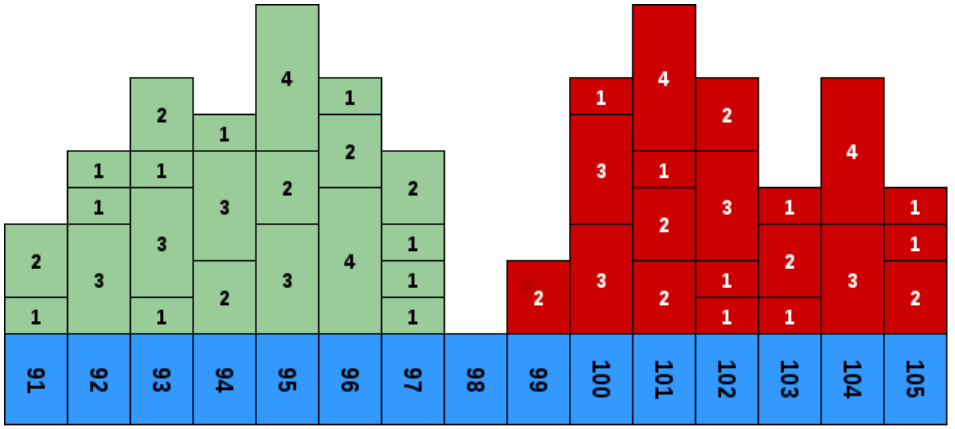
PyXchange

Orderbook & matching engine simulator

Bc. Pavel Schön, July 2016



About author

- Current job: Senior software developer at Deutsche Börse Services
  - ➔ Trading system developer of Eurex exchange, specialities:
    - Python, C++, implementing APIs using both
    - Performance and latency monitoring
    - Network programming, TCP/IP reassembly (pcap), market data reassembly
    - Orderbook visualisation and analysis, market microstructure
    - JavaScript visualisation: D3.js, X3Dom.js

- Past jobs:
  - Python / Django developer at COEX CZ
  - Python developer, linux server administrator at Centrum.cz

PyXchange - overview

- The matching engine is implemented in **C++14** and **Boost** libraries
  - **Boost::Python**
  - **Boost::Multi_index** container
  - **Boost::Format** for string formatting (used for logging)
- The matching engine can be used in two ways:
  1. As Python package for scripting and embedding
     - Some C++ classes are exposed to Python with simple but robust API
     - Messages: ***ping/pong, createOrder, marketOrder, cancelOrder, cancelAll***
  2. As standalone TCP server listening on trading and market data interface
     - TCP server is implemented in **Twisted**, but could also use different networking toolkit

- Almost full **unit-test** coverage plus performance test is included
- High throughput, on authors older workstation ~ 4000 $messages.s^{-1}$

PyXchange – the engine

- Three classes exposed to Python: Client, Trader, Matcher (operating OrderBook)
- OrderBook has two **boost::multi_index** containers of orders: bid orders, ask orders
- Multi_index container is feature rich container with iterator access via configurable indexes
- Not need to use combination of STL containers (vector, map, multimap, set, multiset etc.)
- Each bid or ask container is indexed by:
  - **index by price-time** (used for matching), orders are sorted by price
    - bid orders – descending price, ask orders – ascending price
    - orders at the same price level are sorted by time, lower priority first
  - **index by price** (used for price level aggregation), sort order same as in previous index
  - **index by trader** (used on *cancelAll* to find all orders of Trader)
  - **index by pair [Trader, orderId]** (used on *createOrder* and *cancelOrder* to find order by ID)

