Market Data Line 行情代理使用说明

目录

1. 系统需求

支持 windows 7 32/64, ubuntu 12.04, centos 6.3。
推荐系统 windows 7，有比较好的兼容性。
双核 CPU，1G 以上内存，1G 以上空闲磁盘空间。
如果需要连接较多的客户端(大于 10 个)，内存最好 8G。
如果需要使用 redis 协议建议 4 核 3G 以上 CPU。
如果需要备份消息，则空闲磁盘空间根据要保存的消息类型决定。

2. 安装运行

- 作为 win32 服务

安装服务，在管理员权限下运行 feeder_handler.exe –i feeder_handler
卸载服务，在管理员权限下运行 feeder_handler.exe –u feeder_handler
启动服务，net start feeder_handler
停止服务，net stop feeder_handler

- 作为 linux demon

启动，feeder_handler –d
停止，kill pid

- 作为控制台运行

启动，feeder_handler –c
停止，CTRL+C

3. 参数配置

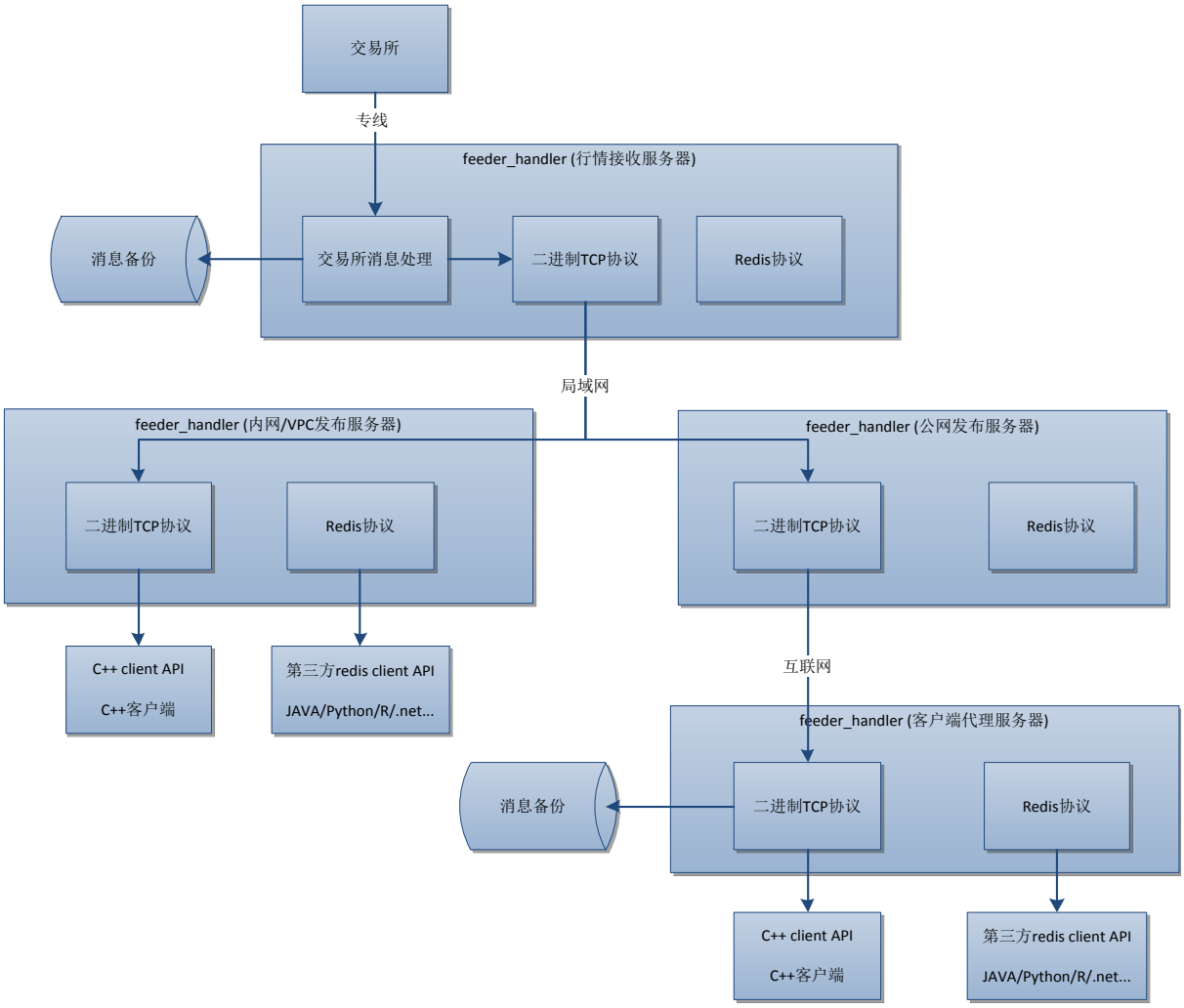
参数配置文件采用 json 格式，包含程序配置和用户私有的认证信息，程序配置已按最佳情况优化一般无需修改。

