openEuler

BPF MAP介绍

目录

CONTENT

01 MAP介绍
创建，使用

02 生命周期
持久化，ref管理

03 MAP类型
常见类型

04 添加新的MAP类型
标题内容

openEuler

BPF MAP介绍

BPF Map是什么

- BPF Map本质上是驻留在内核中的以键/值方式存储的数据结构，它们可以被任何知道它们的BPF程序访问。
- 在用户空间运行的程序也可以通过使用文件描述符（File Descriptors）来访问BPF Map。

- Map可以用来bpf prog之间，bpf prog和用户态直接数据交互

openEuler

BPF MAP介绍

MAP创建

- 两种方式
  - 显示调用bpf(BPF_MAP_CREATE, ...);
  - 加载器帮忙调用(libbpf)

```
tools/bpf/bpftool/prog.c
int load_with_options(int argc, char **argv, bool first_prog_only)
    obj = bpf_object__open_file(file, &open_opts);
        __bpf_object__open(path, NULL, 0, opts)
            bpf_object__elf_collect(obj)
                obj->efile.maps_shndx = idx;
                obj->efile.btf_maps_shndx = idx;
            bpf_object__init_maps(obj, opts);
    bpf_object__load_xattr(&load_attr);
        err = err ? : bpf_object__create_maps(obj);
            bpf_object__create_map(obj, map, false);
                map->fd = bpf_create_map_xattr(&create_attr);
                    fd = sys_bpf(BPF_MAP_CREATE, &attr, sizeof(attr));
kernel/bpf/syscall.c
int map_create(union bpf_attr *attr)
    map = find_and_alloc_map(attr);
        map = ops->map_alloc(attr);
    atomic64_set(&map->refcnt, 1);
    atomic64_set(&map->usercnt, 1);
    bpf_map_alloc_id(map);
        id = idr_alloc_cyclic(&map_idr, map, 1, INT_MAX, GFP_ATOMIC);
    bpf_map_new_fd(map, f_flags);
        anon_inode_getfd("bpf-map", &bpf_map_fops, map, flags | O_CLOEXEC);
```

BPF MAP介绍

MAP创建

- Strace bpftool prog load xxx

```
bpf(BPF_MAP_CREATE, {map_type=BPF_MAP_TYPE_ARRAY, key_size=4, value_size=8, max_entries=10, map_flags=0, inner_map_fd=0, map_n
ame="my_map", map_ifindex=0, btf_fd=0, btf_key_type_id=0, btf_value_type_id=0, btf_vmlinux_value_type_id=0}, 112) = 3
mmap(NULL, 16781312, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9dde02d000
bpf(BPF_PROG_LOAD, {prog_type=BPF_PROG_TYPE_CGROUP_SOCK, insn_cnt=56, insns=0x72fe90, license="GPL", log_level=0, log_size=0,
log_buf=NULL, kern_version=KERNEL_VERSION(0, 0, 1), prog_flags=0, prog_name="sock_release", prog_ifindex=0, expected_attach_ty
pe=BPF_TRACE_FEXIT, prog_btf_fd=0, func_info_rec_size=0, func_info=NULL, func_info_cnt=0, line_info_rec_size=0, line_info=NULL
, line_info_cnt=0, attach_btf_id=0}, 112) = 4
munmap(0x7f9dde02d000, 16781312)        = 0
statfs("/sys/fs/bpf", {f_type=BPF_FS_MAGIC, f_bsize=4096, f_blocks=0, f_bfree=0, f_bavail=0, f_files=0, f_ffree=0, f_fsid={val
=[0, 0]}, f_namelen=255, f_frsize=4096, f_flags=ST_VALID|ST_RELATIME}) = 0
bpf(BPF_OBJ_PIN, {pathname="/sys/fs/bpf/bpf_sock_release", bpf_fd=4, file_flags=0}, 112) = 0
close(3)                                = 0
close(4)                                = 0
```

- 查看BPF Map
  - bpftool map
- Map Id 全局唯一

openEuler

BPF MAP介绍

MAP使用

- 用户态程序
  - int bpf_map_update_elem(int fd, const void *key, const void *value, __u64 flags);
    - sys_bpf(BPF_MAP_UPDATE_ELEM, ...);
  - int bpf_map_lookup_elem(int fd, const void *key, void *value);
    - sys_bpf(BPF_MAP_LOOKUP_ELEM, ...);
  - int bpf_map_delete_elem(int fd, const void *key);
    - sys_bpf(BPF_MAP_DELETE_ELEM, ...);
  - int bpf_map_get_next_key(int fd, const void *key, void *next_key)
    - sys_bpf(BPF_MAP_GET_NEXT_KEY, ...);
  - int bpf_map_freeze(int fd)
    - sys_bpf(BPF_MAP_FREEZE, ...);
- Bpf程序 (helper function)
  - void *bpf_map_lookup_elem(struct bpf_map *map, const void *key)
  - long bpf_map_update_elem(struct bpf_map *map, const void *key, const void *value, u64 flags)
  - long bpf_map_delete_elem(struct bpf_map *map, const void *key)

