本文主要讨论了将bpftime项目适配到centos7时，遇到的问题，以及一些问题的解决方案。

一、在centos7上直接适配

1.更换GCC版本

由于centos7自带的GCC版本为GCC4.8，而bpftime需要支持C++20的编译器，GCC 9版本后便开始支持对C++20 的初步支持。因此，需要将GCC版本升级为GCC 9以上。这里，将GCC版本升级为GCC 11。

步骤：

1.由于通过yum默认仓库无法获取到较高版本的GCC，因此需要加入 SCL 仓库：

```
sudo yum install centos-release-scl
```

2.由于网络问题，因此我们先配置镜像源：

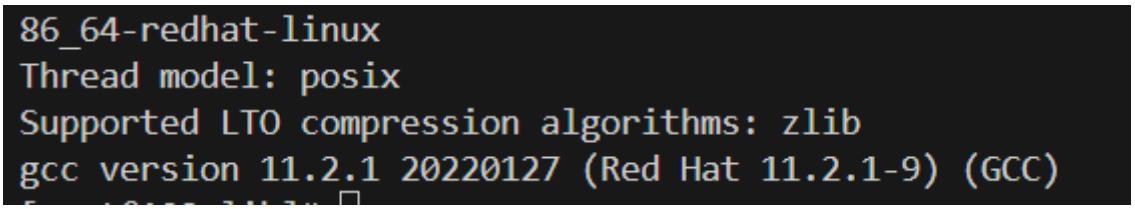
```
#修改以下三个文件：
/etc/yum.repos.d/CentOS-Base.repo
/etc/yum.repos.d/CentOS-SCLo-scl-rh.repo
/etc/yum.repos.d/CentOS-SCLo-scl.repo
```

3.换源后 GPGKEY 报GPG密钥不匹配或者密钥缺失错误：

```
#引入GPGKEY
vi /etc/yum.repos.d/CentOS-SCLo-scl-rh.repo
gpgkey=https://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-
7,https://www.centos.org/keys/RPM-GPG-KEY-CentOS-SIG-SCLo
rpm --import https://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7
rpm --import https://www.centos.org/keys/RPM-GPG-KEY-CentOS-SIG-SCLo
```

4.下载GCC 11

```
sudo yum install devtoolset-11-gcc*
scl enable devtoolset-11 bash
gcc -v
```



到这里，GCC 11升级成功。

2.升级Clang版本

由于编译eBPF程序，需要Clang编译器，因此，将Clang编译器升级到clang 16。这里通过二进制包的方式进行升级（因为使用源码编译方式会报错）。

安装Clang-16：

```
#下载 Clang 16.0.3 二进制包
wget https://github.com/llvm/llvm-project/releases/download/llvmorg-
16.0.3/clang+llvm-16.0.3-x86_64-linux-gnu-ubuntu-20.04.tar.xz
#解压二进制包
sudo tar -xf clang+llvm-16.0.3-x86_64-linux-gnu-ubuntu-20.04.tar.xz -C /usr/local/
sudo mv /usr/local/clang+llvm-16.0.3-x86_64-linux-gnu-ubuntu-20.04 /usr/local/clang-
16
#配置环境变量/etc/profile
export PATH=/usr/local/clang-16/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/clang-16/lib:$LD_LIBRARY_PATH
#验证安装
clang --version
```

这里会报错：

```
clang: error while loading shared libraries: libtinfo.so.6: cannot open shared
object file: No such file or directory
```

但是系统中是有libtinfo.so库，因此，创建符号链接：

```
sudo ln -s /usr/lib64/libtinfo.so.5 /usr/lib64/libtinfo.so.6
sudo ldconfig
```