参数名	说明	默认值
feeder_handler	程序主配置	
feeder_handler. Publishers	行情发布协议列表，可选 1~2 个协议。	
feeder_handler. Publishers[i].Type	发布协议，以下任意一个值 TCP_SERVER REDIS_SERVER	
feeder_handler. Publishers[i].Address	协议监听地址及端口	Type 为 TCP_SERVER 时： 0.0.0.0:9010 Type 为

		REDIS_SERVER 时： 0.0.0.0:9379
feeder_handler. Publishers[i].OutputBufferMax	客户端输出缓存(客户端来不及接收的数据)全局大小上限(KB)，当缓存数据超过指定大小时，程序依次踢出私有缓存最大的客户端，直到全局缓存满足指定大小。	512000
feeder_handler. Publishers[i].Encoding	消息发布编码，Type 为 REDIS_SERVER 时有效，以下任意一个值： 1：二进制 2：FAST 3：JSON 4：protobuf Type 为 TCP_SERVER 时发布编码由客户端登录时动态指定	4
msg_backup	消息备份配置选项，存在表示需要备份	存在
msg_backup.Encoding	消息备份编码，以下任意一个值： 1：二进制 3：JSON 4：protobuf 5：CSV	1
msg_backup.BackupDir	备份目录	msg_backup
msg_backup.FlushFileInterval	以指定时间间隔刷新文件到磁盘(秒)	60
msg_backup. FlushFileSize	写入指定大小数据后刷新文件到磁盘(字节)	104857600
msg_backup. WriteFileIndex	创建文件记录索引	true
msg_backup. DiskspaceReservedInGB	清理备份文件时要得到的空闲磁盘空间（GB)，仅支持 windows 下备份到本地磁盘,不能小于 1。	10
msg_backup. FolderCountReserve	清理备份文件时要保留的备份文件夹数，不能小于 1	3
MSG_FORWARDER	上游服务器配置选项	
MSG_FORWARDER.UpStreams[i].Address	上游服务器地址列表，多个地址用;号隔开。当配	

	置多个地址时程序可以执行自动故障切换。	
MSG_FORWARDER.UpStreams[i].HeartbeatInterval	上游服务器空闲心跳间隔	10
MSG_FORWARDER.UpStreams[i].HeartbeatTimeout	上游服务器数据超时	90

