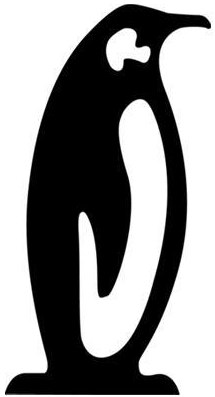


東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

Memory Allocation Profiling upcoming features

東京 2025
LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

New features being considered / in development

- Allocation context capture
- MEMCG support (two options)
- NUMA awareness
- Efficient data querying and filtering via IOCTL API
- Userspace tooling (grouping allocations by categories, issuing ioctl commands, etc)

Allocation context capture (testing)

Allocation tags get a new flag bit indicating context capture is enabled

New ioctl interface to set this flag for a specific allocation

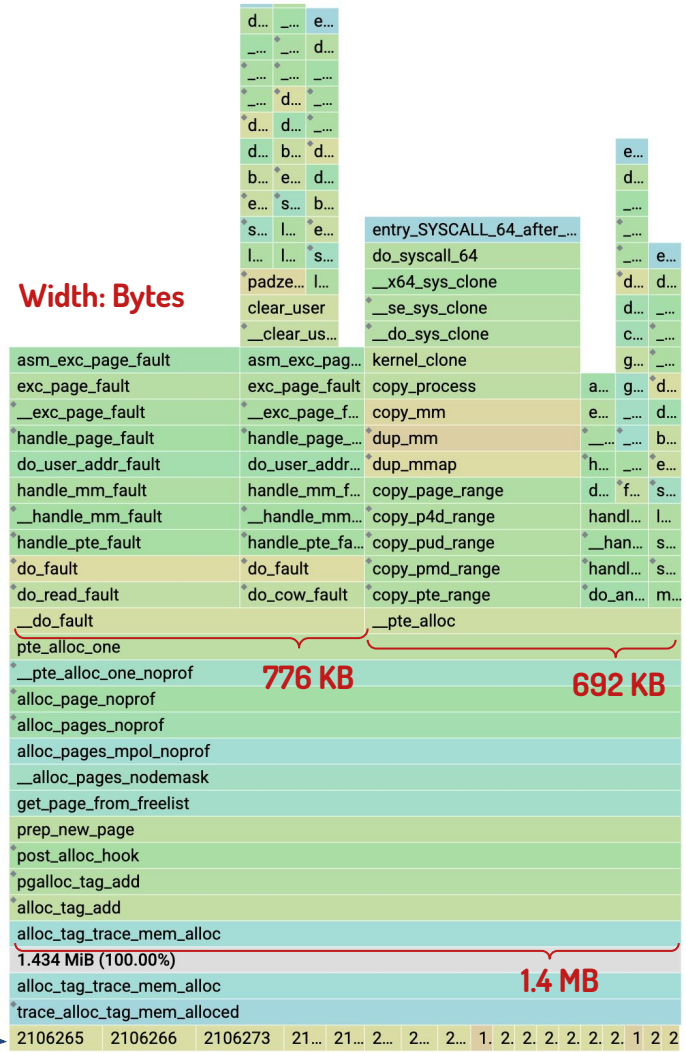
Allocation hooks are modified to check the flag and generate a trace event when it's set

BPF program can be attached to the tracepoing to capture:

