ChatGPT

QMT作为量化数据源的可行性研究

QMT数据内容与频率支持

数据种类与覆盖范围：迅投QMT（迅投终端）提供非常丰富的市场数据，涵盖股票、指数、期货、期权、可转债、场内基金（ETF/LOF）等各类证券品种 [1] 。具体来说：
- **行情数据：**支持沪深A股（含北交所）、股指和商品期货、期权，以及沪港通/深港通标的股票等实时行情 [1] 。对于港股，美股等海外市场，QMT主要支持沪深港通下的港股**（港股通）**交易数据 [1] ；**美股**行情并非QMT核心支持范围，当前QMT数据源尚未覆盖美股（除非通过券商外部接口），因此在海外市场数据上QMT存在局限。

K线频率与历史范围：QMT提供从日线级别到分钟级别乃至Tick级别的历史行情数据 [2] 。基础周期包括**Tick、1分钟、5分钟、日线**等，其他诸如3分钟、15分钟等周期均由基础周期合成 [3] 。数据的历史回溯范围取决于用户权限：

- **券商基础版**（仅使用券商行情，不购买迅投投研数据）：可下载**5分钟K线**约1年历史，**不支持1分钟线**的长历史，**逐笔Tick**仅能获取最近1个月 [4] 。例如，有用户验证**券商版QMT历史Tick数据只能下载最近1个月** [4] 。
- **迅投基础权限**（注册迅投投研账号但未购买VIP）：可获取**1分钟线**约1年历史，Tick约1个月，较券商版略有提升 [5] 。
- **迅投VIP行情权限**（投研版VIP）：提供最完整的数据历史：**1分钟线**和**5分钟线**可追溯至少3年，**Tick数据**可达最近1年 [5] [4] 。日线级别历史通常覆盖更长（一般涵盖上市以来全历史），在数据服务中不受上述短周期限制。

上述历史数据需通过QMT提供的API下载至本地才能使用。需要注意，**ETF基金**、**可转债**等场内基金数据以及**北向资金、沪深港通资金流**等特色数据，QMT VIP也有提供 [6] 。例如，VIP权限支持**北向资金、资金流、沪港通持股数据**等，而基础权限不支持 [6] 。**行业概念板块**信息也由QMT提供，涵盖申万行业分类和概念题材等板块列表 [7] 。**龙虎榜**数据同样在QMT的数据字典中有接口（每日交易龙虎榜信息），VIP用户应可获取该数据。综上，QMT的数据覆盖范围与深度足以涵盖行情、财务、概念、龙虎榜、资金流向等常用量化研究所需的信息。对于港股通标的数据也支持，但完全覆盖港股全市场和美股数据则需要另寻数据源补充。

频率支持与实时性：QMT支持**日线、分钟线（包括1分钟、5分钟等）以及Tick级别**的实时行情推送，满足高频策略需求 [2] 。通过全推行情模式，VIP用户可获得**Tick级逐笔成交**和高达**五档盘口**深度的数据 [8] 。相比之下，基础行情仅提供**最新价或1档盘口** [8] 。这种多档行情和Tick实盘推送能力，使QMT的数据在实时性方面非常突出，适合需要分笔成交、盘口队列等细节数据的量化策略。

小结：QMT提供的行情数据覆盖A股主要证券及衍生品，并延伸至北向资金和概念板块等扩展数据。频率上支持从日线到Tick，VIP权限可获取较长的历史区间（分钟线3年、Tick 1年等 [5] [4] ），足以满足大部分策略回测和研究需求。但若涉及A股之外的海外股票，QMT目前支持有限，需要另行考虑数据来源。

QMT Python接口（xtquant）接入与部署

xtquant接口简介：QMT通过**xtquant**　Python库提供行情和交易API接口。xtquant可以以Python库形式独立使用，用于获取行情数据和执行交易。使用xtquant有两种主要模式：**通过本地MiniQMT终端**对接，或**直接使用Token连接迅投数据中心**。

- **本地MiniQMT模式：**MiniQMT是QMT提供的一个轻量级客户端，可在本地运行以处理数据请求。xtquant默认情况下会尝试连接本地的MiniQMT，由MiniQMT登录券商行情并返回数据给Python [9]。在这种模式下，需要先安装券商QMT客户端或MiniQMT，并使用券商账户登录获取行情。然后Python端调用xtquant时，会**与本地MiniQMT建立连接**，由MiniQMT代理向行情服务器请求数据 [9]。例如，xtdata的各类获取数据接口实际依赖MiniQMT缓存的数据 [10]。因此本地模式要求在同机启动QMT终端或MiniQMT，并确保其登录了行情服务器。**需要安装终端吗？**是的，在不使用Token直连模式时，至少需要运行MiniQMT程序。很多券商提供独立的MiniQMT可执行文件，可在服务器上以命令行方式登录。使用本地模式的优点是数据先下载存储在本地硬盘，之后的策略计算全部在本地进行 [11] ——这与JQData的云端调用不同，**QMT的数据和计算更本地化** [11]。

- **Token直连模式：**迅投提供了使用**API　Token**直接连接数据中心的方式。用户在迅投投研平台获取一个接口Token [12] 后，无需运行本地终端，直接在Python中调用`xtdatacenter.set_token('你的TOKEN')`来配置认证 [13]。然后使用`xtdatacenter.init()`初始化连接，并通过`xtdc.listen()`启动一个本地监听端口 [14]，再调用`xtdata.connect()`接入该端口，即可建立到迅投行情服务器的连接 [15] [16]。这种模式下，xtquant内部相当于起了一个轻量级的数据服务进程（由xtdatacenter驱动）直接连接官方行情站点 [17]。**无需安装QMT完整终端**，只需安装xtquant库即可获取数据。对于服务器部署而言，这非常方便，可以在没有图形界面的纯服务器环境上运行。需要注意在Token模式下，也需要有效的行情权限（如已购买VIP）。通过配置VIP行情站点的地址列表，xtquant会优选高速站点连接，提高数据稳定性 [17]。

