Extending Flamegraphs for Multi-Dimensional Performance Analysis

Gavin Guo
gavinguo@igalia.com

December 11, 2025

igalia

Agenda

- Different flavors of FlameGraphs I've created
- Ongoing projects
  - Easy FlameGraph
  - ScatterPlot + FlameGraph
  - Kernel Callstack Analyzer MCP Server
- Discussion

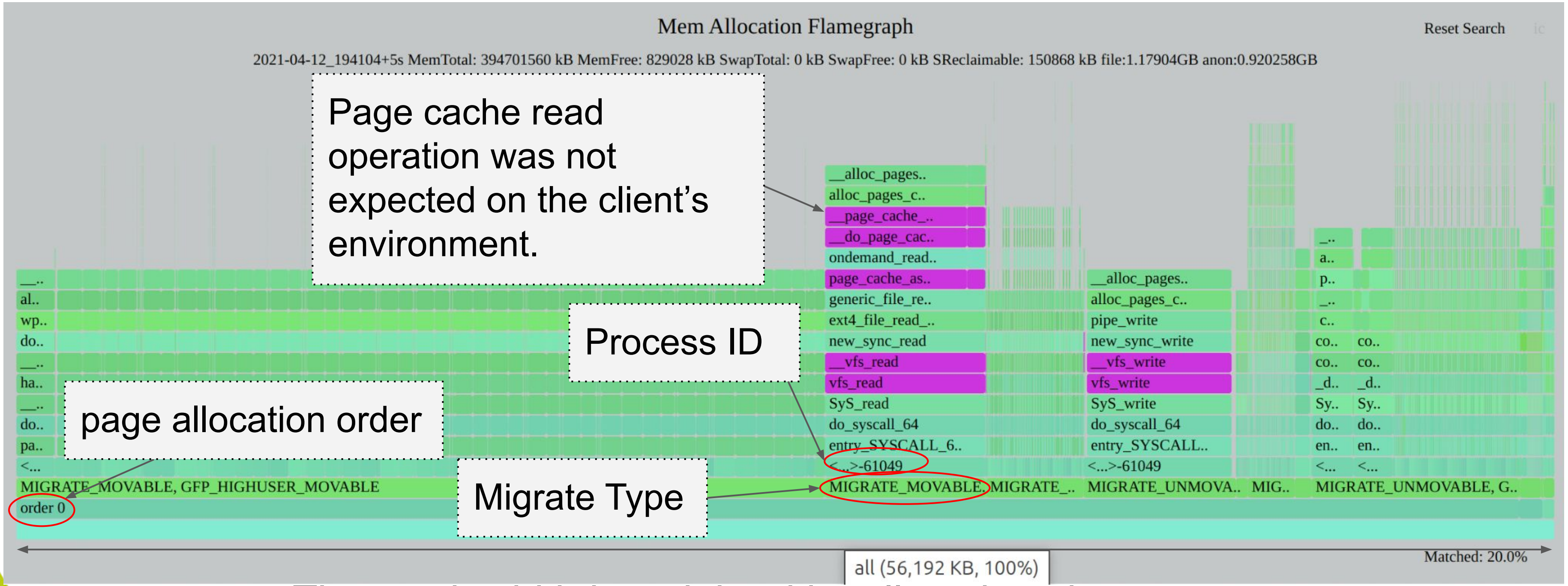
Different Flavors of FlameGraph

- I’ve built four non-CPU *FlameGraph* extensions that proved useful in production debugging, including
  - **Memory**—Who are allocating memory
  - **Latency**—Address the tail latency
  - **Kernel Log**—Categorize the panics
  - **Kdump**—Address the hung tasks
- How can we approach this to the community?

Memory FlameGraph: Unknown Memory Usage

- **Problem Statement and Scenario**
  - A client claims there is unknown memory usage observed.
  - Set up a continuous profiling to capture memory allocation.
  - Because allocation is on the hot path, always-on profiling adds significant overhead.
  - We need to shrink the profiling duration to alleviate the impact.
  - Profile 5 seconds per minute successfully captured the unknown memory usage.

Memory FlameGraph: Unknown Memory Usage



The stack width is weighted by allocation size, not sample count.

Memory FlameGraph: Unknown Memory Usage

- How to capture the memory allocation tracing: mem_ftrace.

