# 股票行情模拟生成器

# StockSim

Author： 张燕

17/07/2014

[git@github.com:yeahzzz/stocksim.git](mailto:git@github.com:yeahzzz/stocksim.git)

## 目录

Contents

[股票行情模拟生成器 1](#_Toc393454833)

[StockSim 1](#_Toc393454834)

[目录 2](#_Toc393454835)

[一、 问题分析 3](#_Toc393454836)

[二、 设计思路 3](#_Toc393454837)

[1. 后台数据生成器 3](#_Toc393454838)

[2. 推送和订阅中间件 3](#_Toc393454839)

[3. 订阅和接收客户端 4](#_Toc393454840)

[三、 系统架构 5](#_Toc393454841)

[四、 实现细节 5](#_Toc393454842)

[1. 行情动态模拟 5](#_Toc393454843)

[2. 订阅和消息推送中间件 6](#_Toc393454844)

[3. 订阅和消息接收客户端 7](#_Toc393454845)

[五、 安装和部署 8](#_Toc393454846)

[1. 环境配置 8](#_Toc393454847)

[2. 编译 11](#_Toc393454848)

[3. 运行 12](#_Toc393454849)

## 一、 问题分析

7月10日收到设计需求文档，经过两天的调查和构思，初步确定了解题思路:

1. 安装和了解股票软件的功能（如： 大智慧等）
2. 确定开发运行环境为Ubuntu-13 64bit Desktop version
3. 使用版本管理工具 Git（repository@ GitHub）
4. 根据文档需求，快速了解中间件Node.js和ProtoBuf等技术
5. 设计系统各个部分的接口、消息格式和交互方式
6. 分别编写后台，中间件和客户端代码，并独立测试
7. 集成各个部分测试，优化和改进
8. 总结和撰写技术文档

## 二、 设计思路

### 1. 后台数据生成器

此部分模拟股票行情数据，实时更新数据并推送。

根据文档需求，所生成每只股票都按随机频率进行数据更新，因此应该使用多线程队列的方式进行数据的更新。在本系统中，只模拟十支股票的信息，因此使用十个线程分别进行数据更新，在真实的系统中，股票数据很多，每个线程可以分别按轮转的方式去更新每个股票的行情。

得到更新股票信息的线程，用线程锁互斥的方式将更新信息放入队列中，由发送线程负责TCP连接和发送数据。

由于股票行情对实时性要求很高，因此发送队列只存放最新更新的数据。

### 2. 推送和订阅中间件

此部分接收后台推送过来的数据，并将更新数据发送给指定客户端。

根据文档需求，中间件有消息订阅的功能，查阅资料可以使用ZeroMQ的PUB-SUB机制来实现此订阅和并发连接客户端的要求。

后台与中间件node.js之间使用JSON这种简单方便的编码格式进行数据传输。

中间件node.js与客户端使用需求指定的ProtoBuf进行数据包的封装。

### 3. 订阅和接收客户端

此部分实现订阅股票的功能，可实时更新被订阅的股票行情。

客户端与中间件在消息传输和消息格式保持对称，即使用ZeroMQ的Subscribe机制和ProtoBuf进行数据包的解码。并将实现此过程的函数封装成动态链接库，由客户端动态调用。

## 三、 系统架构



图 StockSim系统架构

## 四、 实现细节

### 1. 行情动态模拟

#### 多线程更新数据

使用Python多线程并发轮询的方式将每个股票的字典信息按随机频率进行更新。 每只股票起始价格和交易量在区间内随机生成，每次更新比例在正负区间内随机产生。由Update Event为事件触发插入处理队列。

#### 线程锁和处理队列

队列由更新数据的线程进行插入操作，操作时使用线程锁将共享队列占有，处理完成后释放。

当队列满时对之前的包进行丢弃，队列内总是存放最新更新的数据。

#### JSON格式编码

JSON格式是一种互联网广泛采用的轻量级的数据交换格式，方便后台封装和中间件Node.js解码。

#### TCP连接和发送

后台和中间件之间使用TCP单连接方式进行数据传输。后台Server绑定指定IP和Port，进入监听Listening状态，等待客户端的连接。

当队列有数据时，按时间间隔将数据包发送到连接客户端（这个系统中是中间件）。

### 2. 订阅和消息推送中间件

#### TCP连接和接收

中间件和后台建立TCP连接，不停从后台接收推送的更新数据消息。

#### 消息获取和解码

JSON规则简单，轻量级，且格式非常方便JavaScript来解码。

#### ProtoBuf格式组包、编码和序列化

ProtoBuf提供结构化的消息格式，接口简洁，可将消息封装到二进制数据包里，在传输效率和易用上都有优势。 在Node.js使用ProtoBufJS库提供的接口可将消息结构化编码到数据包里。

