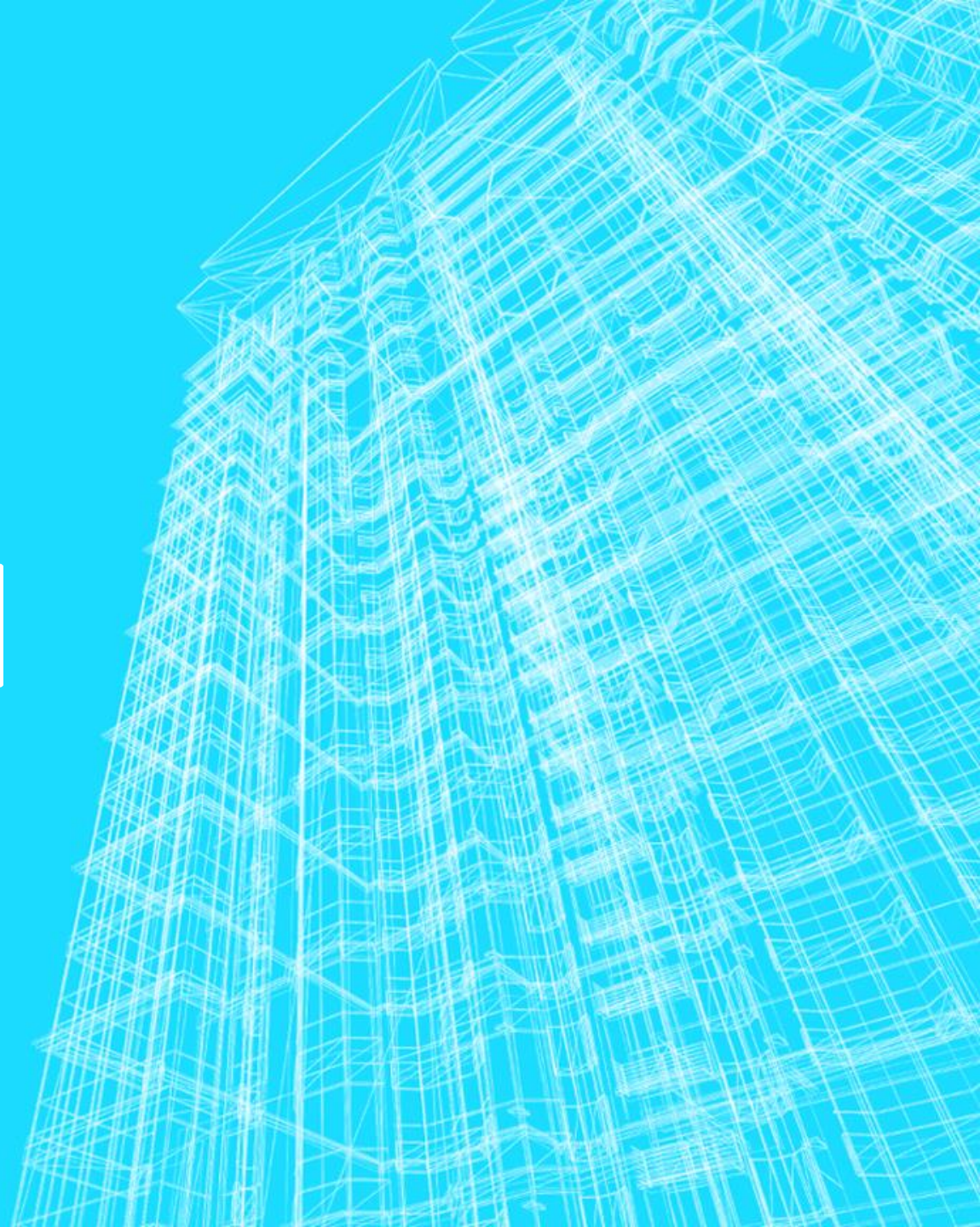
量化交易实战培训

基于开源框架vn.py

Day 3/6

李来佳 QQ/WeChat 28888502

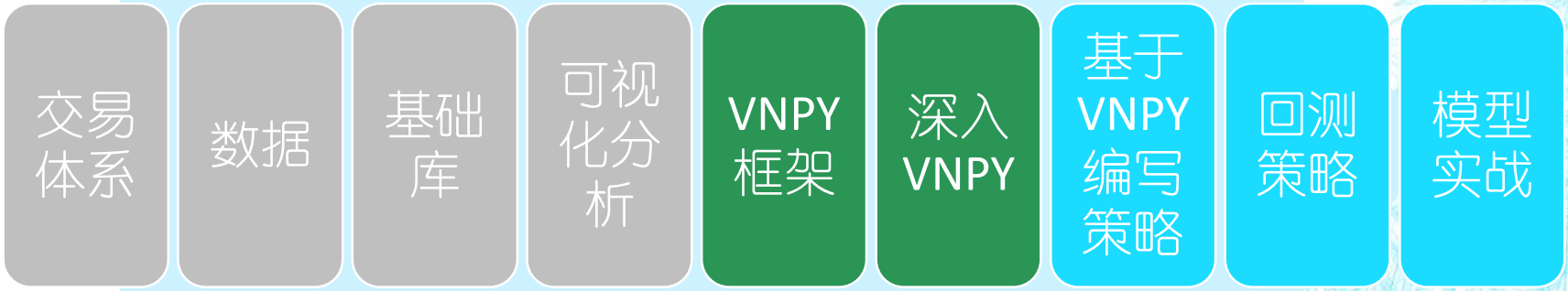


1. 内　容：　较为完整地讲解量化交易体系，并通过vn.py实战量化策略。

2. 听　众：　具有一定编程基础的量化从业人员。

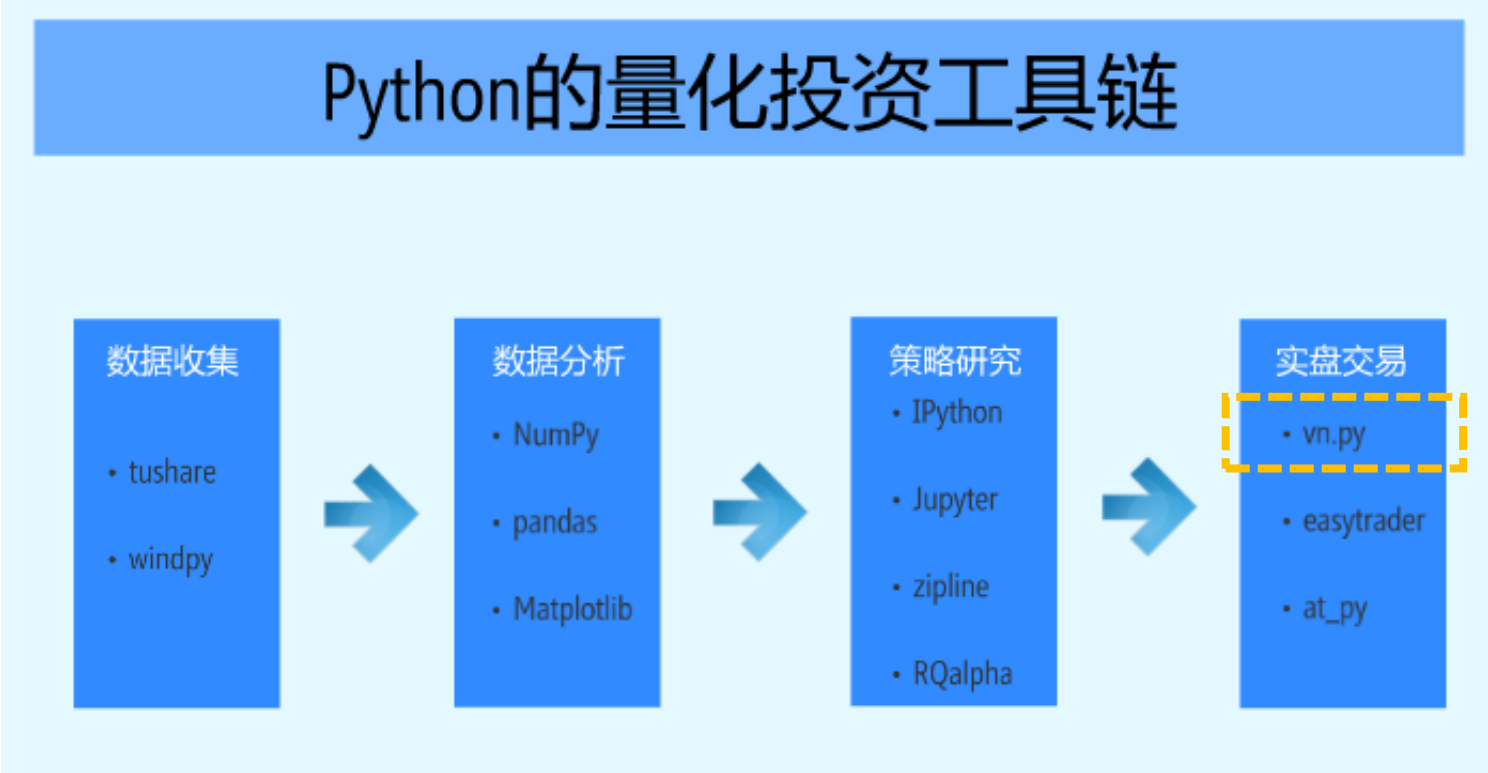
3. 讲　师：　一群爱好交易的程序员。

4. 感　谢：　vn.py的创始人陈晓优和他的开源社区



5.1 VNPY目标与定位

- 成为一套开源的、全面的交易程序开发框架。



PyAlgoTrade

- VNPY围绕交易为主

C/C++

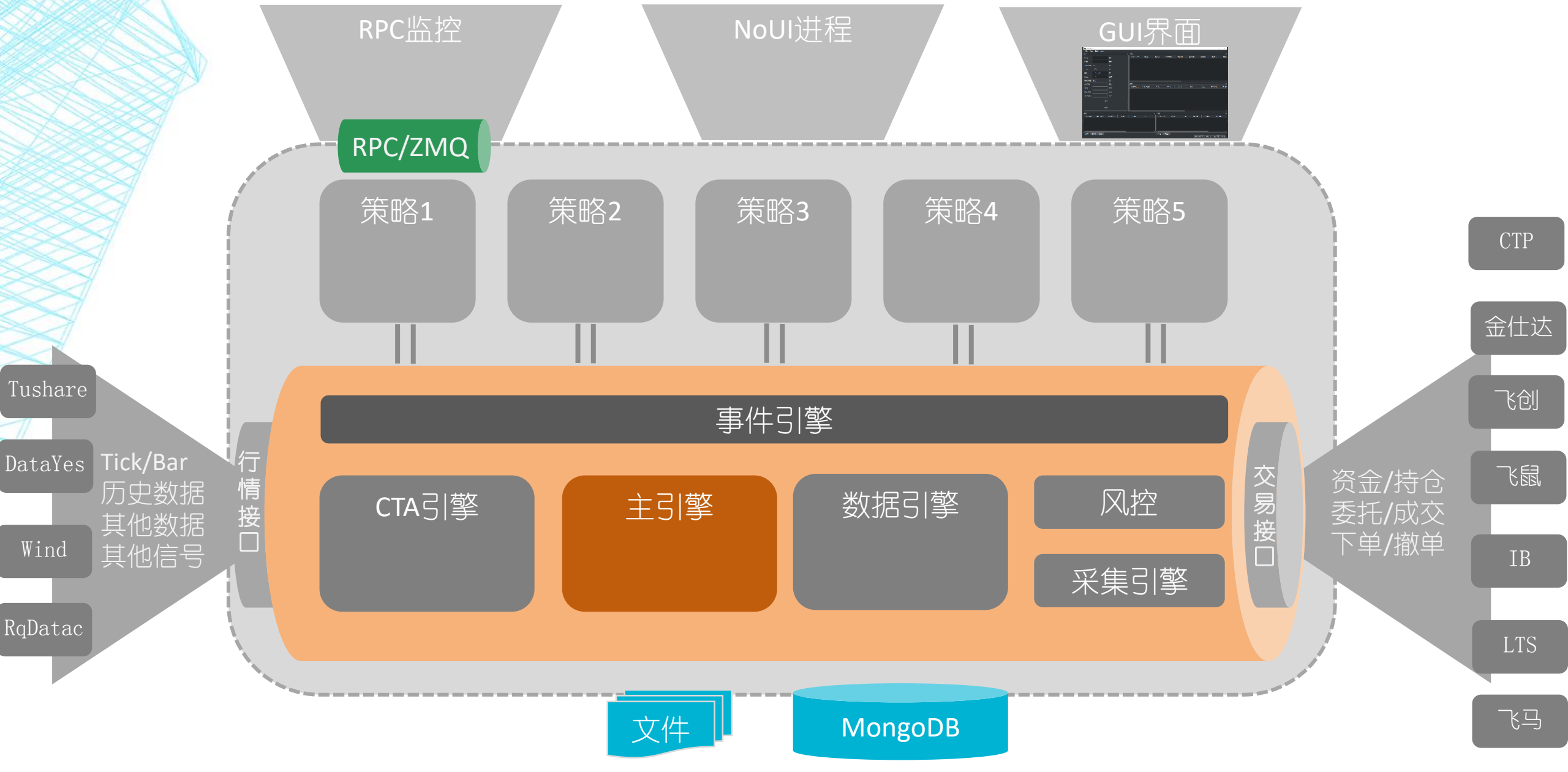
- VNPY Tick2Trade 22.6ms

5.1 VNPY项目组成

- 标准化的交易接口Python封装
  - 将多个金融市场交易与数据接口，进行标准化定义与封装
  - 基本覆盖了国内外常规交易品种（证券、期货、期权、外汇、CFD）
- 基于事件引擎的量化交易平台
  - 参考国外流行的事件驱动交易系统，自建Python高性能事件驱动引擎
  - 整合交易接口
  - 提供完整的CTA策略交易模块和界面
  - 风控体系

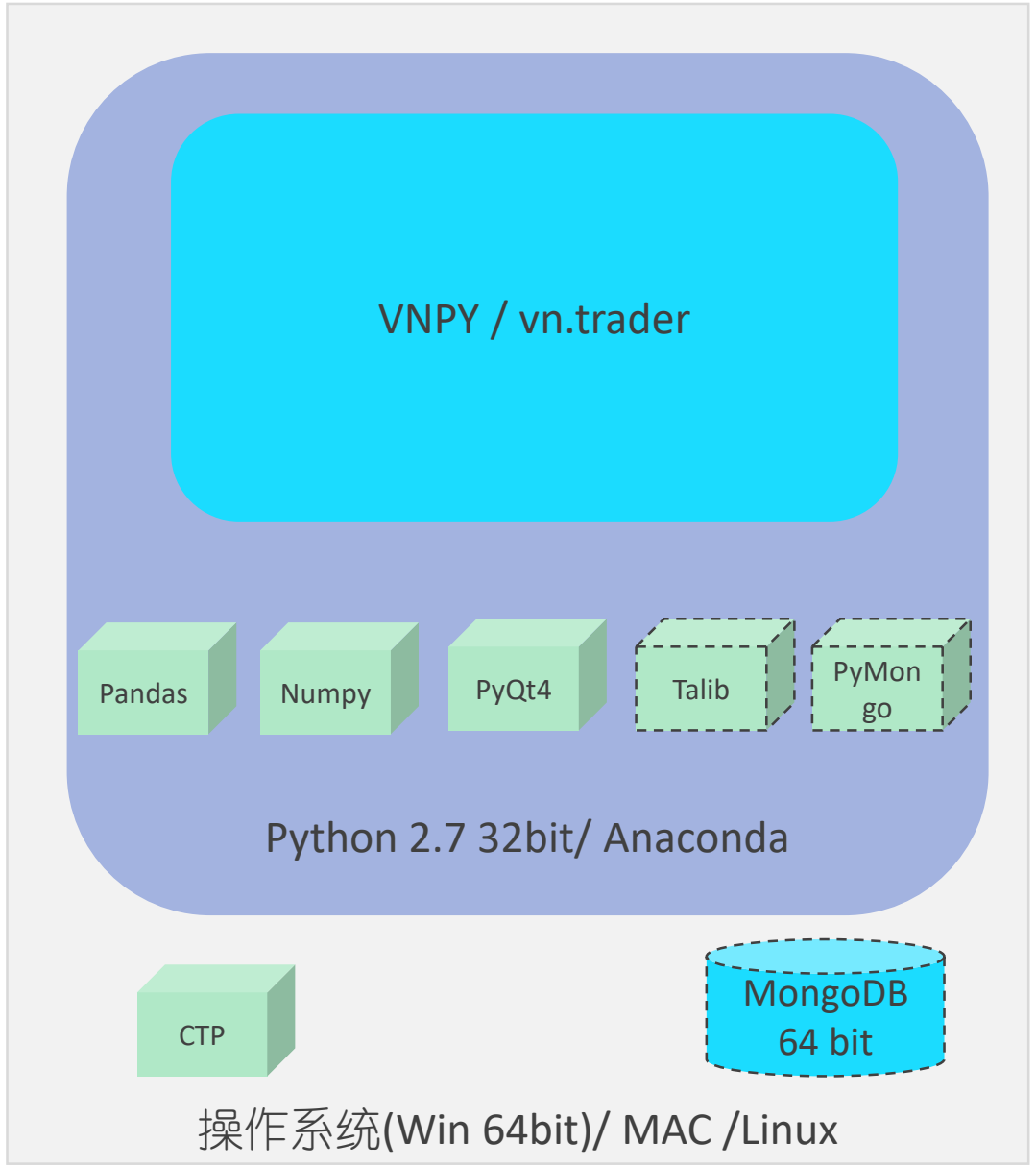
- 活跃的社区
  - http://www.vnpie.com/
  - Q群： 262656087

5.2 VNPY FRAMEWORK



5.3 开发与交易环境

- Python是一个解释类语言，只要有“环境”，就能运行。
- VNPY 标配依托Python 2.7x 32bit环境运行。
- VNPY 依赖PyQt4、Pandas、Numpy等必要的库。
- 交易接口根据厂家提供。