4. Feeder_handler 架构图



5. Feeder_handler 功能

5.1 支持多种通讯协议

支持 TCP 和简化的 Redis 协议，详细描述请参考《通联数据 MDL-Redis 客户端参考.pdf》

5.2 支持多种消息编码

支持标准的 FAST 编码，数据压缩编码，高效的二进制编码，灵活的 Json 编码和通用性

强的 CSV 编码

5.3 支持多种消息接收方式

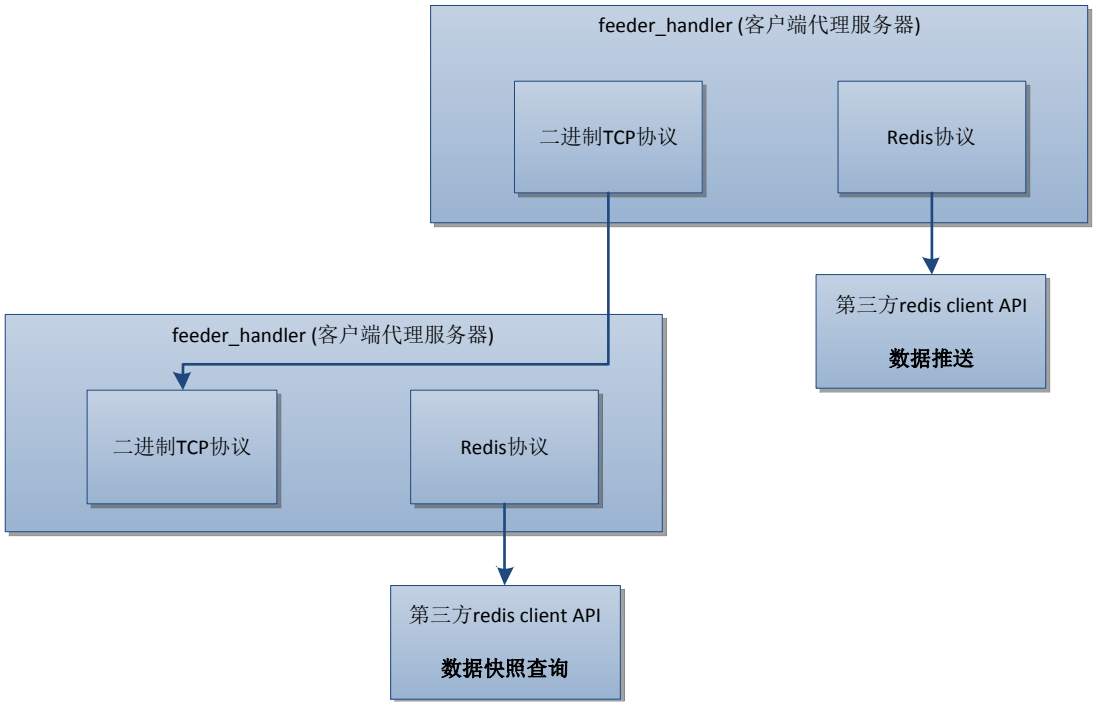
支持 C++客户端接入，配合高效的二进制协议二进制消息编码实现低延迟数据传输。

支持多种高级语言接入，通过简化的 Redis 协议可以支持 40 几种编程语言。

支持文件更新方式接入，客户端通过不断刷新文件取得数据更新，由于数据落地后不会丢失，从而简化了客户端编程。

5.4 消息转发

允许逐层消息转发，请参考配置选项 MSG_FORWARDER。当客户端需要分散服务端处理压力时特别有用，比如说需要频繁的快照查询的同时保持顺畅的数据推送。



5.5 故障切换

在网络传输出现问题时可自动切换到其它服务器，请参考配置选项 MSG_FORWARDER.UpStreams[i].Address

5.6 流量控制

支持客户端传输异常检测，以保证服务端稳定运行，请参考配置选项 feeder_handler.Publishers[i].OutputBufferMax

5.7 消息备份

5.7.1 文件名规则

Feeder_hander 每天在指定备份目录下建立一个文件夹，文件夹名称为 YYYYMMDD，注意由于每个计算机时间不同，对于夜盘行情，当日文件可能包含上一天的最后几条数据。

接收到的消息按类别存入文件，文件名规则 mdl_serviceID_msgID_msgPartID，其中 serviceID 表示服务 ID，msgID 表示消息 ID，msgPartID 表示拆分的消息文件。对于 CSV 编码的 level2 市场行情数据需要拆分为 3 个文件来保存一个消息。

ServiceID		MessageID		文件名
3	上交所 Level1	3	指数	mdl_3_3_0
		4	股票	mdl_3_4_0
		5	基金	mdl_3_5_0
		6	债券	mdl_3_6_0
5	深交所 Level1	1	指数	mdl_5_1_0
		2	股票	mdl_5_2_0
7	中金所期货	1	期货	mdl_7_1_0
8	郑交所期货	1	期货	mdl_8_1_0
9	上期所期货	1	期货	mdl_9_1_0
10	大商所期货	1	期货	mdl_10_1_0
11	港股	2	市场行情	mdl_11_2_0
		3	恒生指数	mdl_11_3_0
12	申万行业分类	1	行业分类指数	mdl_12_1_0
13	衍生数据	1	上交所股票分钟线	mdl_13_1_0
		2	深交所股票分钟线	mdl_13_2_0
		3	港交所股票分钟线	mdl_13_3_0
		4	大商所期货分钟线	mdl_13_4_0
		5	上期所期货分钟线	mdl_13_5_0
		6	郑商所期货分钟线	mdl_13_6_0
		7	中金所期货分钟线	mdl_13_7_0
		8	上交所期权分钟线	mdl_13_8_0
		9	上交所指数分钟线	mdl_13_9_0
		10	深交所指数分钟线	mdl_13_10_0
		11	上交所股票资金流向	mdl_13_11_0
		12	深交所股票资金流向	mdl_13_12_0
		13	行业资金流向	mdl_13_13_0

5.7.2 消息文件编码

5.7.2.1 二进制编码

文件名后缀为.bin

文件格式为

MDLMessageHead	body	MDLMessageHead	body	...

读取时首先应该读取消息头部 MDLMessageHead(定义请参考 mdl_api_types.h)，确定消息 body 大小, body 类型，然后读取消息 body 并做强制类型转换即可。

例如:

```
MDLMessageHead* head = read_head_from_file();
void* body = read_body_from_file ();
If (head->ServiceID == 3 && head->MessageID == 3) {
    mdl_shl1_msg::Indexes* msg = (mdl_shl1_msg::Indexes*)body;
    printf(msg->SecurityID.c_str());
}
```

5.7.2.2 Protobuf 编码

文件名后缀为.pb

文件格式为

MDLMessageHead	body	MDLMessageHead	body	...