在远程服务器运行：xtquant官方支持在**Windows和Linux服务器**上运行数据接口。Windows环境下可直接使用pip安装或将券商QMT安装目录中的xtquant复制到Python环境。Linux环境迅投也推出了专门的xtquant版本（需专业版权限），可以获取行情数据但不支持交易下单 [18]。根据迅投知识库说明：Linux版xtquant仅支持xtdata行情数据功能，不支持xttrade交易模块 [18]。这意味着如果韬睿系统希望部署在Linux服务器上，**可以使用xtquant获取数据**（只要账户权限符合要求），而交易功能若需使用可能需要Windows环境或通过其他接口。实际用户经验也表明，在Linux上使用xtquant需要下载官方提供的压缩包并手动安装，而非纯pip安装 [18] [19]。总的来说，**在远程服务器上运行xtquant获取行情数据是可行的**：对于数据服务，可以在Linux服务器上通过Token模式连接行情；如果需要实盘交易则往往采用云Windows服务器运行MiniQMT+xtquant的方式 [20]。

本地数据接入与缓存：无论通过哪种模式，xtquant都倾向于将历史数据下载到本地缓存以提高调用效率和脱机使用能力。例如，在xtquant的设计中，“大部分历史数据都会以压缩形式存储在本地” [21]。开发者需要在**获取数据前先调用下载接口**将所需历史数据补齐 [21]。比如调用`xtdata.download_history_data('000001.SZ', period='1d', incrementally=True)`来增量下载平安银行的日线数据到本地缓存 [22]；下载完成后，再调用`xtdata.get_market_data_ex()`等接口从本地读取数据 [23]。财务数据、板块数据也有类似的下载函数（如`xtdata.download_financial_data`、`xtdata.download_sector_data`）用于一次性获取全部所需信息到本地 [24]。这种机制使得频繁的历史数据调用不必每次都请求远端，提高了效率。不过也要求我们在初次使用或定期更新时，多调用一步“下载”操作，确保本地缓存最新。

接入配置流程概述：为使用QMT Python API，需要：1）注册迅投投研账号并购买所需行情权限（如VIP）；2）获取API Token [12]；3）在目标机器上安装xtquant库（Windows可直接`pip install xtquant`，Linux需下载专

用包)[25]； 4）编写初始化代码设置Token并建立连接[13][15]。完成上述后，便可使用xtquant的`xtdata`模块函数请求行情与各类数据。在部署时，请确保服务器网络能连通迅投提供的行情站点（如绍兴电信、郑州联通等地址)[17]，并考虑将这些地址加入白名单防火墙以稳定访问。总之，xtquant提供了灵活的接入方式，无需GUI终端即可在本地或云端获取QMT数据，满足远程量化系统的部署需求。

QMT数据与聚宽JQData对比

QMT和聚宽JQData作为国内常用的量化数据源，各有侧重，以下从**数据覆盖、调用性能、实时性、稳定性、接口灵活性**等方面对比二者的优劣：

- **数据覆盖范围：**两者在A股股票及衍生品领域都有较全面的数据。JQData覆盖沪深股票、指数、ETF基金及部分期货等，提供财务报表、行业概念分类、龙虎榜等数据接口。QMT在这些方面同样全面，甚至有过之而无不及。例如QMT提供完整的财务数据（资产负债表、利润表、现金流量表、股东持股等[26]）、行业概念板块列表、沪深港通持股、北向资金流向、逐笔成交和委托队列等**高级行情细节**，而JQData相对缺乏交易所逐笔委托/逐笔成交等细粒度数据。QMT还支持股票期权、国债逆回购、可转债等数据类别[1]（这些要素JQData覆盖有限或没有）。需要指出，**美股和大部分港股**数据，JQData本身也不提供，因此这一方面两者都不涉及。整体而言，在A股及其相关市场，QMT数据覆盖面与深度不逊于JQData，甚至**多出了如期权、可转债、席位资金流**等特殊数据[6]。

- **调用速度与性能：**JQData采用远程API调用模式，每次数据请求需要通过网络访问聚宽服务器，这在大批量、频繁调用时会受到网络延迟和聚宽API限频的影响。QMT则强调本地化：数据先下载到本地，然后读取本地文件[21]。在实际使用中，获取同样的数据，QMT初次下载可能稍耗时，但之后本地读取非常快速。此外，QMT支持实时推送模式，订阅后数据源主动推送更新，几乎无延迟；而JQData主要是轮询式获取，**不具备实时推送**。对于需要高频更新的场景，如逐笔行情，每秒成百上千笔Tick，QMT能够通过全推接口源源不断提供，而JQData并不提供Tick级实时数据服务。因此**在低延迟和高吞吐**方面，QMT更具优势。实践中，QMT允许设定连接高性能的VIP行情主站[17]，并支持并行下载多只股票数据，合理配置后批量数据获取的效率颇高[27][28]。反观JQData在批量历史数据下载时可能有每分钟请求次数限制，需要控制节奏。总的来说，**QMT通过本地缓存和推送机制在速度上占优**。

- **实时性与数据更新：**QMT定位于策略实盘交易平台，其行情与交易数据均为实时更新、毫秒级推送。只要行情连通，QMT可提供交易时段内持续的最新行情、逐笔成交数据，满足实时监控和交易需求。而JQData更多用于**研究与回测**，虽然也能获取当日最新报价，但通常有一定延迟，不支持像QMT那样的毫秒级持续推送。举例来说，使用QMT的五档全推行情，用户可以实时获取盘口队列和最新逐笔成交[29][8]；但JQData无法提供盘口逐笔的持续推送。对于**高频或日内策略**，QMT的数据时效性显然更好。即便是日线级别数据，QMT在收盘后即可通过接口获取当日数据，龙虎榜等数据也会在交易所公布后及时更新；JQData通常也能在当晚更新日线和财务数据，但实时行情方面无法用于盘中策略执行。