- VNPY使用PyMongo访问MongoDB
- Talib为可选，建议安装
- 32bit 组件能运行在64bit操作系统中，反之则不行



5.3.1 安装基础运行环境(1/3) PYTHON

- 准备一台Windows 7/10 64位系统的电脑
- 安装Anaconda：下载Python 2.7 32位版本，注意必须是32位

http://www.continuum.io/downloads

- 安装Visual C++ Redistributable Packages for VS2013，中英文随意，为了未来使用方便把x86和x64的都给装了

https://www.microsoft.com/en-gb/download/details.aspx?id=40784

- 安装QDarkStyleSheet（非常漂亮的PyQt黑色主题）：在cmd中运行pip install qdarkstyle
- 为conda增加国内镜像,在cmd中运行以下命令

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/

conda config --set show_channel_urls yes

- 降级PyQt为4.x版本:在cmd中运行 conda install pyqt=4
- 安装Talib库，在cmd中运行

conda install -c https://conda.anaconda.org/quantopian ta-lib

5.3.2 安装基础运行环境(2/3) MONGODB

- 数据库
  - 安装MongoDB：下载Windows 64-bit 2008 R2+版本

    http://www.mongodb.org/downloads
    - 创建 c:\MongoDB 和两个子目录c:\MongoDB\Data c:\MongoDB\Log
    - 用CMD（管理员身份），运行以下指令，将MongoDB注册为Windows服务并启动：

