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 |
| V2.0.5 | 2023.11.28 | 黄金鹏 | 播放指定数据时，过程中出现卡顿问题，已修复 |
| V2.1.0 | 2023.11.30 | 黄金鹏 | 交易策略，根据任务文件名控制开始时间，根据任务内容控制交易完成时间；可以动态地加入新文件，然后提取交易任务 |
| V2.1.1 | 2023.12.01 | 黄金鹏 | 配置文件新增workPattern项，可以设置要执行的功能。  优化了播放时的多线程调度方式，播放速率提高 |
| V2.1.2 | 2023.12.13 | 黄金鹏 | 存储数据时直接把内存的数据作为二进制数据存储到文件中，而且是在程序结束时一次性存入。播放时也是把文件里的二进制数据一次性读入内存，然后按顺序播放。 |
| V2.1.3 | 2023.12.14 | 黄金鹏 | 存数据时的文件名，改为数字1234，避免暴露文件功能 |
| V2.1.4 | 2023.12.15 | 黄金鹏 | 落数据存文件时，使用lz4算法压缩。播放数据时，将数据读入内存，解压缩后再播放 |
| V2.2.0 | 2023.12.22 | 黄金鹏 | 接收快照数据和委托成交数据后转化为自己的数据类型，然后传给下个模块去存储或分析，播放二进制文件和csv文件时也按照新的数据类型读取文件 |
| V2.2.1 | 2023.12.26 | 黄金鹏 | 读取snapshot的csv文件播放数据时，内存数组容量扩大 |
| V2.3.0 | 2023.12.27 | 黄金鹏 | 读取的orderBook数据转化为自定义数据类型quickSnapInfo，然后传给其他模块。播放数据时也按照自定义数据类型读取；  修复了写csv文件最后一列多一个逗号的bug；  修改了配置文件workPattern代表的含义 |
| V2.3.1 | 2023.12.27 | 黄金鹏 | 实时接收委托簿数据转化为自定义数据，数据放在函数栈局部空间，避免不同线程的数据冲突 |
| V2.3.2 | 2024.01.02 | 黄金鹏 | 执行策略的模块，接收QuickSnapInfo数据类型，分析数据，执行策略；自动化的脚本放入doc目录，安装cx\_Oracle的详细步骤，放入doc目录 |
| V2.3.3 | 2024.01.08 | 黄金鹏 | 执行策略的模块，通过证券代码判断交易市场类型，更新对应的时间戳；扩大落数据时开辟的空间，减少中途开辟空间的次数；quickSnap数据里某些字段不够准确，已修复; |
| V2.4.0 | 2024.01.12 | 黄金鹏 | 新增工作模式105，实时收数据时，生成k线数据文件 |
| V2.5.0 | 2024.01.15 |  | 新增工作模式106，实时收数据，生成K线数据文件，执行交易策略 |
| V2.5.1 | 2024.01.15 |  | 交易任务文件应该有表头，将程序取任务时的代码，修改适配 |
| V2.6.0 | 2024.01.16 |  | 接收实时数据，登录成功时记录日志；存储二进制数据时，每隔10秒记录一下回调触发的次数，记到日志里 |
| V2.7.0 | 2024.01.23 |  | 使用华锐新版本库文件 |
| V3.0.0 | 2024.01.29 |  | 1. 记录证券代码用九位数，首位代表市场类型，1是上交所，2是深交所 2. 新增MDApplication类，凡是创建它的子类并注册反射，都可以通过修改配置文件调用新增应用的功能,所有的应用放在单独目录里 3. 把已有的应用（存文件，执行交易策略，生成k线）也统一改为MDApplication的子类 |
| V3.0.1 | 2024.02..02 |  | 存csv的类，盘前处理函数没有正确返回，修复此问题 |
| V3.0.2 | 2024.02.08 |  | 模拟播放数据存储csv文件时，文件日期与播放的数据日期不一致，修复此问题。 |

# 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

策略平台主要有两大部分，一部分是数据平台，提供数据，数据可以是实时的((MarketDataSpi))，可以是从旧文件播放出来的(VirtualMarket, VirtualMarketBin)；另一部分是应用，是接收数据然后进行处理，主要有四个应用，存csv文件(StoreMarketData),压缩存储二进制文件(StoreMarketDataBin)，执行交易任务(TradingStrategy)，生成分钟k线(EnhanceMinuteKLine)。

## 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，转化为自定义的数据类型，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 SetOnOrderAndExecu(const OrderAndExecuFunc& func)

void SetOnQuickSnap(const QuickSnapFunc& func)

private:

//把收到数据转化为自定义的类型

void ConvertToSnapshotInfo(const amd::ama::MDSnapshot\* snapshot, SnapshotInfo\* snapInfo);

//把tickOrder转化为orderAndExecutionInfo

void AnalyseTickOrder(const amd::ama::MDTickOrder\* ord, OrdAndExeInfo\* ordAndExe);

//把tickExecution分析转化位OrdAndExeInfo

void AnalyseTicExecution(const amd::ama::MDTickExecution\* execu, OrdAndExeInfo\* ordAndExe);

//把OrderBook转化为QuickSnapInfo

void AnalyseOrderBook(const amd::ama::MDOrderBook& orderBook, QuickSnapInfo\* quickSap);

}

### 2.2.3配置信息

workPattern = 1

设为1时接收实时数据

usrname = huangjinpeng

password = huangjinpeng

ip = 192.168.144.92

port = 8200

接收数据的环境信息

## 2.3 模拟行情（VirtualMarketBin）

### 2.3.1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。播放的文件存的是二进制数据，读到内存，解压缩，然后放入数组里。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。