- **稳定性：**这里分两层含义：**数据稳定性**和**服务稳定性**。在数据质量方面，QMT直接对接交易所行情（通常由券商系统提供底层行情），数据准确完整，并提供除权除息处理等功能，历史数据经过验证，**准确性高**。JQData的数据源亦来自交易所公告和行情，但用户偶有反馈某些冷门数据有缺失或延迟，需要及时反馈修正。不过总体两者数据质量都较高。在服务可用性方面，JQData依赖聚宽云服务，一旦网络不畅或服务端维护，用户请求可能失败或超时；QMT则主要由本地客户端或直连服务器获取数据，可自行选择备用行情服务器提高可用性[17]。即使一个站点故障，xtquant会自动尝试其它站点，从而**降低服务中断概率**[17]。当然，QMT模式下也依赖用户本地环境稳定，如本地网络、终端运行状况等。整体而言，在国内环境下两者服务都比较稳定，但**QMT允许多线路热切换、数据缓存本地**，因此在网络波动情况下成功率可能更高[30]。对于韬

睿系统这样的生产环境，可以通过编写代码监测QMT连接状态并快速切换站点或转向备用源，从而实现更健壮的数据服务。

- **接口灵活性：**JQData的Python SDK使用相对简洁，上手容易，比如一行`get_price`即可获取多标的历史K线，`get_fundamentals`即可查询财务指标，返回Pandas DataFrame方便后续处理。QMT的xtquant接口则稍显底层，需要先下载再获取，调用步骤多一些。另外在QMT环境中，内置Python策略与原生Python调用有所差别（内置模式下有Context对象封装，原生模式需要使用xtdata模块函数）。迁移初期，开发者需要熟悉QMT接口。例如获取历史行情，在JQData中是`get_price('000001.XSHE', count=5, unit='1d')`，而在xtquant需调用`xtdata.download_history_data('000001.SZ','1d')`然后`xtdata.get_market_data_ex(...)`，返回的数据结构也可能是dict嵌套Pandas。**换言之，JQData接口更高级抽象，而QMT接口提供更底层的控制**。从灵活性看，QMT接口因为底层化，反而允许自定义很多行为，比如订阅实时数据回调、自定义板块等 [31]（QMT支持创建自定义板块分组 [32]），这些是JQData不具备的。QMT甚至提供一些高级功能接口（如获取席位成交、订单流信息）满足特定需求。而JQData的优势是简单明确的函数调用，更快速地获取常规数据。对于韬睿团队，如果希望**最大程度掌控数据获取过程和本地存储**，QMT的接口更为灵活；如果偏好**快速原型和简单调用**，需要封装QMT接口使之类似JQData的用法。

综合评价：QMT在数据的广度（期权可转债等）、深度（Tick和逐笔、资金流）以及实时推送能力上明显胜出，是实盘交易和日内量化的不二选择 [2]。JQData则胜在成熟易用，适合快速取数与回测。在稳定性和性能上，QMT通过本地部署降低了对第三方服务的依赖，批量调用性能优于JQData。但相应地运维成本稍高（需管理本地数据下载、环境配置）。在成本方面，JQData属于商业付费数据服务，费用不低（通常按年订阅不同模块）；QMT的行情VIP权限费用相对低廉，而且很多数据随券商接口免费或一次购买即可长期使用。这意味着从**性价比**看，QMT很有吸引力。因此，对于韬睿量化系统，如果追求更实时丰富的数据且愿意投入一定的开发精力，那么QMT完全可以作为主要数据源，替代大部分JQData的功能。下一步需要评估哪些模块可直接切换。

韬睿量化系统中QMT可替代聚宽的模块

根据上述比较，QMT几乎覆盖了聚宽JQData提供的所有数据类别。具体到韬睿量化系统，目前使用聚宽数据的模块，大概率包括：**行情历史数据模块、实时行情订阅模块、财务数据模块、行业概念模块、龙虎榜数据模块、资金流监控模块**等。以下分析每一部分QMT是否可以替代，以及替代方案：

- **行情/行情历史模块：**QMT可以完全替代聚宽的行情数据接口。对于**日线和分钟线历史**，QMT提供同样甚至更长的历史跨度（如日线全历史、分钟线近3年） [5]。通过xtquant的`xtdata.get_market_data_ex`或`get_market_data`接口，可以获取单个或多个标的在指定周期的历史行情，与JQData的`get_price`功能等价。**实时行情**方面，QMT更具优势：可使用`xtdata.subscribe_quote`订阅实时行情推送并在回调中接收最新数据 [33]。这比聚宽需要轮询最新价格的方式高效且实时。韬睿系统中如有**实时监控或交易信号模块**，可以用QMT的订阅机制取代JQData的轮询，获得毫秒级行情更新。

- **财务数据模块：**QMT的数据字典提供了丰富的财务指标和财报数据接口 [34] [26]。支持按季度、年度提取上市公司财报科目，涵盖资产负债表、利润表、现金流量表、主要财务指标、股本变动、股东人数、十大股东等 [26]。这些与聚宽的财务数据接口（如`get_fundamentals`返回的一系列财务字段）对应。QMT甚至还提供了**“问董秘”**信息和**交易所公告**数据接口 [35] [36]，这些是聚宽未必涵盖的。韬睿系统中如有定期财务因子计算或财报事件分析模块，可以直接改用QMT的`xtdata.get_financial_data`接口获取所需数

据。唯一需要注意的是字段命名和取值可能与聚宽有差异，需要在替换时对照字段含义，保证一致性。整体而言，**财务数据模块可以由QMT无缝接管**，数据范围和时效都能满足要求。

- **行业与概念板块模块：**聚宽常用的行业概念数据（如获取某概念成分股列表、行业分类信息）可以由QMT替代。QMT提供`xtdata.download_sector_data()`下载全部板块分类[37]和`xtdata.get_sector_list()`获取板块列表[38]。板块列表中包含行业板块（如申万行业，以代码`1000SW1`开头[39]）和概念题材板块。随后可使用`xtdata.get_sector_components(板块代码)`或`get_sector_list`结合`get_market_data`来获取板块成分股[40][41]。事实上，QMT维护了自己的**迅投行业分类**和概念分类[42]（迅投行业类似申万或证监会行业，自有编码）。韬睿系统中比如**"主线成分股"**模块，应当指当前市场热点主线题材的板块成分，这可以通过QMT的概念板块数据实现：获取相关概念板块的成分股列表，再用于策略。QMT还支持**自定义板块**，可将一组股票保存成自定义板块方便管理[31]。因此，不论标准的行业概念，还是自定义主题板块，QMT均可胜任，**可以全面取代聚宽的行业/概念模块**。

