

证券交易系统架构设计

—— 挑战与实践



上海证券交易所 技术开发部

陈晨

cchen@sse.com.cn

+86 138-0199-1611

- 上交所交易系统介绍
- 交易系统技术架构
- 挑战及解决之道
- 交易系统的未来

上交所交易系统介绍

发展历史



- 1990 年 11 月 26 日成立，同年 12 月 19 日正式营业。

发展历史

上交所

1990.12.19

- 开业第一天即采用电子撮合系统
- 每秒处理 3 笔，月处理 2

1992.12

- 系统升级，采用 UNIX 小型机
- 每秒 200 笔，日处理 200 万

1993.1

- 采用卫星广播行情
- 双向卫星接收订单

1997, 1999

- 系统两次升级，性能提升至每秒 2 万笔，日处理 800 万

❖ 开业的第一天就采用电子撮合系统进行交易撮合

- 基于 Novell 服务器的局域网络
- 每秒处理 3 笔业务，月处理成交 2 万笔
- 市场的委托、行情、成交回报等环节仍需要手工完成

❖ 1992 年 12 月，系统升级

- Novell 主机更换为基于惠普小型机 UNIX 操作系统
- 每秒 200 笔，日处理能力 200 万笔
- 1993 年，采用单向卫星广播行情，双向卫星接收报单

❖ 1997 年和 1999 年进行了两次设备和应用的重大升级

- 系统处理能力提高到每秒 2 万笔，日处理能力 800 万笔
- 后随着不断的扩容和改造，性能和容量不断被刷新

上交所

- ◆ 2009.11.23
 - 新一代交易系统上线
 - 使用多主机并行撮合
 - 最高支持 10 万笔每秒
 - 全天容量 1 亿笔订单
 - 账户容量 1 亿