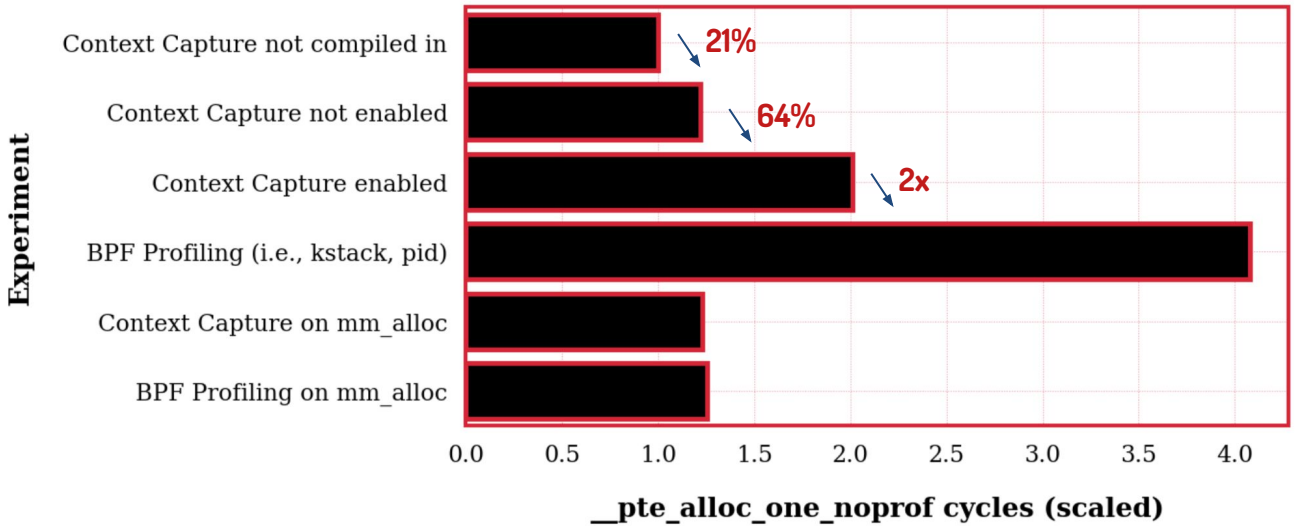
- pid of the caller
- timestamp
- callstack

BPF based context capture

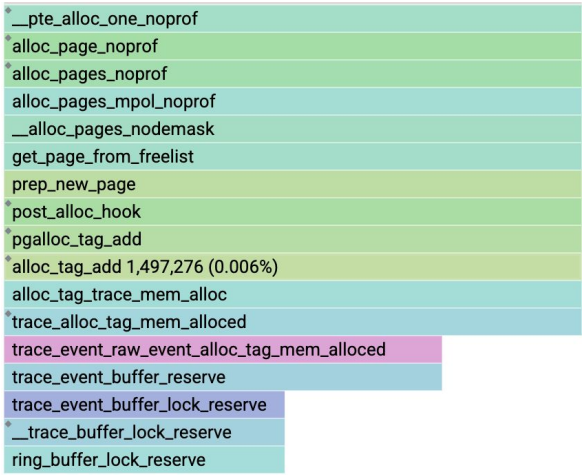
```
SEC("tp_btf/alloc_tag_mem_alloced")
int BPF_PROG( alloc_tag_mem_alloced, struct codetag_ref *ref,
                struct alloc_tag *tag, unsigned int bytes)
{
        __u32 kstackid = bpf_get_stackid(ctx, &kernel_stacks, 0);
        __u32 pid = bpf_get_current_pid_tgid() & 0xFFFFFFFF;

        const AllocTagKey key = {.kstackid = kstackid, .pid = pid};
        Counters *valp = bpf_map_lookup_elem((void *)&alloc_tag_map, &key);
        if (valp != NULL) {
                __sync_fetch_and_add(&valp->bytes, bytes);
        }
        return 0;
}
```