    “C:\program files\mongodb\server\3.4\bin\mongod” -dbpath “c:\MongoDB\Data” -logpath “C:\MongoDB\Log\MongoDB.log” -install -serviceName “MongoDB”
- 客户端
  - 安装pymongo：在cmd中运行pip install pymongo
  - [可选]下载可视化客户端工具 http://robomongo.org/

5.3.3 安装基础运行环境(3/3) 运行配置

- 在vn.py项目的Github主页选择Download ZIP下载项目代码，并解压到C:\vnpy

http://github.com/vnpy/vnpy

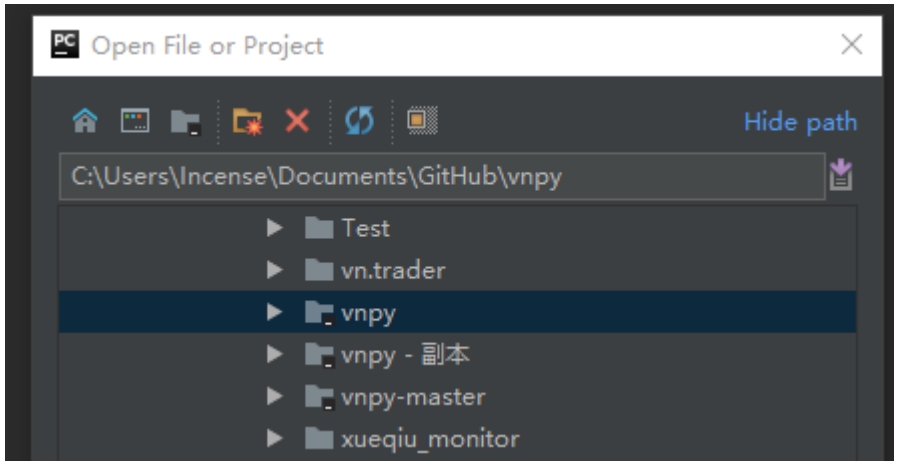
- 打开C:\vnpy\vn.trader文件夹，准备进行下一步的交易接口配置
- 在SimNow注册CTP仿真账号，记下你的**账号、密码、经纪商编号**，然后网站“常见问题”查询你的**交易和行情服务器地址**

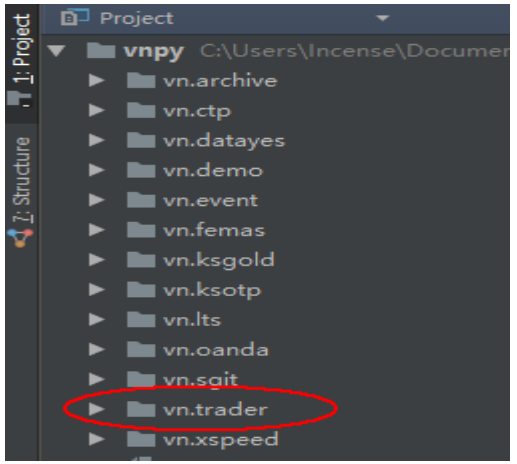
http://www.simnow.com.cn/

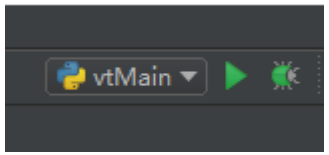
- 把C:\vnpy\vn.trader\ctpGateway\CTP_connect.json中的账号、密码、服务器等修改为上一步注册完成后你的信息（注意使用专门的编程编辑器，如Sublime Text等，防止json编码出错）
- 双击运行C:\vnpy\vn.trader\vtMain.py，开始交易！

5.3.4 安装开发工具 PYCHARM

- 下载Pycharm 社区版（免费）
  https://www.jetbrains.com/pycharm/download/
- 运行Pycharm， File->Open， 选择 c:/vnpy 目录即可
- 检查Python环境 File->Setting->Project VNPY->Project Interpreter,选项为Anaconda,Python为2.7
- 展开从项目文件, c:\vnpy\vn.trader\vtmain.py， 为主运行文件。选择”Run”进行运行模式， 或选择” Debug”， 开始调试模式







5.3.4 其他安装问题

- 出现缺少IB模块等提示
  - 因为缺省vnpy加载了很多gateway，这些gateway需要驱动或编译好的模块，若不使用到，可以在vnpy代码中进行注释取消。
  - 在vn.trader/vtEngine.py的initGateway()中，注解不需要使用的gateway
  - 在vn.trader/vtMainWindow.py的initMenu()中，注释不需要使用的连接。

```
"""
try:
    from ltsGateway.ltsGateway import LtsGateway
    self.addGateway(LtsGateway, 'LTS')
    self.gatewayDict['LTS'].setQryEnabled(True)
except Exception, e:
    print e
"""
```

```
#connectLtsAction = QtGui.QAction(u'连接LTS', self)
#connectLtsAction.triggered.connect(self.connectLts)

#connectKsotpAction = QtGui.QAction(u'连接金仕达期权', self)
#connectKsotpAction.triggered.connect(self.connectKsotp)

#connectFemasAction = QtGui.QAction(u'连接飞马', self)
#connectFemasAction.triggered.connect(self.connectFemas)
```

5.3.5 交易环境

- 主机托管/云主机/个人PC/公司服务器
- Windows 2012 64bit
  - 执行1~3安装配置步骤
  - [可选]Pycharm 远程调试模块（除非你很牛x）

- Linux 服务器（非可视化界面）
  - 安装Python环境和相关基础库
  - 安装交易接口API
  - 部署代码
  - 运行非界面主程序 vn.trader/NoUiMain.py [讲师提供源代码，见下链接]

  - https://github.com/msincenselee/vnpy

6.1 EVENT ENGINE事件引擎

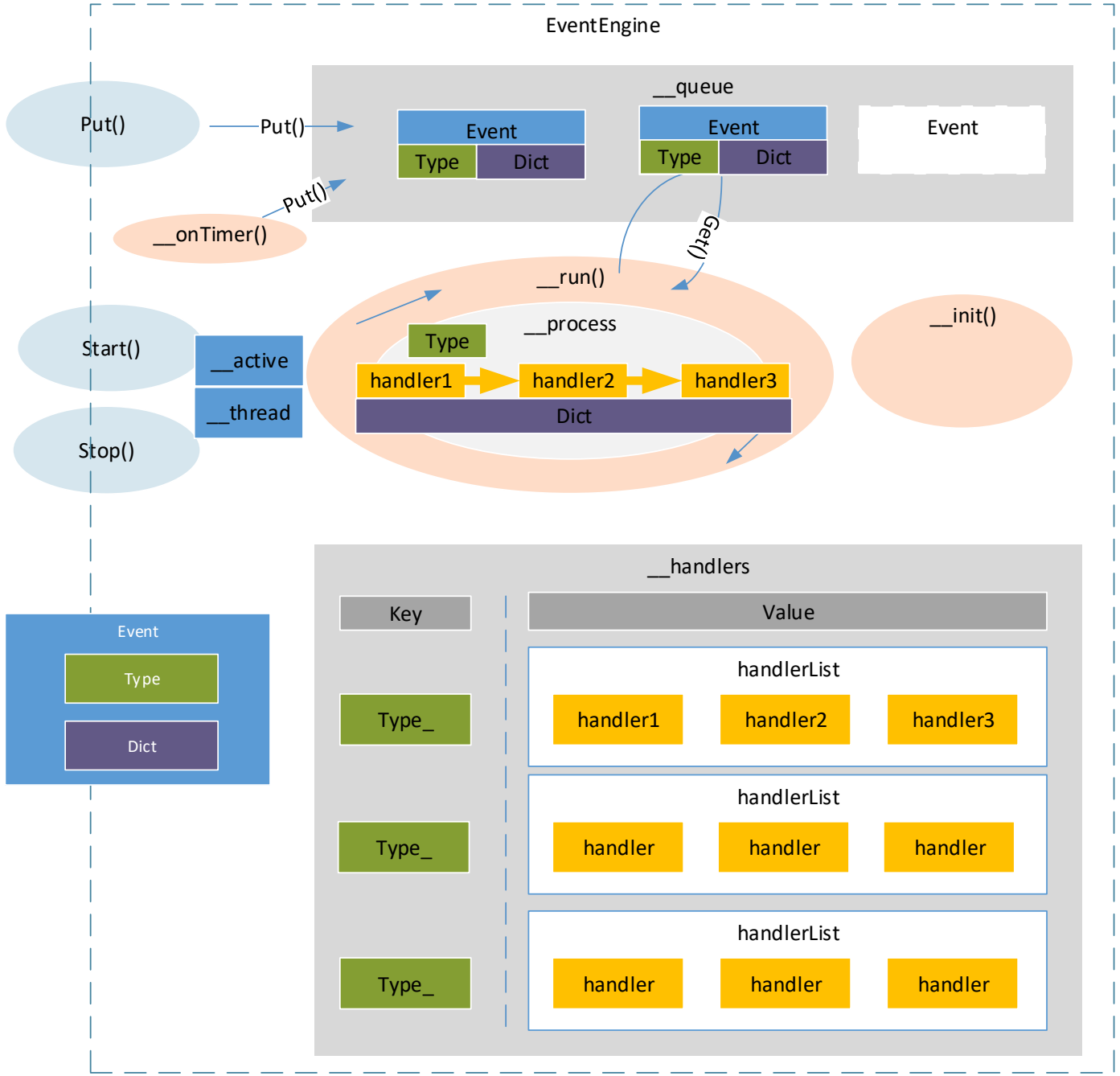
- 通俗的理解：
  - 想象一下，一个作战部队，每个小分队都有“通讯员”。
  - 当有作战指令发布，或者战情时，“通讯员”把指令/战情，通知所有相关的其他“通讯员”。小分队收到后分别执行指令。
- 什么是事件（Event）
  - 有类型和内容的消息。（消息：例如部队调动，军情等）
- 什么是订阅（Register/Subscript）
  - 关心某类型的消息，指定其监听函数。（例如指定谁是通讯员）
- 什么是事件通知（OnEvent）
  - 有消息到达，逐一对登记的监听函数进行调用，发送消息。

