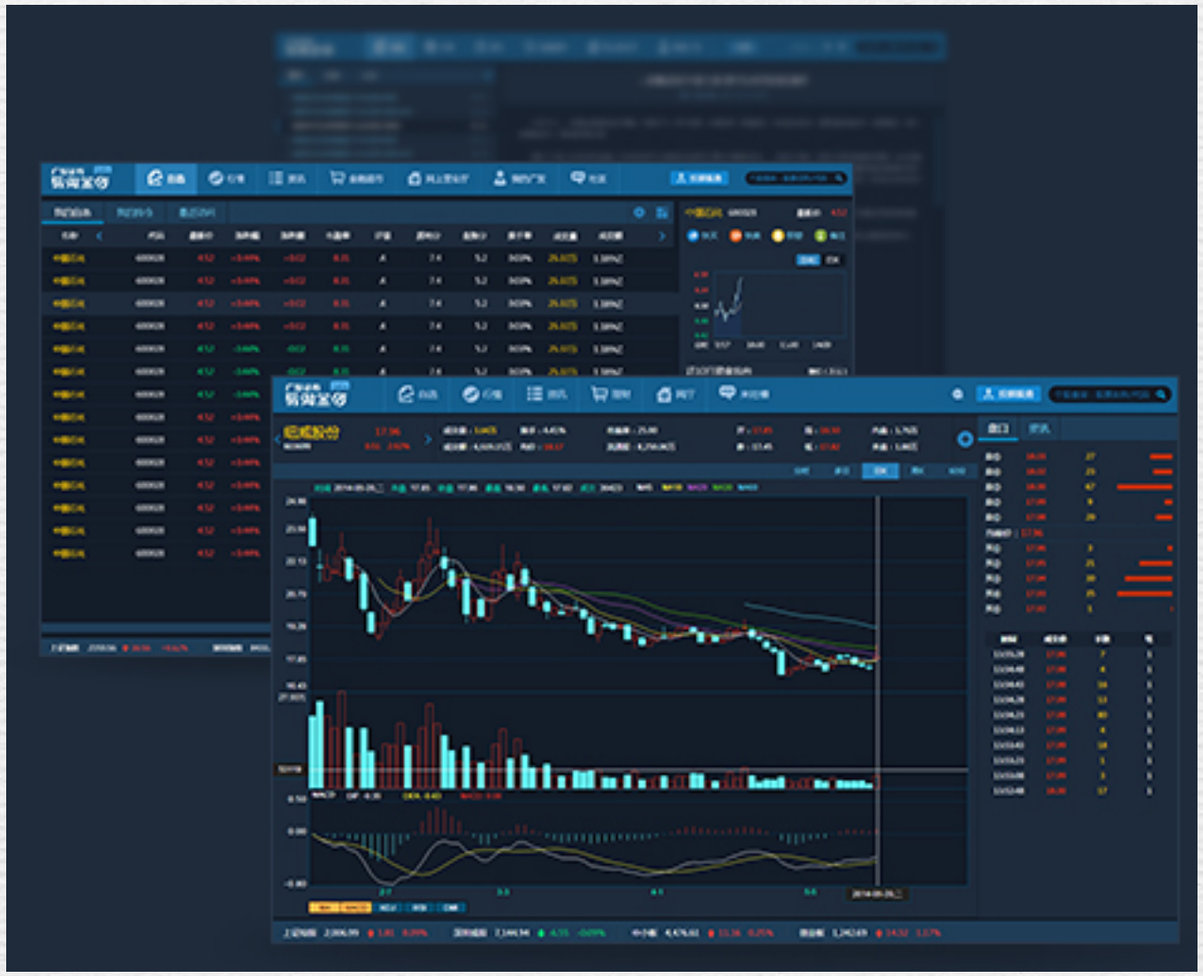
- Location unaware
- Scale up
- Scale out

Reactive to Resilience

- 反脆弱
- 错误也是一种事件
- Supervisor：监控，日志，动态路由，大数据

Reactive to User

- 用户体验
- 实时



Q&A

- 个人博客：liebo.github.io

- http://it.gf.com.cn