Profiling Overhead

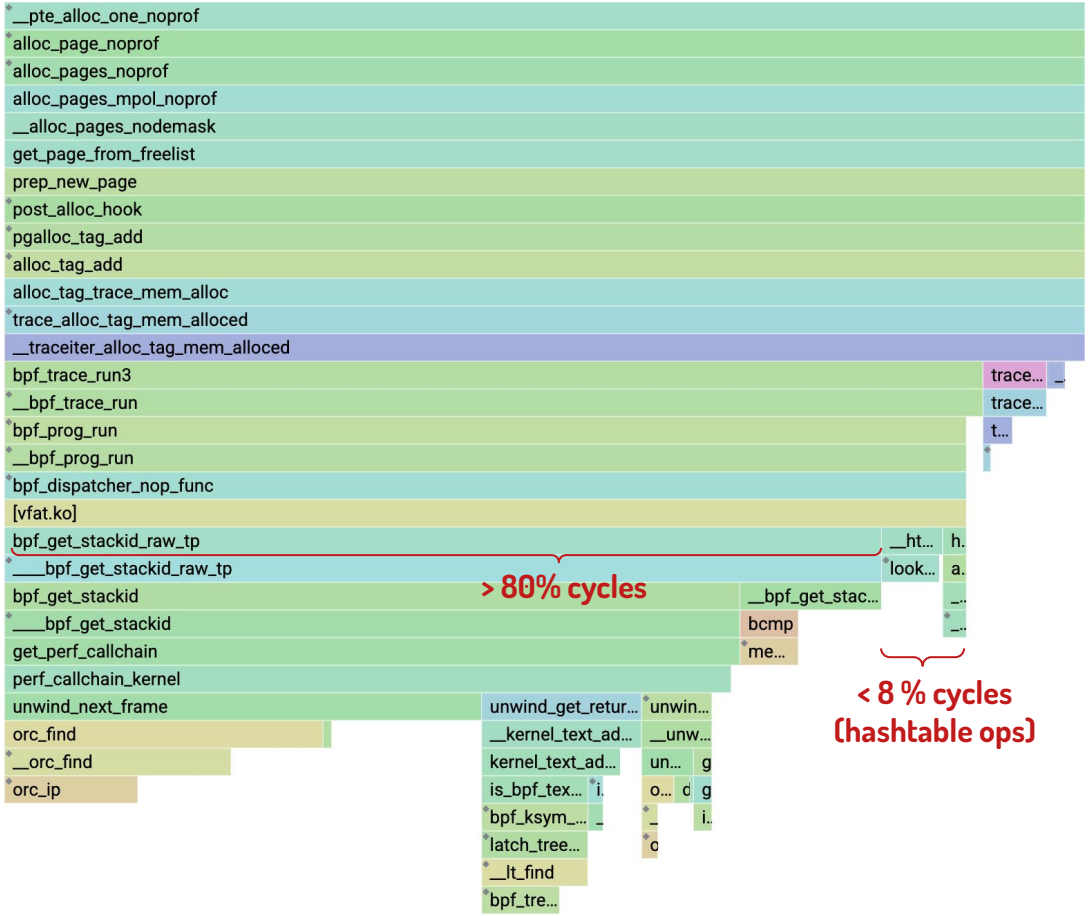


Context Capture enabled



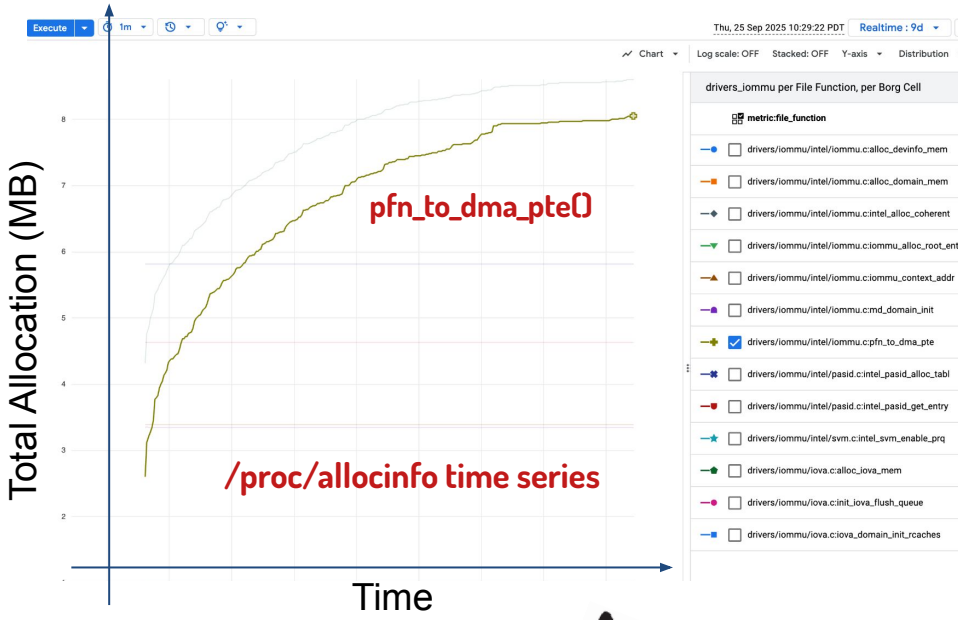
2x

BPF Profiling (kstackid, pid)



Width: Cycles

Context Capture in action



Context Capture

- scsi_queue_rq
- scsi_dispatch_cmd
- ata_scsi_queuecmd
- __ata_scsi_queuecmd
- ata_scsi_translate
- ata_qc_issue
- ata_sg_setup
- dma_map_sg_attrs
- intel_map_sg
- domain_sg_mapping
- domain_mapping
- __domain_mapping
- pfn_to_dma_pte
- iommu_alloc_page_node_noprof
- iommu_alloc_pages_node_noprof
- __iommu_alloc_pages_node_noprof
- alloc_pages_node_noprof
- __alloc_pages_node_noprof
- __alloc_pages_noprof
- __alloc_pages_nodemask

`grep nr_iommu_pages /proc/vmstat` **is High!**

Context Capture

- scsi_queue_rq
- scsi_dispatch_cmd
- ata_scsi_queuecmd
- __ata_scsi_queuecmd
- ata_scsi_translate
- ata_qc_issue
- ata_sg_setup
- dma_map_sg_attrs