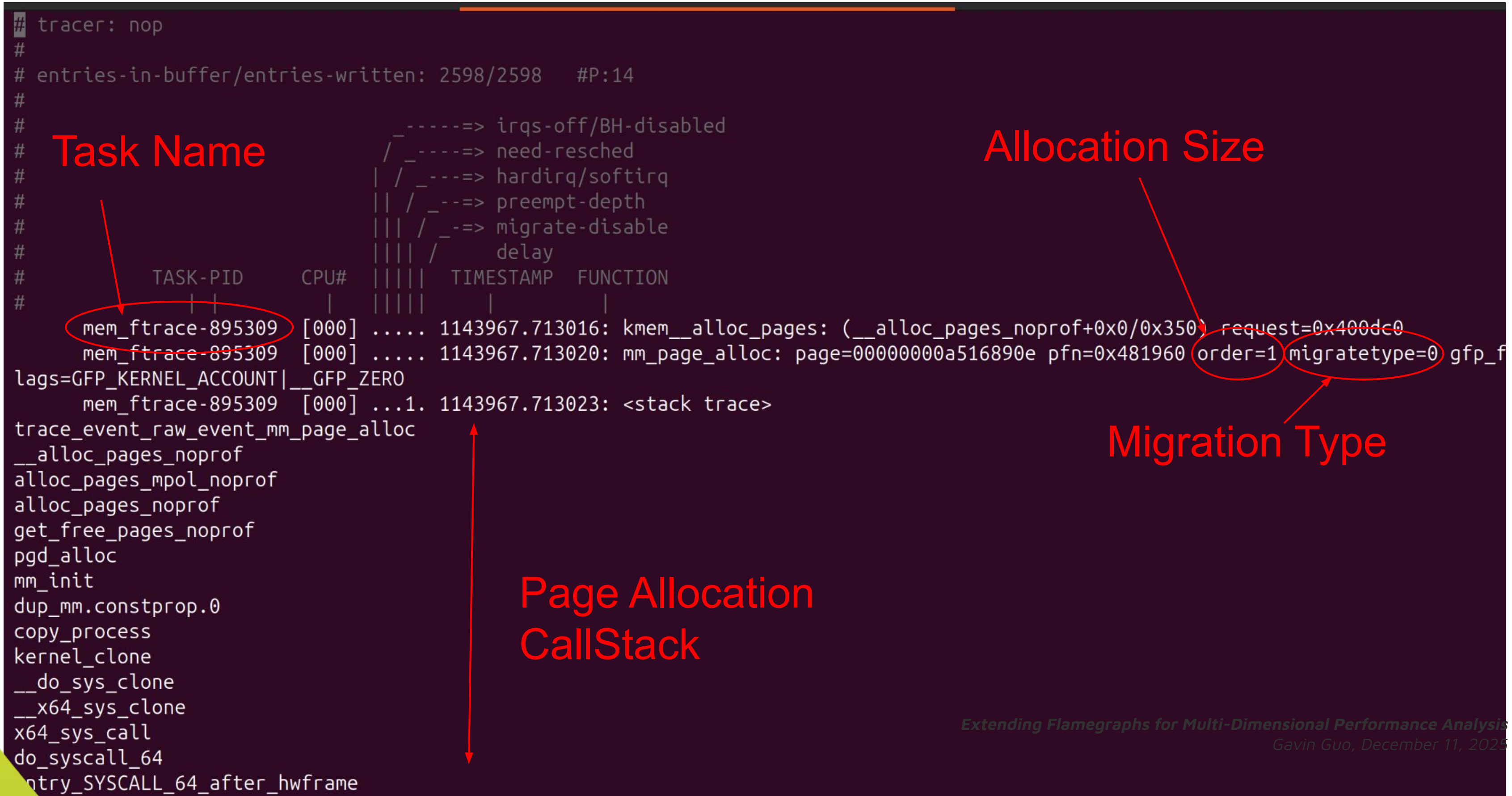
```
$ echo 'p:kmem__alloc_pages __alloc_pages_noprof request=%di' >>
/sys/kernel/debug/tracing/kprobe_events

$ echo 'kmem:mm_page_alloc' >> /sys/kernel/debug/tracing/set_event

$ echo stacktrace > /sys/kernel/debug/tracing/events/kmem/mm_page_alloc/trigger
```

- How to parse the log to flamegraph input format: stackcollapse-tracecmd.pl.