6.1 .2 VN.EVENT

1、Queue，先进先出的消息队列
- Put()：把需要推送的事件，放进队列。
- Get()：提取最前一个事件。

2、Handlers，事件注册表
- Type：事件的类型
- Dict：内容体，数据字典
- Handler List：监听函数清单
- Register()：添加监听函数

3、Thread，事件推送线程
- 逐一提取事件，根据类型找出Handler list
- 对List的监听函数，逐一调用推送



6.1.3 事件类型

```
# 系统相关
    EVENT_TIMER = 'eTimer'          # 计时器事件，每隔1秒发送一次
    EVENT_LOG = 'eLog'              # 日志事件，全局通用

# Gateway相关
    EVENT_TICK = 'eTick.'           # TICK行情事件，可后接具体的vtSymbol
    EVENT_TRADE = 'eTrade.'         # 成交回报事件
    EVENT_ORDER = 'eOrder.'         # 报单回报事件
    EVENT_POSITION = 'ePosition.'   # 持仓回报事件
    EVENT_ACCOUNT = 'eAccount.'     # 账户回报事件
    EVENT_CONTRACT = 'eContract.'   # 合约基础信息回报事件
    EVENT_ERROR = 'eError.'         # 错误回报事件

# CTA模块相关
    EVENT_CTA_LOG = 'eCtaLog'               # CTA相关的日志事件
    EVENT_CTA_STRATEGY = 'eCtaStrategy.'    # CTA策略状态变化事件

# 行情记录模块相关
    EVENT_DATARECORDER_LOG = 'eDataRecorderLog' # 行情记录日志更新事件
```

6.1.4 扩展事件类型

扩展

- # 在vn.trader/eventType.py中添加扩展事件
- EVENT_ACCOUNT_LOSS = 'eAccountLoss' # 账户亏损事件
- EVENT_FULL_MONITOR = 'eFullMonitor' # 全周期监控
- EVENT_ON_BAR = 'eOnBar' # OnBar事件

发送

1、填充数据字典
data[‘key1’] = value1
data[‘key2’] = value2
…
2、创建对应类型的事件，绑定数据
event = Event(type = EVENT_ON_BAR)
event.dict_[‘data’] = data

3、调用事件引擎，发送事件
eventEngine.put(event)

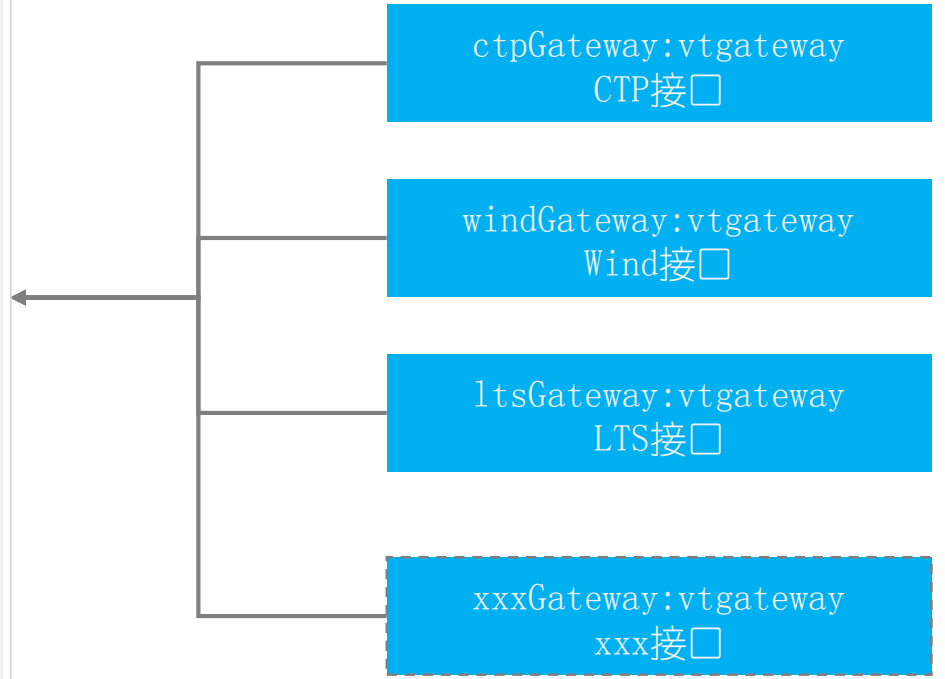
使用

1、注册事件，绑定事件的监听函数
eventEngine.register(EVENT_ON_BAR,self.display)

