策略平台需求跟踪

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订人** | **备注** |
| V1.0.0 | 2023.10 | 黄金鹏 | 创建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 项目需求收集情况

|  |  |
| --- | --- |
| **提交时间** | **进展情况** |
| 2023.10.16 | 1.连接Linux服务器，VPN账户和服务器地址账户已记录  2.仿照《策略平台需求跟踪》文档，记录每次开会更新的需求内容  3.MarketData这是已写好的demo，修改IP端口账户等信息，看能否在测试和开发环境运行，之后重构成cmake项目，使它能够跨平台  4.在之后的开发中遵从代码规范  5.demo如果能够运行，再进行改进，它输出的数据只留下我们需要的行情信息，并且不再打印而是写入csv文件里 |
| 2023.10.17 | 1.华锐的redhat/c++版本代码经验证可以运行;安装cmake，make（已完成）  2.将MarketData项目的源文件，需要的头文件和库集合到一起，编写cmakelists文件，在Linux服务器上编译运行（已执行）  3.编译过程中出现未定义的引用错误，但是所有库文件都已链接，需要排查原因 |
| 2023.10.18 | 已完成：  解决编译出错问题，demo在Linux上成功运行  了解代码逻辑，如何设置交易市场，接收数据类型，证券种类，证券号码,  以及收到数据时调用的回调函数，处理的数据的位置。  待办：  重写OnMDSnapShot()回调函数和订阅行情数据的函数，订阅和接收需要的行情，把接收到的行情数据写入csv文件 |
| 2023.10.20 | 已完成：  重写OnMDSnapshot()函数，订阅行情写入csv文件；  编写shell脚本压缩csv文件并分卷，编写脚本合卷解压缩  在MySpi类中添加string成员存储要写的文件名，添加init()函数，把当前日期加入文件名（如SnapShot20231020.csv），给函数使用  在MySpi类添加int32\_t成员变量，init()函数将其初始化为0，执行回调函数时自增并写入文件  重写OnMDSnapshot()函数，所有字段都写入  待办：  除了OnMDSnapshot()函数外，还需要重写OnMDTickExecution()函数和OnMDTickOrder()函数，文件里除了本身字段外新增三列，写入回调函数第几次调用，本行数据是此次接收到的数组里第几个，本地系统当前时间戳  程序出现各种异常情况时，写入Log文件 |
| 2023.10.23 | 已完成：  MarketData1.0版本，接收Snapshot数据和OrderBook数据，写入Snapshot.csv文件和OrderBook文件  撰写设计文档，描述项目功能  待办：  测试实际接收和写文件效果 |
| 2023.10.24 | 已完成：订阅和接收委托簿数据  待办：调整委托簿的价格档位个数，以及每个价格档位的委托队列的长度  测试接收和写文件的效果，按照版本1.0.0提交到github，设计文档和需求文档一并上传 |
| 2023.10.25 | 已完成：  StrategyPlatform V1.0.0版本完成并上传  待办：  2.0版本新增接收次序和时间戳表列，  根据已有的行情文件，播放行情数据 |
| 2023.10.26 | 已完成：  2.0版本，新增两个表TickExecution.csv和TickOrder.csv，记录逐笔成交和逐笔委托数据，表结构新增三列，回调次数，数据次序，本地时间  待办：  测试接收文件写文件效果  实现需求，根据文件里数据的时间顺序和分组情况，播放数据，重现行情 |
| 2023.10.27 | 已完成：  2.0版本 四类数据可以正常接收与接收次序一起写入表  待完成：  打包库文件，把执行逻辑放在库中，与可执行文件相独立  实现日志记录功能，记录程序异常结束时的原因和位置  数据播放 |
| 2023.11.06 | 播放功能，用足够大的std::array存储所有数据，调用回调函数时，传递array中的数据地址，不需要另外申请空间；  从文件读数据时，使用c语言的fscanf()格式化读文件；  1.在build目录下写一个例行执行的脚本，如果lib目录下静态库文件有更新，就重新链接，链接时是main.o文件和库文件（今天）  2.继续调试播放功能代码，修复bug；（本周三）  3.实现交易策略功能。（本周五）  4. 3.0版本接dma交易柜台，实现接入（下周五11.17）  调试细节（11.20~11.24） |