Memory FlameGraph: Unknown Memory Usage

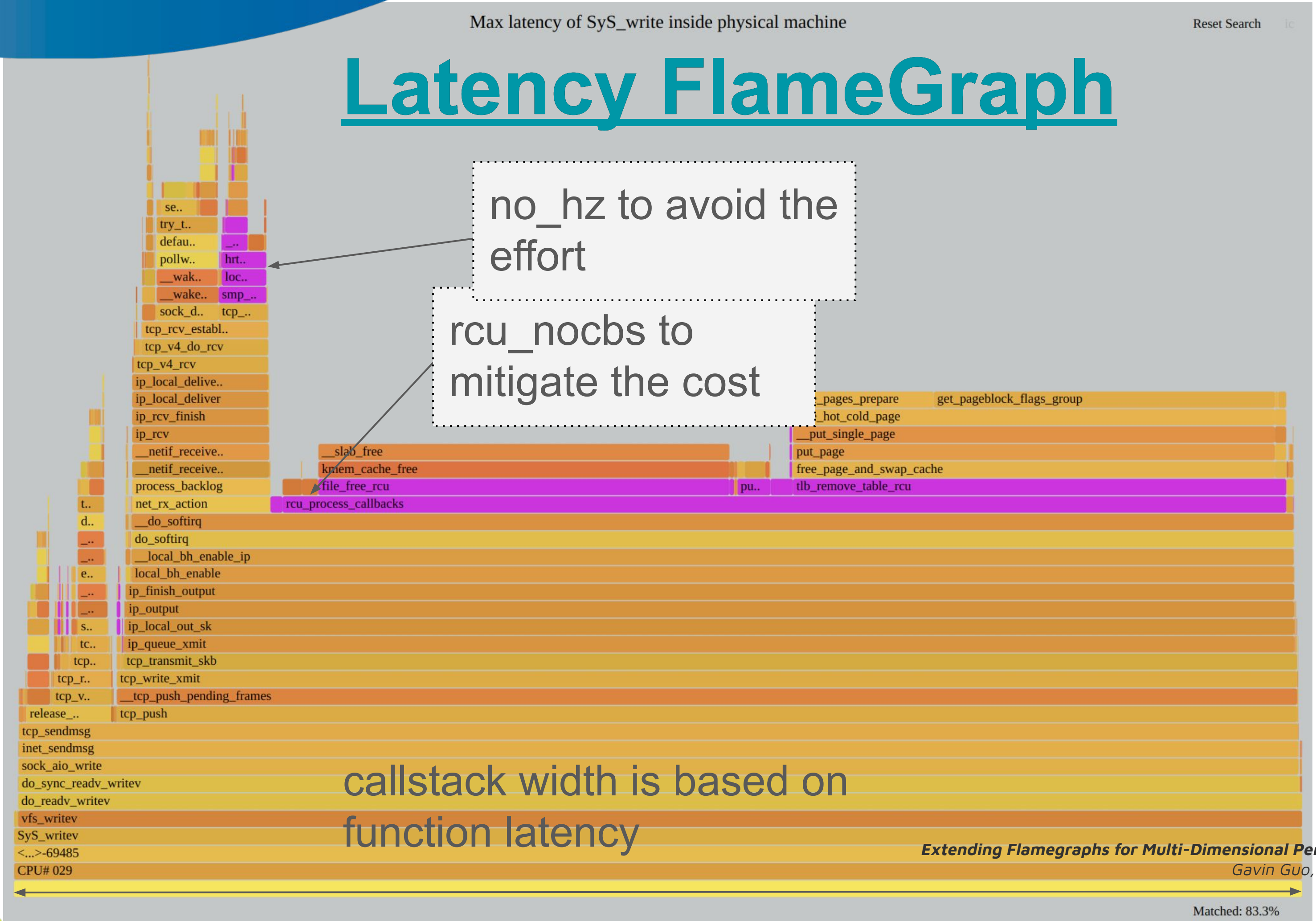


```
# tracer: nop
#
# entries-in-buffer/entries-written: 2598/2598   #P:14
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
      mem_ftrace-895309  [000] ..... 1143967.713016: kmem__alloc_pages: (__alloc_pages_noprof+0x0/0x350) request=0x400dc0
      mem_ftrace-895309  [000] ..... 1143967.713020: mm_page_alloc: page=00000000a516890e pfn=0x481960 order=1 migratetype=0 gfp_f
lags=GFP_KERNEL_ACCOUNT|__GFP_ZERO
      mem_ftrace-895309  [000] ...1. 1143967.713023: <stack trace>
trace_event_raw_event_mm_page_alloc
__alloc_pages_noprof
alloc_pages_mpol_noprof
alloc_pages_noprof
get_free_pages_noprof
pgd_alloc
mm_init
dup_mm.constprop.0
copy_process
kernel_clone
__do_sys_clone
__x64_sys_clone
x64_sys_call
do_syscall_64
ntry_SYSCALL_64_after_hwframe
```

Latency FlameGraph

- **Problem Statement and Scenario**
  - A useful scenario would be, for example, a client has a strong requirement to meet a hard limit with the sys_write latency.
  - Traditional sampling often misses latency outliers.
  - Enabling *function_graph* filter would be helpful in gathering the calltraces and the corresponding latency number with specific functions.
  - To identify the outlier, the approach is to **sort** the callstack by function latency and generate the latency *FlameGraph* apiece rather than a collective *FlameGraph* with all callstacks.