- intel_map_sg
- domain_sg_mapping
- domain_mapping
- __domain_mapping
- pfn_to_dma_pte
- iommu_alloc_page_node_noprof
- iommu_alloc_pages_node_noprof
- __iommu_alloc_pages_node_noprof
- alloc_pages_node_noprof
- __alloc_pages_node_noprof
- __alloc_pages_noprof
- __alloc_pages_nodemask

```
perf record -e intel_iommu:map_sg
        ❌ Cancels Out
perf record -e intel_iommu:unmap_sg

// No further investigation needed
```

trace_mm_page_free

- ata_sg_clean
- intel_unmap_sg
- intel_unmap
- queue_iova
- fq_ring_free
- iova_entry_free
- iommu_free_pages_list
- __iommu_free_page
- __iommu_free_pages
- free_unref_page
- free_unref_page_prepare
- free_pcp_prepare
- free_pages_prepare

Focus investigation here!

MEMCG support option 1: single group tracking

Pros:

Enabled for a single cgroup only

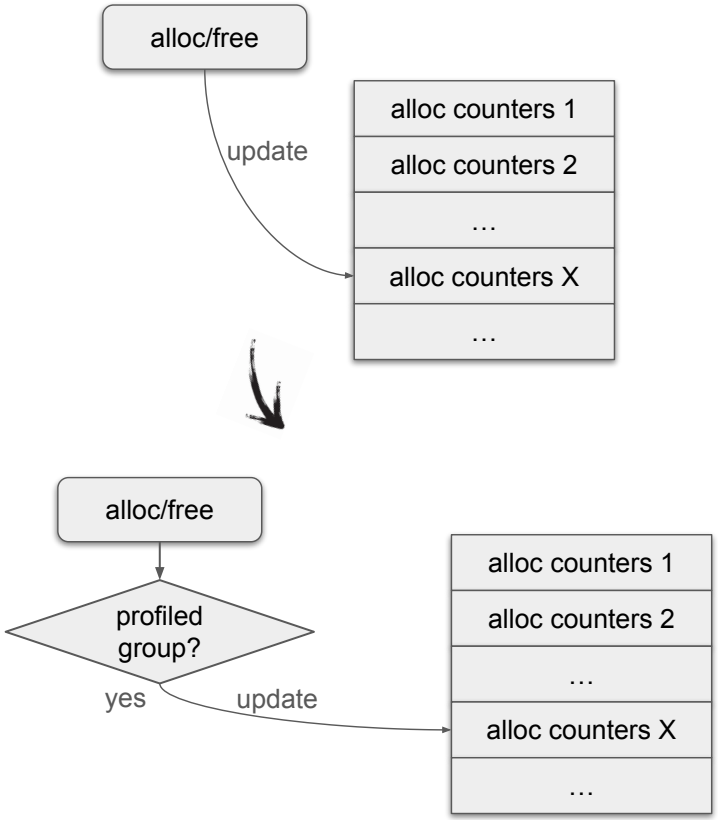
No memory and minimal performance overhead

