策略平台设计

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订人** | **备注** |
| V2.0.3 | 2023.11 | 黄金鹏 | 创建 |
| V2.0.4 | 2023.11.27 | 黄金鹏 | 可以通过配置文件选择要播放哪些类型的数据，以及哪个日期的数据 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 概述

策略平台主要功能是接收行情数据，利用数据执行交易策略，得到交易结果。收到的数据除了直接使用之外，还可以存到文件里，在需要的时候按照接收时的顺序从文件里读取，进行播放，模仿当天的行情，去执行策略。除了策略之外，利用收到的数据也可以生成增强分钟K线，统计每分钟的委托情况和交易情况，写入文件中。

# 2、策略平台设计

## 2.1 核心逻辑

策略平台主要有五个模块，即五个类，实时行情(MarketDataSpi)，模拟行情(VirtualMarket)，存储数据(StoreMarketData)，交易策略（TradingStrategy），增强分钟K线（EnhanceMinuteKLine）,五个模块相互独立但又能够建立联系，实现不同的功能。实时行情和模拟行情是两种不同的数据来源，无论哪一个数据来源，收到数据时都可以调用交易策略，存储数据和分钟k线的回调，执行相应的功能。

## 2.2实时行情（MarketDataSpi）

### 2.2.1主要功能点

使用了华锐AMD高速行情传输平台的API，订阅数据之后就能在自己的函数(OnMDSnapshot，OnMDTickOrder等)收到相应的数据。接收数据的环境信息，在配置文件里设置。接收到数据之后，传给下个模块的回调函数去执行对应的处理，回调函数的设置有专门的成员函数。

### 2.2.2部分核心接口

class MarketDataSpi : public amd::ama::IAMDSpi

{

public:

// 定义快照数据回调处理方法

virtual void OnMDSnapshot(amd::ama::MDSnapshot\* snapshot, uint32\_t cnt) override;

//定义逐笔委托数据回调处理方法

virtual void OnMDTickOrder(amd::ama::MDTickOrder\* ticks, uint32\_t cnt) override;

// 定义逐笔成交数据回调处理方法

virtual void OnMDTickExecution(amd::ama::MDTickExecution\* tick, uint32\_t cnt) override;

//定义委托簿数据回调处理方法

virtual void OnMDOrderBook(std::vector<amd::ama::MDOrderBook>& order\_book) override;

//设置处理数据的回调函数

void SetOnMySnapshot(const SnapshotFunc& func)

void SetOnMyTickOrder(const TickOrderFunc& func)

void SetOnMyTickExecution(const TickExecutionFunc& func)

void SetOnMyOrderBook(const OrderBookFunc& func)

}

## 2.3存储行情数据(StoreMarketData)

### 2.3.1主要功能点

把实时收到的行情数据，存入文件，并记录收到数据的回调次序。示例：在MarketDataSpi的SetOnMyTickOrder()函数里传入StoreMarketData的OnMyTickOrder()函数地址，就能把收到的逐笔委托数据存储起来。写数据的文件路径通过配置文件来配置，写出的文件名带有当天日期(YYYYMMDD)。

### 2.3.2部分核心接口

class StoreMarketData

{

public:

//收到数据时调用的回调函数

void OnMySnapshot(amd::ama::MDSnapshot\* snapshot, uint32\_t cnt);

void OnMyTickOrder(amd::ama::MDTickOrder\* ticks, uint32\_t cnt);

void OnMyTickExecution(amd::ama::MDTickExecution\* tick, uint32\_t cnt);

void OnMyOrderBook(std::vector<amd::ama::MDOrderBook>& order\_book);

}

## 2.4 模拟行情（VirtualMarket）

### 2.4.1主要功能点

把存储的行情数据文件的内容播放出来，按照写文件时记录的顺序严格播放，模拟当天的行情。要播放的文件的路径在配置文件里设置。跟MarketDataSpi类似，有设置回调函数的接口，可以把播放的数据通过回调函数传递出去，给其他模块使用。

### 2.4.2 部分核心接口

class VirtualMarket

{

public:

//设置回调函数

void SetSnapshotCallback(const SnapshotFunc& snapshotCallback);

void SetTickOrderCallback(const TickOrderFunc& tickOrderCallback);

void SetTickExecutionCallback(const TickExecutionFunc& tickExecutionCallback);

void SetOrderBookCallback(const OrderBookFunc& orderBookCallback);

//播放

void Play();

}

### 2.4.3配置文件

VirtualMarketFilePath = ../data/VirtualMarket 播放的文件的路径

playFileDate = 20231124 根据日期找到对应文件

playSnapshot = 1 是否播放snapshot文件数据，1播放，0不播放

playTickOrder = 0 同上

playTickExecution = 1 同上

playOrderBook = 0 同上

OrderBookThreadNum = 6 orderBook文件的个数，也是播放时的线程数

## 2.5交易策略（TradingStrategy）

### 2.5.1主要功能点

按照twap策略去完成一定的交易任务，把总共需要交易的数量平均分散到每个分钟交易一次。目前只是通过OnOrderBook函数根据收到的委托簿数据来执行交易。需要把TradingStrategy的OnOrderBook函数设置到实时行情或模拟行情里作为回调函数。

### 2.5.2配置

TradingTaskFile = \*\*\*，在配置文件中是设置交易任务文件的路径

TradeResultFilePath = \*\*\*，在配置文件里是交易结果写入文件的路径

## 2.6 增强分钟K线(EnhanceMinuteKLine)

### 2.6.1主要功能点

根据收到的snapshot数据和tickExecution数据，统计每个分钟的委托数据和成交数据，生成一个以当前分钟命名的csv文件。需要把自己的回调函数设置给实时行情或模拟行情，用于接收数据。

### 2.6.2部分核心接口

**class EnhanceMinuteKLine**

**{**

**public:**

**//回调函数，收到snapshot行情时调用**

**void OnSnapshot(amd::ama::MDSnapshot\* snapshot, uint32\_t cnt);**

**//回调函数，收到tickExecution行情时使用**

**void OnTickExecution(amd::ama::MDTickExecution\* tick, uint32\_t cnt);**

**};**

# 3、配置文件