- ◆ 2014.11.17
 - 沪港通业务上线

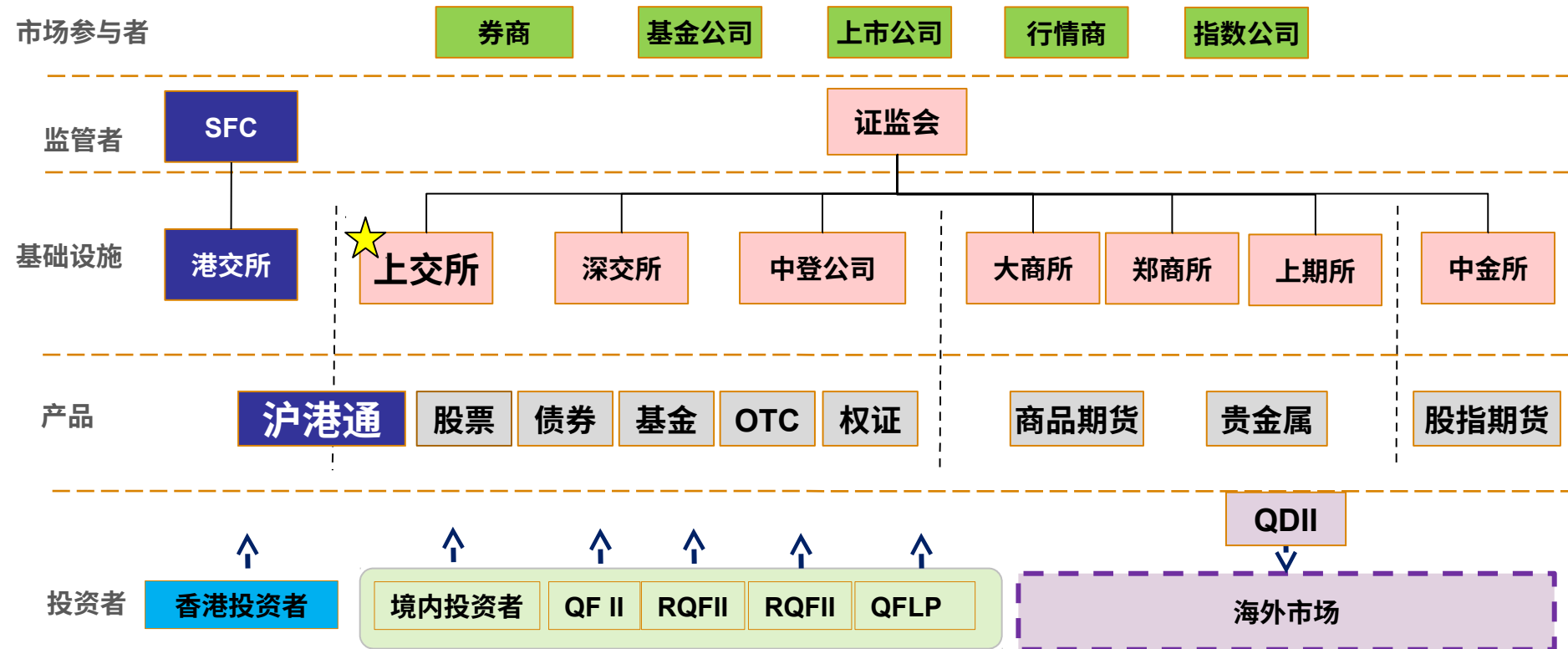
❖ 新一代交易系统的上线

- 2009 年 11 月 23 日，新一代交易系统上线
- 使用多主机并行撮合
- 最高支持 10 万笔每秒
- 全天容量 1 亿笔订单
- 帐户容量 1 亿

❖ 基于新一代交易系统，2014 年 11 月 17 日沪港通业务上线

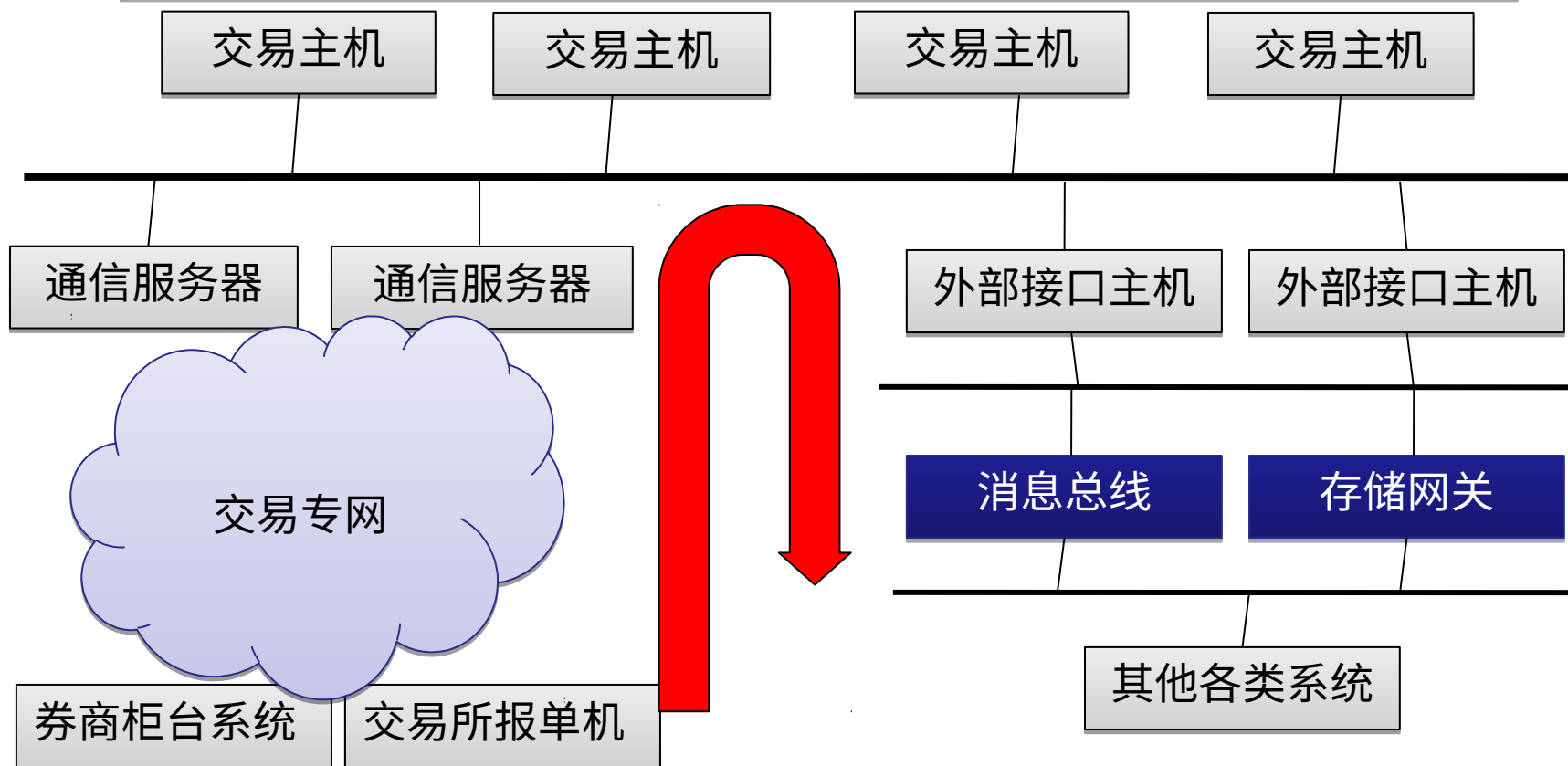


市场结构图



系统结构图

核心交易系统内部结构



交易系统技术架构

交易系统技术架构

交易系统的三层式划分

❖ 交易层

撮合器 1

