基于eBPF的异常容器检测

报告人：肖晗溪

一 技术选择

- 本项目选择了bpftrace来进行数据采集。bpftrace是一种高级跟踪语言，提供了丰富的跟踪指令和语法，使跟踪和分析更加灵活和高效。 bpftrace基于Linux的eBPF技术，通过探针机制采集内核和程序运行的信息，具有轻量级和低性能开销的优点。
- 算法预测部分采用python语言编写。

二 代码文件说明

runqlen.bt

主要功能是监控和统计当前系统的 CFS 运行队列长度。代码说明如下：

（1）初始化提示：在脚本开始时，打印一条消息告知用户采样频率和终止方式。

（2）定时采样：以 50Hz 的频率定期采样。

（3）获取运行队列长度：获取当前任务的 task_struct->从 task_struct 中提取所属的 CFS 运行队列->获取运行队列中正在运行的任务数量，并减去自身任务，得到实际等待运行的任务数。

（4）记录数据：将运行队列长度记录到直方图 @runqlen 中，以便后续分析和可视化。

二 代码文件说明

runqlen.bt生成的log文件示例：

```
Attaching 2 probes...
Sampling run queue length at 99 Hertz... Hit Ctrl-C to end.


@runqlen:
[0, 1)              1918 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[1, 2)                 2 |                                                    |
```

二 代码文件说明

2. oepnsnoop. bt

主要功能是追踪系统调用 open 和 openat 的执行情况，并输出每次调用的详细信息，包括进程 ID、进程名、文件描述符、错误码和打开的文件路径。代码说明如下：

（1）打印提示信息和表头，用于显示追踪结果。

（2）追踪 open 和 openat 系统调用进入事件。

（3）追踪 open 和 openat 系统调用退出事件。

（4）清除关联数组 @filename，释放资源。

二 代码文件说明

opensnoop.bt生成的log文件示例：

```
Attaching 6 probes...
Tracing open syscalls... Hit Ctrl-C to end.
PID    COMM               FD ERR PATH
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-co
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-sc
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-sc
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/kernel/sound/pci/hda/snd-hda-sc
7630   depmod              2   0 /lib/modules/5.15.0-130-generic/modules.builtin.bin
```

二 代码文件说明

3.biosnoop.bt

主要功能是追踪块设备的 I/O 操作，记录每个 I/O 请求的开始时间和结束时间，并计算其延迟（以毫秒为单位）。

代码说明如下：

（1）打印表头，用于显示追踪结果的列标题。

（2）追踪开始：当 I/O 操作开始时，记录当前时间、进程 ID、进程名和相关磁盘名称。

（3）追踪结束：当 I/O 操作结束时，计算延迟，打印详细信息，并清理记录的数据。

（4）结束清理：在脚本结束时，清除所有关联数组。

二 代码文件说明

biosnoop.bt生成的log文件示例：

```
1 Attaching 6 probes...
2 cannot attach kprobe, probe entry may not exist
3 Warning: could not attach probe kprobe:__blk_account_io_done, skipping.
4 cannot attach kprobe, probe entry may not exist
5 Warning: could not attach probe kprobe:__blk_account_io_start, skipping.
6 TIME(ms)     COMM             DISK    PID    LAT(ms)
7 428          NetworkManager   sda     794          2
8 429          NetworkManager   sda     794          0
9 430          jbd2/sda5-8      sda     343          1
10 431         jbd2/sda5-8      sda     343          1
11 431         jbd2/sda5-8      sda     343          0
12 958         kworker/u256:3   sr0     5562         0
13 1858        snapd            sda     809         13
14 1860        snapd            sda     809          1
15 1860        snapd            sda     809          0
16 1861        snapd            sda     809          1
17 1862        snapd            sda     809          0
18 1864        snapd            sda     809          1
19 1864        snapd            sda     809          1
20 1864        snapd            sda     809          2
```

二 代码文件说明

4. run_bpftrace

代码的自动运行脚本脚本，可定义脚本路径，实现框架的灵活拓展。

```
# 定义bpftrace脚本路径，可选opensnoop.bt/runqlen.bt/biosnoop.bt
BPFTRACE_SCRIPT="./opensnoop.bt"
```

二 代码文件说明

5. runqlen.py

读取log文件，并使用逻辑回归对数据进行预测。该文件读取的数据可以归结为一个二分类问题，因此采用逻辑回归算法检测数据是否异常。逻辑回归是一个线性模型，容易理解和实现。并且该算法对于较小的特征空间和线性可分的数据集，训练和预测速度较快。代码通过sklearn使用了这一算法。输出结果为0/1，其中0为正常，1为异常。

```
Model accuracy on training set: 0.94
Predictions for new samples: [0]
```

```
Model accuracy on training set: 0.94
Predictions for new samples: [1]
```

二 代码文件说明

6. opensnoop. py

读取log文件，将数据向量化，并使用随机森林对数据进行预测。该文件读取的数据有多个特征，随机森林算法可以处理具有很多特征的数据的数据，并且不用降维，无需做特征选择，判断特征的重要程度，判断出不同特征之间的相互影响，并且实现起来比较简单。代码通过sklearn使用了这一算法。输出结果为normal/abnormal，其中normal为正常，abnormal为异常。

```
cooler@ubuntu:~/Desktop/bpf_tools$ python3 opensnoop.py
Model Accuracy: 92.31%
Predicted Behavior for Example Process: Normal
```

三 使用说明

- 打开一个已安装好bpftrace的容器：其中，ebpf-for-mac是已安装好bpftrace的容器，启动该容器，进入到含有bpftrace程序运行脚本的文件夹中。

```
cooler@ubuntu:~/Desktop/bpf_tools$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS     NAMES
0a15fb516985   ebpf-for-mac   "/bin/sh -c 'mount -…"   43 hours ago   Up 23 hours             ebpf-for-mac
cooler@ubuntu:~/Desktop/bpf_tools$ sudo docker exec -it 0a15fb516985 /bin/bash
```

- 运行执行脚本run_bpftrace.sh

```
root@0a15fb516985:~/bpf_tools# ./run_bpftrace.sh
bpftrace has been terminated and output saved to bt.log
```

- 输出日志，使用python利用容器输出的日志文件bt.log，在主机中进行预测。

```
cooler@ubuntu:~/Desktop/bpf_tools$ python3 runqlen.py
```

四 可改进的地方

- 使用容器以及主机的cron服务来进行定时收集数据与预测。
- 使用网络协议（如SSH）来对容器进行远程控制和输出。
- 利用其它BPF技术来对容器进行非侵入式的检测。

Thank You for Listening!