### 2.3.2 部分核心接口

class VirtualMarketBin

{

public:

//设置回调函数

void SetSnapshotCallback(const SnapshotFunc& snapshotCallback);

void SetOrdAndExecuCallback(const OrderAndExecuFunc& func);

void SetQuickSnapCallback(const QuickSnapFunc& quickSnapCallback);

//播放

void Play();

}

### 2.3.3配置文件

示例：

workPattern = 2 设为2时播放二进制文件数据

VirtualMarketBinFilePath = ../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

compressLevel = 1 为0时直接读取，大于0时需要解压缩

## 2.4播放csv文件(VirtualMarket)

### 2.4.1主要功能

跟VirtualMarketBin类似，读取数据文件，播放模拟实时数据，只不过它播放的数据来源是csv文件

### 2.4.2部分核心接口

//设置回调函数

void SetSnapshotCallback(const SnapshotFunc& snapshotCallback);

void SetOrdAndExecuCallback(const OrderAndExecuFunc& ordAndExeCallback);

void SetQuickSnapCallback(const QuickSnapFunc& func);

### 2.4.3配置信息

workPattern = 3 设为3时播放csv文件数据

VirtualMarketFilePath = 播放csv文件的数据来源目录

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

playQuickSnap = 0 同上

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

## 2.5应用类基类(MDApplication)

### 2.5.1主要功能

是所有应用类的基类，包括盘前处理，盘后处理，接收各类数据的回调函数。新增的应用只需要继承它，按照需要重写一些函数，然后注册反射，就可以在配置文件里设置使用

### 2.5.2核心接口

//盘前处理

virtual int BeforeMarket()

{

return 0;

}

//如果子类不重写此函数，函数内无任何操作

virtual void OnSnapshot(SnapshotInfo\* snapshot, uint32\_t cnt)

{

}

virtual void OnOrderAndExecu(OrdAndExeInfo\* info, uint32\_t cnt)

{

}

virtual void OnQuickSnap(QuickSnapInfo\* info, uint32\_t cnt)

{

}

//盘后处理

virtual int AfterMarket()

{

return 0;

}

## 2.6存储行情数据(StoreMarketDataBin)

### 2.6.1主要功能点

MDApplication类的子类，把收到的行情数据，存入文件，并记录收到数据的回调次序。在程序结束时，把内存里的数组以二进制形式放入文件，放入文件前经过lz4算法进行压缩，使得文件在磁盘里占据空间更小。

### 2.6.2部分核心接口

//收到数据时调用的回调函数

void OnSnapshot(SnapshotInfo\* snapInfo, uint32\_t cnt);

void OnOrdAndExecu(OrdAndExeInfo\* info, uint32\_t cnt);

void OnQuickSnap(QuickSnapInfo\* info, uint32\_t cnt); }

### 2.6.3配置文件相关内容

示例：

application = …,1,... application队列里有1，说明要使用存储二进制文件的功能

StoreMarketDataBinFolder = /mnt/disk4/StrategyPlatformData/BinData/today

接收的数据存放的目录

配置文件里compressLevel一项，设为0时不压缩，大于0时，数字越大，压缩率越低，但是压缩的更快。一般设置为1

## 2.7交易策略（TradingStrategy）

### 2.7.1主要功能点

MDApplication类的子类，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OnQuickSnap函数根据收到的委托簿数据来执行交易。

根据任务文件名控制开始时间，根据任务内容控制交易完成时间；在实时行情下，可以动态地加入新文件，然后提取交易任务，当前时间开始执行交易，规定的时间完成任务；在模拟行情下，根据文件名时间开始执行交易，在规定时间完成任务。

### 2.7.2部分核心接口

//QuickSnap数据的回调函数

void OnQuickSnap(QuickSnapInfo\* infos, uint32\_t cnt);

### 2.7.3配置

application = 3,... application队列里有3，说明要使用交易策略的功能

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

## 2.8 增强分钟K线(EnhanceMinuteKLine)

### 2.8.1主要功能点

MDApplication类的子类，根据收到的快照数据和成交数据，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。

### 2.8.2部分核心接口

**class EnhanceMinuteKLine**

**{**

**public:**

//回调函数，收到snapshot行情时调用

void OnSnapshot(SnapshotInfo\* snapshot, uint32\_t cnt);

//回调函数，收到tickExecution行情时使用

void OnOrdAndExecu (OrdAndExeInfo\* execu, uint32\_t cnt);

### 2.8.3配置文件

application = …,4,... application队列里有4，说明要使用生成k线的应用

KLineFilePath = … 生成k线文件的位置

## 2.9存储csv文件(StoreMarketData)

### 2.9.1主要功能

跟StoreMarketDataBin类似，有自己的回调函数接收数据，存储数据，只不过它存储的是csv可读文件

### 2.9.2部分核心接口

//收到数据时调用的回调函数

void OnSnapshot(SnapshotInfo\* snapInfo, uint32\_t cnt);

void OnOrdAndExecu(OrdAndExeInfo\* ordAndExe, uint32\_t cnt);

void OnQuickSnap(QuickSnapInfo\* infos, uint32\_t cnt);

### 2.9.3配置

application = …,2,... application队列里有2，说明要使用存储csv的应用

StoreMarketDataFolder = 存储csv文件的目录

## 2.10、配置文件

### 2.10.1

workPattern = 1

后面跟一个数字，等于1是实时行情数据，2是播放二进制文件，3是播放csv文件

### 2.10.2

application = 1,3,NumCount

后面跟要使用的应用，用逗号隔开，1234有特殊意义，新增的应用应该把类名写入队列。