- 分配股票（银行、汽车）
- 分配基金、ETF

撮合器 2

- 分配股票（制造业、零售）
- 分配债券

❖ 定序层

定序器 1

定序器 2

❖ 接入层

接入点 A

接入点 B

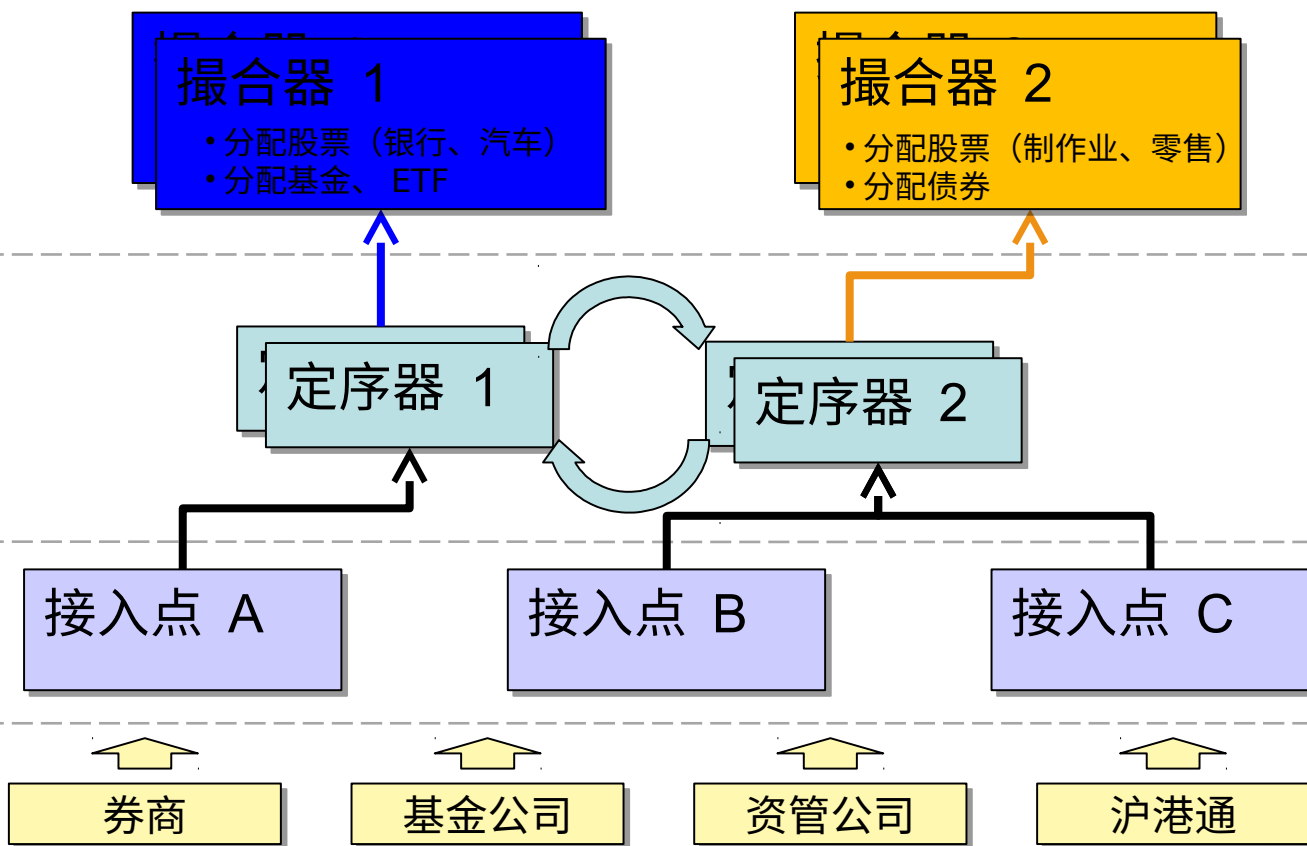
接入点 C

券商

基金公司

资管公司

沪港通



功能

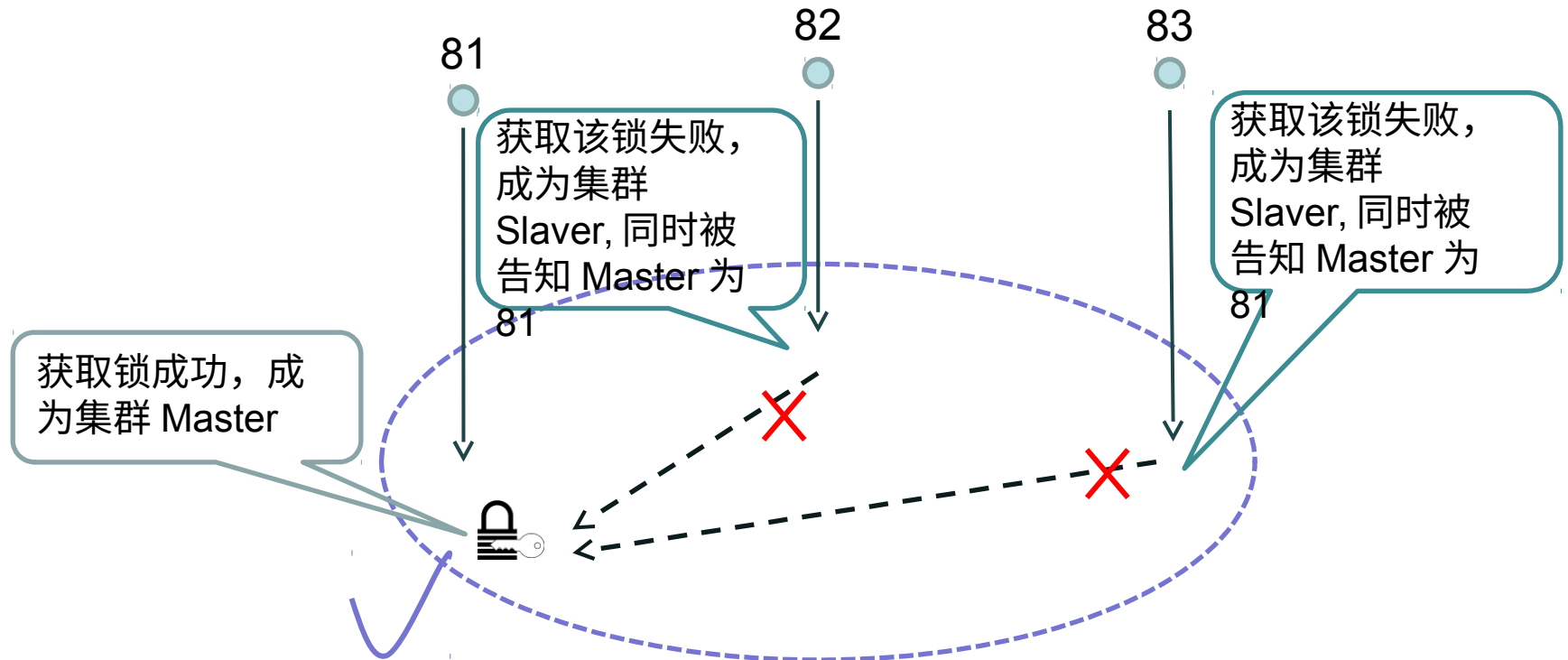
- 执行交易业务逻辑

实现模式

- 同组交易主机为接收单一输入序列的状态
- 从一系列独立运作的交易主机节点中自动选举产生主节点
- 可利用集群锁服务来选举主节点
- 备机可选择“重演”或者“重放”模式

