Scribe 编译与非超级权限运行

关于编译 *Scribe* 需要注意的问题

屈庆磊

quqinglei@pwrd.com

2013 年 *8* 月 *11* 日

目录

1 编译须知

Scribe 依赖一些第三方的包，所以在编译的时候需要一些预先配置，官网上所说的编译过程并不全面，下面是一个较为全面的编译方法。但仍然存在需要改进的地方。

libevent libevent 是一个事件触发的网络库，适用于 windows、linux、bsd 等多种平台，内部使用 select、epoll、kqueue 等系统调用管理事件机制。著名分布式缓存软件 memcached 也是 libevent

based，而且 libevent 在使用上可以做到跨平台，而且根据 libevent 官方网站上公布的数据统计，似乎也有着非凡的性能。

boost Boost 库是一个可移植、提供源代码的 C++ 库，作为标准库的后备，是 C++ 标准化进程的发动机之一。Boost 库由 C++ 标准委员会库工作组成员发起，其中有些内容有望成为下一代 C++ 标准库内容。在 C++ 社区中影响甚大，是不折不扣的“准”标准库。Boost 由于其对跨平台的强调，对标准 C++ 的强调，与编写平台无关。大部分 boost 库功能的使用只需包括相应头文件即可，少数（如正则表达式库，文件系统库等）需要链接库。但 Boost 中也有很多是实验性质的东西，在实际的开发中实用需要谨慎。boost 在一些播放软件和音效中指增强，比如 Bass Boost，低音增强。

thrift 是一个软件框架，用来进行可扩展且跨语言的服务的开发。它结合了功能强大的软件堆栈和代码生成引擎

2 本例中下载的包

boost_1_54_0.tar.bz2 boost 库，可以从其官网下载，地址为：www.boost.org/

thrift-0.9.0.tar.gz thrift 包，官网地址：http://thrift.apache.org/

scribe-master.zip scribe 包，可以从 github 上下载，地址为：https://github.com/facebook/scribe/

3 编译并安装

不建议在配置时使用自定义的安装路径，这样在编译的时候可能会带来一些不必要的困难和错误。

3.1 安装编译依赖包

基本的编译环境需要的依赖包：

```
yum install automake flex bison libevent-devel #libevent-devel 不属于基本开发包，但必须安装
```

4 安装 boost

不建议设置 *boost* 的安装路径，采用默认的安装方式反而更容易编译 *scribe*。*boost* 默认情况下会把 *boost* 的动态库安装到*/usr/local/lib* 所以我们在编译完成后可以直接把它拷贝到我们的路径下，因为二进制文件在运行时只需要 *boost* 的动态链接库，而不需要头文件和代码。

```
# tar xjvf boost_1_54_0.tar.bz2
# cd boost_1_54_0
# ./bootstrap.sh
# ./b2
# ./b2 install
```

5 安装 thrift 和其下的 fb303

thrift 的编译需要 *boost* 的支持，所以我们在它之前安装了 *boost*，下面是编译步骤。

```
# tar xzvf thrift-0.9.0.tar.gz
# cd thrift-0.9.0
# ./configure
# make
# make install

# cd contrib/fb303
# ./bootstrap.sh
# make
# make install
```

6 编译 scribe

我们在前面编译并安装一些依赖的时候全部采用默认方式，这样在编译 *scribe* 的时候就能省一些不必要的麻烦，下面是步骤。

```
# ./bootstrap.sh
# ./configure
# make # 在编译的时候就可能会出错，这是因为它设置的依赖库的原因，总之它存在编译的 bug。

# 可以到 src 路径直接使用命令编译
# g++  -Wall -O3 -L/usr/local/lib/ -lboost_system -lboost_filesystem  \
       -o scribed store.o store_queue.o conf.o file.o conn_pool.o scribe_server.o \
       network_dynamic_config.o dynamic_bucket_updater.o  env_default.o \
        -L/usr/local/lib -L/usr/local/lib -L/usr/local/lib -lfb303 -lthrift \
        -lthriftnb -levent -lpthread  libscribe.a libdynamicbucketupdater.ae.a \
       libdynamicbucketupdat
```

编译完成后会生成一个可执行的 scribed 文件，它就是我们想要的东西！

7 非超级权限运行

绝大多数应用软件是不需要超级用户权限运行的，*scribe* 并不例外，我们没必要采用静态编译的方式解决问题，导出一个库的环境变量就解决了这个问题。我们把 *scribed* 依赖的动态库放到一个路径，并导出这个路径的环境变量，它就可以使用了，而每个普通用户也维护着一份环境变量，所以我们可以把环境变量写在脚本上，在程序执行之前导出。

8 打到一个文件夹里

为了方便使用，我们把 *scribed* 打到一个文件夹里，方法如下：

```
# mkdir -p scribe-bin/bin/
# mkdir -p scribe-bin/lib/
# cp -r /usr/local/lib/* scribe-bin/lib/
# cp scribed scribe-bin/bin/ # 把生成的二进制文件放到文件夹的 bin 路径
# tar czvf scribe-bin.tar.gz scribe-bin
```

9 环境变量

以下环境变量是用来告诉程序去哪里找动态链接库的东西。

```
export LD_LIBRARY_PATH="/usr/lib:path-to-scribe-bin/lib"
```