读取时首先应该读取消息头部 MDLMessageHead(定义请参考 mdl_api_types.h)，确定消息 body 大小, body 类型，然后读取消息 body 然后由 protobuf 库进行转换。

例如:
```
MDLMessageHead* head = read_head_from_file();
void* body = read_body_from_file ();
If (head->ServiceID == 3 && head->MessageID == 4) {
    mdl_shl1_pbmsg:: Equity msg;
    if (msg.ParseFromArray(body, head->MessageSize – head->HeadSize)) {
        printf(msg. Securityid().c_str());
    }
}
```

5.7.2.3 Json 编码

文件名后缀为.json

文件格式为

MDLMessageHead	body	MDLMessageHead	body	...

读取时首先应该读取消息头部 MDLMessageHead(定义请参考 mdl_api_types.h)，确定消息 body 大小, body 类型，然后读取消息 body 即可。

例如:
```
MDLMessageHead* head = read_head_from_file();
void* body = read_body_from_file ();
If (head->ServiceID == 3 && head->MessageID == 4) {
    printf("read shanghai level1 equity message\n");
    printf(body); // json 编码以 null 结尾
}
```

5.7.2.4 csv 编码

文件名后缀为.csv

文件格式为

head\n	line\n	line\n	line\n	...

注意 csv 编码为 gbk

5.7.3 消息索引文件

消息索引文件允许快速消息定位和读取，其文件名为: 消息文件名_index。

每个消息在写入文件后会写入摘要到索引文件，摘要大小为固定的 12 字节，按顺序依次为:

字段名	字段含义	
文件偏移	消息在文件中的首字节偏移	8 字节
消息大小	消息大小	4 字节

5.7.4 服务器状态文件

Feeder_handler 每隔 1 秒更新状态文件，以反映最新写入的数据并表示服务器存活。状

态文件名为 state.csv，保存在每天的备份文件夹中。状态文件是 CSV 文件格式，没有头部，第一行是状态文件信息，其它行是数据文件信息。例如:

文件名	文件状态 0: 表示正常操作 1: 数据文件读写错误 2: 索引文件读写错误 3: 文件全部读写错误	最后写入的文件大小(字节)	最后更新时间(HHMMSSsss)
state.csv	0	0	113030000
mdl_3_3_0.csv	0	100	112030000
mdl_3_4_0.csv	0	200	112031000

5.7.5 备份文件清理

服务器在创建新一天的备份文件夹时对已保存的旧文件夹进行清理，清理规则由 msg_backup. DiskspaceReservedInGB，msg_backup. FolderCountReserve 两个配置参数控制。FolderCountReserve 决定要保留的文件夹个数，即至少需要保留的文件夹个数，DiskspaceReservedInGB 决定要保留的磁盘空间，即最多可保留的文件夹个数。

6. 日志输出

Feeder_handler 输出两种类型日志，trace 类型包含每隔 3 秒输出的当前流量信息，写入 feeder_handler.trace.log。其它类型包含客户端登入登出日志，以及其它错误和警告，写入 feeder_handler.log。

如果 feeder_handler 以控制台方式运行，则日志在写入文件的同时还显示在屏幕上。

Feeder_handler 的 trace 日志格式如下图:

```
2015-10-14 16:34:42[140215985334016][TRACE]mdl - Sub conn(1) in(3.9kmsg/s 0.3mb/s) thd(0,0,0,0,0,0,0,0)
2015-10-14 16:34:42[140215985334016][TRACE]mdl - TcpPub conn(0) in(3.9kmsg/s 0.8mb/s) enc(0.00, 0B, 0.0mb/s) out(0.0kmsg/s 0.0kb/s) buf(0B)
2015-10-14 16:34:42[140215985334016][TRACE]mdl - RdsPub conn(0) in(3.9kmsg/s 0.8mb/s) enc(0.83, 0B, 0.8mb/s) out(0.0kmsg/s 0.0kb/s) buf(0B)
```

Trace 日志格式说明如下:

- Sub 表示上游服务器流量
  conn(连接的上游服务器数量)
  in(输入消息速率，输入流量)
  thd(每线程未编码的消息个数) 进程共享的编解码线程池信息
- TcpPub 表示 TCP 协议的流量，如果启用了 TCP 协议会出现
  conn(接入的客户端数量)
  in(输入消息速率，输入流量)，由于此时消息已经从上游消息解码，实际消息流量会小于上游输入流量。
  enc(编码压缩率，未编码的数据大小，编码速度)
  out(输出消息速率，输出流量)
  buf(未及时发送到客户端的数据大小)
- RdsPub 表示 Redis 协议的流量，如果启用了 Redis 协议会出现
  conn(接入的客户端数量)
  in(输入消息速率，输入流量)，由于此时消息已经从上游消息解码，实际消息流量会小于上游输入流量。
  enc(编码压缩率，未编码的数据大小，编码速度)

out(输出消息速率，输出流量)

buf(未及时发送到客户端的数据大小)

- 注意日志时间为 UTC 时间，转化为中国时间需要+8