交易层 集群锁管理模式

基于 OpenVMS 的 Lock 机制，实现了一套用于集群 (Group) 管理，集群内各主机同步、通信的工具库



功能

- 点对点的路由转发
- 将券商端的订单提交给定序层实例

实现模式

- 提供一组彼此对等的实例同时提供对外服务
- 不需要考虑主从划分和失效接管

定序层

功能

- 高可用设计的关键、确定全序
- 持久化及保持动态一致性

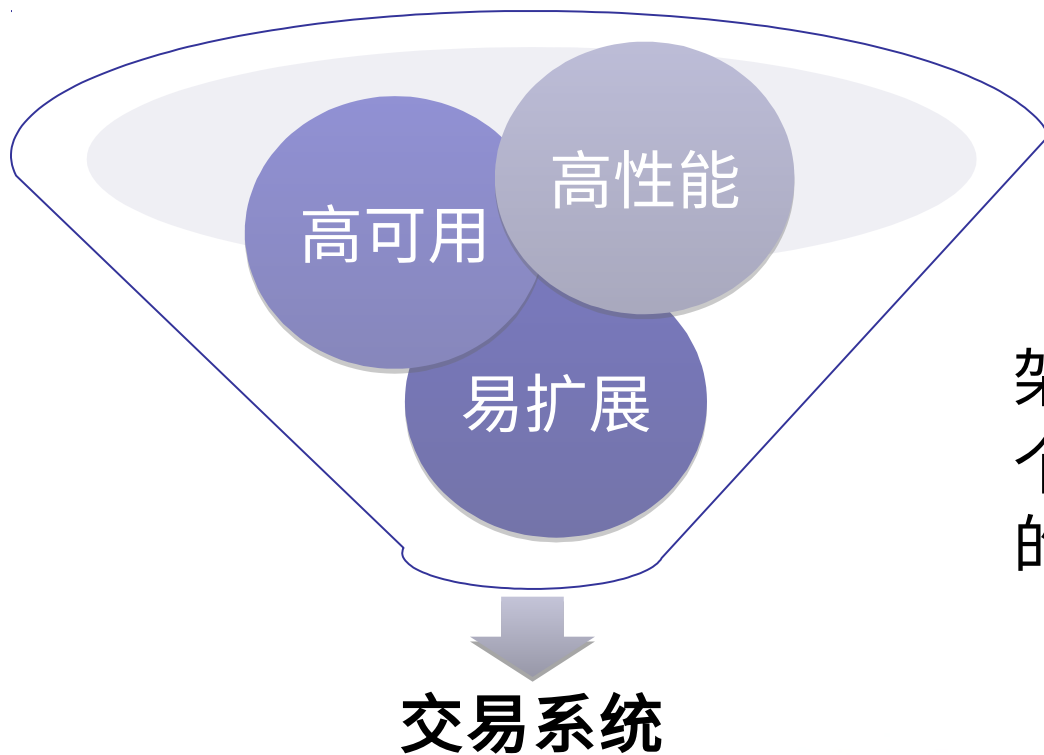
实现模式

- 多播通信机制
- Paxos 算法、虚同步
- 自主研发、商业软件、开源软件
- 逻辑概念，可采用单独排队机定序或者主撮合定序

挑战及解决之道

交易系统面临的挑战

交易系统设计之初就要考虑到如何满足和平衡各方面的技术需求

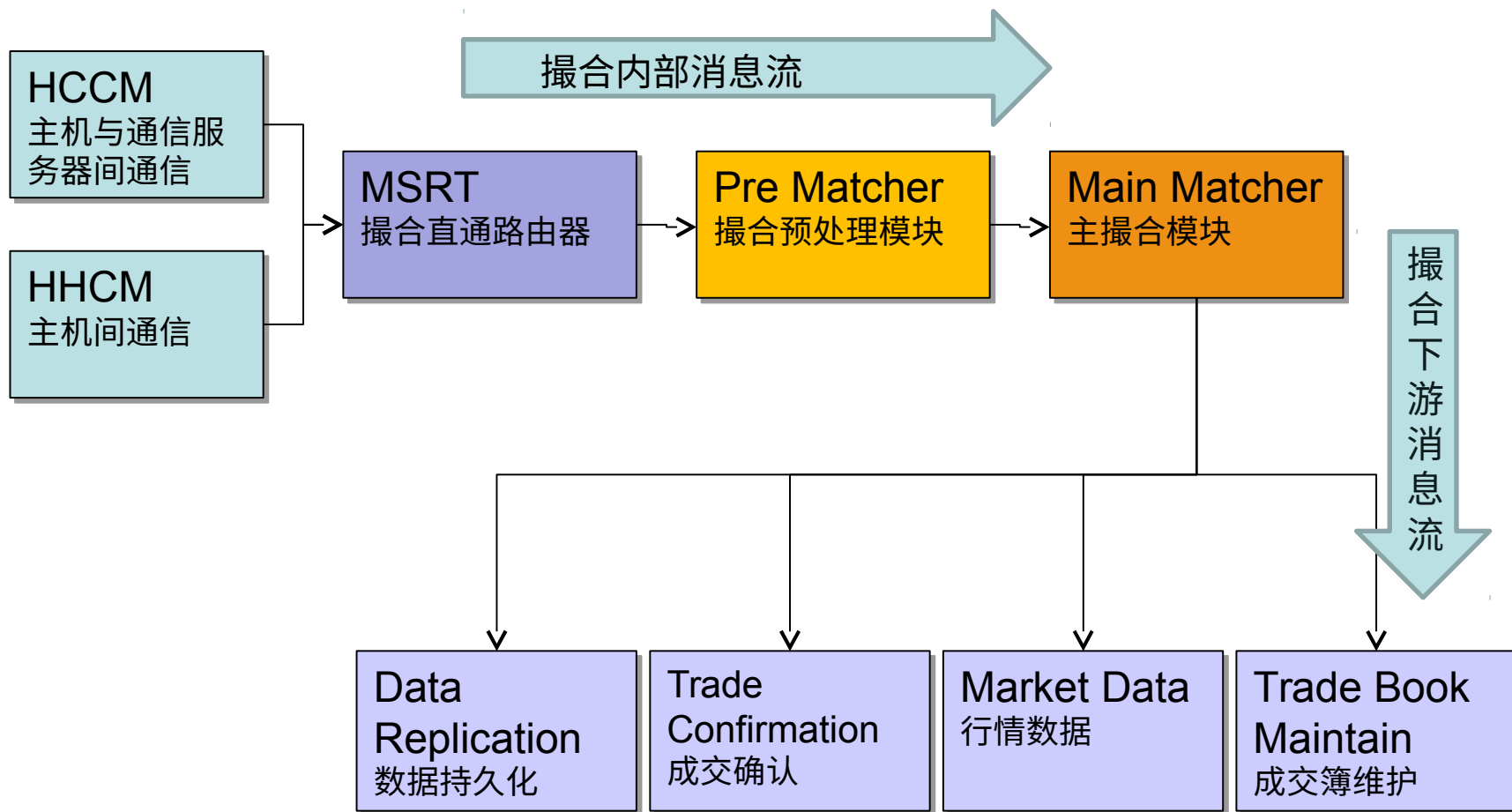


架构设计是一个**平衡**和**抉择**的艺术

■ 衡量交易系统性能主要指标

- 吞吐量
- 订单时延
- 系统容量

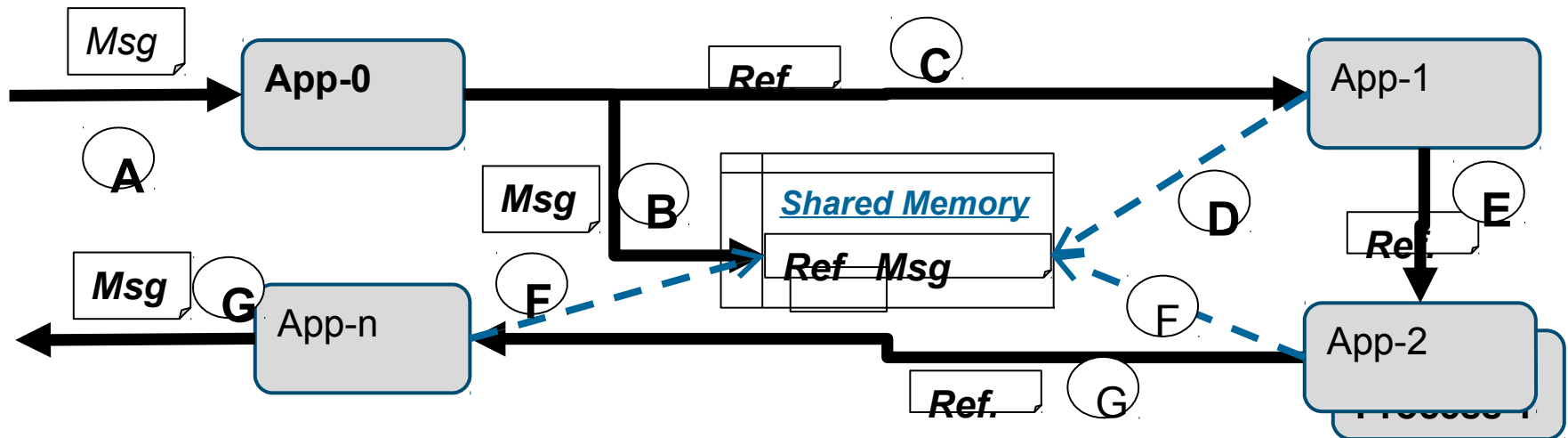
1. 流水线化内存撮合



2. 内容和键值分离

■ 精简的进程间通信消息

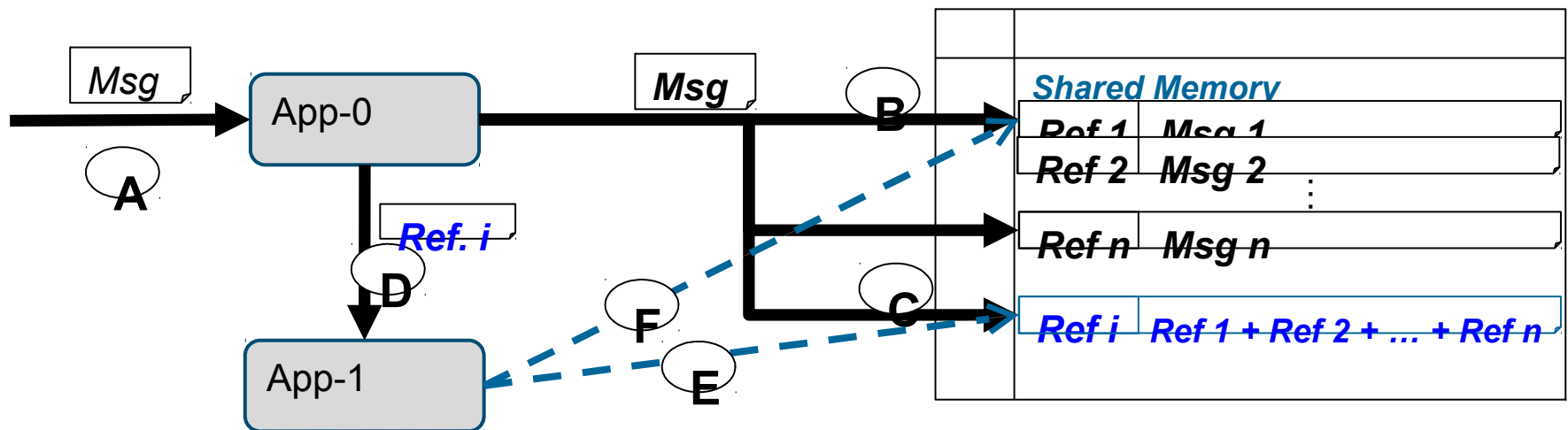
- 消息 body 通过内存缓存；
- 进程间传递短小的消息 header ；
- 进程通过 header 信息，访问内存获取消息实体；