2、定义监听函数
def display(event)
    data = event.dict_[‘data’]
    print data

6.2.1 行情/交易接口

vn.trader/vtGateway.py



vn.trader/xxxGateway/

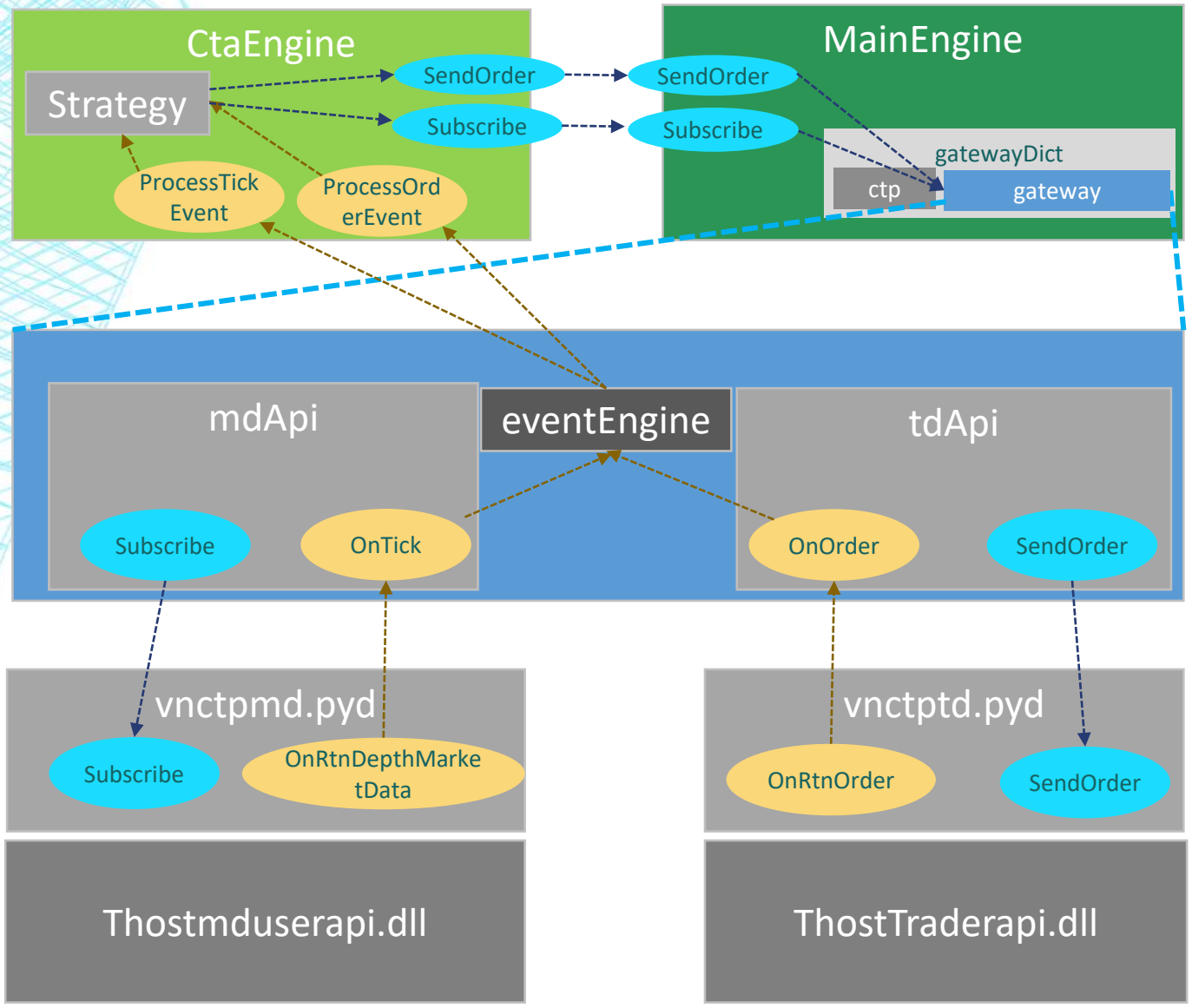
CtpMdApi

CtpTdApi

LtsQryApi

LtsMdApi

6.2.2 CTP 行情/交易接口



CtpGateway：vtGateway
CTP接口

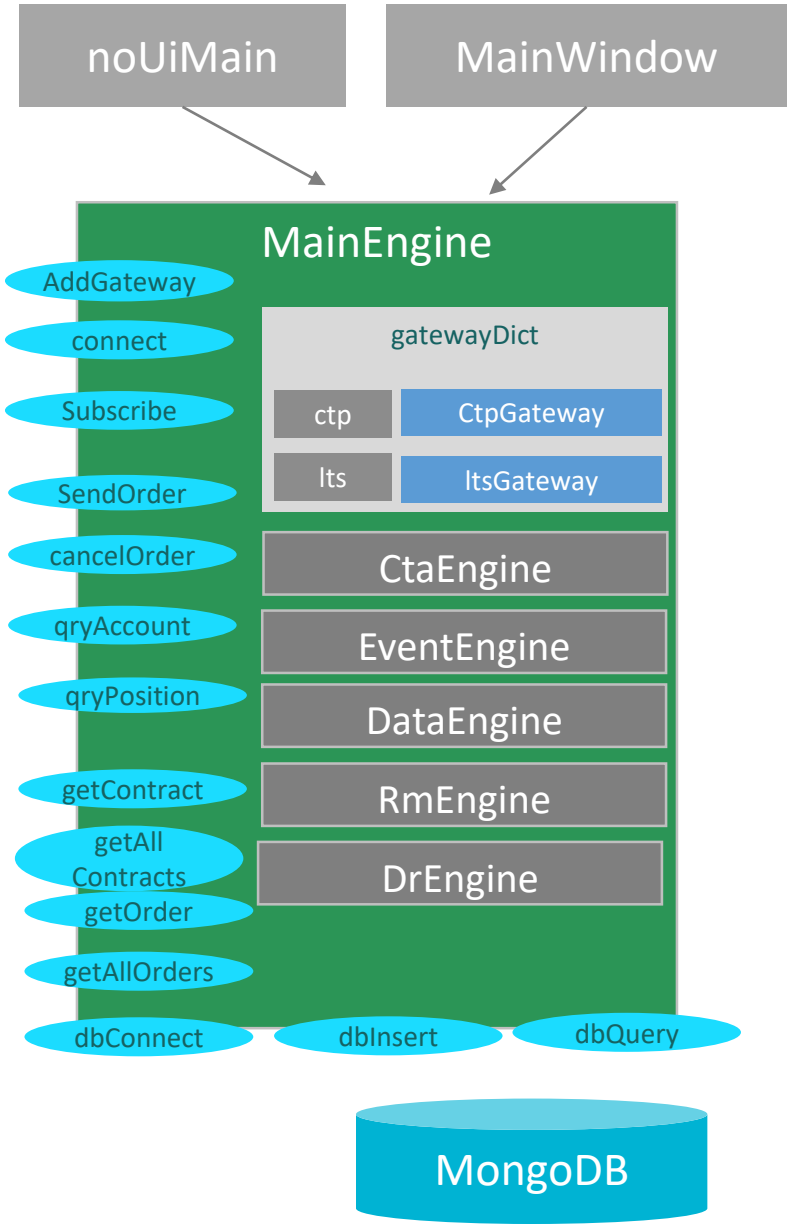
eventEngine	#事件引擎
mdApi	#行情Api
tdApi	#交易Api
mdConnected	#行情Api连接状态
tdConnected	#交易Api连接状态
onTick()	#市场推送行情
onTrade()	#成交信息推送
onOrder()	#订单变化推送
onPosition()	#持仓信息推送
onAccount()	#账户信息推送
onError()	#错误信息推送
onLog()	#日志推送
onContract()	#合约基础信息推送
connect()	#连接
subscribe()	#订阅行情
sendOrder()	#发单
cancelOrder()	#撤单
qryAccount()	#查询账户资金
qryPosition()	#查询持仓
close()	#关闭
initQuery()	#初始化连续查询
query()	#注册在事件引擎的查询函数
startQuery()	#启动连续查询
setQryEnabled()	#设置启动循环查询

6.2.3 对CTP接口的改进

- 问题：
  - 期货公司的CTP服务器，不是常开放的，仅在开盘前开放，开盘后（清算时）关停。
  - 会出现“前端不活跃”异常。
  - Ctpapi原生接口连接服务器成功后，若断开，会不断尝试重连，有时会导致vn.trader整个Crash。
- 改进
  - 修改ctpGateway中的TdApi、MdApi实例化方法和断开机制。
  - 修改ctpGateway中的各类方法调用，增加TdApi、MdApi实例化判断。
  - 修改ctpGateway中的订阅，重连后，自动重新订阅行情。

6.3.1 主引擎MAINENGINE

- 负责实例化
  - 实例化策略引擎（CtaEngine）
  - 实例化事件引擎（EventEngine）
  - 实例化数据引擎（DataEngine）
  - 实例化风控引擎（RmEngine）
  - 实例化采集引擎（DrEngine）
  - 连接后端数据库（MongoDb）
- 提供gateway的通用方法封装
  - 添加gateway,连接/断开
  - 订阅/取消,委托/撤单
  - 查询账号/持仓
- 提供dataEngine的方法封装
  - 查询合约/所有合约
  - 查询委托/活跃委托
- 提供对数据库的方法封装
- 清空数据/保存数据