Latency FlameGraph



no_hz to avoid the effort

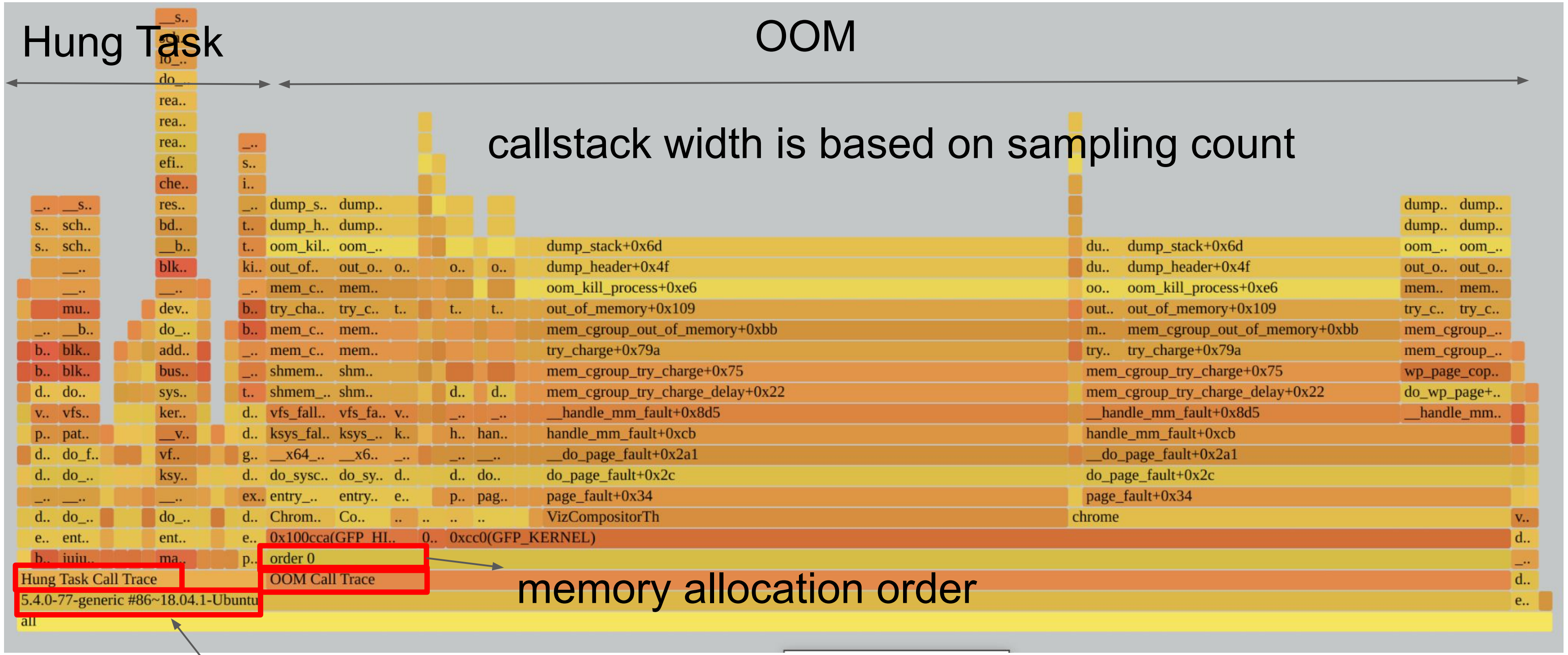
rcu_nocbs to mitigate the cost

callstack width is based on function latency

Kernel log FlameGraph

- **Problem Statement and Scenario**
    - In production system, there could be complicated panics mixing various error types of callstacks—up to hundreds or thousands of callstacks—in the kern.log, exacerbating the diagnosis to untangle the root cause.
    - Using *FlameGraph* to categorize the panics into kernel version, CPU numbers, error types, and further details to narrow down the root cause.