- **龙虎榜数据模块：**沪深交易所每日公布的龙虎榜数据，聚宽通过`get_billboard_list`等接口提供。QMT数据服务同样涵盖**龙虎榜**信息[43]。用户可通过QMT获取每日上榜个股及其上榜原因、买卖金额、营业部列表等数据（具体接口如`xtdata.get_top_list`等，需查阅QMT文档确定）。由于QMT定位实盘，一些游资席位数据、龙虎榜监控也是其特色功能的一部分（例如迅投社区有席位数据接口介绍）。因此，**龙虎榜模块完全可以用QMT替换**。QMT可能提供更及时或更详细的席位成交数据，比聚宽更深入。韬睿系统如需每天收盘后自动抓取龙虎榜股并存储分析，用xtquant可实现相同效果。

- **资金流数据模块：**若系统中包含对资金流向、主力资金监控的功能，QMT提供了**多维度的资金流数据**。例如，有**大单净流入**、超大单净流入等逐级资金流统计，以及北向资金净买入等[44]。QMT的行情字段中直接计算了净流入额（如总买额、总卖额），在Level2扩展数据中可以获取[44]。另外还有专门的北向资金（沪股通/深股通）持股统计接口[43]。聚宽对于资金流的支持相对有限（可能仅有简单的指标，或需要用户自行根据逐笔交易计算）。因此使用QMT可以更便利地得到这方面数据。例如，QMT有"资金流量数据"接口，提供日内逐分钟的主力资金流入流出量。韬睿系统的**资金流模块**可以改为定时调用QMT的资金流API，将得到的数据写入数据库供分析。综上，**资金流相关功能QMT能够较好地替代**，并且有更实时的推送（QMT可订阅资金流实时变化）。

- **其他可能的模块：**如果韬睿系统有**交易日历**、**停复牌信息**、**股票基本信息**等需求，QMT也都有相应接口。比如`xtdata.get_trade_days()`可以获取交易日历，`xtdata.get_instrument_detail()`可查证券基本信息（上市日期、是否ST等）[45]。这些都可以用来替换聚宽的`get_trade_days`、`get_security_info`等函数。此外，假如系统有使用聚宽的**宏观经济数据**或**指数成分**，QMT也提供指数成分股权重下载接口[46]和部分宏观指标（这点需确认QMT是否开放宏观数据，如果没有可暂时保留聚宽该部分）。但从提问列出的重点看，宏观数据不在主要考虑范围。

替代总结：韬睿量化系统中依赖聚宽的**行情、财务、板块、龙虎榜、资金流**等核心数据模块，均可以用QMT提供的对应接口加以替换，而且**功能更强或更实时**。这意味着从数据功能角度，QMT足以覆盖系统需求。需要仔细规划的是**替换过程**，包括代码改造和数据校验，在正式切换前建议进行充分的对比测试（例如对比同一天聚宽与QMT的财务指标、价格数据，确保一致）。除了极少数聚宽特有的数据（如若有宏观或特定指数计算），大部分模块迁移不会导致数据缺失。

QMT数据写入MongoDB的方案与同步机制

采用QMT作为数据源后，需将获取的数据写入现有的MongoDB，以供韬睿系统各组件查询使用。整体思路与目前聚宽数据入库类似，但由于QMT数据获取方式的特点，需要做相应调整。以下是建议的**数据入库与自动同步方案**：

1. **初始历史数据导入：**在切换数据源时，首先利用QMT接口将所有必要的历史数据一次性下载并存储至MongoDB。例如：
2. **股票列表获取：**调用QMT接口获取全部股票代码清单（可通过下载板块分类信息后提取沪深主板/创业板所有股票代码，或通过`download_history_contracts()`获取全部合约列表[47][48]）。整理出股票代码列表用于批量处理。
3. **行情数据导入：**对每只股票，调用`xtdata.download_history_data(code, period='1d')`下载日线历史到本地缓存，然后用`xtdata.get_market_data_ex`读取并得到Pandas DataFrame，再写入MongoDB对应的日线行情集合。分钟线（如1分钟、5分钟）数据量较大，可按需导入近几年数据（QMT支持1分钟3年历史[49]）。也可以通过多线程或批量接口提升下载效率（xtquant允许一次获取多只股票的数据，但需注意内存和频率限制）。初次导入建议分市场或分批次进行，避免一次请求过大。
4. **财务数据导入：**利用`xtdata.download_financial_data(stock_list)`一次性下载所有标的的财务数据至本地[24]。然后使用`xtdata.get_financial_data`提取各公司各期财务指标，按财报日期写入MongoDB相应集合（可设计库表结构与现有聚宽财务数据表一致，例如按季度存储）。QMT财务数据字段较多，可选择常用指标存储，并确保字段命名清晰。
5. **板块数据导入：**调用`xtdata.download_sector_data()`获取行业与概念分类信息[37]。随后调用`xtdata.get_sector_list()`拿到所有板块代码与名称[38]。再对每个板块调用`xtdata.get_sector_components(sector_code)`获取成分股列表，写入板块成分集合（或按板块名称存储成分列表）。特别关注主要使用的概念板块以及申万行业板块，保证和系统原有分类对应。
6. **龙虎榜及资金流初始数据：**历史龙虎榜可以考虑近几个月的重要数据，通过调用QMT的龙虎榜接口按日期获取并存储（日频数据）。资金流向历史数据（如个股每日主力净流入）也可批量获取一定区间的数据填充数据库，以备策略参考。

通过以上步骤，可建立起MongoDB中**从QMT获取的完整历史数据库**。这个过程可能需要数小时到数天（取决于数据量和带宽）。建议在非交易时段进行，并做好进度日志记录，方便中断续传。

