# X-trader(Extreme Trader)

## 架构

图 1 架构图

Tunn\_producer

Md\_producer

Pending\_sig\_producer

uni\_consumer

Write ord\_rpt data

Writer market data

Writer pending signal

Read all types of data and process them

如上图，系统采用disruptor。共有3个生产者和一个消费者。每个消费者都拥有一个私有的环形队列（可以用数组模拟），这些私有环形数据是实际存放数据的地方。当消费者要生产一笔数据时，它首先将数据放到自己的私有环形队列中，然后将其对应的索引和数据类型写入disruptor queue中。

而消费者收到数据后，读取数据类型和索引，然后到指定的生产者的私有队列中读取实际的数据，并根据数据执行具体的业务逻辑。

3个生产者和一个消费者如下：

* Md\_producer

全称：market data producer。

该生产者接收行情后，将行情数据写入自己私有的环形队列，然后将在私有队列中的索引和数据类型写到disruptor queue中

* Tun\_rpt\_producer

全称：tunnel report data producer。

该生产者接收柜台系统发送来的报告后，将报告数据写入自己私有的环形队列，然后将在私有队列中的索引和数据类型写到disruptor queue中

* Pending\_sig\_producer

全称：pending signal producer。

如果有下单信号因对应策略的仓位存在冻结仓位而暂时不能发送时，该生产者将该下单信号数据写入自己私有的环形队列，然后将在私有队列中的索引和数据类型写到disruptor queue中

* Uni\_consumer

全称：unity consumer。

x-trader中只有一个消费者，该消费者从dispruptor queue中读取数据的索引和类型，然后从对应的生产者的私有队列中读取实际的数据，根据不同数据类型执行如下功能：

* Market data

将行情数据推送对应的策略，如果策略有信号产生，则处理这些信号

* Pending signal

如果某个策略存在冻结仓位，那么新的下单信号是不能发送到通道的，所以该信号会发送到dispruttor queue中等待处理。

当unity consumer从读到pending signal数据后，会看对应策略是否存在冻结仓位：如果存在，则再次把数据放回disruptor queue；否则，将信号发送到通道

* Tunnel report

收到通道报告后，更新对用策略的仓位信息，然后把报告推送给对用的策略。如果策略有信号产生，则处理这些信号

## 版本

## 关键功能

## 推送行情

## 关键类

Cont\_stra\_map\_table

Stra\_table

Uni\_consumer

Md\_buffer

Md\_producer

Tunn\_producer

Strategy

sig\_table

Sigid\_sigandrptidx\_map\_table

sigrpt\_table

localorderid\_sigandrptidx\_map\_table

图 2 行情处理关键类

## 行情处理流程

1. Uni\_consumer从disruptor queue接收到数据后，读取其数据类型和索引值。如果其类型是market data，则通过md\_producer获取指定索引存储的实际行情数据
2. Uni\_consumer根据行情的合约到Cont\_stra\_map中找到订阅该合约的策略的存储索引，然后从Stra\_table中获取策略对象
3. Uni\_consumer推送行情给策略，如果有信号产生，则将信号赋值到sig\_table，并在Sigid\_sigandrptidx\_map\_table增加映射记录，同时在sigrpt\_table创建对用的信号报告
4. 如果还有要处理的策略，则执行第2)步；否则，退出

*提示：*

*对于锁仓版，因一个帐号一个品种只有一个策略，故可以在有冻结仓位情况下，撤单和下单都立即发送*

*对于非锁仓版本：在将信号发送到通道前，必须计算仓位，保证仓位不超过最大限制*

## 处理交易信号

## 关键类

Cont\_straidx\_map\_table

Stra\_table

Uni\_consumer

Pending\_sig\_producer

Sig\_buffer\_

## 处理流程

对于撤单信号：

1. Strategy如果产生撤单信号，但是策略没有冻结仓位，则抛弃该撤单信号否则需要将信号发给Uni\_consumer处理
2. Uni\_consumer将撤单信号转换成Tunn\_producer需要的格式，发给Tunn\_producer执行撤单操作

对于下单信号：

1. 如果有冻结仓位，则不能执行下单信号，将该信号作为pending signal（pendingsig\_table存放paire<strategyid, signalid>）的索引放进disruptor中，结束流程；否则，进入下一步。

处理pending signal时，从disruptor queue中取pendingsig\_table的索引，然后从pendingsig\_table取pair，通过strateyid去Strategy对象，然后从Stategy对象，传signalid可以取signal对象

1. Uni\_consumer调用Strategy.HasFreezedPosition，如果返回true，则将信号放回disruptor queue，然后结束流程；否则，执行下一步
2. Strategy.PrepareToExecute，PrepareToExecute 更新信号委托量（Strategy需要检查委托量，保证开仓仓位不超过最大仓位限制；平仓不超过持仓量，开仓：单边 持仓量 +开仓 冻结量 <= max；平仓：平仓量不能超过可平量），并更策略新仓位信息
3. Uni\_consumer调用Tunn\_producer.NewLocalOrderID，返回LocalOrderID，调用Strategy.UpdateLocalOrderID(LocalOrderId, signalid)，向localorderid\_sigandrptidx\_map\_table增加映射记录
4. 调用Tunn\_producer.Place()