| 2023.11.07 | 接收数据：订阅接收股票和基金两类证券的行情数据  测试程序：订阅接收除股票基金以外所有证券的数据  代码版本更新时上传到github，在commit信息里标注版本号 |
| 2023.11.10 | 1.适当调整main.o文件的大小  2.修复播放过程中出现的最后多写一行的问题  3.修复接收的数据写文件时回调次数偶尔不连续的问题  4.交易策略，接收到QuickSnapshot数据时触发回调，在回调函数里根据收到的证券代码，时间戳和价格行情执行对应证券和时间点的交易行为，本时间点已经执行过交易的不再执行，收到的行情数据时间戳相比最新的全局时间戳更早更旧，超过一秒钟的不使用。  不同交易市场有自己单独的最新时间戳。  调整构建委托簿的线程数量，使速率最佳，如果有多个线程更新同一个市场的最新时间戳，注意线程安全。  5.记录异常时，回溯的函数栈层数调整为50层 |
| 2023.11.14 | 1.接收QuickSnap数据写文件时，不同的线程写入不同的文件，减少线程的加锁阻塞。  2.交易策略中OnOrderBook()回调函数会在不同的线程中并行调用，所以在记录交易次数自增时加锁，把交易结果信息写入文件时，各个线程单独写文件，避免加锁；  3.交易策略中如果收到的行情数据时间戳比最新的时间戳更早，允许的延迟范围是10秒，超过10秒之前的不使用。 |
| 2023.11.17 | 1.调整main.o文件大小  2.交易策略：不仅能执行买任务，也能执行卖任务  3.交易策略：交易的任务事先写在文件里，程序从配置文件里获得要交易的信息，然后去执行  4.接dma交易柜台（下周实现） |
| 2023.11.20 | 1.交易策略将交易记录写文件，加上回调的次数和本次回调的第几条数据，时间戳用交易所的。  2.交易任务文件，统一写成逗号隔开的csv文件  3.增强分钟K线，统计每个分钟内的OrderEvaluate委托数据和逐笔成交数据，写文件，每分钟一个文件。（本周完成） |
| 2023.11.21 | 1.github上的代码，提交时附上版本号和版本改动，每个版本新增一个设计文档，描述新版本详细信息。  2.同一档位的价格不同，需要考虑是新增委托的情况，如果是靠后的档位，对方相同价格的委托已经不在十档内，可以忽略不统计。  3.tickExecution数据判断主动买还是主动卖，不同的交易所用不同的方法判断。 |
| 2023.11.23 | 1.编译步骤写成文档，放入项目doc目录  2.增强分钟k线代码，修复bug;  3.接收华锐实时数据，播放文件数据，写文件，执行策略，生成分钟K线，各部分功能代码相互独立，解耦合；  4.播放数据的功能，配置文件增加一条要播放的数据的日期，根据日期合成文件名，播放对应的文件。  5.OnMDOrderBook()被不同线程调用时，数据写入不同的文件，避免每次回调都获取一遍线程id，可以根据证券代码对应固定的线程，从而写入对应的文件。  6.多线程播放数据功能，可以继续优化，一个线程触发了回调后，继续触发下一个回调的线程，更接近真正的并发，效率更高 |
| 2023.11.24 | 1.接收QuickSnap数据时，不存储委托队列数据order\_queue\_size和order\_queue[i],避免访问时出现段错误，保证文件里回调次数连贯，数据个数完整；  2.交易策略回测，落数据，播放数据执行交易策略，得到交易结果，并进行分析，实现每天按时自动执行（第一周）；  3.执行交易策略时接DMA（第二周）；  4.监控程序运行的主机的状态，包括磁盘空间，内存空间是否充足，cpu运行情况，监控程序运行是否出错，并及时修复（第三、四周）。 |
| 2023.11.27 | 1.播放功能，通过配置文件来选择播放哪些数据  2.自动化的功能，使用shell脚本，定时启动某些程序  3.播放时，检测文件里的回调次数是否有重复 |
| 2023.11.28 | 1.交易策略：交易文件里含有开始时间和结束时间，在规定时间内完成交易任务，在运行过程中有新的任务文件加入，及时监测到并提取新任务 |
| 2023.11.30 | 1.从配置文件里能够配置跑哪个模式  2.分钟K线，tickExecution数据主买主卖交易方向未知的直接忽略，不用打印  3.交易策略：实时场景下，判断交易开始时间，以当前的交易所时间戳为准  4.播放数据时能不能更快？写文件时压缩数据，播放时读入内存再解压缩，或者其他方案 |