1. **定时增量更新机制：**完成初始导入后，需要设计每日或实时的同步策略，使MongoDB中的数据保持最新：
2. **日线/分钟线行情增量：**可以在每日收盘后运行一个任务，调用`xtdata.download_history_data(code, period='1d', incrementally=True)`对所有股票进行增量下载[22]。使用incrementally参数，xtquant只会下载自上次下载之后的新数据，大幅减少数据量[22]。然后读取当日K线数据并写入MongoDB（日线表）。分钟线数据可采用类似方法，在盘后下载当日所有分钟数据写库。如果需要盘中更新分钟线，则可以在交易时段定时（如每5分钟）运行一次下载当天分钟线最新数据并合并入库。由于QMT支持实时推送，另一种选择是在盘中通过subscribe订阅分钟行情，在每根分钟线完成时通过回调获取数据并直接写库，这样实时性更强。不过实现稍复杂，可根据需求选择**定时轮询**还是**事件驱动**。
3. **实时行情更新：**对于需要在MongoDB中维护**最新报价**或Tick数据的场景，可使用QMT的订阅接口：在开盘时订阅全市场或关心标的的Tick/逐笔数据，写入缓存。当达到批量后（比如每累计1000条成交）批量写入MongoDB Tick集合。这种做法可实现Tick级别的数据同步。但要注意Tick数据量巨大，需要评估MongoDB承载能力和存储成本，通常只对特定策略标的记录Tick。
4. **财务数据更新：**财报发布一般在每季度末至下季度初。可以设置一个季度任务：如每季度末和初分别调用`xtdata.download_financial_data()`更新本地财务数据，再对比MongoDB中最新财报日期，插入

新增季度的数据。QMT财务数据接口提供当最新财报数据公布后会更新，因此确保每季同步。对于一些高频变动的指标（如股东人数每月公布），也可按月抓取更新。

5. **板块和概念更新：**行业分类一般不会频繁变化，但概念题材会有新增或调整。可以每周或每月调用一次 `download_sector_data` 刷新板块信息，检查是否有新板块或成分变动。如果有则更新MongoDB相应集合。例如新上市股票会归入某行业板块，应更新其成分。还有用户自定义的"主线"概念可能变化，需人工确认板块名单调整并更新数据库。
6. **龙虎榜与资金流更新：**每日收盘后，交易所公布龙虎榜。当晚可调用QMT龙虎榜接口获取当天数据并写库，包括上榜股及金额、席位等。主力资金流向数据可每日盘后从QMT获取当日总计的净流入额写库，或在盘中累积计算。北向资金每日增减也可通过接口取得并存储。

为实现上述定时任务，可以结合调度系统（如cron或Airflow等）安排脚本运行。在脚本设计中，充分利用xtquant的增量下载特性，避免重复拉取大批历史。对于实时部分，采用**异步推送+批量写库**的策略，以平衡时效和性能。

1. **数据校验与回滚：**在切换到QMT数据后，建议实现一定的数据校验机制。例如每日更新完毕后，对比MongoDB中当日几项关键数据（收盘价、成交额等）和券商行情或交易所公布数据，确保无误。如发现异常（例如某些股票停牌导致无数据等），要能识别并处理（如跳过或补充特殊标记）。另外，也可以暂时保留聚宽的数据获取作为备份，用于交叉验证。在最初上线阶段，每天对比部分股票的QMT数据和聚宽数据，确认一致后再投入正式使用。

2. **性能考虑：**一次性导入和每天增量更新都要考虑执行效率。MongoDB写入可采用批量操作（如一次插入多行），提高速度。xtquant支持批量获取一定数量股票的数据，但官方未明确限制数量，建议分批，比如每次请求50只股票的日线数据，然后循环。对于上千只股票的数据下载，合理的并发可以显著缩短总耗时。然而并发过多可能触发xtquant的**请求限制**（知识库提到行情请求有并发数上限 50 51 ）。可以逐步测试调整批处理大小，使下载和写库在可接受时间窗口内完成（如每日增量更新尽量在几分钟内完成）。

总而言之，**QMT的数据入库流程**可以概括为："先全量导入，后定期/实时增量更新，结合缓存和校验"。完成这套机制后，MongoDB将持续维护最新的行情和财务信息，为上层策略和分析模块提供支持。需要强调的是，这套机制实际上和之前聚宽数据的入库流程类似，只是换了数据来源和接口，因此实现起来不会脱离团队既有的经验。

是否采用双数据源架构（聚宽+QMT）的建议

考虑到数据源切换涉及稳定性和覆盖面的权衡，一个**双数据源架构**也许是过渡期内稳妥的选择。即同时接入聚宽JQData和迅投QMT，将二者优势互补、相互备份。以下是对此架构的分析和调度策略建议：

1.　双数据源的必要性：　如果QMT已经完全满足所有数据需求，理论上可以独立承担生产任务，无需并行使用JQData。但在实际运营中，**备用数据源**可以提高系统可靠性：当主要数据源服务中断或数据异常时，备用源可及时接管，避免因单一数据源故障导致系统功能瘫痪。例如，万一迅投行情服务器网络异常，系统仍可暂时切换到聚宽的数据以继续基本功能。这对实时性要求不高的模块尤为适用（如日线级别分析可以稍微延迟用备用源补足）。此外，在切换初期，双源并行还有助于**验证数据一致性**，及时发现差异。综上，建议至少在QMT上线初期保留聚宽作为辅助，形成双数据源架构。在运行一段时间、验证QMT稳定可靠且无数据缺项后，可根据成本考虑逐步停用聚宽。

2. 调度策略：双数据源下，需要定义清晰的**主次顺序**和任务分配。推荐方案是以**QMT作为主数据源**、聚宽为辅：
- **正常调度下：**所有数据更新任务优先调用QMT接口获取。如QMT返回正常，则将结果写库并提供给业务使用。聚宽接口仍可定时调用，但作为校验或备用，不直接覆盖主数据。例如每天收盘后，系统先用QMT更新日线数据，并将部分结果与同时调用的JQData数据做对比。如果一致或差异在可接受范围，则以QMT数据为准。如发现某股票

