使用多进程实现多账户no_ui下并行执行，并分配各自独立的工作环境和策略

2021年12月11日 00:30 (由 张国平 最后编辑：2021年12月11日 00:45)
#1

看到群里有人问vnpy下多账户同时运行，并且给每个账户分配不同工作内容，就是执行不同策略。

考虑了一下，实现方式很多，在vnTrader 实例代码中 no_ui 启动vnTrader就是用使用多进程模块multiprocessing，在子进程中运行。因为python GIL全局锁的存在，每个子进程都有独立的实例化环境，其实就是一个独立运行vnTrader。那么多开几个子进程，分配给对应的账户参数，就可以实现 vnpy下多账户同时运行。

这里我用json文件存储账户配置信息，每个账户配置信息，

其中包括

- 账户名，后面用账户名命名子进程，这样可以在log文件中区加入子进程信息，区分是那个账户的日志信息，方便跟踪。
- 接口名，默认是CTP，也可以支持其他接口。
- 工作路径，就是放.vnTrader文件夹和策略的路径；给每个账户分配不同配置信息和策略参数等。这里要注意，必须在路径下创建 .vnTrader文件夹，否则 vnpy的 utility 中_get_trader_dir方法就会使用home路径作为工作路径。
- 登录信息，就是登录密码一类东西。

具体代码这里有几个涉及技术点，先说说，具体的在代码看注释就可以了。

- 第一个，同时运行account 不应该超过cpu核心数，因为vntrader 运行可以看成是一个持续死循环，在main_engine.close之前不会退出cpu占用； 如果超过核心数，等待的运行事务将一直等待，所以不可以超过。
- 第二个，其实一开始我是用multiprocessing.Pool，再用pool.map_async(run_child, account_detail_list) 来实现批量创建进程，但是这样没法给每个进程命名，而且 map_async不带阻塞，没法跟踪每个子进程返回。最后还是使用队列一个一个创建process进程方法，同时用账户信息给每个进程命名，方便看console日志

multiprocessing.Process(target=run_child, name=account_detail["account_name"], args=(account_detail,))。

最后这里不用pool.map，或者jion()；这样阻塞主进程的操作，刚刚说的因为vnTrader子进程是一个持续死循环，一旦阻塞，下次主进程只用等到完全停止完毕才能开始。

- 第三个，工作环境切换，这里使用os.chdir更改工作路径到配置文件中的指定工作路径；但是因为在引用vnpy 模块时候，就已经有全局数据使用静态方法去获取工作路径了，所以代码中把vnpy模块的import放在 进程方法中，更改工作路径之后。
- 第四个，在启动进程方法中，我又定义了两个嵌套方法，用于创建threading.Timer子线程来定时启动监控策略初始化是否完成，和交易时间是否结束。原来代码都是用time.sleep 结合 while 来停止当前线程的，但是之前界面化运行vnTrader时候，sleep会停止pyQt前端界面，阻塞主代码执行，当然策略初始化和事件传输引擎都是独立线程运行，不会影响。 不过保证交易安全，改成Timer子线程定时运行监控。

这里简单介绍下python线程，虽然由于GIL锁，一个python环境只能用到一个核，但是在IO密集情况，比如网络爬虫或者请求等待时候，使用多线程可以提高效率，让可以把CPU资源给其他子线程。不过在等待的时候，直接sleep主线程还是不太稳妥；其可以利用Timer另加线程定时查看；或者传入callback方法为参数，让子线程任务完成后调用callback返回，还有python新特性Future协程等，具体待研究。