Kernel log FlameGraph

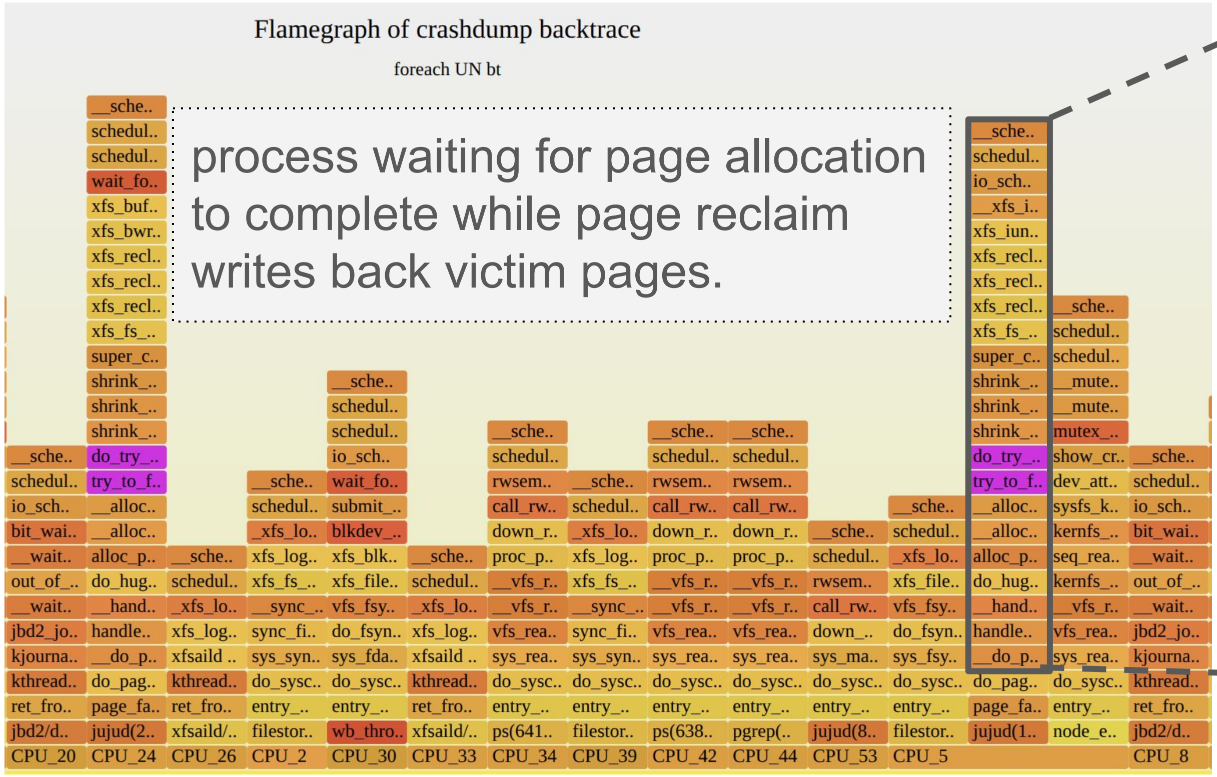


kernel version

Kdump FlameGraph

- **Problem Statement and Scenario**
  - When analyzing a *crashdump* with hung-task issues, people frequently encountered a challenge to locate the related hung tasks by navigating within hundreds or thousands of uninterruptible processes.
  - *FlameGraph* is useful to condense the similar callstacks, facilitating the analysis with a few dominant patterns.

Kdump FlameGraph



process waiting for page allocation to complete while page reclaim writes back victim pages.

```
__schedule <0xffffffff859c21a6>
schedule <0xffffffff859c26b6>
io_schedule <0xffffffff859c2ab6>
__xfs_iunpin_wait [xfs] <0xffffffffc0d2d49c>
xfs_iunpin_wait [xfs] <0xffffffffc0d306d9>
xfs_reclaim_inode [xfs] <0xffffffffc0d23d53>
xfs_reclaim_inodes_ag [xfs] <0xffffffffc0d2419c>
xfs_reclaim_inodes_nr [xfs] <0xffffffffc0d25383>
xfs_fs_free_cached_objects [xfs] <0xffffffffc0d38699>
super_cache_scan <0xffffffff8528818a>
shrink_slab <0xffffffff851f21bb>
shrink_slab <0xffffffff851f2459>
shrink_node <0xffffffff851f7458>
do_try_to_free_pages <0xffffffff851f774e>
try_to_free_pages <0xffffffff851f7ab1>
__alloc_pages_slowpath <0xffffffff851e5099>
__alloc_pages_nodemask <0xffffffff851e5e19>
alloc_pages_vma <0xffffffff85248152>
do_huge_pmd_anonymous_page <0xffffffff8525fcc7>
__handle_mm_fault <0xffffffff8521c0a7>
handle_mm_fault <0xffffffff8521c88c>
__do_page_fault <0xffffffff850783b1>
```

