Clion Remote Debug

前言

曾经我在MacOS上编译clickhosue踩了很多的坑 issue，最后决定放弃使用在MacOS上编译clickhouse进行debug的想法，原因是当我们在MacOS进行编译clickhouse的时候会有很多功能例如：S3Disk这些功能是不会被编译到代码当中的，另外还有很多兼容性问题，很让人烦恼。所以，我选择在VirtualBox上安装ubuntu（clickhouse社区也是采用ubuntu编译的代码）虚拟机,本地通过Clion配置Remote debug进行打断点源码分析。

安装Clion

官方下载Clion 2020.3, 破解方式请参考：http://www.520xiazai.com/soft/jetbrains-eval-reset.html

安装Ubuntu

虚拟机选择virtualbox, ubuntu选20.04, 这里就不在赘述了

编译ClickHouse

在remote端进入clickhouse源码根路径下：

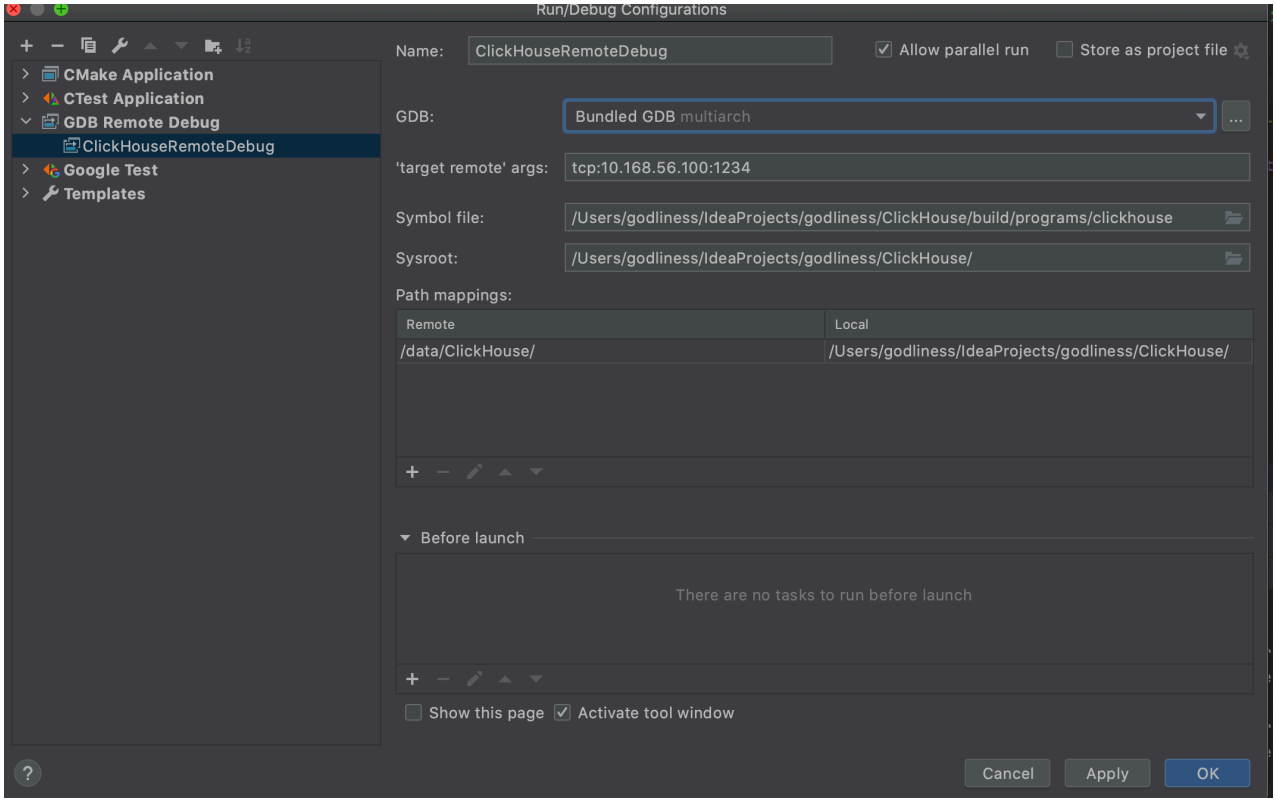
```
mkdir build && cd build

cmake .. \
    -DCMAKE_C_COMPILER=/usr/bin/clang-11 \
    -DCMAKE_CXX_COMPILER=/usr/bin/clang++-11 \
    -DCMAKE_BUILD_TYPE=Debug \
    -DENABLE_JEMALLOC=0 \
    -DENABLE_TESTS=OFF

ninja clickhouse-server
```

配置remote debug

可以参考：https://cloud.tencent.com/developer/article/1406250



Clickhouse代码准备两套，一套在ubuntu上主要负责编译与运行，一套在本地主要负责debug时候clion gdb查找debug info, 以及Clion源码跳转跟踪，这两套代码均可以通过clion进行同步。

remote端：

```
sudo gdbserver :1234 --attach `pidof clickhosue-server`
```

local端

```
GDB: Clion自带的gdb工具，可以支持多平台
Symbol file: 就是你本地从远端同步过来的clickhouse二进制文件
Sysroot: Clickhouse源码根路径
Path mappings: 远端与本地的项目根路径
```

断点分析

本地启动debug 

显示connected以后，发送请求，断点锁定，就可以开始debug了。

这里如果跳转经常跳转到一些信号回调函数的话，请在Clion GDB console上输入：

```
handle SIGUSR1 noprint nostop
handle SIGUSR2 noprint nostop
```

参考

https://cloud.tencent.com/developer/article/1406250

http://www.520xiazai.com/soft/jetbrains-eval-reset.html

https://github.com/ClickHouse/ClickHouse/issues/21904