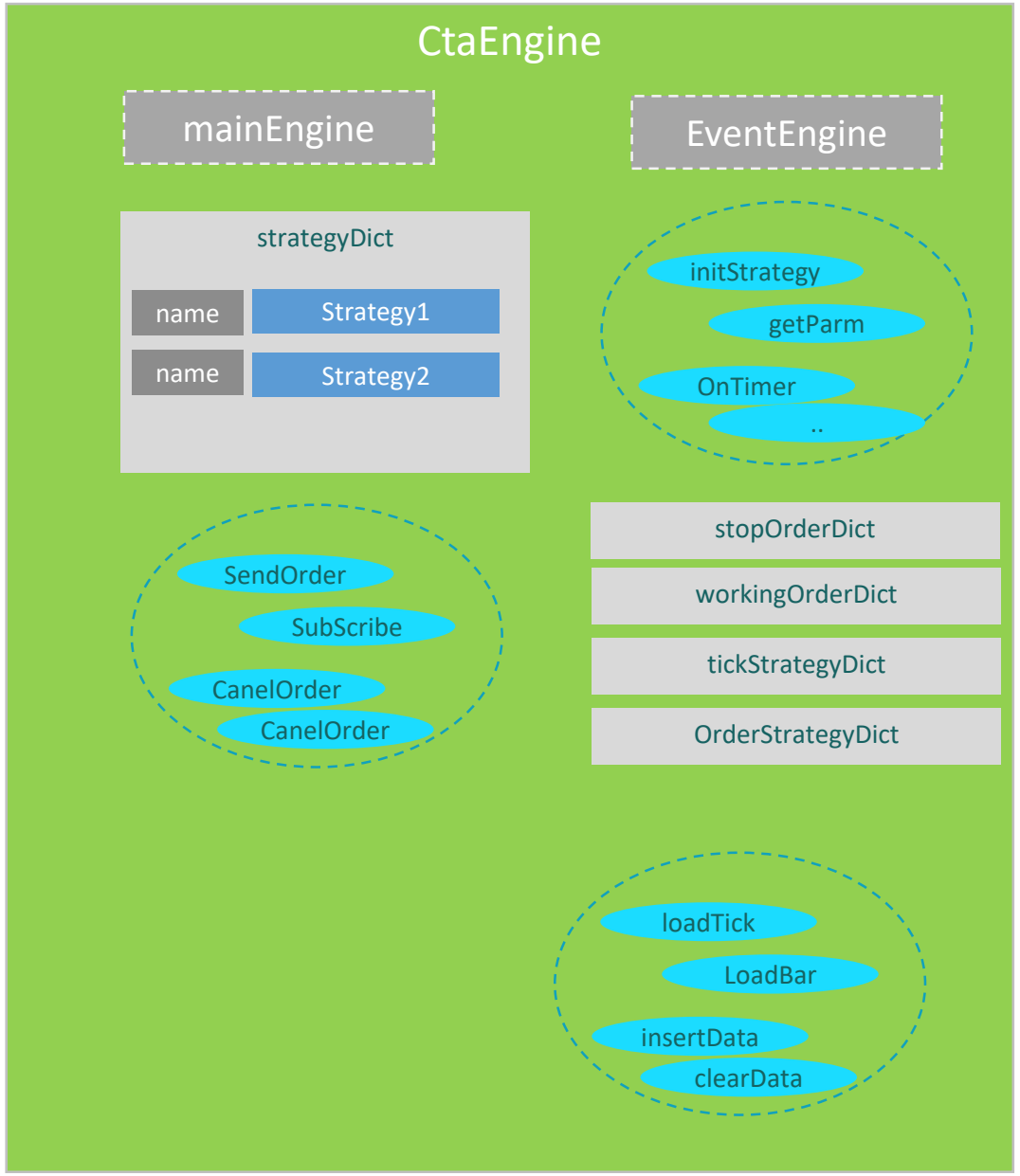


6.3.1 使用主引擎无人值守

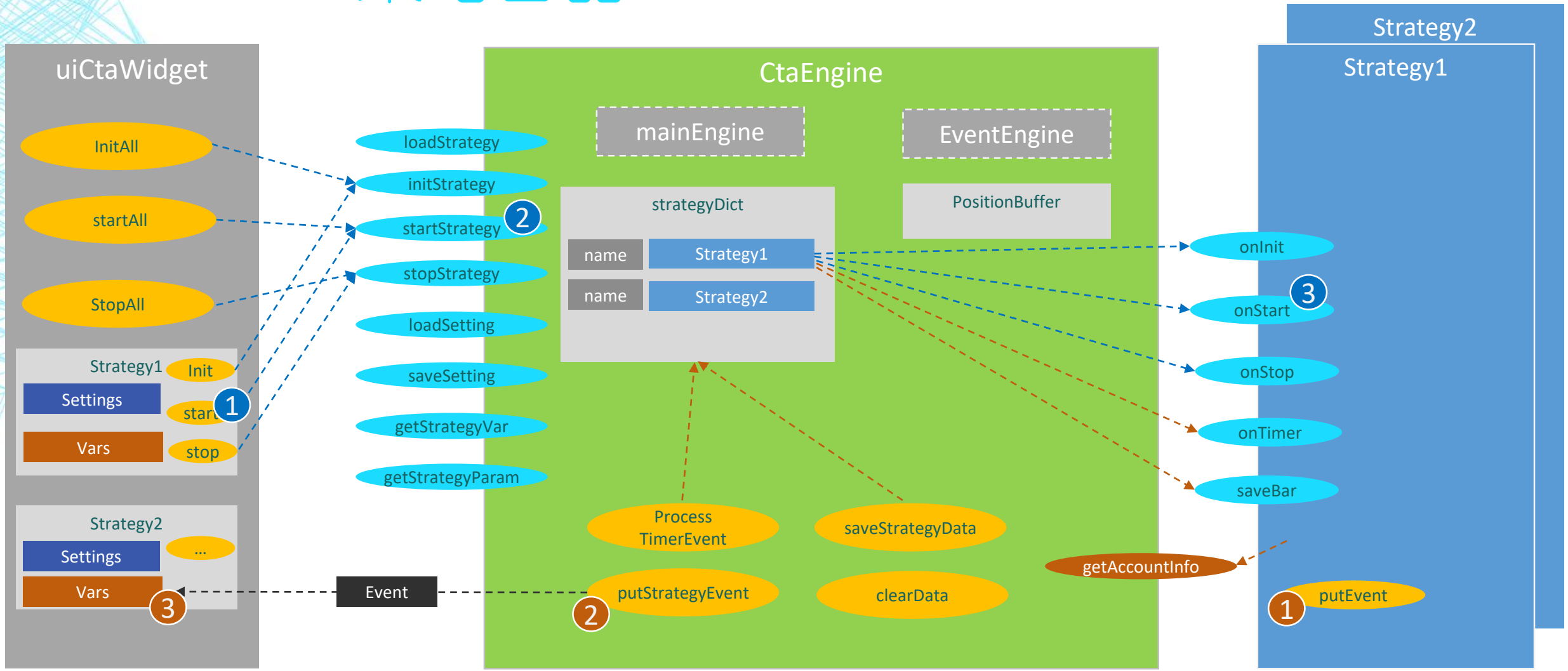
- 自动连接，断开自动重连
- 账单确认
- 策略加载、初始化与启动
- 程序异常，自动重启
- 告警信息发送
- Log日志

6.4.1 CTA引擎

- 策略容器
  - 加载/初始化/启动/停止
  - 策略参数与运行监控
  - 持久化策略数据
  - 定时触发器
  - 风控
- gateway接口封装
  - 委托/撤单/全撤
  - 本地停止单
  - 行情订阅
- 数据支持
  - 合约数据
  - tick/Bar数据



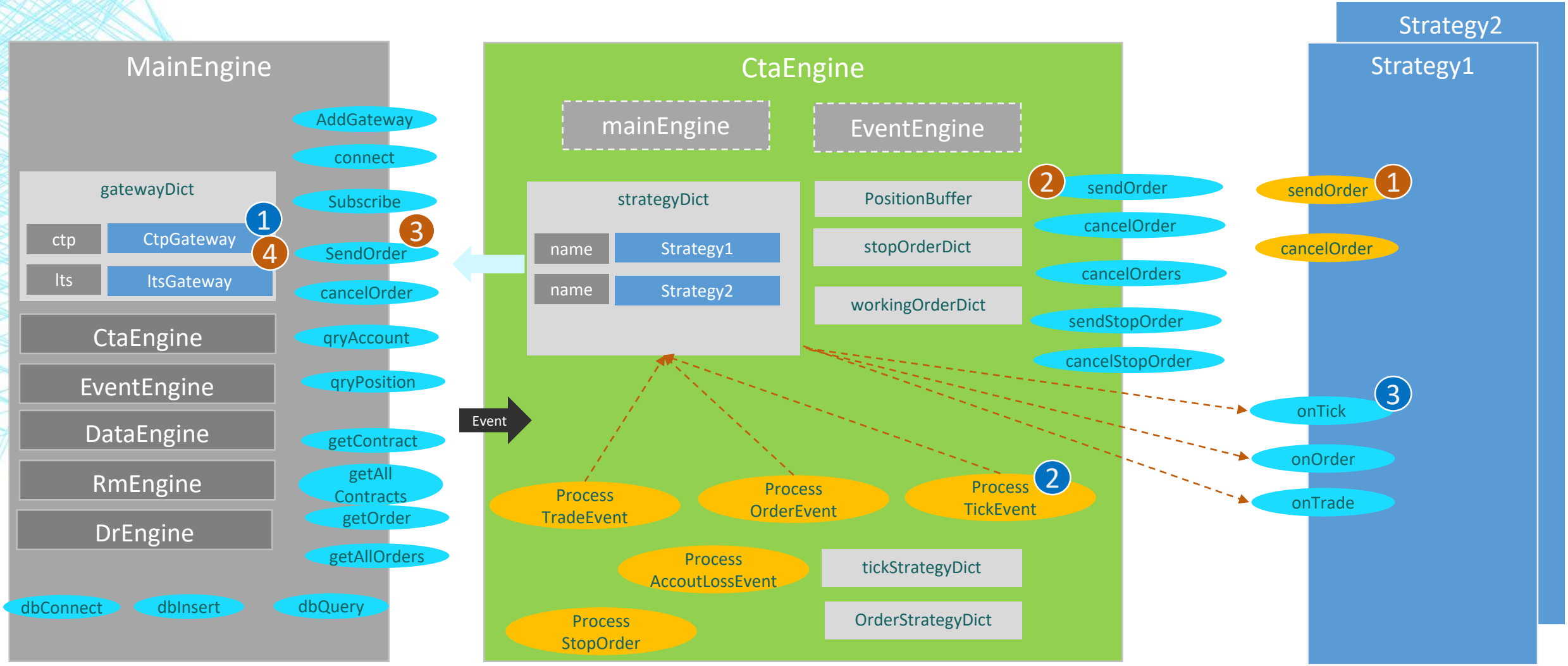
6.4.2 策略容器



启动过程 1 界面启动 2 CtaEngine.StartStrategy() 3 Strategy1.onStart()

监控过程 1 变量更新putEvent 2 CtaEngine.putStrategyEvent() 3 uiCtaWidgate. updateMonitor()

6.4.3 CTA策略-行情/交易

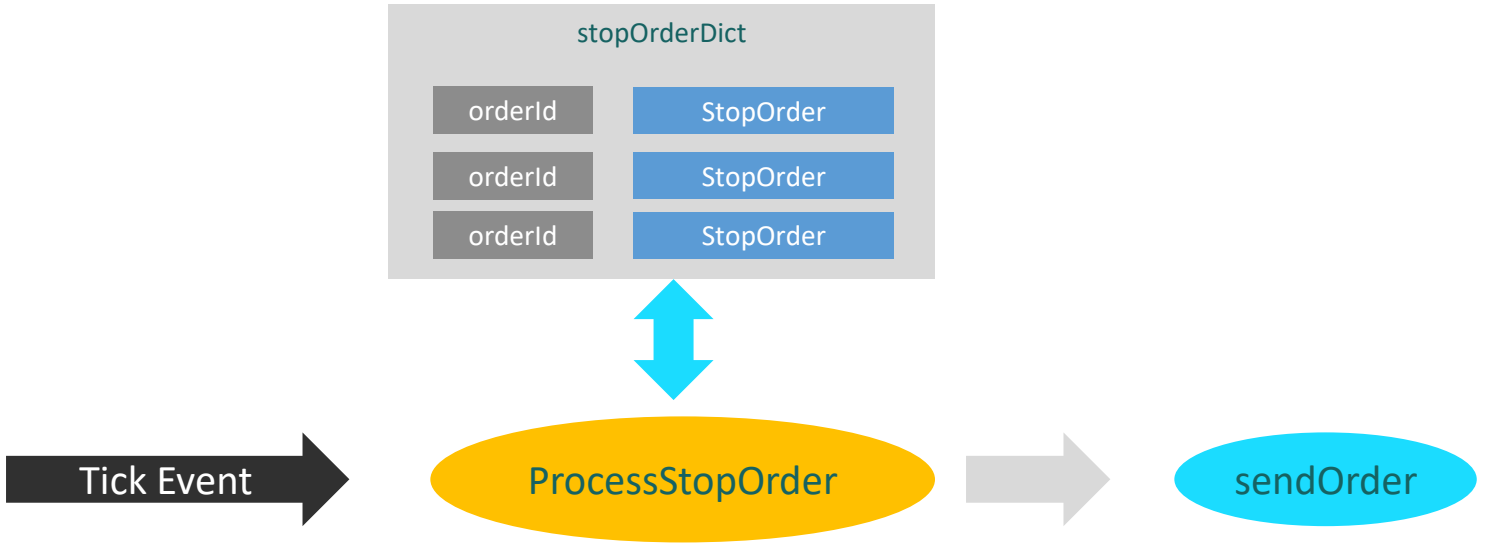


Tick推送 1 收到tick,发送Event 2 CtaEngine.ProcessEvent()处理分发 3 Strategy1.onOnTick()

委托过程 1 发送委托 2 CtaEngine.SendOrder() 3 mainEngine.sendOrder () 4 gateway.sendOrder ()