QMT数据缺失或明显异常，则记录告警并考虑启用聚宽数据补足。
- **负载分担：**在某些高频场景，可以让两个源各担一部分任务，以降低单源压力。例如Tick级别实时推送用QMT（因为聚宽无该功能），而日终某些冗长指标计算用聚宽取历史数据（如果聚宽有更长历史或更方便的指标接口）。不过这种划分要确保最终数据口径一致。通常还是以一个源为主更简单。

3. 故障切换机制： 需要制定**自动故障切换**策略。当检测到主源无法提供服务时，系统应平滑切换到次源。实现上，可以封装一层数据获取模块：每次请求数据时，先尝试QMT，设定短超时；如果失败或超时，则fallback到JQData接口获取，并发出警报提示运维人员。同时可以标记当前切换状态，定期重试主源，恢复后再切回。举例：行情订阅过程中，如与QMT服务器连接断开，可立即调用聚宽的`get_price`获取最新价格以应急；对于财务数据，如季度末QMT数据更新稍慢，可暂时用聚宽同类数据填充，待QMT数据完整后再更新覆盖。在这种机制下，**双源保证了数据连续性**。不过也要注意同步：一旦使用备用源获取了数据，要在主源恢复后检查是否有偏差，必要时修正数据库中暂存的数据（例如不同数据源计算口径略有不同，需要统一）。

4. 策略与数据一致性： 双数据源意味着可能同一指标来自不同渠道，必须确保不会因为来源不同造成策略逻辑不一致。为此应该：在同一时间窗口内，同一数据应只来自一个来源，不要混杂。例如单只股票的一段历史行情要么全部用QMT，要么全部用聚宽，不要半段用QMT半段用JQ，以免因前复权等处理差异出现跳变。通过设置数据优先级和**明确的数据合并规则**，可以避免此类问题。

5. 双源架构的生命周期： 不建议长期依赖双份付费数据，因为这会增加成本和系统复杂度。更实际的做法是：**将双数据源作为过渡方案**。在QMT稳定运行若干月且证明覆盖全面后，逐步减少对聚宽的依赖，最终完全停用聚宽以节省开支。但保留一套紧急备用方案（可能无需实时订阅，仅需关键数据的备用获取能力）。这样既有容错又不浪费资源。

总结：采用双数据源架构在切换初期是稳健的选择。可以设计**"主用QMT+备用JQData"**的模式，通过智能调度确保服务不断。关键是实现自动监测和切换，以减少人工干预。长远看，如果QMT表现可靠，则双源架构的必要性降低，可以简化为单源架构，从而达到降低成本的目标。

QMT接入韬睿系统的技术集成建议书

综合以上分析，下面给出一个**QMT接入韬睿量化系统**的完整技术方案，涵盖安装配置、API使用、模块对接示例以及数据入库与更新流程。此方案旨在帮助决策和指导实施，让韬睿系统平稳过渡到以QMT为主的数据源架构。

1. 系统架构概览与安装配置

目标架构：采用QMT作为主要数据源，所有数据先通过xtquant接口获取并进入本地MongoDB，韬睿系统各模块从MongoDB读数据供策略和分析使用。聚宽JQData作为备用，在需要时补充。系统应包括：QMT数据获取服务、数据存储层（MongoDB）、量化策略/分析应用层。

环境准备：
- **账号与权限：**注册迅投投研账户并购买所需的"行情用户VIP"权限（以及期权等附加数据权限，如果策略涉及）[6]。获取账号的API Token [12] 并妥善保存。
- **软件安装：**在服务器（推荐64位Linux或Windows Server）上安装Python环境（建议3.8~3.10，xtquant最新版本支持到3.12）[52]。通过`pip install xtquant`安装xtquant库。如果使用Linux，按照迅投知识库提供的Linux版包进行安装 [18]。安装完成后，在Python中执行`import xtquant`测试是否成功导入。若导入失败，检查依赖（Windows可能需VC++运行库，Linux需glibc等）。

- **QMT终端/组件：**如果使用Token模式，不需要完整安装QMT终端。但可以下载一个QMT交易端用于调试（Windows下安装券商QMT客户端或迅投投研终端，以便获取MiniQMT可执行文件）。对于Windows服务器，若需要实盘交易，安装券商QMT后可在其`bin.x64`目录找到`XtMiniQmt.exe`以备使用 [19]。对于数据获取，**不强制要求运行MiniQMT**，因为我们将使用Token直连。
- **配置文件：**创建一个配置文件（如`qmt_config.py`或配置项）存储QMT相关参数：API Token，行情服务器列表（可使用迅投VIP行情站点列表 [17]），本地数据缓存目录等。设置`xtdatacenter.set_token(TOKEN)`以及`xtdatacenter.set_allow_optimize_address([...VIP站点列表...])`以限定使用高速服务器 [17]。如果需要指定本地缓存路径，调用`xtdatacenter.set_data_home_dir('指定缓存目录')`。
- **初始化连接：**编写初始化脚本，在系统启动时执行：

```
from xtquant import xtdatacenter as xtdc, xtdata
xtdc.set_token('YOUR_API_TOKEN')
vip_servers = ['115.231.218.73:55310','115.231.218.79:55310',  # 迅投绍兴VIP
               '42.228.16.211:55300','42.228.16.210:55300',  # 郑州联通VIP
               '36.99.48.20:55300','36.99.48.21:55300']      # 郑州电信VIP
xtdc.set_allow_optmize_address(vip_servers)
xtdc.init()
port = xtdc.listen(port=(58620,58650))  # 自动挑选可用端口
xtdata.connect(port=port)
```

上述代码将启动xtquant数据服务并建立行情连接 [53] [15]。打印输出确认已"连接上了"VIP行情服务器 [16]。此初始化应在任何数据获取前完成。可以将其做成守护进程，使其一直运行以接受行情推送（也可结合`xtdata.run()`进入消息循环 [54]）。

2. API使用示例与模块接口对接

行情获取示例：以下展示如何通过xtquant获取行情数据，并对比聚宽调用：
- *例1*：获取单只股票日线历史（平安银行，最近5日收盘价）：
聚宽：

```
import jqdatasdk
jqdatasdk.auth('user','pass')
df = jqdatasdk.get_price('000001.XSHE', count=5, unit='1d', fields=['close'])
print(df)
```