callstack width is based on sampling count

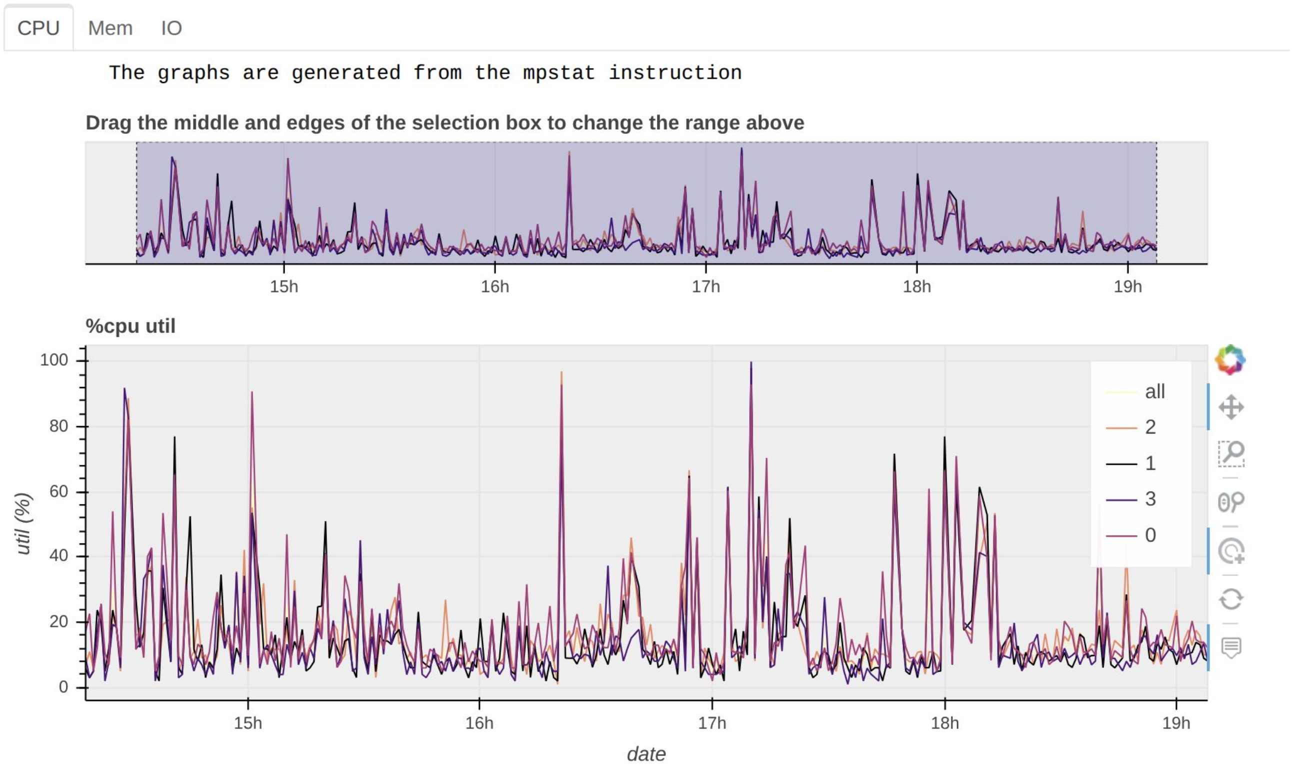
On-going Projects

- My two ongoing projects to facilitate *FlameGraph* debugging
  - Easy FlameGraph
  - ScatterPlot + FlameGraph
  - Kernel Callstack Analyzer MCP Server

Easy FlameGraph

- **Problem Statement and Scenario**
  - In production system, a sudden stuttering could happen in a specific time without any records or logs.
  - Continuous profiling is designed to capture and profile what happens for this specific event.
  - Easy FlameGraph is designed with continuous profiling to capture CPU and memory FlameGraph when a specific threshold is triggered.