| 2023.12.07 | 1.尝试更高效的压缩算法，使文件占空间小，读取和解压缩的综合效率最高  2.自己定制数据结构，使占据空间更小，之后如果进行计算时，需要注意不要溢出  3.从外部接收的数据先转化成自己定制的数据结构，回调函数接收自己定制的结构体类型，需要存文件时就直接存接收到的数据。播放时，把文件的数据读出来，可以直接传给接收数据的回调函数，不需要经过任何转化拷贝。  4.如果需要进一步提升效率，可以考虑解压缩时分多个线程共同解压缩 |
| 2023.12.18 | 1.定制自己的数据结构，在接收行情后处理成自己的结构类型，用于存储，播放，执行交易等。（周一周二）  2.测试修改后的版本，接收数据。  3.开发环境的自动化脚本（周三）  4.生产环境的自动化脚本。实现自动执行定时任务的脚本，每天的定时任务包括：开盘前启动，盘后压缩，传输（最终传到56服务器上），实现全自动（周四）  3.接入柜台（周五） |
| 2023.12.20 | * 自动化流程   （生产环境）   1. 每天定时启动 2. 闭市后结束程序   等待一会，压缩存储二进制数据   1. shell脚本压缩数据   （本地）   1. 下载到本地 2. 上传到56服务器   （服务器）   1. 脚本解压缩 2. 生成K线 3. twap策略  * MLFlag   A,B,C,MLU分别代表上海成交数据，上海委托数据，深圳成交，深圳委托   * 日期dt暂时没用，删掉 * 生成K线的类，修改适配新的数据类型，先判断是成交数据再执行后面 |
| 2024.01.05 | 1. 交易策略模块，根据代码数字来判断数据来源的交易所，记录各自的时间戳  2.写一个监控的小程序，每隔一段时间记录当前内存可用空间，磁盘可用空间，最好再记录下网络传输状态  3.添加一个105工作模式，在生产上运行，能够接收数据的同时生成K线数据。闭市后将K线数据传回56服务器，这需要修改自动化流程的脚本。  4添加一个106工作模式，接收实时数据，同时生成K线，执行策略。在自动化流程里，每天把策略任务从56服务器传到生产，启动前检查有没有传输好任务文件 |
| 2024.01.15 | 1.程序启动时是否成功连接和登录，写日志，运行过程中是否接收到数据，写日志  2.华锐的API更新到4.3版本了，我的也应该适配4.3版本，使用新版本的库 |
| 2024.01.18 | 1.自定的数据类型，证券代码写成9位数，最高位1代表上交所，2代表深交所  2.平台能同时调用多个业务，业务模块根据类名反射出创建实例的函数，主函数根据配置文件创建业务类对象，注册回调函数给平台。  3.自动化流程中，交易日的分割线，由当天的24点改为次日15点之前 |
| 2024.01.19 | Windows传输数据的脚本，如果出现异常情况，文件遗留，会被新一天文件覆盖，或者在新一天被当作新文件上传，修复此问题，把每天的数据放在单独的日期命名的目录里 |
| 2024.02.22 | * 把原有的k线的统计数据加进来 * 对于一些可以用其他指标四则运算得到的，可以不统计 * 原始数据，每分钟存一次，除了日期之外的所有字段 * 每个指标都有一个编号 * 委买/委卖价格跨度，买一卖一价格跨度，以切片为颗粒度，统计最大最小值，开始到最后的变化值，均值，标准差 * 以切片为颗粒度，主买成交金额/委卖价格跨度=成交金额强度，累加一分钟，新增卖委托金额/委卖价格跨度=委托金额强度，累加一分钟 * 成交价格平均值=成交金额的和/成交数量的和 * 成交价格标准差，计算时考虑权重，权重就是每个成交价的成交数量 * 一分钟内，切片个数，切片间隔时间的最大最小值，均值，标准差 * 在一个全局的位置，存储开盘价，昨收价，每天只存一次 * 从配置文件里获得历史数据里的各个分钟成交金额占全天比例 |
| 2024.02.26 | * 主买主卖成交金额占全市场的比例，以一秒为颗粒度，记录最大最小值，开始值，结束值。计算均值，标准差，计算时乘以权重（成交金额），除以一分钟内总成交金额 * 主买主卖成交金额占市场金额的比例，一分钟为颗粒度，全天为时间窗口，统计均值，标准差，计算时乘以权重（成交金额） * 在配置文件里写历史数据的每个证券和成交量分布 * 查看生成k线的python代码，对照代码逻辑 |
| 2024.02.28 | * 检查服务器磁盘空间，根据每天的数据量估算还能存储多少天数据。 * 由于磁盘空间不足导致某些数据没有传完整，单独执行传输脚本补全数据 |
| 2024.03.01 | * 每个交易任务文件，单独执行不与其他任务合并，并且可能会用不同的算法执行 |
| 2024.03.04 | * 每天传完数据回测时，不再写csv数据文件。写一个单独的脚本，执行时会根据某个日期播放数据写csv文件 * 尝试寻找新算法，进一步提高压缩率，减少占用空间 |