## 处理通道报告

## 关键类

图 3 处理通道报告关键类

Tunn\_producer

uni\_consumer

Rpt\_buffer

Straid\_straidx\_map\_table

Stra\_table

Sigrpt\_table

Localorderid\_sigandrptidx\_table

Strategy

* Rpt\_buffer：array容器，存储通道报告对象。Uni\_consumer从disruptor queue获取报告的索引，然后通过索引从rpt\_buffer获取报告对象。
* Stra\_table：arrary容器，存储策略对象。
* Straid\_straidx\_map\_table：unordered\_map容器，key：策略id；value：策略对象在stra\_table中的索引
* Sigrpt\_table：array容器，存储信号报告对象。
* Localorderid\_sigandrptidx\_table：unordered\_map容器，key:LocalOrderId；value：信号报告对象在sigrpt\_table中的索引;信号在sig\_table中的索引。

在根据信号下单时，在sigrpt\_table初始化信号对应的信号报告对象，并根据其在sigrpt\_table的索引和下单返回的LocalOrderId，填充localorderid\_sigandrptidx\_table

* sigid\_sigandrptidx\_table：unordered\_map容器。Key：信号id；value：信号对象在Sigrpt\_table中的索引;信号报告在sigrpt\_table中的索引。

## 处理流程

1. uni\_consumer收到数据后，读取类型和索引，如果类型是tunnel report，则通过tunn\_producer获取指定索引的报告数据
2. uni\_consumer根据报告本地委托号计算出策略id，然后通过straid\_straidx\_table获取策略的索引值后，通过stra\_table获取策略对象
3. 根据报告的状态和成交更新策略仓位信息

根据Localorderid\_sigrptidx\_table和 Sigrpt\_table获取信号报告，并更新

1. 推送仓位和信号报告给策略
2. 如果策略有信号产生：处理见行情处理流程的第2)步

## 仓位管理

## 逻辑

保证逻辑仓位 <= Max,即

|long - short| <= Max

1. 可平量

可平多仓（即可开空仓）： short + 可平多仓 - long <= max ==>可平多仓 = Max - short + long，且：

* 可平多仓 > 0
* 可平多仓 <= long
* 可平多仓 <= Vol

可平空仓（即可开多仓）: long + 可平空仓 - short <= max ==> 可平空仓 = Max - long + short，且：

* 可平空仓 > 0
* 可平空仓 <= short
* 可平空仓 <= Vol

2. 可开量

可开多仓: 可开多仓 + long - short <= Max ==> 可开多仓 = Max - long + short，且：

* 可开多仓 > 0
* 可开多仓 <= Vol

可开空仓: 可开空仓 + short - long <= Max ==> 可开空仓 = Max - short + long，且：

* 可开空仓 > 0
* 可开空仓 <= Vol

## 日间交易

日间交易分建仓和平仓2种场景，但某一瞬间，必须保证：

|long – short| <= Max Position

## 建仓

## 平仓

## 日内交易

frozen close short

frozen open short

frozen close long

frozen open long

current long

current short

available close short =

## 合约比较方案

## unordered\_map

key：contract字符串

value：订阅该合约的策略so的索引

## strcmp

接收到行情，遍历策略so列表，比较合约

## 二级索引(two-dimensional array)

一级索引：品种字符和（要验证三个市场所有品种是否有重复只）

二级索引：年月的atoi(“1708”)

在行情入口将非主力合约扔掉

## 问题列表

1. 撤单问题

撤单必须用同步撤单，否则仓位会异常。

如果撤单信号后边跟了【开/平信号】，则执行撤单信号，开平信号放回队列。

哪些信号需要放回队列？

如果该信号所属的策略仓位有对应冻结量，说明有信号没执行完，则该信号要放回队列（需要考虑开平）

哪些信号需要丢弃？

如果信号对应的仓位已满，且没有对应的冻结仓位，则该信号需要抛弃（需要考虑开，平）

要考虑一个线程内既包含 生产者，也包含消费者场景

1. 信号处理问题
2. 策略日志输出，有trader在idle时持久化
3. 需要确定接收行情一直是同一个线程
4. 需要确定委托响应和回报是否是同一个线程
5. 分析X1回报，看状态，成交，部分成交等场景
6. 如何推送行情

x-trader只有一个消费者。

在初始化时，根据合约创建索引表，可以根据合约快速找到需要该合约行情的策略列表。索引表使用map。Key:合约； value:vector

## 性能考虑

1. 落地行情数据，看其按合约的时间分布特点
2. 字符串比较如何做高效

合约比较：采用将合约各个字符相加值作为比较值和索引值

1. Map，unordered\_map选择

选择unordered\_map

1. Array, a[],vector比较

尽可能多采用a[]:

1. 想办法用信号id作为数组索引
2. 想办法用合约字符累加值作为数组索引
3. Move使用
4. 接口加入日志输出功能

策略通过接口已结构形式将日志传递给trader，trader负责输出。

有专门的写日志线程。日志结构的传递考虑使用disruptor queue，然后在空闲时写日志

1. 返回值优化等
2. 对策略的约束

策略信号ID从0或1开始递增

## 方案

Disruptor queue：存储行情，回报，待执行信号3中类型数据