Easy FlameGraph Example



The graphs are generated from the mpstat instruction

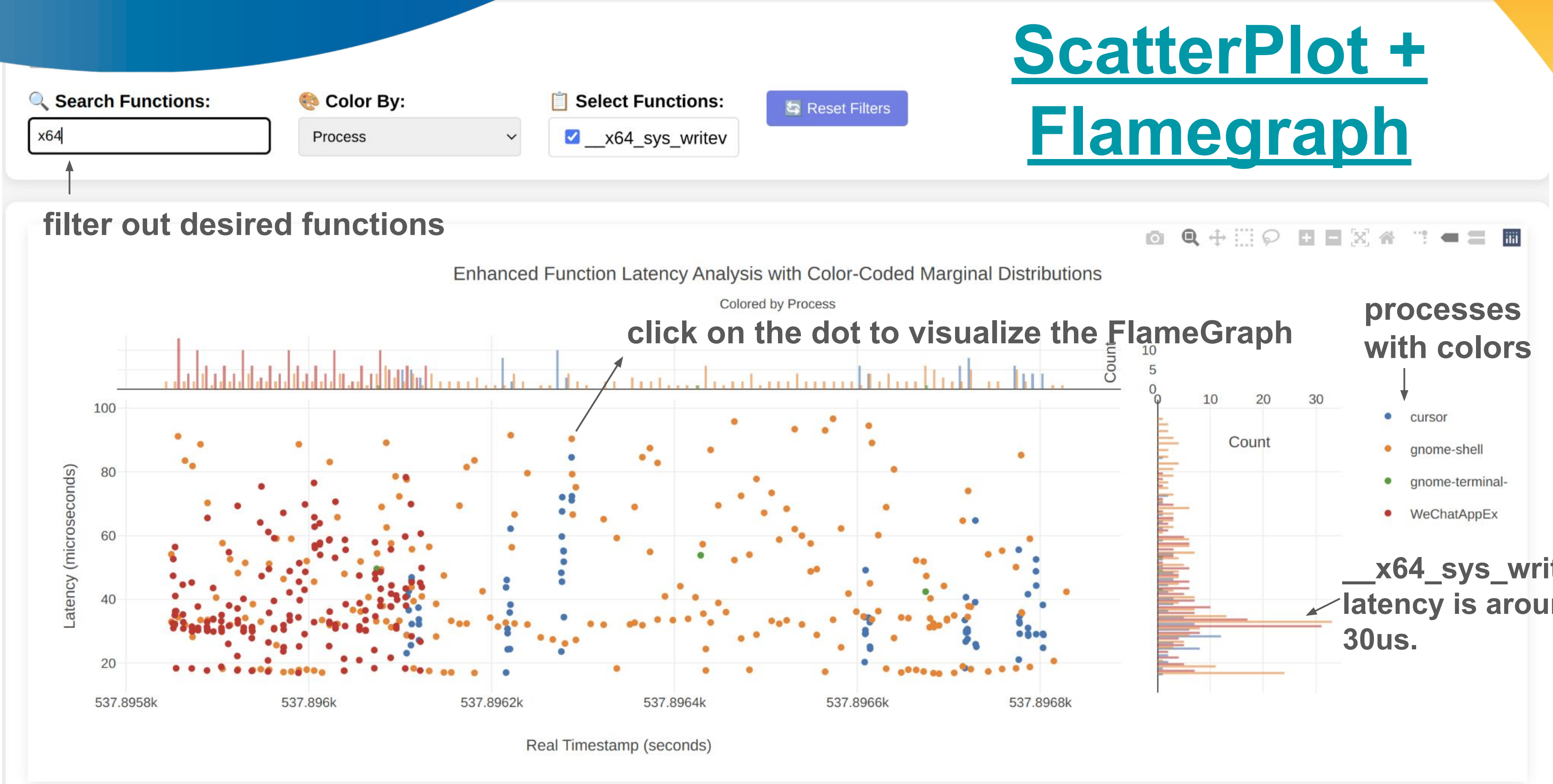
Drag the middle and edges of the selection box to change the range above

%cpu util

ScatterPlot + FlameGraph

- **Problem Statement and Scenario**
  - Performance engineers often want to observe the performance impact among processes and functions.
  - How can we filter the *FlameGraphs* with high-latency functions?
  - *ScatterPlot* is useful to visualize the time and latency distribution.
  - When investigating a high-latency function call, and wanting to explore further, just clicking on the point—the corresponding *FlameGraph* will pop out.

