TRACE_EVENT 宏

功能	链接
宏的背景及实现	https://lwn.net/Articles/379903/
DECLARE_EVENT_CLASS 宏节省空间	https://lwn.net/Articles/381064/
在核心内核之外添加跟踪点	https://lwn.net/Articles/383362/

BPF SCHED 特性

https://lore.kernel.org/all/20210916162451.709260-1-guro@fb.com/

一、https://lore.kernel.org/all/20210916162451.709260-2-guro@fb.com/

引入调度BPF程序基本定义和设施，定义 BPF_PROG_TYPE_SCHED 程序类型和 BPF_SCHED 附加类型

二、https://lore.kernel.org/all/20210916162451.709260-3-guro@fb.com/
添加三个帮助函数处理调度实例（sched entities）

```
u64 bpf_sched_entity_to_tgidpid(struct sched_entity *se)
```

功能：如果调度实体是任务，则返回任务编码的 tgid 和 pid

Return： 如果 **se** 是一个任务，tgid 和 pid 编码为 gid << 32 | pid 。其他情况返回 (u64) - 1。

```
u64 bpf_sched_entity_to_cgrpid(struct sched_entity *se)
```

功能：如果调度实体是 cgroup，则返回 cgroup id

Return： 如果 **se** 是 cgroup，返回 cgroup id，其他情况返回 (u64) - 1。

```
long bpf_sched_entity_belongs_to_cgrp(struct sched_entity *se, u64 cgrpid)
```

功能：检查是否调度实例属于 cgroup 或它的 sub-tree。它不需要使能 控制组（cgroup）CPU 控制器

Return： 如果调度实例属于 cgroup 则返回 1，否则返回 0。

三、https://lore.kernel.org/all/20210916162451.709260-4-guro@fb.com/

引入专用静态键和 bpf_schd_enabled() 包装器保护调度器代码中对BPF程序的所有调用。在没有调度器 bpf programs attached 的情况下，有助于避免任何潜在的性能退化。

四、https://lore.kernel.org/all/20210916162451.709260-5-guro@fb.com/

添加了三个补丁来控制唤醒和时钟抢占

- cfs_check_preempt_tick

第一个钩子允许在时钟中断上下文中强制或抑制抢占。一个明显的使用示例是通过（有条件的）为任务或任务组提供延长的执行片段，来最小化非自愿上下文切换的次数并降低相关的延迟惩罚。它可以用来替代调整 sysctl_sched_min_granularity 的方式。

hook 返回负数避免发生抢占，返回正数通知调度器应该把它切换到另一进程，返回 0 表示由调度器给管理。

sysctl_sched_min_granularity：在 check_prempt_tick 函数中，sysctl_sched_min_granularity 可用来表示：delta_exec <= ideal_runtime && delta_exec > sysctl_sched_min_granularity成立，即一个调度实体运行时间未达到理论运行时间且运行时间大于sysctl_sched_min_granularity，那么这个任务就可以被认为是可以被抢占的，这种设置的目的是为了确保那些未能及时抢占的任务不会被长时间地推迟执行。https://juejin.cn/post/7054069979615854599

- cfs_check_preempt_wakeup

第二个钩子在唤醒抢占代码中调用，允许重新定义新唤醒任务是否应该抢占当前执行的任务。这对于延迟敏感任务的抢占次数非常有用。它比 sysctl_sched_wakeup_granularit 更灵活。

hook 返回负数表示阻止这个进程抢占当前运行的进程，返回正数会强制抢占，返回 0 则由调度器自己决定。

sysctl_sched_wakeup_granularit： 在唤醒新任务时，只有在当前任务 A 的虚拟时间比被唤醒任务 B 的虚拟时间多于sysctl_sched_wakeup_granularity 参数的加权平均值时才会考虑让新唤醒任务 B 抢占当前任务 A。避免频繁的切换导致大量上下文的开销。https://blog.csdn.net/wennuanddianbo/article/details/118345922

- cfs_wakup_preempt_entity

第三个钩子类似于第二个，但它调整了 wakeup_preempt_entity() 函数，该函数不仅从唤醒上下文调用，还从 `pick_next_task()` 中调用，这允许影响决定下一个运行的任务是哪一个任务。

hook 返回负数表示没有抢占，返回正数会强制抢占，返回 0 则由调度器的其他决策部分决定。

五、https://lore.kernel.org/all/20210916162451.709260-7-guro@fb.com/

让 bpftool 工具可以对调度器 bpf programs 进行识别

三、https://lore.kernel.org/all/20210916162451.709260-4-guro@fb.com/