3. 数据打包处理

■ 消息的打包处理

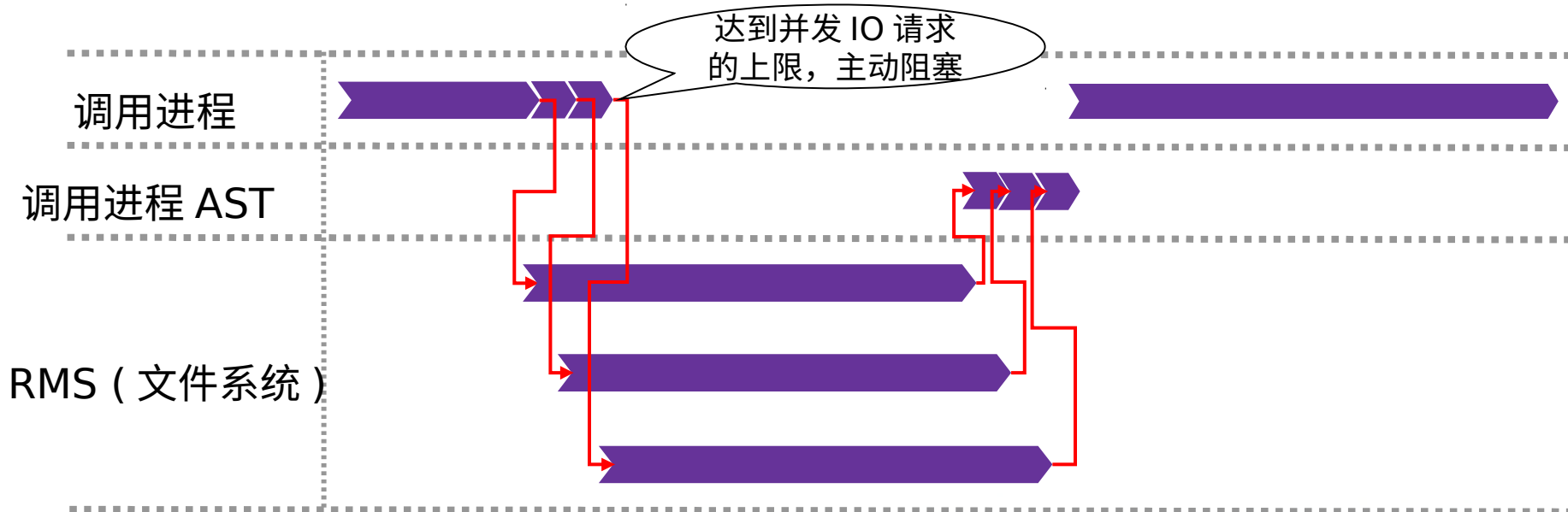
- 请求消息通过用 header 表示后很短小，支持多条打包模式；
- 申请新的共享内存消息，消息体中包含多个请求实体的 header ；
- 实际消息通信中传递打包消息的消息 header 即可；
- 进程接收消息后，根据打包消息 body 中的多个实际消息 header ，逐一处理。



4. 多线程异步 IO

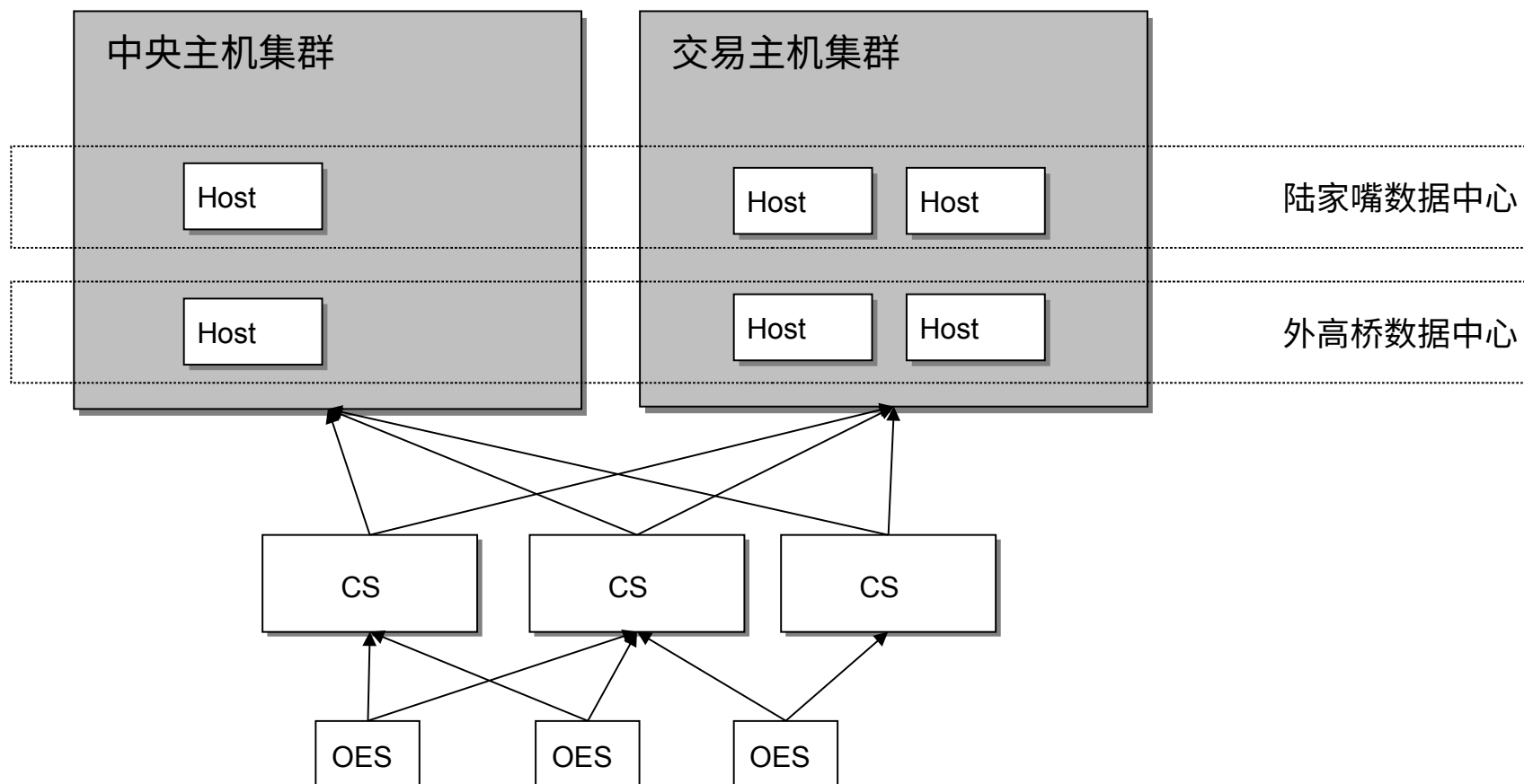
■ 应用异步 IO 提升性能的典型案例

- 调用进程通过异步 IO 连续的抛出一组 IO 请求，RMS 可以并行地处理这些请求，成倍地提升 IO 吞吐量；
- 连续发出的请求应当有一定限制，当未完成的请求数量达到限制时，调用进程可以主动将自己阻塞。