PyXchange – scripting API

```
>>> from pyxchange import engine, utils
>>> matcher = engine.Matcher()
INFO:pyxchange:OrderBook is ready
INFO:pyxchange:Matcher is ready
>>> t1 = utils.DequeHandler()
>>> trader1 = engine.Trader(matcher, 'trader1', t1)
INFO:pyxchange:trader1 connected
>>> trader1.createOrder({ 'side': 'BUY', 'price': 100, 'quantity': 10, 'orderId': 1 })
DEBUG:pyxchange:trader1 added order bid:10@100
>>> t1.messages.popleft()
{'report': 'NEW', 'orderId': 1, 'message': 'executionReport', 'quantity': 10}
>>> t2 = utils.DequeHandler()
>>> trader2 = engine.Trader(matcher, 'trader2', t2)
INFO:pyxchange:trader2 connected
>>> trader2.createOrder({ 'side': 'SELL', 'price': 90, 'quantity': 20, 'orderId': 1 })
DEBUG:pyxchange:Execution 10@100
DEBUG:pyxchange:trader2 added order ask:10@90
>>> t2.messages.popleft()
{'report': 'FILL', 'orderId': 1, 'message': 'executionReport', 'price': 100, 'quantity': 10}
>>> t2.messages.popleft()
{'report': 'NEW', 'orderId': 1, 'message': 'executionReport', 'quantity': 10}
```

Boost::Python interface

```
BOOST_PYTHON_MODULE( engine )
{
    using namespace ::boost::python;
    using namespace ::pyxchange;

    class_<Trader, TraderPtr, boost::noncopyable>( "Trader", no_init )
        .def( "__init__",      make_constructor( &make_shared_<Trader, const MatcherPtr&, const std::string&, const object&> ) )
        .def( "__str__",       &Trader::toString )
        .def( "ping",          &Trader::notifyPong )
        .def( "logDisconnect", &Trader::logDisconnect )
        .def( "handleMessage", &Matcher::handleMessageJson, args( "data" ) )
        .def( "handleMessage", &Matcher::handleMessageDict, args( "data" ) )
        .def( "createOrder",   &Matcher::handleCreateOrder, args( "data" ) )
        .def( "marketOrder",   &Matcher::handleMarketOrder, args( "data" ) )
        .def( "cancelOrder",   &Matcher::handleCancelOrder, args( "data" ) )
        .def( "cancelAll",     &Matcher::handleCancelAll,   args( "data" ) );

    class_<Client, ClientPtr, boost::noncopyable>( "Client", no_init )
        .def( "__init__", make_constructor( &Matcher::makeClient ) )
        .def( "__str__",  &Client::toString )
        .def( "logDisconnect", &Client::logDisconnect );

    class_<Matcher, MatcherPtr, boost::noncopyable>( "Matcher", no_init )
        .def( "__init__", make_constructor( &make_shared_<Matcher> ) );
}
```

Boost::multi_index container

```
template<typename ComparePriceTime>
struct OrderContainer
{
    typedef ComparePriceTime cmp_price_time;
    typedef typename boost::tuples::element<0, typename cmp_price_time::key_comp_tuple>::type cmp_price;

    typedef boost::multi_index::multi_index_container<
        OrderPtr, boost::multi_index::indexed_by<

            boost::multi_index::ordered_unique<                              // iterate orders during match event
                boost::multi_index::tag<tags::idxPriceTime>,
                    extractors::keyPriceTime, cmp_price_time>,

            boost::multi_index::ordered_non_unique<                          // iterate orders by price levels
                boost::multi_index::tag<tags::idxPrice>,
                    extractors::keyPrice, cmp_price>,

            boost::multi_index::ordered_non_unique<                          // find orders of trader
                boost::multi_index::tag<tags::idxTrader>,
                    extractors::keyTrader >,

            boost::multi_index::hashed_unique<                               // find order by id
                boost::multi_index::tag<tags::idxTraderOrderId>,
                    extractors::keyTraderOrderId > > > type;

    typedef std::set<price_t, cmp_price> price_set;
};
```

Contact

- E-mail: pavel@schon.cz
- Linkedin: https://linkedin.com/in/pavelschon
- GitHub: https://github.com/pavelschon

Questions?