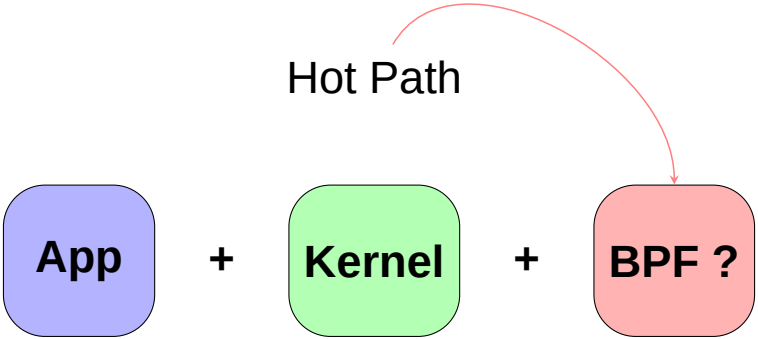
Performance Insights into eBPF Step by Step

Dmitrii Dolgov
Senior Software Engineer

12-09-2022

App + Kernel


Hot Path
App
+
Kernel
+
BPF ?

Current state of things

BPF Instruction Set

```
if (delta < ts)
```

```
; if (delta < ts)
  cmp    %rsi,%rdi
  jae    0x0000000000000068
```

```
; if (delta < ts)
  cmp    %rdi,%rsi
  jbe    0x0000000000000065
```

eBPF Instruction Sets

Map batch operations

```
BPF_MAP_LOOKUP_BATCH
BPF_MAP_LOOKUP_AND_DELETE_BATCH
BPF_MAP_UPDATE_BATCH
BPF_MAP_DELETE_BATCH
```

Bloom filter map

```
bpf_map_create(
    BPF_MAP_TYPE_BLOOM_FILTER, NULL,
    0, sizeof(value), 100, NULL);
```

BPF program pack allocator

```
struct bpf_prog_pack {
    struct list_head list;
    void *ptr;
    unsigned long bitmap[];
};
```

Task local storage

```
ptr = bpf_task_storage_get(
        &start, t, 0,
        BPF_LOCAL_STORAGE_GET_F_CREATE);
```

Which perf analysis methods could work for BPF?

Talk to compiler

```
-Rpass=.*
-Rpass-analysis=.*
-Rpass-missed=.*

remark: load of type i32
not eliminated [-Rpass-missed=gvn]
```

Talk to compiler

```
static __always_inline
int bpf_example_fn(void * restrict ctx)
```

Global kernel stats / Uptime

```
$ sysctl -w kernel.bpf_stats_enabled=1
$ bpftool prog

379: raw_tracepoint [ ... ]
run_time_ns 35875602162 run_cnt 160512637
```

Printk / manual instrumentation

```
// somewhere inside your BPF prog
bpf_trace_printk("Timestamp: %lld", ts);

$ cat /sys/kernel/debug/tracing/trace_pipe
$ bpftool prog tracelog
```

fentry & fexit / Topdown?

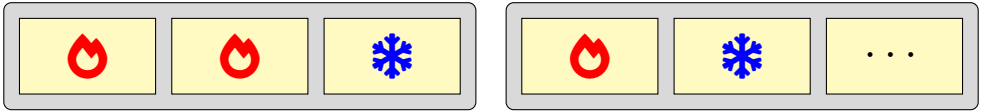
```
$ perf stat -b 5 --topdown
```

```
SEC("fentry/XXX")
int BPF_PROG(fentry_XXX)
{
    // ...
}
```

```
SEC("fexit/XXX")
int BPF_PROG(fentry_XXX)
{
    // ...
}
```

BPF program pack allocator

```
struct bpf_prog_pack {
    struct list_head list;
    void *ptr;
    unsigned long bitmap[];
};
```



Profiling

```
Percent | uops_retired.stall_cycles
        :
        : if (duration_ns < min_duration_ns)
   0.00 :    9f:movabs $0xffffc9000009e000,%rdi
   0.00 :    a9:mov    0x0(%rdi),%rsi
        :
        : e = bpf_ringbuf_reserve( ... )
  21.74 :    ad:movabs $0xffff888103e70e00,%rdi
   0.00 :    b7:mov    $0xa8,%esi
   0.00 :    bc:xor    %edx,%edx
   0.00 :    be:callq  0xffffffffc0f9fbb8
```

Profiling

```
$ perf record -e intel_pt// \
    --filter 'filter bpf_prog_9baac7ecffdb457d'

$ perf record -e intel_pt// \
    --filter 'start 0xffffffffc1612d64'
```

Modeling?

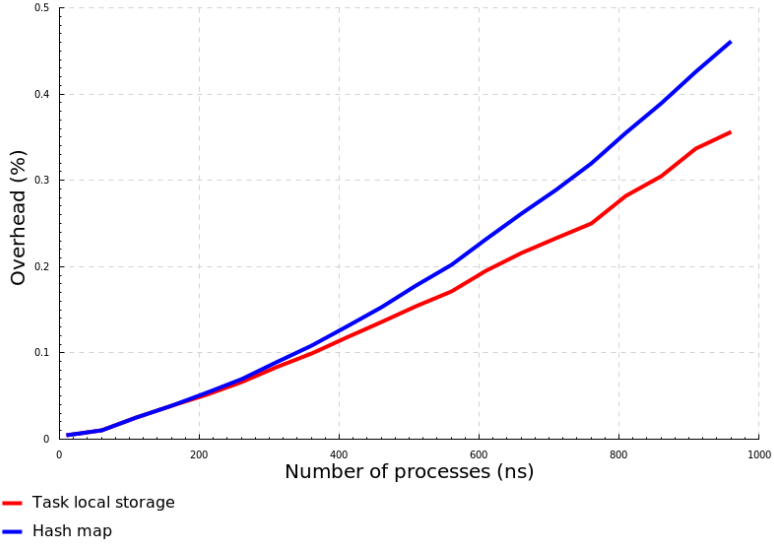
Modeling

```
bpf: runqslower: Use task local storage

Replace hashtab with task local storage in runqslower.
This improves the performance of these BPF programs.
The following table summarizes average runtime of
these programs, in nanoseconds:

                 task-local hash-prealloc hash
sched_wakeup     125        340           3124
sched_wakeup_new 2812       1510          2998
sched_switch     151        208           991
```

Runqslower overhead simulation



Questions?

- @erthalion
- dmitrii.dolgov at redhat dot com