#### ZeroMQ Publisher发布

TCP socket连接的监听和客户端连接的管理工作都由ZeroMQ负责。

ZeroMQ提供PUB/SUB机制。 在发送时，把订阅的topic放在数据包的头部， 由ZeroMQ发送数据包。

### 3. 订阅和消息接收客户端

#### ZeroMQ Subscriber订阅

与中间件的连接由ZeroMQ负责管理。

在接收消息时， ZeroMQ会根据topic对数据包进行过滤， 只接受被订阅的消息。ZeroMQ支持多个Subscriber并发连接和订阅不同消息， 客户端可同时收到被订阅的消息。

#### ProtoBuf格式解码

ZeroMQ收到消息后将消息体放入一个buffer中，buffer包括两部分：topic部分和按ProtoBuf格式编码的消息体body部分。计算偏移量将topic部分去掉，调用ProtoBuf接口按消息格式解码body部分。

#### 使用动态链接库

ProtoBuf由C++编写，因此C++很方便编译和生成动态链接库。C语言有处理二进制数据包的优势。因此生成动态链接库负责接收消息和解码消息，由客户端传入网络信息和订阅内容。

## 五、 安装和部署

### 1. 环境配置

#### Python

Linux机器一般默认安装有Python，无需特别安装，2.6或更高版本都可。

#### Node.js

版本0.10.29

下载地址：

<http://nodejs.org/download/>

解压：

tar -zxf node-v0.10.29-linux-x64.tar.gz

cd node-v0.10.29-linux-x64

编译安装：

./configure --prefix=/home/node\_installdir

make

make install

环境配置：

打开/etc/profile文件，并在末尾添加：

export NODE\_HOME=/home/node\_installdir

export PATH=$PATH:$NODE\_HOME/bin

export NODE\_PATH=$NODE\_HOME/lib/node\_modules

使配置生效：

source /etc/profile

这时输入node –v会显示版本号，恭喜你安装成功！

用npm安装node.js的一些库：

npm install zmq –g

npm install protobufjs –g

（npm依赖网络环境安装，若网络环境不好，也可以下载源文件进行安装：

<https://github.com/JustinTulloss/zeromq.node>

<https://github.com/dcodeIO/ProtoBuf.js>）

#### ZeroMQ

版本3.2.4

下载地址：

<http://zeromq.org/intro:get-the-software>

解压：

tar –zxf zeromq-3.2.4.tar.gz

cd zeromq-3.2.4

编译安装：

./configure

sudo make

sudo make check

sudo make install

sudo ldconfig

#### ProtoBuf

版本2.5.0

下载地址：

<https://code.google.com/p/protobuf/>

编译安装：

./configure --prefix=xx

make

make check

make install

ldconfig

环境配置：

打开/etc/profile文件，并在末尾添加：

export PATH=$PATH:/home/amy/stock/test/protobuf-2.5.0/installdir/bin

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/home/amy/stock/test/protobuf-2.5.0/installdir/lib

export PKG\_CONFIG\_PATH=/home/amy/stock/test/protobuf-2.5.0/installdir/lib/pkgconfig

使配置生效：

source /etc/profile

### 2. 编译

源码可以在GitHub download：

<https://github.com/yeahzzz/stocksim>

用protoc生成消息格式头文件：

$ cd stocksim/MessageFormat

$ protoc -I=./ --cpp\_out=../SubscribeClient ./message.proto

编译动态链接库：

$ cd stocksim/SubscribeClient

$ g++ libsub.cpp message.pb.cc -I$PROTOBUF\_HOME/include -L$PROTOBUF\_HOME/lib -lprotobuf -lzmq -pthread -fPIC -shared -o libsub.so

编译客户端：

$ cd stocksim/SubscribeClient

$ g++ client.cpp -L. -lsub -o client

### 3. 运行

运行后台数据模拟器：

$ cd stocksim/DataSim

$ python DataSim.py

运行订阅中间件：

$ cd stocksim/PushPublish

$ node PushPublish.js

运行客户端：

$ cd stocksim/SubscribeClient

$ ./client