生命周期

持久化

- Bpffs
  - https://lore.kernel.org/patchwork/project/lkml/list/?series=252972&state=%2A&archive=both

```
[root@localhost ~]# mount | grep bpf
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
[root@localhost ~]# ls /sys/fs/bpf/
bpf_test
```

- eBPF map 和 eBPF program 可以 pin（固定）到这个文件系统，这个过程称为 object pinning。
- 新增如下系统调用
  - int bpf_obj_pin(int fd, const char *pathname)
    - sys_bpf(BPF_OBJ_PIN, ...);
  - int bpf_obj_get(const char *pathname)
    - fd = sys_bpf(BPF_OBJ_GET, ...);
- Bpftool
  - bpftool prog help
  - bpftool map help
  - bpftool map pin MAP FILE
  - bpftool prog pin PROG FILE
- 提供了不同bpf prog使用同一个map的方式

生命周期

Ref管理

- Bpf map的生命周期通过refcnt管理

```
创建map时，置 1
int map_create(union bpf_attr *attr)
    atomic64_set(&map->refcnt, 1);
close mapfd时，  减 1
int bpf_map_release(struct inode *inode, struct file *filp)
    bpf_map_put_with_uref(map);
加载prog，校验时，加 1
int bpf_prog_load(union bpf_attr *attr, bpfptr_t uattr)
    bpf_check(&prog, attr, uattr);
        resolve_pseudo_ldimm64(env);
            bpf_map_inc(map);
prog free时，减 1
void bpf_prog_free(struct bpf_prog *fp)
    INIT_WORK(&aux->work, bpf_prog_free_deferred);
        bpf_free_used_maps(aux);
            __bpf_free_used_maps(aux, aux->used_maps, aux->used_map_cnt);
                bpf_map_put(map);
pin时，加 1
int bpf_obj_pin(const union bpf_attr *attr)
    bpf_obj_pin_user(attr->bpf_fd, u64_to_user_ptr(attr->pathname));
        bpf_fd_probe_obj(ufd, &type);
            bpf_map_get_with_uref(ufd);
                bpf_map_inc_with_uref(map);
            3
get pin时，加 1 (对应close，  -1)
int bpf_obj_get(const union bpf_attr *attr)
    bpf_obj_get_user(u64_to_user_ptr(attr->pathname), attr->file_flags);
        bpf_obj_do_get(pathname, &type, f_flags);
            bpf_any_get(inode->i_private, *type);
                bpf_map_inc_with_uref(raw);
unpin时，减 1
void bpf_free_inode(struct inode *inode)
    bpf_any_put(inode->i_private, type);
        bpf_map_put_with_uref(raw);
```

openEuler

生命周期

Ref管理

- prog对map的引用

```
struct bpf_map_def SEC("maps") my_map = {
        .type = BPF_MAP_TYPE_ARRAY,
        .key_size = sizeof(int),
        .value_size = sizeof(long),
        .max_entries = 10,
};

__section("cgroup/sock_release")
int sock_release(struct bpf_sock *sk)
{
        int key = 0;
        long *p_value;
        long value= 0;

        if (sk->type != SOCK_STREAM)
                return 1;

        bpf_printk("---->protocol:%d, state:%d\n", sk->protocol

#if 0
        p_value = bpf_map_lookup_elem(&my_map, &key);
        if (p_value == NULL) {
                bpf_printk("can not get key0\n");
        } else {
                value = *p_value;
        }

        value++;

        bpf_map_update_elem(&my_map, &key, &value, BPF_ANY);
#endif
        return 1;
}
```

```
[root@localhost bpf_test]# bpftool prog
81: cgroup_sock  name sock_release  tag d4d6fdc31f5e5fc9  gpl
        loaded_at 2021-07-18T01:07:26+0800  uid 0
        xlated 200B  jited 172B  memlock 4096B
[root@localhost bpf_test]# bpftool map
```

```
[root@localhost bpf_test]# bpftool prog
84: cgroup_sock  name sock_release  tag d9d0636e21860ab0  gpl
        loaded_at 2021-07-18T01:08:23+0800  uid 0
        xlated 528B  jited 342B  memlock 4096B  map_ids 144
[root@localhost bpf_test]# bpftool map
144: array  name my_map  flags 0x0
        key 4B  value 8B  max_entries 10  memlock 4096B
```