高性能

5. 多机并行



■ 衡量交易系统可用性主要指标

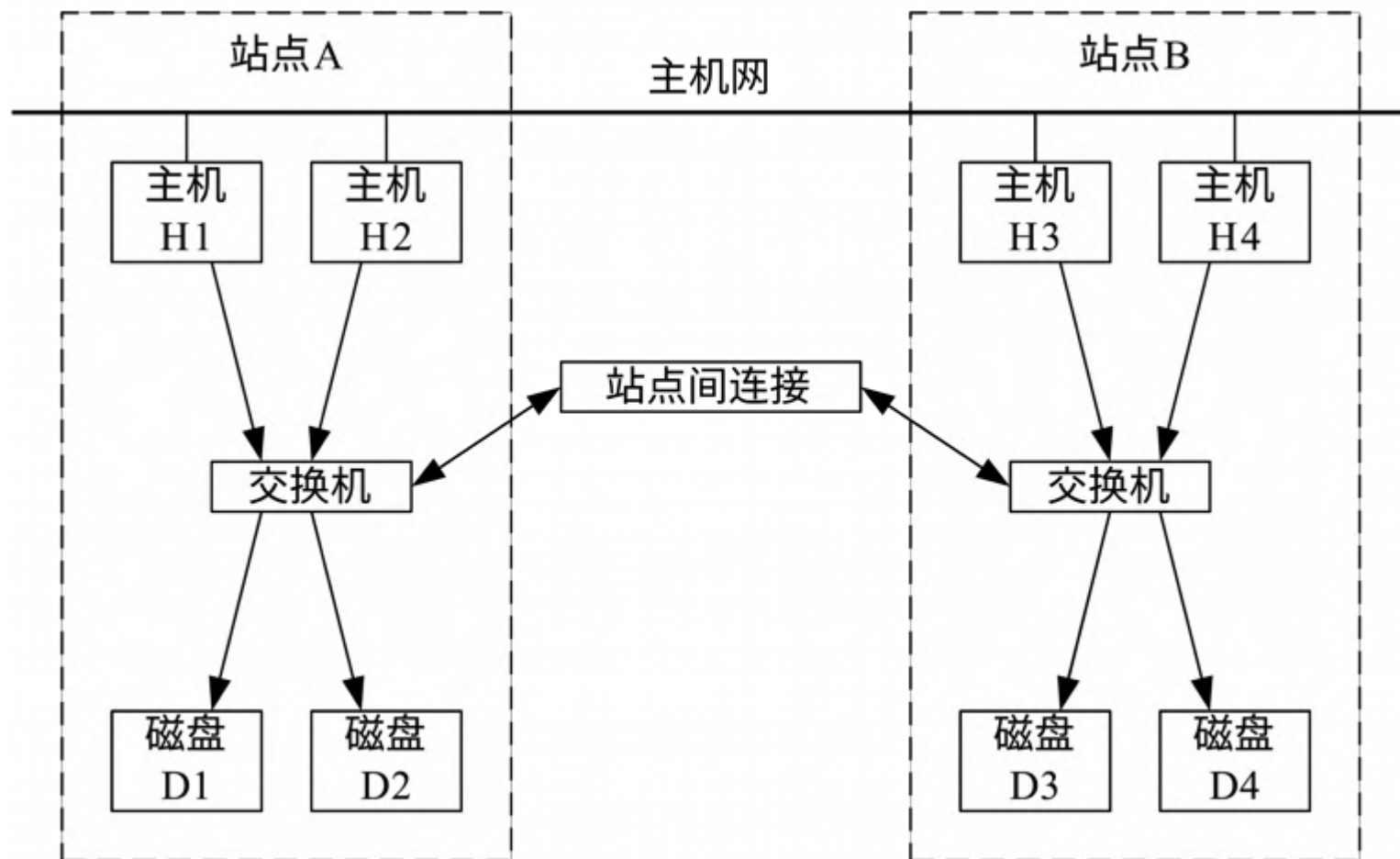
- 恢复时间目标 RTO （ Recovery Time Objective s ）
- 恢复点目标 RPO （ Recovery Point Objectives ）