运行效果如下：

两个账户TEST_1_ACCOUNT；TEST_2_ACCOUNT；各自独立工作路径；进程名也输出界面，而log文件也按照账户分开输出。

```
启动CTA策略守护父进程
启动子进程
子进程启动成功
2021-12-10 16:53:51,896  INFO: TEST_1_ACCOUNT  主引擎创建成功
2021-12-10 16:53:51,896  INFO: TEST_1_ACCOUNT 工作路径: (WindowsPath('C:/TEST_ACCOUNT_1_WORKSPACE'), WindowsPath('C:/TEST_ACCOUNT_1_WORKSPACE/.vntrader'))
2021-12-10 16:53:51,897  INFO: TEST_1_ACCOUNT 注册日志事件监听
2021-12-10 16:53:51,906  INFO: TEST_1_ACCOUNT 连接CTP接口
2021-12-10 16:53:51,948  INFO: TEST_2_ACCOUNT  主引擎创建成功2021-12-10 16:53:51,948  INFO: TEST_1_ACCOUNT 交易服务器连接成功

2021-12-10 16:53:51,949  INFO: TEST_2_ACCOUNT 工作路径: (WindowsPath('C:/TEST_ACCOUNT_2_WORKSPACE'), WindowsPath('C:/TEST_ACCOUNT_2_WORKSPACE/.vntrader'))
2021-12-10 16:53:51,949  INFO: TEST_2_ACCOUNT 注册日志事件监听
2021-12-10 16:53:51,959  INFO: TEST_2_ACCOUNT 连接CTP接口
2021-12-10 16:53:51,961  INFO: TEST_1_ACCOUNT 行情服务器连接成功
2021-12-10 16:53:52,035  INFO: TEST_1_ACCOUNT 行情服务器登录成功
2021-12-10 16:53:52,058  INFO: TEST_2_ACCOUNT 行情服务器连接成功
2021-12-10 16:53:52,073  INFO: TEST_2_ACCOUNT 交易服务器连接成功
2021-12-10 16:53:52,145  INFO: TEST_2_ACCOUNT 行情服务器登录成功
auth success
2021-12-10 16:54:02,240  INFO: TEST_1_ACCOUNT jqdata数据接口初始化成功
auth success
2021-12-10 16:54:02,278  INFO: TEST_2_ACCOUNT jqdata数据接口初始化成功
2021-12-10 16:54:02,332  INFO: TEST_1_ACCOUNT CTA策略引擎初始化成功
2021-12-10 16:54:02,333  INFO: TEST_1_ACCOUNT CTA策略初始化完成
2021-12-10 16:54:02,333  INFO: TEST_1_ACCOUNT AG201开始执行初始化
2021-12-10 16:54:02,362  INFO: TEST_2_ACCOUNT CTA策略引擎初始化成功
2021-12-10 16:54:02,363  INFO: TEST_2_ACCOUNT CTA策略初始化完成
2021-12-10 16:54:02,364  INFO: TEST_2_ACCOUNT AG201开始执行初始化
2021-12-10 16:54:03,298  INFO: TEST_1_ACCOUNT [AG201]  行情订阅失败，找不到合约ag2201.SHFE
2021-12-10 16:54:03,299  INFO: TEST_1_ACCOUNT AG201初始化完成
2021-12-10 16:54:03,310  INFO: TEST_1_ACCOUNT agDne开始执行初始化
2021-12-10 16:54:03,356  INFO: TEST_2_ACCOUNT [AG201]  行情订阅失败，找不到合约ag2201.SHFE
2021-12-10 16:54:03,356  INFO: TEST_2_ACCOUNT AG201初始化完成
2021-12-10 16:54:03,356  INFO: TEST_2_ACCOUNT agDne开始执行初始化
```

---

https://billyzhangguoping.github.io/CPPLearningManual/
2021年12月11日 00:34
#2

参数文件如下，名称Mutiple_Accounts_Config.json 放在no_ui文件夹下就可以

```
[
    {
    "account_name": "TEST_2_ACCOUNT",
    "gateway": "CTP",
    "workspace": "C:/TEST_ACCOUNT_2_WORKSPACE/",
    "logon_detail": {
      "用户名": "",
      "密码": "",
      "经纪商代码": "",
      "交易服务器": "",
      "行情服务器": "",
      "产品名称": "",
      "授权编码": ""
    }
  },
  {
    "account_name": "TEST_1_ACCOUNT",
    "gateway": "CTP",
    "workspace": "C:/TEST_ACCOUNT_1_WORKSPACE/",
    "logon_detail": {
      "用户名": "",
      "密码": "",
      "经纪商代码": "",
      "交易服务器": "",
      "行情服务器": "",
      "产品名称": "",
      "授权编码": ""
    }
  }
]
```