问题得到解决，但是接下来又会报错：

```
[root@192 project]# sudo ln -s /usr/lib64/libtinfo.so.5 /usr/lib64/libtinfo.so.6
[root@192 project]# sudo ldconfig
[root@192 project]# clang --version
clang: /lib64/libtinfo.so.6: no version information available (required by clang)
clang: /lib64/libm.so.6: version GLIBC_2.27' not found (required by clang)
clang: /lib64/libm.so.6: version GLIBC_2.29' not found (required by clang)
clang: /lib64/libc.so.6: version GLIBC_2.34' not found (required by clang)
clang: /lib64/libc.so.6: version GLIBC_2.32' not found (required by clang)
clang: /lib64/libc.so.6: version GLIBC_2.33' not found (required by clang)
clang: /lib64/libstdc++.so.6: version GLIBCXX_3.4.30' not found (required by clang)
clang: /lib64/libstdc++.so.6: version GLIBCXX_3.4.29' not found (required by clang)
clang: /lib64/libstdc++.so.6: version CXXABI_1.3.13' not found (required by clang)
clang: /lib64/libstdc++.so.6: version CXXABI_1.3.11' not found (required by clang)
clang: /lib64/libstdc++.so.6: version GLIBCXX_3.4.26' not found (required by clang)
clang: /lib64/libstdc++.so.6: version GLIBCXX_3.4.20' not found (required by clang)
clang: /lib64/libstdc++.so.6: version GLIBCXX_3.4.21' not found (required by clang)
```

错误信息表明当前系统中的 **GLIBC** 和 **libstdc++** 版本较低，不满足 Clang 的运行要求。GLIBC 是 GNU C 库，是许多程序运行的核心组件。需要将它升级到至少 **2.34** 版本。

在这里，通过源码的方式进行GLIBC的升级，但是由于系统的make和bison版本过旧，因此需要升级。升级好以后，设置 `LD_LIBRARY_PATH` 环境变量，来使用新的GLIBC库。但是这里会报错：

```
relocation error: /opt/glibc-2.34/lib/libc.so.6: symbol __tunable_get_val, version
GLIBC_PRIVATE not defined in file ld-linux-x86-64.so.2 with link time reference
```

这个错误表明 `ls` 命令在运行时使用了新的 `glibc` 库（`/opt/glibc-2.34/lib/libc.so.6`），但该库和当前使用的动态链接器 `ld-linux-x86-64.so.2` 不兼容。

这个问题目前一直没有更好的方式去解决。。。

```
clang: /lib64/libstdc++.so.6: version GLIBCXX_3.4.30' not found (required by clang)
上面这个问题可以通过升级GCC版本到GCC 13得到解决。
```

通过查阅资料，最终，本机可以成功安装gcc 11 和 clang 7，并且升级boost为1.78.0版本。

3.开始编译：

错误一：

```
编译过程中，会报错`__u32` 和 `uint32_t`数据类型未定义，通过issue中的解决方案（Try replacing
`__u32` to `uint32_t`, `__u64` to `uint64_t`, etc）得到了解决。但是后面也会报其他类型未定义
```

错误二：

```
link.c:579:10: error: ‘NFPROTO_NETDEV’ undeclared here (not in a function); did you
mean ‘NFPROTO_DECNET’?
  579 |          [NFPROTO_NETDEV] = "netdev",
      |           ^~~~~~~~~~~~~~
      |           NFPROTO_DECNET
link.c:579:10: error: array index in initializer not of integer type
link.c:579:10: note: (near initialization for ‘pf2name’)
make[4]: *** [link.o] Error 1
```

修改代码，编译可以通过。

错误三：

```
boost库中的<boost/container_hash/hash.hpp>找不到
```

通过升级boos库可以解决。

错误四：

```
In file included from /root/project/bpftime/daemon/kernel/bpf_tracer.bpf.c:6:
/root/project/bpftime/third_party/vmlinux/x86/vmlinux.h:5:15: error: attribute
'preserve_access_index' is not supported by '#pragma clang attribute'
#pragma clang attribute push (__attribute__((preserve_access_index)), apply_to =
record)
```