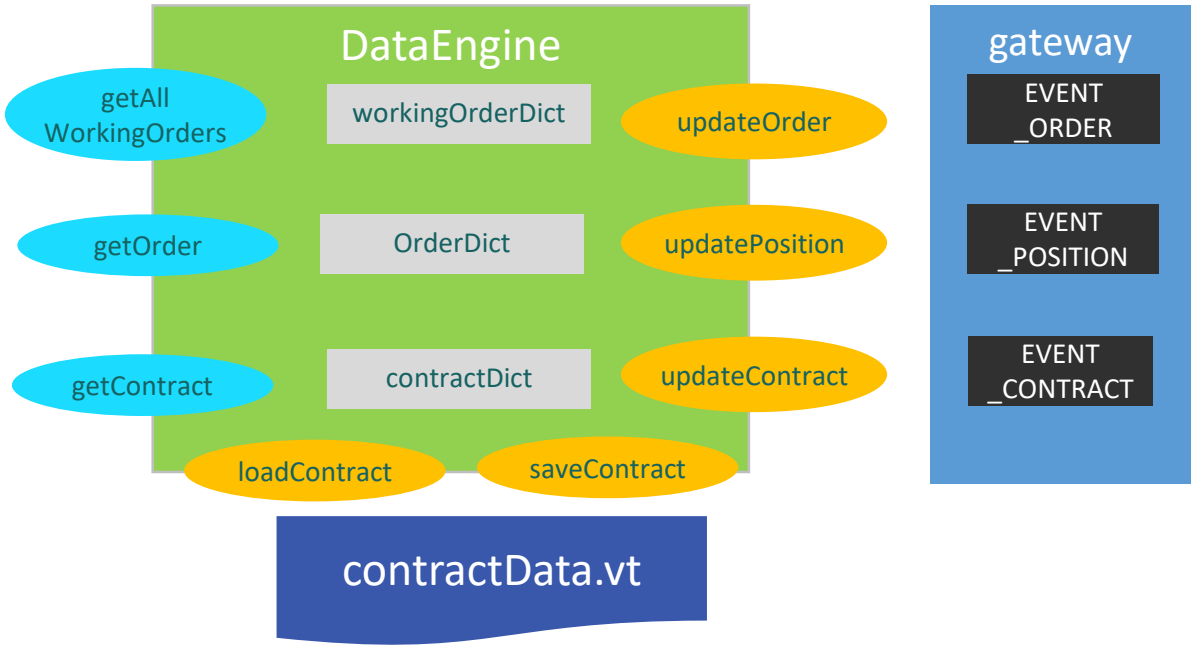
6.4.4 本地停止单

- 本地停止单，为策略提供了更灵活的条件单开仓方式。



6.5 DATAENGINE数据引擎

- 为CTA引擎/主引擎，提供合约/委托/持仓的查询服务
  - 更新合约数据updateContract
  - 查询合约对象getContract
  - 查询所有合约对象getAllContracts
  - 保存所有合约saveContracts
  - 更新委托updateOrder
  - 查询委托getOrder
  - 查询所有委托getAllWorkingOrders
  - 更新持仓updatePosition

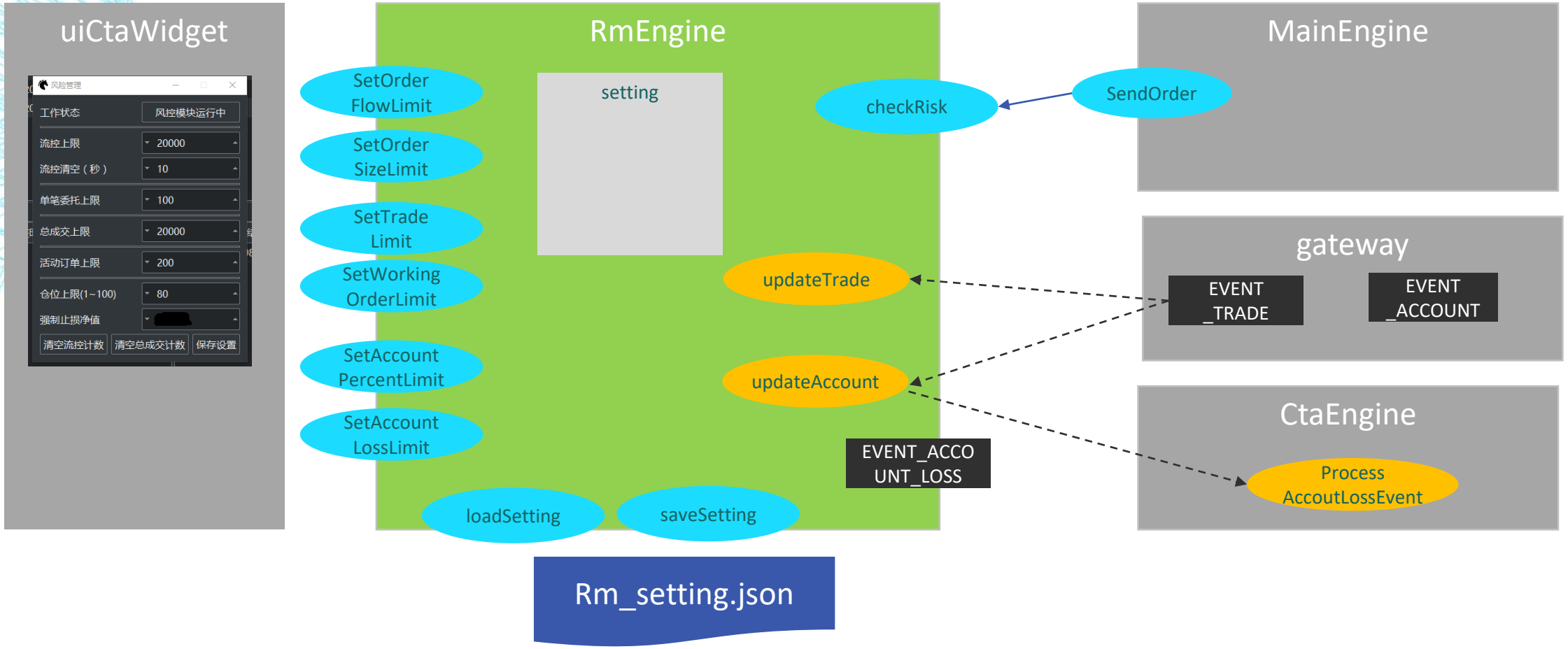


6.6.1 RMENGINE风控引擎

- Vnpy侧重交易，其风控引擎专注在执行的风控，包括
  - 每秒委托流量上限，限制每秒的订单发送数量
  - 单笔委托上线，限制每笔委托内的合约下单数量
  - 总成交上线，限制每日成交合约总数量
  - 活动合约上限，限制委托指令发送

- 我对vnpy的风控扩展，主要为资金的风控
  - 总仓位比例，超过时，限制所有策略/界面指令的开仓
  - 总资金清盘线，当账号当前权益低于指定金额时，强制全平仓