QMT：

```
from xtquant import xtdata
xtdata.download_history_data('000001.SZ', period='1d', count=5)  # 先下载5日数据
data = xtdata.get_market_data(['close'], ['000001.SZ'], period='1d', count=5)
print(data['000001.SZ'])  # QMT返回一个dict，键为股票代码，值为数据DataFrame
```

在QMT中，我们明确指定字段['close']和标的列表['000001.SZ']来获取收盘价数据。需要先用`download_history_data`下载所需范围的数据，然后`get_market_data`（或`get_market_data_ex`）返回结果。**结果结构：**QMT的`get_market_data`返回一个字典，value通常是numpy数组或DataFrame；`get_market_data_ex`更高级一些，可以直接返回带日期index的DataFrame。调用后，要整理成和聚宽DataFrame类似的格式再写库。

- 例*2*：获取多只股票分钟线数据：可利用QMT批量接口：

```
codes = ['600000.SH','600001.SH', '600002.SH']
xtdata.download_history_data2(codes, period='1m', begin_time='20231201')
# 批量增量下载这些股票1分钟数据
df_dict = xtdata.get_market_data_ex(['open','close','high','low','volume'],
codes, period='1m', start_time='20231201')
# df_dict是股票代码到DataFrame的映射，可以将其逐个写入数据库
```

以上`download_history_data2`可一次下载多只股票的数据，简化循环 [55] 。数据获取后字典中每个DataFrame包含所请求字段的分钟线序列。写入MongoDB时，可按股票逐条插入，或将DataFrame直接转为JSON批量插入。

- 例*3*：获取板块成分股：（例如获取"半导体"概念板块的所有股票）

```
xtdata.download_sector_data()  # 下载最新板块分类
sectors = xtdata.get_sector_list()  # 获取板块列表
# 找到名称包含"半导体"的板块
target_sectors = [s for s in sectors if '半导体' in s]
for sec in target_sectors:
    comp = xtdata.get_sector_components(sec)  # 获取该板块成分，返回股票列表
    print(sec, comp)
```

QMT的板块编码如`885916芯片产业`等都会出现在sector_list中，拿到后即可获取对应股票列表，进而用于行情数据下载或存库。聚宽类似的是调用`get_concept`/`get_industry`。使用QMT需先下载板块分类才能保证本地有数据 [37] 。

- 例*4*：获取财务指标：（如获取招商银行2023年三季度净利润）

```
code = '600036.SH'
xtdata.download_financial_data([code])  # 下载财务数据到本地
# 获取净利润（归母净利）字段值，xtquant提供多种财务数据接口
data = xtdata.get_financial_data(code, 'ProfitStatementTOT.NETPROFIT')  #
ProfitStatement表中的NETPROFIT字段
print(data.tail(1))  # 打印最新一期净利润
```

QMT将财务数据划分为不同表，可以通过字段路径指定[56]。例如这里`ProfitStatementTOT.NETPROFIT`表示合并利润表的净利润。结果`data`通常是一个时间索引的Series或DataFrame，包含历史净利润。我们可以将其插入MongoDB财务表。如果需要多指标一起拿，也可以使用`get_financial_data(code, [field1, field2, ...])`。聚宽的获取方式是`get_fundamentals(query(profit.net_profit, balance.total_assets).filter(...), statDate=...)`，相对复杂且分次取每期；QMT一次下载后多期数据均在本地，可以更高效批量获取。

- 例*5*：实时订阅与回调：（获取五档盘口与逐笔成交）

```
def on_tick(data):
    # 假设data结构为 { '600000.SH': { 'bidPrice':[...],'askPrice':
[...], ... } }
    tick = data.get('600000.SH', {})
    if tick:
        best_bid = tick['bidPrice'][0]
        best_ask = tick['askPrice'][0]
        print("600000 最新一档买卖:", best_bid, best_ask)
xtdata.subscribe_quote('600000.SH', period='tick', count=-1,
callback=on_tick)
xtdata.run()  # 开始监听回调
```

这一段展示了如何订阅一只股票的Tick行情。QMT的订阅函数可指定回调函数，当新Tick到达时自动调用[33]。我们的回调将提取盘口一档买卖价打印。实际应用中，可在回调里将Tick数据存入MongoDB或推送给策略模块。需要注意`xtdata.run()`会阻塞线程以持续监听，因此可在单独线程运行，或采用非阻塞模式（例如自行用while循环+sleep检查`xtdata.get_market_data`）。相比聚宽，聚宽没有这样的推送接口，只能不断轮询`get_price`，实时性较差。因此韬睿系统可以利用QMT订阅，将**实时交易模块**的行情源切换为QMT，获得更及时的信号触发。

模块接口对接：在将上述接口融入韬睿系统各模块时，可以考虑**封装一层适配**。即编写与原聚宽接口同名的函数，但在内部调用QMT的实现，从而尽量减少上层代码改动。例如，定义`get_price_qmt(security, start_date, end_date)`内部按照QMT逻辑获取数据并返回DataFrame，函数签名和行为与聚宽`get_price`类似。这样，上层模块只需替换导入的库或函数名称，不用重写全部逻辑。类似地，可封装`get_fundamentals_qmt`，`get_index_weights_qmt`（对应指数成分权重QMT接口[46]），`subscribe_quote_qmt`等。在适配函数内部处理QMT的数据格式并转化为策略所需格式。经过这一层包装，模块替换将更平滑。测试时可以同时调用聚宽和QMT的适配函数对比输出，确保一致。

3. 数据存储与更新流程

前文已经详细描述了数据如何入库与同步，这里将其纳入整体方案强调关键点：

- **本地缓存 vs 数据库：**xtquant本身会将下载的数据存于本地硬盘（默认在用户目录下的xtquant_data文件夹，或自定义目录）。这份缓存可视为数据库的源头，但由于韬睿系统已有MongoDB集中存储，建议仍以MongoDB为最终数据源。可以定期清理xtquant本地缓存，避免重复占用磁盘。不过第一次全量下载后缓存保留，后续增量下载速度更快，所以缓存也有用。权衡之下，可保留主要品种的缓存，加快下载，而把非实时访问的数据下载完即删除。