本错误是clang版本太低导致的。

因此，还是需要高版本的Clang。这里升级Clang 到Clang 12版本。

```
[root@192 local]# clang -v
clang: /lib64/libtinfo.so.5: no version information available (required by clang)
clang: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.20' not found (required by clang)
clang: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.21' not found (required by clang)
```

针对这个问题，libstdc++变成GCC 13的库，解决了。

当前环境为GCC 13、Clang 12，继续开始编译：编译过程中还是报了很多错误，尤其在编译libbpf时，错误很多。

关闭libbpf的编译，又会报错：

```
/root/project/bpftime/runtime/src/bpf_map/userspace/prog_array.cpp:8:10: fatal error: bpf/bpf.h: No such file or directory
    8 | #include "bpf/bpf.h"
      |          ^~~~~~~~~~~
compilation terminated.
gmake[2]: *** [runtime/CMakeFiles/runtime.dir/src/bpf_map/userspace/prog_array.cpp.o] Error 1
```

centos7中缺少libbpf的头文件，将libbpf的头文件引入系统，又会报错类似上述的一些错误（类型未定义等），目前这些问题没有得到解决。

二、尝试在Docker中编译好并放入Centos7中

将所有的库文件和依赖库都放入Centos7中，还是报错：

```
[yys@192 lib]$ bpftime
bpftime: /lib64/libm.so.6: version `GLIBC_2.38' not found (required by bpftime)
bpftime: /lib64/libm.so.6: version `GLIBC_2.27' not found (required by bpftime)
bpftime: /lib64/libm.so.6: version `GLIBC_2.29' not found (required by bpftime)
bpftime: /lib64/libm.so.6: version `GLIBC_2.35' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.22' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.25' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.38' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.32' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.28' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.33' not found (required by bpftime)
bpftime: /lib64/libc.so.6: version `GLIBC_2.34' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.31' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.32' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.30' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.29' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.22' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `CXXABI_1.3.9' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.26' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.20' not found (required by bpftime)
bpftime: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.21' not found (required by bpftime)
```

继续尝试升级GLIBC库，但还是失败了（系统中的命令会出现问题）。

通过上述两种方式部署bpftime项目，未能成功。因此，尝试在容器中进行部署：

```
12:05:42                                                    malloc called from pid 505
12:05:43                                                    continue malloc...
12:05:44                                                    malloc called from pid 505
12:05:45                                                    continue malloc...
12:05:46                                                    malloc called from pid 505
12:05:47                                                    continue malloc...
12:05:48                                                    malloc called from pid 505
12:05:49                                                    continue malloc...
12:05:50                                                    malloc called from pid 505
        pid=505         malloc calls: 9                     continue malloc...
12:05:51                                                    malloc called from pid 505
        pid=505         malloc calls: 10                    continue malloc...
12:05:52                                                    malloc called from pid 505
        pid=505         malloc calls: 9                     continue malloc...
12:05:53                                                    malloc called from pid 505
        pid=505         malloc calls: 10                    continue malloc...
```

通过容器方式来部署bpftime是没有问题的。

三、总结

通过以上适配过程中出现的问题可以发现，最主要的问题是难以升级GLIBC库。GLIBC是系统的基础库，几乎所有的程序和工具都依赖于它，升级 `glibc` 会直接影响整个系统。并且许多已安装的软件都是基于旧版本的 `glibc` 编译的，升级 `glibc` 后，它们可能会出现兼容性问题。 如果 `glibc` 更新失败，系统可能无法完成启动过程，甚至可能完全无法进入系统。后期我会寻找更好的方法去解决这个问题。

因此，在旧版本的系统中适配bpftime项目，更推荐的方式是通过容器来部署，这种方式可以更好的安装bpftime需要的一些依赖，并且不和系统库发生冲突，而且这种方式可以更好的兼容内核中的eBPF的特性，更好的发挥bpftime的功能。