openEuler

MAP类型

- enum bpf_map_type {
- BPF_MAP_TYPE_UNSPEC,
- BPF_MAP_TYPE_HASH,
- BPF_MAP_TYPE_ARRAY,
- BPF_MAP_TYPE_PROG_ARRAY,
- BPF_MAP_TYPE_PERF_EVENT_ARRAY,
- BPF_MAP_TYPE_PERCPU_HASH,
- BPF_MAP_TYPE_PERCPU_ARRAY,
- BPF_MAP_TYPE_STACK_TRACE,
- BPF_MAP_TYPE_CGROUP_ARRAY,
- BPF_MAP_TYPE_LRU_HASH,
- BPF_MAP_TYPE_LRU_PERCPU_HASH,
- BPF_MAP_TYPE_LPM_TRIE,
- BPF_MAP_TYPE_ARRAY_OF_MAPS,
- BPF_MAP_TYPE_HASH_OF_MAPS,
- BPF_MAP_TYPE_DEVMAP,
- BPF_MAP_TYPE_SOCKMAP,
- BPF_MAP_TYPE_CPUMAP,
- BPF_MAP_TYPE_XSKMAP,
- BPF_MAP_TYPE_SOCKHASH,
- BPF_MAP_TYPE_CGROUP_STORAGE,
- BPF_MAP_TYPE_REUSEPORT_SOCKARRAY,
- BPF_MAP_TYPE_PERCPU_CGROUP_STORAGE,
- BPF_MAP_TYPE_QUEUE,
- BPF_MAP_TYPE_STACK,
- BPF_MAP_TYPE_SK_STORAGE,
- BPF_MAP_TYPE_DEVMAP_HASH,
- BPF_MAP_TYPE_STRUCT_OPS,
- BPF_MAP_TYPE_RINGBUF,
- BPF_MAP_TYPE_INODE_STORAGE,
- BPF_MAP_TYPE_TASK_STORAGE,
- };

- 5.13

openEuler

MAP类型

Array Maps

- 所有数组key为 4 字节，并且不支持删除值
- BPF_MAP_TYPE_ARRAY
  - 简单数组。Key 是数组索引，不能删除元素。
- BPF_MAP_TYPE_PERCPU_ARRAY
  - 同上，precpu
- BPF_MAP_TYPE_PROG_ARRAY
  - bpf_tail_call()用作跳转表的BPF程序数组， samples/bpf/sockex3_kern.c
- BPF_MAP_TYPE_PERF_EVENT_ARRAY
  - 内核在 bpf_perf_event_output() 中使用的数组映射，用于将跟踪输出与特定键相关联。用户空间程序将 fds 与每个键关联，并且可以 poll() 这些 fds 以接收数据已被跟踪的通知
- BPF_MAP_TYPE_ARRAY_OF_MAPS
  - Map in map， 外层map的vaule是内层map的fd，samples/bpf/test_map_in_map_kern.c

MAP类型

Hash Maps

- 通过key的hash索引查找。与数组情况不同，可以从hash map中删除值
- BPF_MAP_TYPE_HASH：
  - 简单的哈希映射
- BPF_MAP_TYPE_PERCPU_HASH
  - 同上， percpu
- BPF_MAP_TYPE_LRU_HASH
  - 每个hash为每个bucket维护一个LRU（最近最少使用）列表，当hashbucket填满时通知删除。
- BPF_MAP_TYPE_HASH_OF_MAPS
  - 类似ARRAY_OF_MAPS

MAP类型

其它Maps

- BPF_MAP_TYPE_STACK_TRACE
  - 内核程序可以通过 bpf_get_stackid() 帮助程序存储堆栈
- BPF_MAP_TYPE_LPM_TRIE
  - 最长前缀匹配，例如，用于存储/检索 IP 路由
- BPF_MAP_TYPE_SOCKMAP
  - sockmaps 主要用于套接字重定向
- BPF_MAP_TYPE_DEVMAP
  - 与 sockmap 做类似的工作，使用 XDP 的 netdevices 和 bpf_redirect()

MAP类型

Map operation definitions

- include/linux/bpf_types.h
  - BPF_MAP_TYPE(BPF_MAP_TYPE_ARRAY, array_map_ops)
  - BPF_MAP_TYPE(BPF_MAP_TYPE_PERCPU_ARRAY, percpu_array_map_ops)
  - …

- 使用参考samples/bpf/

添加新的MAP类型

- 在enum bpf_map_type添加类型
  - enum bpf_map_type {}
    - include/uapi/linux/bpf.h
- 注册ops
  - BPF_MAP_TYPE(BPF_MAP_TYPE_XXX, yyy_ops)
    - include/linux/bpf_types.h
  - BPF_MAP_TYPE宏，会添加这个type到全局bpf_map_types数组中
- 合适的位置实现struct bpf_map_ops yyy_ops
- 允许的helper函数
  - 在check_map_func_compatibility中添加兼容的helper函数

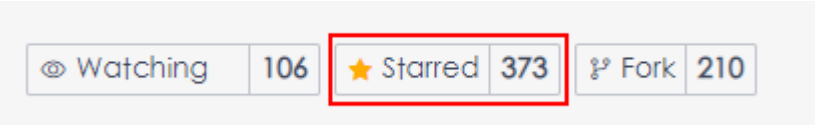
openEuler

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

openEuler

Thank you

openEuler