---

https://billyzhangguoping.github.io/CPPLearningManual/
2021年12月11日 00:35
#3

run代码如下，因为改动较多，建议直接覆盖原来的run.py：

```
import multiprocessing
import json
from threading import Timer
import sys
import os
from time import sleep
from datetime import datetime, time
from logging import INFO
# Chinese futures market trading period (day/night)
DAY_START = time(8, 45)
DAY_END = time(15, 0)
NIGHT_START = time(20, 45)
NIGHT_END = time(1, 30)
def check_trading_period():
    """"""
    current_time = datetime.now().time()
    trading = False
    if (
        (current_time >= DAY_START and current_time <= DAY_END)
        or (current_time >= NIGHT_START)
```

```
            or (current_time <= NIGHT_END)
        ):
            trading = True
        return trading
    def run_child(account_detail):
        """
        Running in the child process.
        """
        account_name = account_detail["account_name"]
        ctp_setting = account_detail["logon_detail"]
        # 更改工作路径
        os.chdir(account_detail["workspace"])
        # 把对vntrader 包的引用放在工作路径更改后，不然工作路径更改无法生效,
        from vnpy.event import EventEngine
        from vnpy.trader.setting import SETTINGS
        from vnpy.trader.engine import MainEngine
        from vnpy.gateway.ctp import CtpGateway
        from vnpy.app.cta_strategy import CtaStrategyApp
        from vnpy.app.cta_strategy.base import EVENT_CTA_LOG
        SETTINGS["log.active"] = True
        SETTINGS["log.level"] = INFO
        SETTINGS["log.console"] = True
        from vnpy.trader.utility import TRADER_DIR, TEMP_DIR
        # 结束引用
        SETTINGS["log.file"] = True
        event_engine = EventEngine()
        main_engine = MainEngine(event_engine)
        main_engine.add_gateway(CtpGateway)
        cta_engine = main_engine.add_app(CtaStrategyApp)
        main_engine.write_log(f" 主引擎创建成功")
        main_engine.write_log(f"工作路径: {TRADER_DIR, TEMP_DIR}")
        log_engine = main_engine.get_engine("log")
        # 更新log 格式。
        event_engine.register(EVENT_CTA_LOG, log_engine.process_log_event)
        main_engine.write_log(f"注册日志事件监听")
        main_engine.connect(ctp_setting, account_detail["gateway"])
        main_engine.write_log(f"连接CTP接口")
        sleep(10)
        cta_engine.init_engine()
        main_engine.write_log(f"CTA策略初始化完成")
        def recheck_thread():
            #每个10秒检查所有策略是否初始化完成，使用Timer线程每10秒检查，
            all_strategise_inited = False
            for strategy in cta_engine.strategies.values():
                if strategy.inited == False:
                    all_strategise_inited = False
                    break
                all_strategise_inited = True
            if all_strategise_inited:
                main_engine.write_log(f"CTA策略全部初始化=====")
                cta_engine.start_all_strategies()
                main_engine.write_log(f"CTA策略全部启动=====")
            else:
                newTask = Timer(10, recheck_thread)
                newTask.start()
        cta_engine.init_all_strategies()
        newTask = Timer(5, recheck_thread)
        newTask.start()
        def recheck_trading_period():
            # 每隔10秒检查是否交易时段，否则退出，使用Timer线程每10秒检查，
            trading = check_trading_period()
            if not trading:
                main_engine.write_log(f"关闭子进程")
                main_engine.close()
                sys.exit(0)
            else:
                closeTask = Timer(10, recheck_trading_period)
                closeTask.start()
        closeTask = Timer(5, recheck_trading_period)
        closeTask.start()
    def run_parent():
        """
        Running in the parent process.
        """
        print("启动CTA策略守护父进程")
        with open('Mutiple_Accounts_Config.json', mode="r", encoding="UTF-8") as f:
            account_detail_list = json.load(f)
        child_process_list = []
        while True:
            trading = check_trading_period()
            # Start child process in trading period
            if trading and child_process_list == []:
                print("启动子进程")
                # 同时运行account 不应该超过cpu核心数，因为vntrader 可以看成是一个持续循环，在main_engine.close之前不会退出cpu占用；
                # 等待的运行事务将一直等待
                for account_detail in account_detail_list:
                    new_process = multiprocessing.Process(target=run_child, name=account_detail["account_name"], args=(account_detail,))
                    new_process.start()
                    child_process_list.append(new_process)
                # # 使用进程池更加方便，但是无法给进程命名去查看log 是那个account的，所以不建议
                # pool = multiprocessing.Pool(multiprocessing.cpu_count())
                # child_process_list = pool.map_async(run_child, account_detail_list)
                print("子进程启动成功")
```

```
        # 非记录时间则退出子进程
        if not trading and child_process_list:
            for process in child_process_list:
                if not process.is_alive():
                    child_process_list.remove(process)
            if child_process_list == []:
                print("子进程关闭成功")
        sleep(10)
if __name__ == "__main__":
    run_parent()
```

还有一个小修改，为了输出各自子进程名，在LogEngine中，trader/engine.py 中，加入这个processName：

```
self.formatter = logging.Formatter(
    "%(asctime)s  %(levelname)s: %(processName)s %(message)s"
)
```

不过如果账户太多，还是有点乱，那时候可以用其他工具来执行；另外策略代码也可以按照工作路径区分，这里没有展示。

https://billyzhangguoping.github.io/CPPLearningManual/