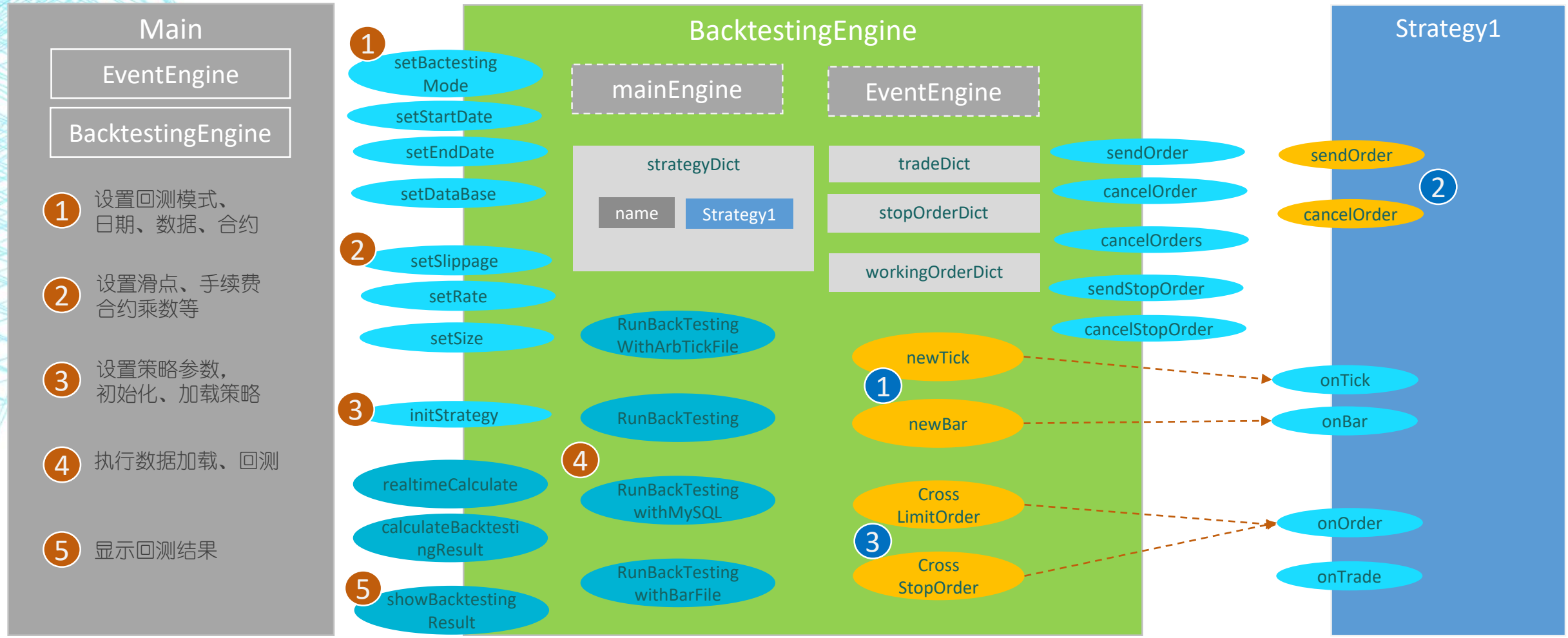
6.6.2RMENGINE风控引擎



6.7.1 BACKTESTINGENGINE回测引擎

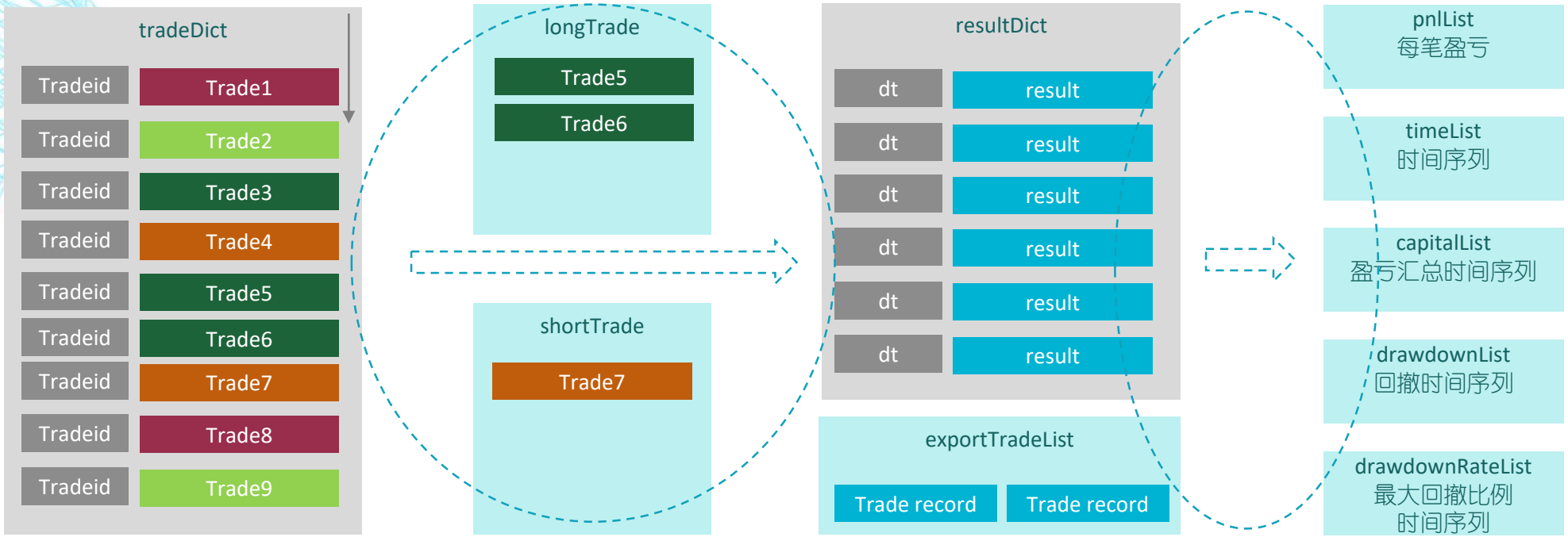
- 回测引擎，模拟了CTA引擎，进行：
  - 加载、初始化、运行策略
  - 加载（历史）数据，推送至策略
  - 接收从策略发出的交易指令，进行本地执行、撮合、反馈
  - 数据加载结束后，统计交易记录，计算策略的综合表现。
- 回测引擎支持的数据，包括
  - Tick数据
  - K线数据
- 数据源，由你决定：
  - 来自原始tick文件，或csv的K线数据
  - 来自数据库（MySQL、MongoDB）
  - 来自数据供应商在线获取（Tushare、Wind、DataYes、Rqdatac）

6.7.2 回测过程

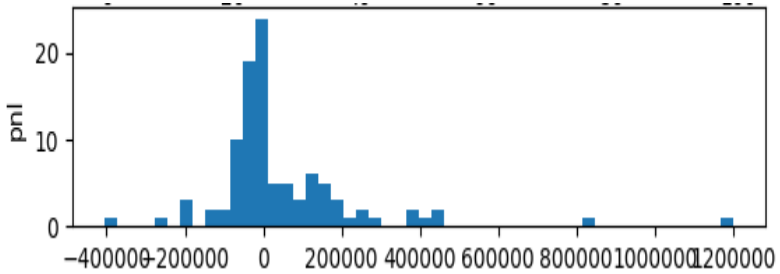


模拟过程 1 推送tick/bar 2 策略逻辑运算，产生交易信号 3 下一Tick/bar执行信号，更新交易记录

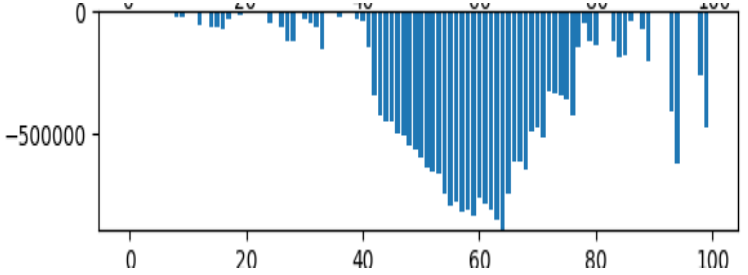
6.7.3 计算盈亏、分析输出



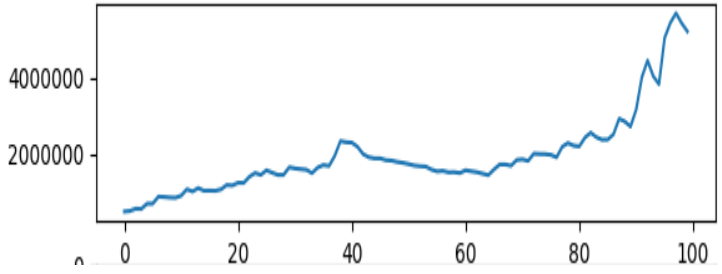
盈亏的正态分布



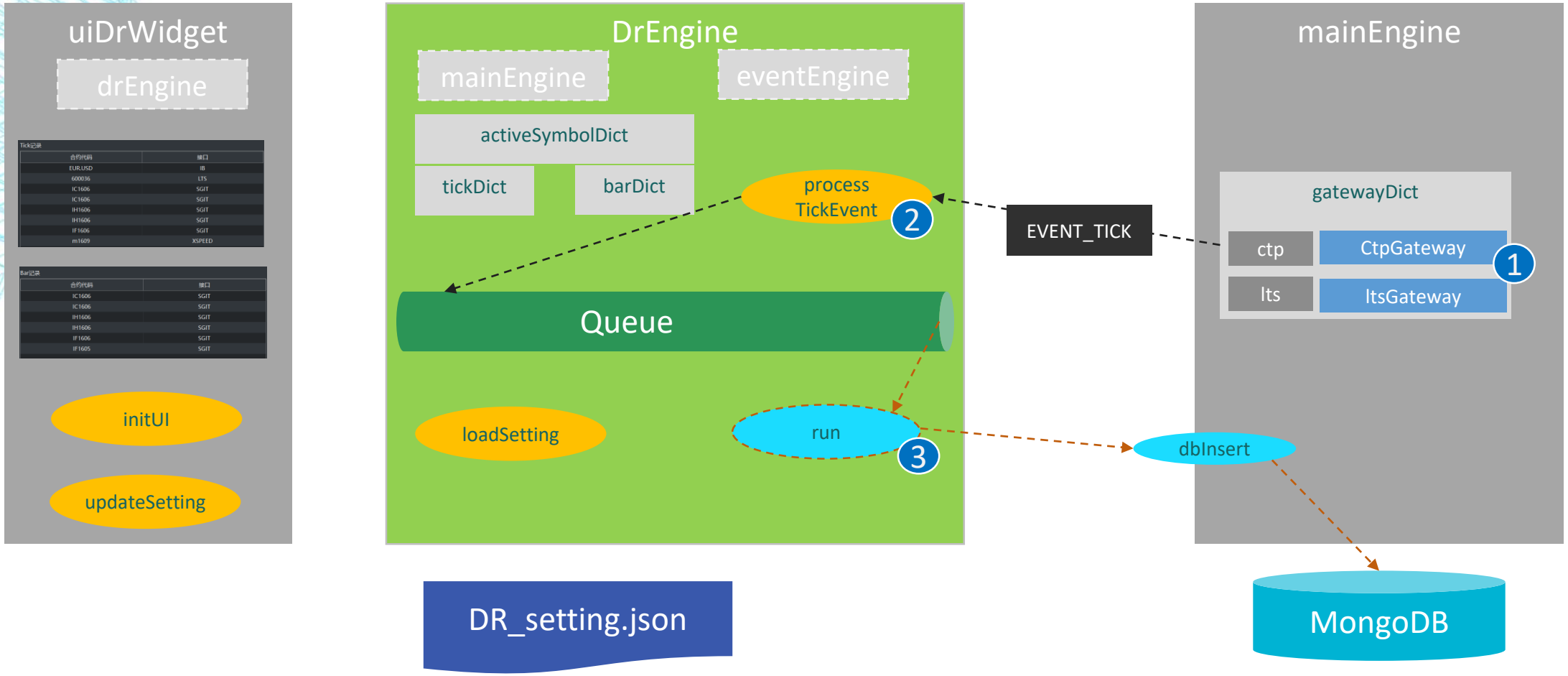
每笔回撤



资金曲线



6.8 DATARECORDER数据收集



记录过程 1 推送tick Event 2 生成tick，合成Bar，分别放置进缓存Queue 3 逐一提取，入库

课后作业

- 1.安装环境
- 2.使用PyCharm运行调试vnpy，启动demo策略，跟踪Tick的事件

THANKS

微信：28888502