- **自动同步架构：**实现一个调度脚本（或服务）模块，专职负责调用xtquant接口并更新MongoDB。采用**分任务调度**：

- 交易日每日的收盘后任务（T+0 日线数据更新，龙虎榜更新等），
- 交易日每晚的财务数据检查（是否有新公告发布，需要更新财务库），
- 每月/季度任务（如板块分类更新、股本变动数据更新等），

- 盘中实时任务（如分钟线更新，Tick流写库等）。
  这些任务可用定时器触发，也可由事件（如检测到新公告）触发。

- **故障与补救：**当数据同步出现异常（如某天行情缺失），设计补救措施：可人工或程序检测后，使用聚宽数据进行补录，或者次日调大同步范围补齐前一天缺口。MongoDB中的数据需有字段标记来源或更新时间，以便跟踪。

- **数据验证：**推荐在MongoDB层面做一些**数据一致性验证**。例如每天对所有股票日线进行完整性检查（应有交易日就有一条记录，缺漏则报警），对财务数据的资产=负债+权益等恒等式平衡关系做抽查，确保入库数据正确。利用双数据源，可抽样对比聚宽与QMT的关键指标，如总市值、PE等，看是否一致，来校验QMT计算逻辑正确。通过这些质量控制，保证用QMT替换后系统结果不受负面影响。

4. 项目实施步骤与决策建议

实施步骤：
1. **测试验证阶段：**先在测试环境接入QMT，对比聚宽数据。选取几个交易日的数据，分别用QMT和JQData获取行情、财务、龙虎榜等，比较差异。如果发现指标定义或算法不同，及时调整处理逻辑（例如复权方式不同，就统一设为后复权等）。这一步确保QMT数据可用且理解透彻。
2. **并行运行阶段：**在生产环境中部署QMT数据获取服务，但暂不切断聚宽。让两个源同时跑一段时间。观察QMT的数据延迟、稳定性。如果两者结果一致，则逐步让业务模块切换读取QMT提供的数据（从MongoDB）。此时聚宽仍获取但作为备份。
3. **正式切换阶段：**确认所有模块都已成功从QMT获取所需数据并正常运行后，可停止聚宽数据的日常获取，仅保留应急调用。监控系统在完全使用QMT后的表现，包括日志中的数据错误率、网络性能等。如果一切正常，考虑不再续费聚宽服务，实现成本节约。
4. **优化完善阶段：**针对QMT的特点优化系统。如充分利用QMT实时推送简化策略触发逻辑，利用其自定义板块功能优化选股模块等。也可以探索QMT提供的**因子库**（需额外购买）和指标计算接口，将部分原本在本地计算的内容交给QMT完成，提升效率 [6] 。

风险与应对：决策上，需要考虑QMT作为新数据源可能的风险：例如QMT由第三方公司运营，其服务质量和持续性需要关注。不过目前迅投QMT用户群广泛，稳定运营多年，风险不高 [1] 。另一个风险是完全依赖本地运行，需要对我们自己的系统稳定性提出更高要求（相当于把数据任务从聚宽云转移到自己手中）。对此应加强系统监控，确保服务器性能和网络满足7x24运行。

结论：综合各方面，迅投QMT作为量化系统数据源是**可行且有优势**的。它能够提供聚宽JQData的大部分乃至全部数据，且在实时性和数据细节上更胜一筹 [2] 。对于韬睿量化系统而言，引入QMT可以降低长期数据服务成本，提高实时交易能力，并整合更多数据类型（期权、资金流等）的分析。建议采取**主用QMT、辅以聚宽过渡**的方案，在保证平稳过渡的同时尽快发挥QMT数据源的价值。通过上述技术方案的实施，韬睿系统将能够优雅地整合QMT，实现从研究到实盘的一体化量化数据支持，支撑更复杂高效的量化策略开发。

(完) 4 5 1 26

---

1 2 迅投QMT量化软件怎么样？做个人量化策略哪家券商支持QMT？_交易_能力_用户
https://www.sohu.com/a/788016555_121750917

3 行情函数 - 迅投知识库
https://dict.thinktrader.net/innerApi/data_function.html

4 券商版QMT的历史tick数据是不是只能下载一个月的？ - 有问必答 - 迅投QMT社区 - Powered by Discuz!
https://www.xuntou.net/forum.php?mod=viewthread&tid=2021&mobile=no

5 6 8 12 25 29 49 快速开始 | 迅投知识库
http://dict.thinktrader.net/dictionary/

7 37 38 39 40 41 42 47 48 行业概念数据 | 迅投知识库
http://dict.thinktrader.net/dictionary/industry.html

9 10 13 14 15 16 17 21 22 23 24 27 28 33 53 54 完整实例 | 迅投知识库
https://dict.thinktrader.net/nativeApi/code_examples.html

11 聚宽策略迁移至QMT | 迅投知识库
https://dict.thinktrader.net/strategy/JoinQuant2QMT.html

18 Linux版xtquant快速开始指南 - 迅投知识库
https://dict.thinktrader.net/nativeApi/
Linux%E7%89%88xtquant%E5%BF%AB%E9%80%9F%E5%BC%80%E5%A7%8B%E6%8C%87%E5%8D%97.html

19 量化交易的库- 微信公众号--共鸣圈 - 博客园
https://www.cnblogs.com/welhzh/p/17679348.html

20 【零基础部署】QMT策略24小时自动交易| 云服务器+xtquant保姆级教程
https://zhuanlan.zhihu.com/p/1959970965472655022

26 34 35 36 43 45 55 56 股票数据 | 迅投知识库
http://dict.thinktrader.net/dictionary/stock.html

30 xtquant 中切换IP 实现更稳定地获取数据
https://kaihu51.com/quant/1423

31 32 46 50 51 XtQuant.XtData 行情模块 | 迅投知识库
https://dict.thinktrader.net/nativeApi/xtdata.html

44 数据结构 - 迅投知识库
https://dict.thinktrader.net/innerApi/data_structure.html

52 miniQMT & pyKX_miniqmt linux-CSDN博客
https://blog.csdn.net/ffossil/article/details/139989931