高可用 技术抉择的难题

- 人工侦测故障 OR 应用程序自动侦测故障？
- 应对单点故障 OR 应对双点故障？
- 同城灾备同步复制 OR 异步复制？

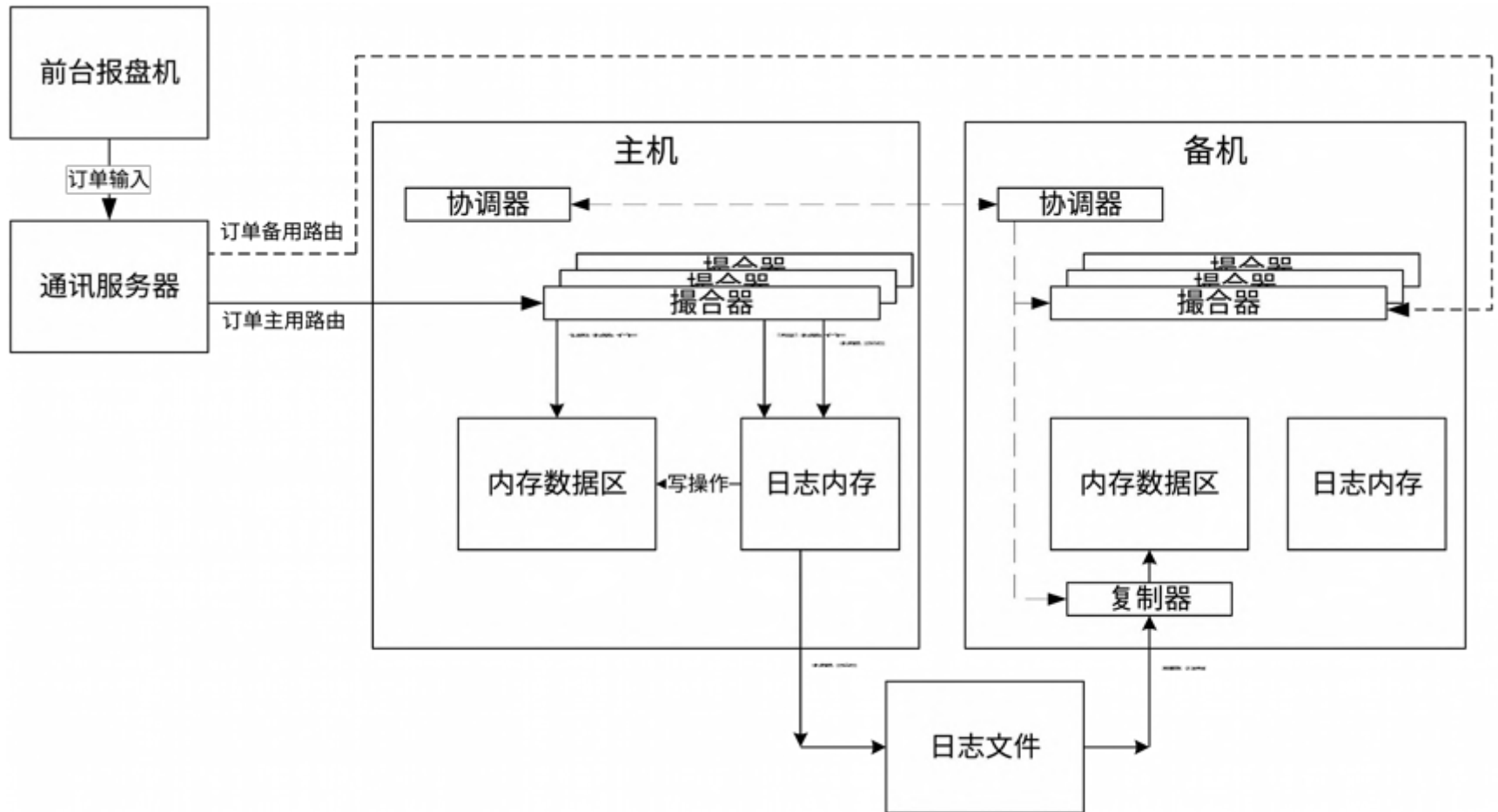
高可用

1. 站点备份



高可用

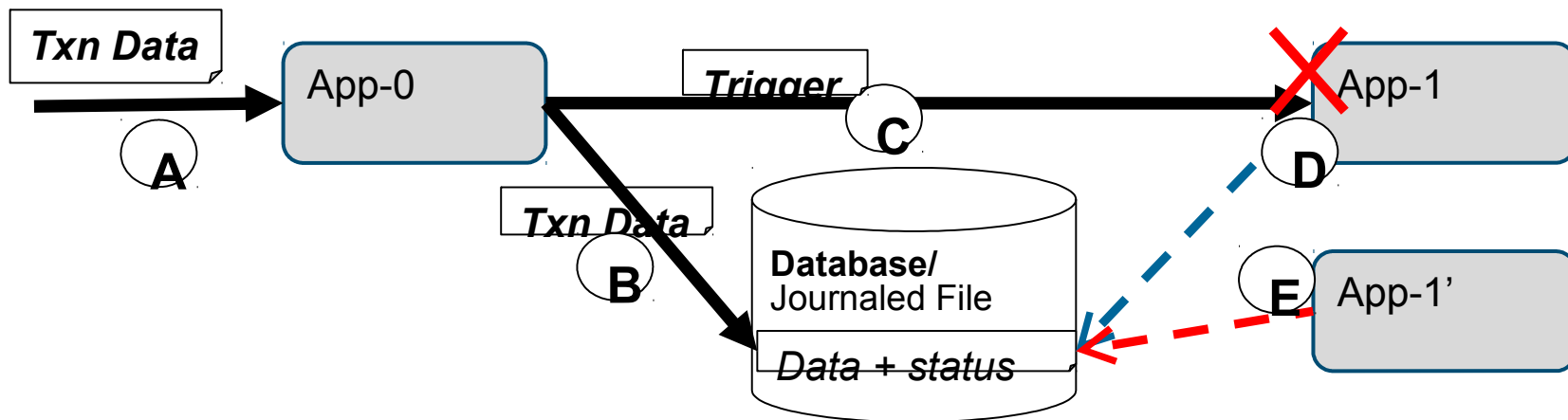
2. 主机备份



3. 进程恢复

■ 进程的恢复机制

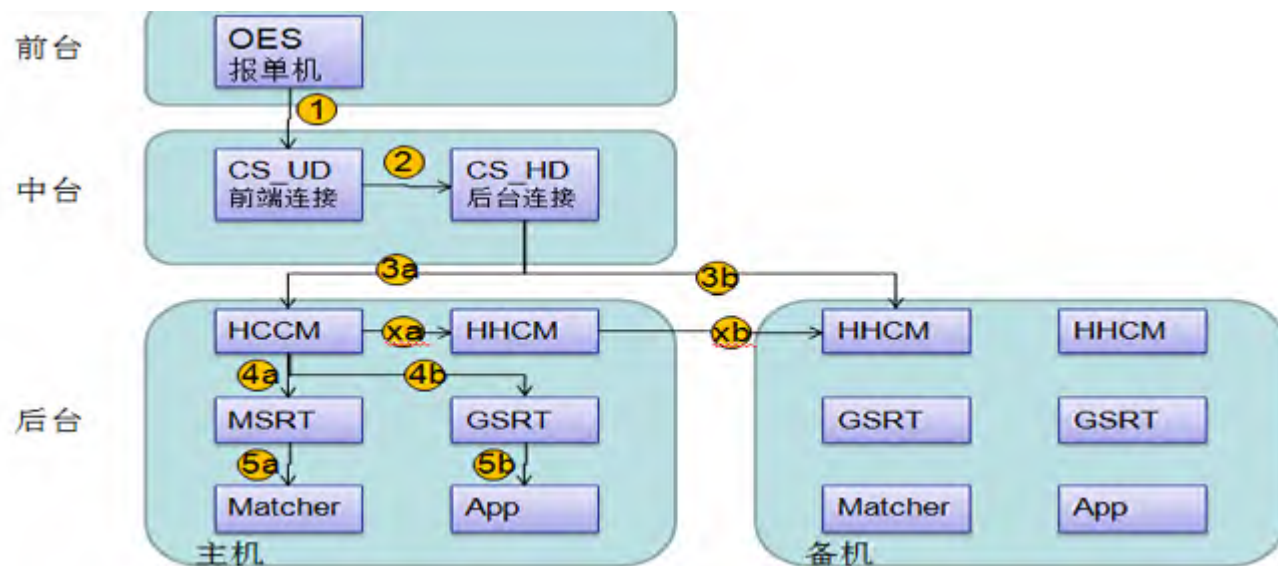
- 事务 / 请求数据带事务的文件存储，包含相应的状态位；
- 任何一个进程异常，根据事务文件中数据状态重演恢复；
- 无法应对程序本身逻辑错误



4. 消息重发 / 防重处理

■ 系统发生主备切换

- 切换完成后的自动通知机制
- 未响应消息重新路由机制
- 消息防重复处理机制



高可用

5. 流控机制

■ 流量 / 负载控制：

- ✓ 系统必须提供自保护机制来处理异常的大量或者突发交易量
- ✓ 包括主动控制和被动控制

■ 主动控制：

- ✓ 主动控制实现于前端（请求源端）会员，根据尚未响应的订单数量控制请求发送速度
- ✓ 系统可配置成会员端总体的请求数量不超过后台的处理能力，因此从源端控制整个系统的负载