Simple implementation

Cons:

Limited to a single cgroup at a time

No global profiling when cgroup is profiled



MEMCG support option 2: multi-group tracking

Pros:

Can be enabled for multiple cgroups

Global profiling is not affected

Cons:

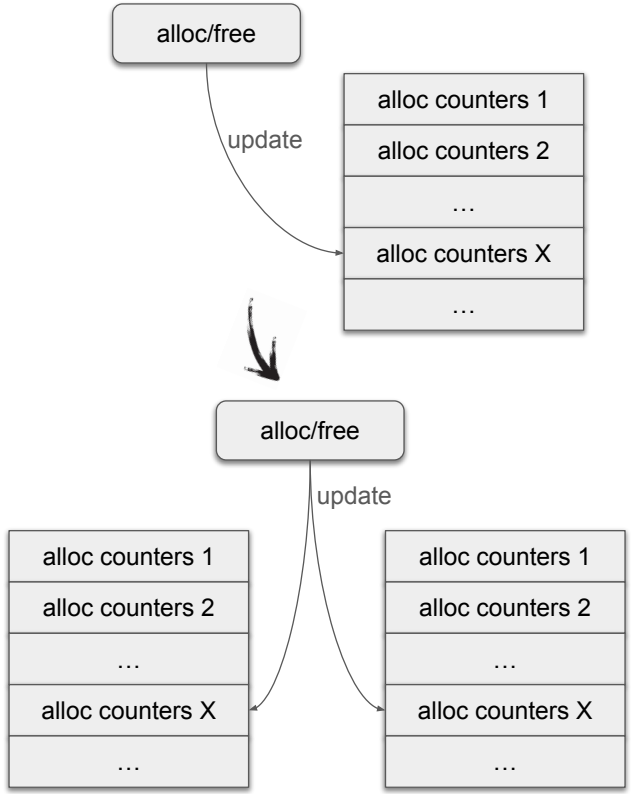
Memory overhead is linear to the number of profiled groups

(each selected memcg gets a new set of per-cpu counters)

Performance profiling overhead for profiled groups roughly doubles

(two sets of counters need to be updated)

Complex implementation



NUMA awareness (proposal)

Proposed at:

https://lore.kernel.org/all/20250711002322.1303421-1-cachen@purestorage.com/

Each tag gets per-NUMA node counter

Concerns:

- High memory overhead (linear to the number of nodes)
- Output format changes (unless queried via IOCTL)

Is this useful enough?

Efficient data querying and filtering via IOCTL API

Enabling context capture

Filtering

- by size
- by allocation type (page/slab/per-cpu)

Reporting extra allocation data

- gfp_flags
- unaccounted memory size (early boot or no extension to account)
- per-NUMA information

Userspace tooling (alloctop)

Supported basic features:

- Limiting the number or lines in the output
- Sorting by allocations size, call count, tag name
- Continuous polling

Planned features:

- Grouping allocations by categories (mm/net/drivers/etc.)
- Command-line support for IOCTL commands

Thank you!