| 2024.03.11 | * 颗粒度没写的，补全 * 价格，恢复到浮点数原值，然后参与计算 * 关于成交金额的标准差统计，先把每个金额变成ln(金额)，再计算 |
| 2024.03.13 | 监控系统资源的日志，定时查看dmesg日志，记录下一些异常情况的日志。目前只记录了内存不足杀死进程的日志。 |
| 2024.3.19 | 安装好gdb调试功能，争取之后调试代码能够使用 |
|  |  |
| 2024.3.21 | 修改平台框架，MDApplication新增回调函数OnUnchangedData()；接收实时数据的类新增MDApplication\*数组，在第一次收到昨收价或开盘价时，传给数组里所有对象的OnUnchangedData()函数；重写StatisticalData类的OnUnchangedData()，存储收到的数据，用于统计 |
| 2024.3.22 | 1. 调试平台代码，修改报错，运行，统计各项指标，发现得到的全天收益数据是0，不符合预期  2. 关于开盘价的产生时机，在集合竞价阶段只有委托，集合竞价结束(9:25)会撮合成交产生开盘价，少数情况下没有成交产生开盘价，小概率异常事件。  关于昨收价的产生时机，一开始就有。  3. 播放数据模拟行情时，类似开盘价昨收价等数据可以从外部配置文件获得，自己只需要拿来用  4. 生产上已经完整传下来的数据，删掉一些之后，磁盘空间占用77% |
| 2024.03.25 | 1. 实时模式统计的全天收益数据总是0，问题已经修复；  2. 56服务器，如果传过来的数据解压出文件数量没有缺失，删掉压缩包文件，写入脚本里，自动操作。删掉压缩包是因为它可以再生，用分片文件再合成出来  3.生产环境账号没了，以后就用root账号，文件放在/mnt/tqnas目录  重新部署接收数据的程序，运行时只要占用内存一多，就被终止 |
| 2024.03.26 | 1.生产环境32009端口重新部署程序，成功接到数据并存储，数据放到/mnt/tqnas磁盘下  修改监控系统资源的脚本，监控的磁盘空间改为/mnt/tqnas对应的空间  修改传输脚本的连接信息，运行脚本把今天的数据传回56服务器  生产上重新设置每天定时启动  2.历史数据已经找回  3. 新增一个类，读取一个配置文件，把每行数据所有字段，读入内存，按照double的形式读入，可以通过第几行和字段名获得数据，可以获得总数据行数  4.把新功能统计各项指标的版本，部署到生产上，明天开始运行 |
| 2024.03.27 | 1. 自动脚本需要用到expect工具，定时向程序输入0结束，安装expect，安装它的依赖tcl  2. 生产，修改自动脚本，检测每天生成的统计数据，放入新建的日期文件  3.尝试从Linux服务器向windows本地传输文件，经研究需要在windows本地建立sftp服务器，在Linux上调用sftp功能传输。安装fileZilla软件在本地建立了服务器，设置端口号，用户名和主目录，在linux上用telnet可以连接到服务器，但是用sftp连接时没能成功  4.写一个应用例子，用到分析配置文件的类，利用文件内容和收到的成交信息统计一些值  5.手动执行脚本，处理今天的数据 |
| 2024.03.28 | 1.在windows本地建立ftp服务端，在Linux远程主机调用ftp命令可以成功传送文件到本地。sftp/rsync命令涉及到ssh协议，暂时没有成功。  2.更新git版本库 |

# 待办事项

|  |  |
| --- | --- |
| 优先级 | 待办事项 |
| P1 | * 把新应用生成的文件，以100ms的频率传到本地，在程序代码里调用ftp工具，传送文件。 * 调研，hdf5格式存文件，用c++实现，读取播放的效率如何，压缩空间如何？   以及parquet格式存文件的效果 |
| P2 | * 记录dmesg中异常情况的日志，如果记录过的日志，不要重复记录 * 记录系统启动了多久，一小时记录一次 * 服务器上gdb工具无法使用，重新安装一下 * 每周五提醒宋老师帮我固化环境，以免数据被清理 |
| P3 | * 对比python代码和自己的k线代码，看看计算过程中有什么区别 * 编号相同的多个小单，它们是由一个大单拆分得来，可以再次还原成一个大单 * 配置文件，记录历史数据每分钟成交额占全天的比例 * snapshot和ordexe数据合并对齐，每个切片间有哪些委托成交 * 改进压缩算法，增加压缩率，使存储文件占据空间更小 * 执行交易任务的模块，读取多个交易任务文件然后分开执行，不要合到一起 |
| 未知 |  |