■ 被动控制：

- ✓ 路由架构内置的自我保护机制
- ✓ 每个路由架构组件跟踪其输入 / 输出差异，并且根据监测的差异触发自我保护机制来或者阻塞消息流，或者弹回新的请求

■ 主要指标

- 扩展性衡量系统适应业务发展与变更的能力，既包括业务容量的扩展又包括业务模式的扩展

■ 应对方案

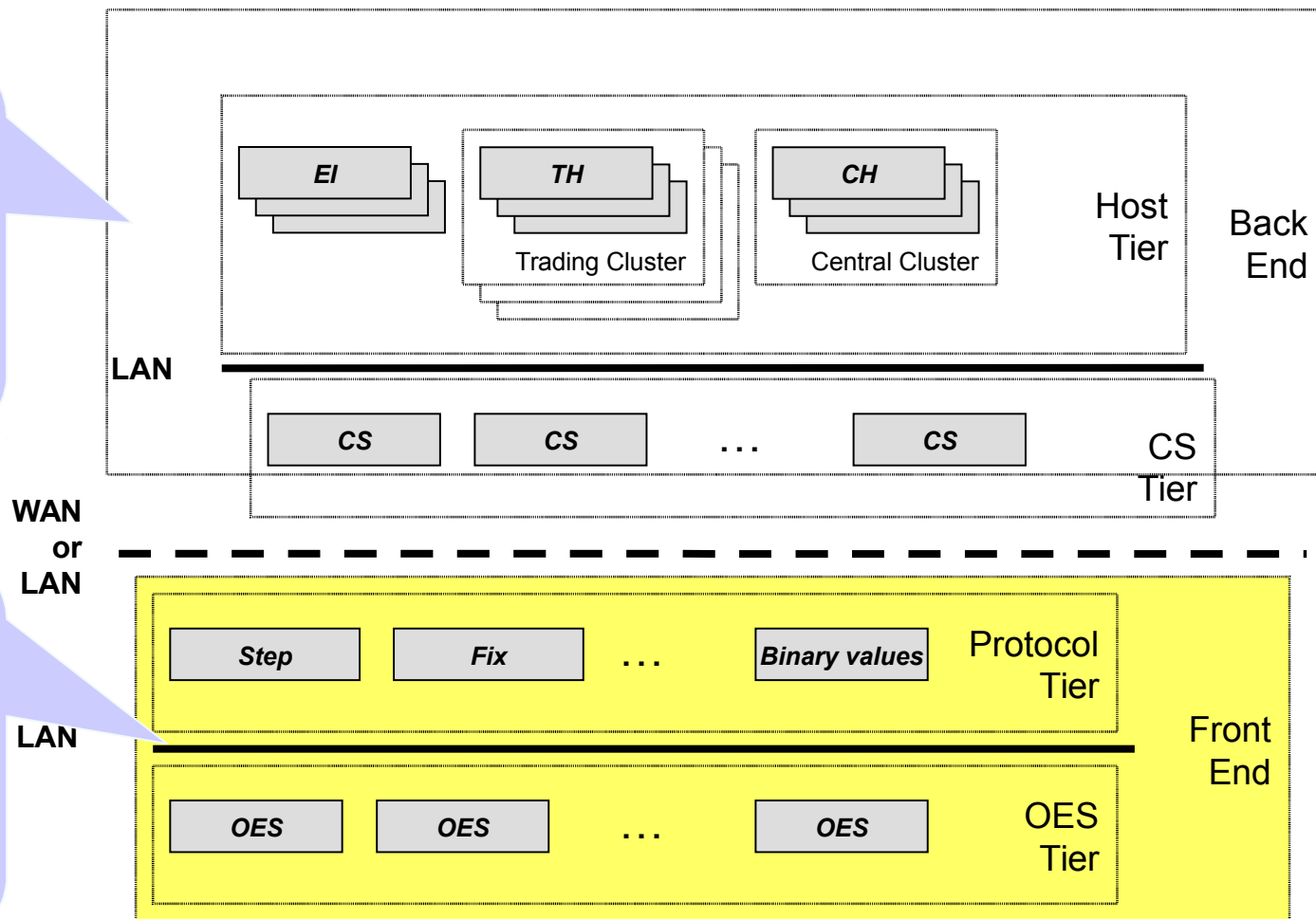
- 高扩展性需要在内部核心数据结构和接口定义上预留足够的扩展空间。
- 系统内部结构上，通过分层抽象服务使得某一个层次的升级更新不影响到全局架构，通过模块化设计使得某一个模块的变更不影响到整体稳定。

易扩展

1. 系统架构的扩展

后台的扩展

- 交易层中的各个平台，可以平行扩展设备，支持业务的容量和品种的发展
- 接入层中，可以根据网段规模和接入点无缝地进行平行扩展；每个接入点属于无状态设备

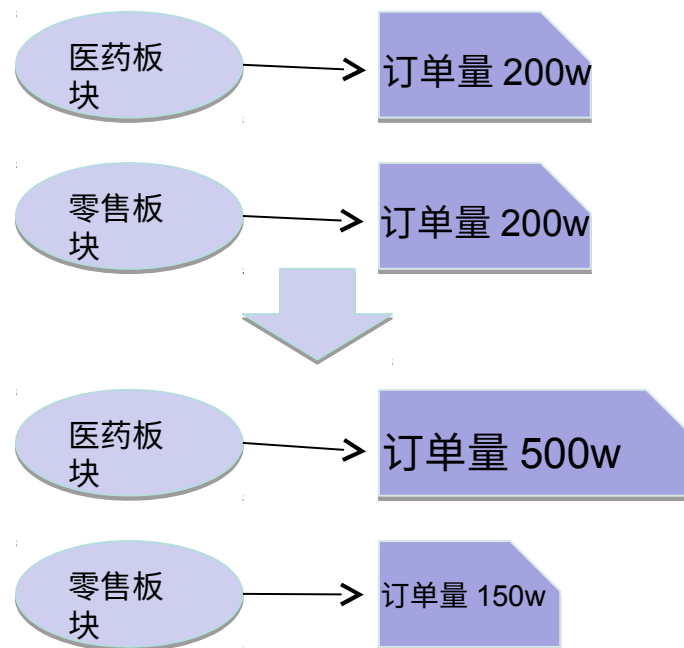
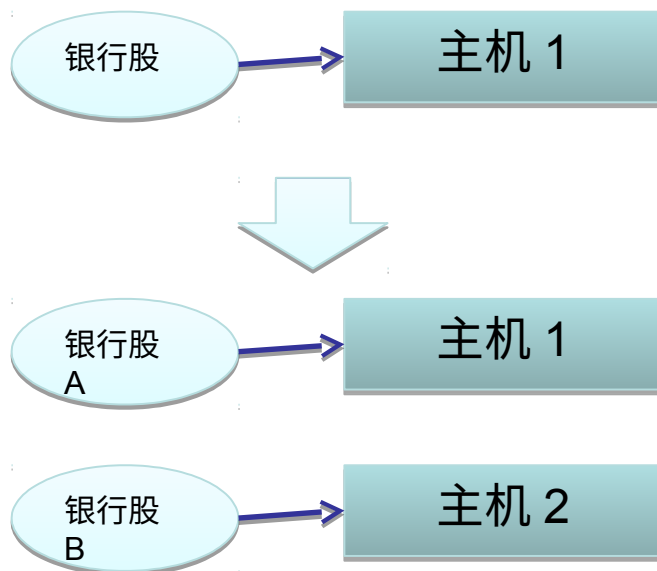


前台的扩展

- 对于市场参与者的接入，既提供客户端的模式，也支持消息协议和API的模式扩展
- 交易所提供的接入端可以无差异的多地不少和彼此备份、分流业务数据等

2. 应用配置的扩展

- 基于产品的不同类别配置，可根据负载均衡的原则，重新进行划分或扩展
- 单一类别的处理容量可以通过配置参数进行调整。调整后的容量在系统重启后自动生效



3. 设计模式的分层设计



交易系统的未来

交易系统的未来

■ 轻量化的技术系统

- 小型机 - 》 PC 服务器

■ 开放性的转变

- 封闭操作系统 - 》 开源操作系统
- 商业应用软件 - 》 开源应用软件

■ 形成企业级的基础库

- EzWEB
- EzSOFT
- HAP

■ 新技术新算法的研究

- FPGA
- Infiniband
- 全内存备份

互联网格局下的交易系统

- 更多的外部攻击
- 更长的交易时间
- 更复杂的业务处理
- 更紧迫的竞争压力

THANKS!