

策略平台架构图

1. 系统组成部分
2. 该交易系统主要由web服务器、中控服务器、策略服务器、下单服务器、风控服务器、系统监控服务器、策略监控服务器、行情计算服务器、回测服务器、DB/Cache组成；
3. 各部分主要功能
4. web服务器

用户通过网页与web服务器交互，通过web服务器，用户可实现行情的订阅及实时查看、策略的相关操作、下单/撤单等操作；

1. 中控服务器

该服务器作为整个系统的核心，采用高速事件驱动模式，内部使用高性能的Disruptor队列来实现系统内部间的线程通信。用户在网页上的一切行为可通过该服务器进行转换操作。

1. 策略服务器

该服务器承载用户的策略计算服务，采用多线程的方式实现多策略的并行高速处理。

1. 风控服务器

主要对用户的交易行为进行监控。

1. 系统监控服务器

监控主系统的运行情况，并及时报警。

1. 策略监控服务器

监控策略的运行状况

1. 行情计算服务器

对实时行情进行保存，以及指标的计算。

1. 回测服务器

对策略进行回测

1. MongoDB

保存行情历史数据

1. MySql

保存用户基本信息以及其他数据（数据量较小）

1. Redis

保存热点数据

1. 下单服务器

策略触发后，生成下单指令

1. 系统组成详解
2. Web服务器
3. 用户登录
4. 用户策略信息
5. 用户交易账户登录
6. 用户交易信息
7. 行情订阅信息
8. 下单/撤单操作
9. 系统日志

* web服务器可使用nginx做负载均衡，部署tomcat集群，Netty作为通信框架，系统内部功能避免使用单点，以便于扩展移植；
* 可将tomcat与netty部署在同一机器上，tomcat实现页面与用户交互，netty实现行情推送；
* Tomcat+netty多机部署实现集群功能。
* Mysql数据库存储数据如下
* 用户基本信息（用户名、密码、手机号等）
* 操作日志
* 其他信息
* Redis存储：
* 策略信息（策略名称、策略标的、开始时间、结束时间）
* 行情订阅信息
* 其他信息

1. 中控服务器
2. 采用高速事件驱动引擎，对外部事件进行适配，并通过Disruptor进行系统间通信（单线程）

* 交易
* 策略
* 回测

1. 数据引擎

* 交易数据处理（下单、撤单、成交回报等）

（3）系统日志

1. 策略服务器
2. 多线程方式运行，每个策略开启单独的线程；
3. 策略与用户的映射信息保存至Redis，以便策略触发时能发送正确的指令；
4. 策略触发记录保存至Redis（可配置定时任务，将该数据移植到Mysql数据库）；
5. 策略驱动方式
   1. 行情tick驱动
   2. 指标驱动

策略订阅行情 Disruptor队列接收行情（事件模式） 计算 触发 发送指令至中控服务器

1. 集群部署
2. Disruptor策略对比

#BlockingWaitStrategy 低效 低CPU（延迟在250微秒，但是不稳定）

#SleepingWaitStrategy 对生产者影响较小

#YieldingWaitStrategy 高性能 ,适当CPU占用，建议开启BIOS超线程(稳定延迟在75微秒)

#BusySpinWaitStrategy 高性能,高CPU占用,需要CPU实际物理核心数量高于FastEventDynamicHandler数量。建议最低使用14核物理机,关闭BIOS超线程!

注：

1. 一个策略开启一个线程订阅Disruptor的行情
2. 使用YieldingWaitStrategy时，handler要小于核心数，否则CPU会飙至100%
3. 策略监控服务器
4. 使用插件方式挂载至策略服务器；
5. 对策略触发情况进行监控（单位时间内触发次数、下单数量/价格等）
6. 行情计算服务器

（1）使用Netty接收行情，Disruptor队列转发行情（多事件订阅模式）

（2）接收实时行情

* 推送给策略服务器/WebSocket
* 使用MongoDB保存为历史行情供回测使用
* 计算K线数据
* 计算指标
* MongoDB存储数据时，期货和股票分库，每个合约保存为一个集合

1. 第三方数据

进行分析计算（此部分具体功能未知）

1. 回测服务器
2. 单独部署，供策略回测使用；
3. 从历史行情数据库采集数据，提供tick、K线、指标的回测。
4. 下单服务器
5. 策略触发后，该服务器生成下单指令
6. 下单至交易服务器，并接收成交回报
7. 将成交信息保存至Redis，并告知中控服务器取数据显示至页面
8. 系统未知问题、瓶颈及解决办法
9. 服务器间通信
10. 策略服务器与中控服务器间通信
11. 中控服务器与交易服务器间通信
12. 数据中心服务器与策略服务器间通信

使用Netty进行通信，测试在同一机器上的两个服务间通信大约在200-600微秒之间，中位数300微秒，偏离值1.3ms

1. 系统内部通信
2. 数据中心服务器中行情接收及转发
3. 中控服务器中指令接收及转发
4. 策略服务器中策略触发后指令的发送

系统内部间通信使用Disruptor队列，可将延迟控制在40微秒左右，正常情况下，偏离值在1.5ms以下。

1. 系统内部数据处理
2. 行情接收后的解析

多消费者模式

1. 策略计算

接收到行情并进行计算，所耗时在1ms以下，最大偏离值2.5ms，因此纯计算时间在百微秒。

1. 数据的适配

* 中控服务器中多账户的指令及成交回报适配

多账户登录根据本地系统号进行区分

* 策略服务器策略与账户，以及系统用户间的适配

使用Redis建立映射关系

1. MongoDB占用内存过高
2. 单独配置在一台服务器上，避免同服务器有高内存占用的服务；
3. 配置高容量内存
4. 限制内存使用
5. 用文本存储代替