ScatterPlot + Flamegraph



filter out desired functions

click on the dot to visualize the FlameGraph

processes with colors

__x64_sys_writev latency is around 30us.

Kernel Error Analysis AI Agents

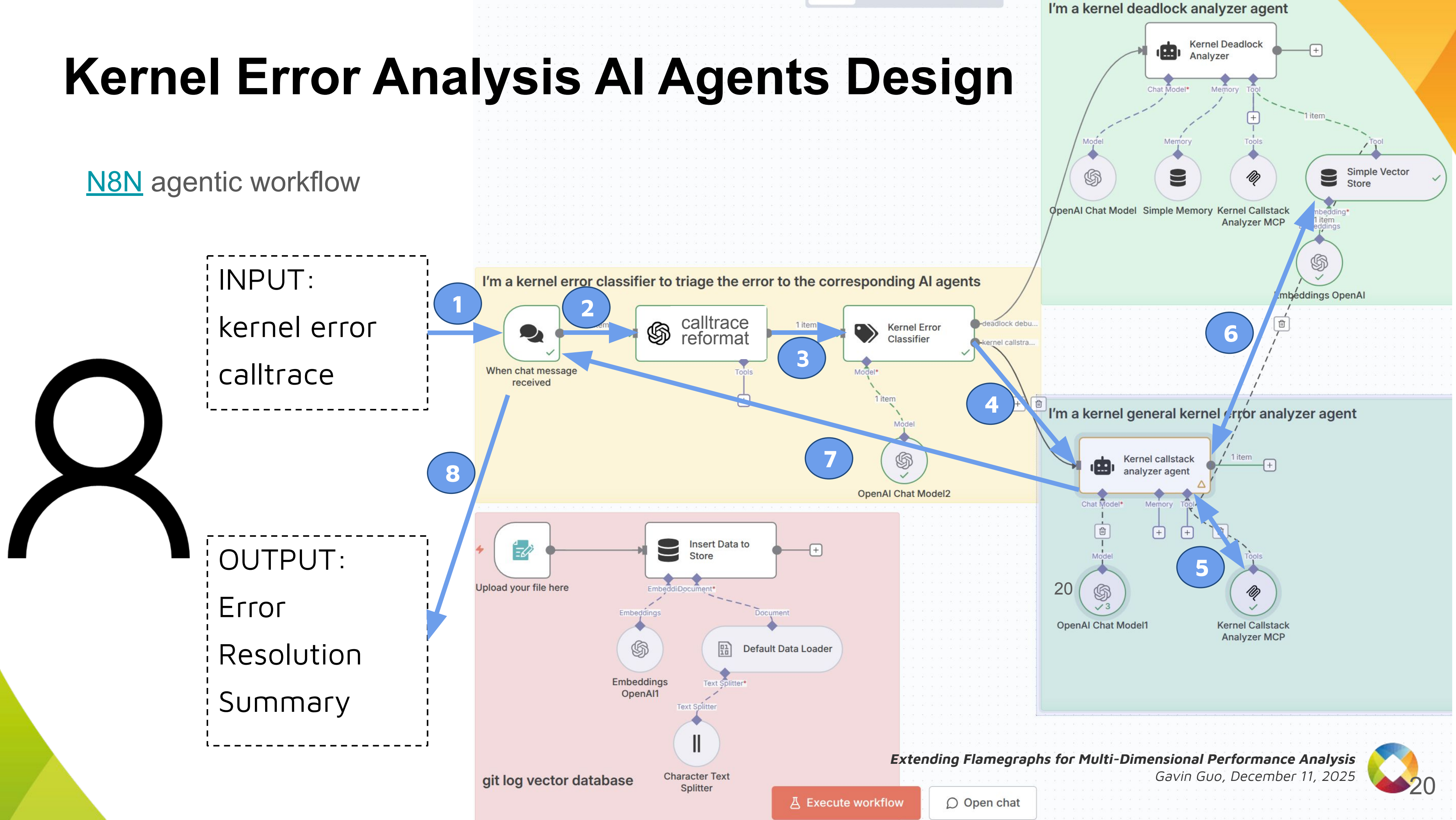
- **Problem Statement and Scenario**
  - Analyzing the callstack is always a laborious effort and requires broad knowledge to comprehend and narrow down the culprits.
  - Is it possible to leverage the LLM’s capability to accelerate the debugging?
  - An AI agent analyzes kernel-panic-related source code provided by a kernel-source MCP server and searches a git-log vector database for potential fixes.
  - Presented in Decoding Kernel Callstack With MCP Tools—China Linux Kernel Conference 2025.

Kernel Error Analysis AI Agents Design

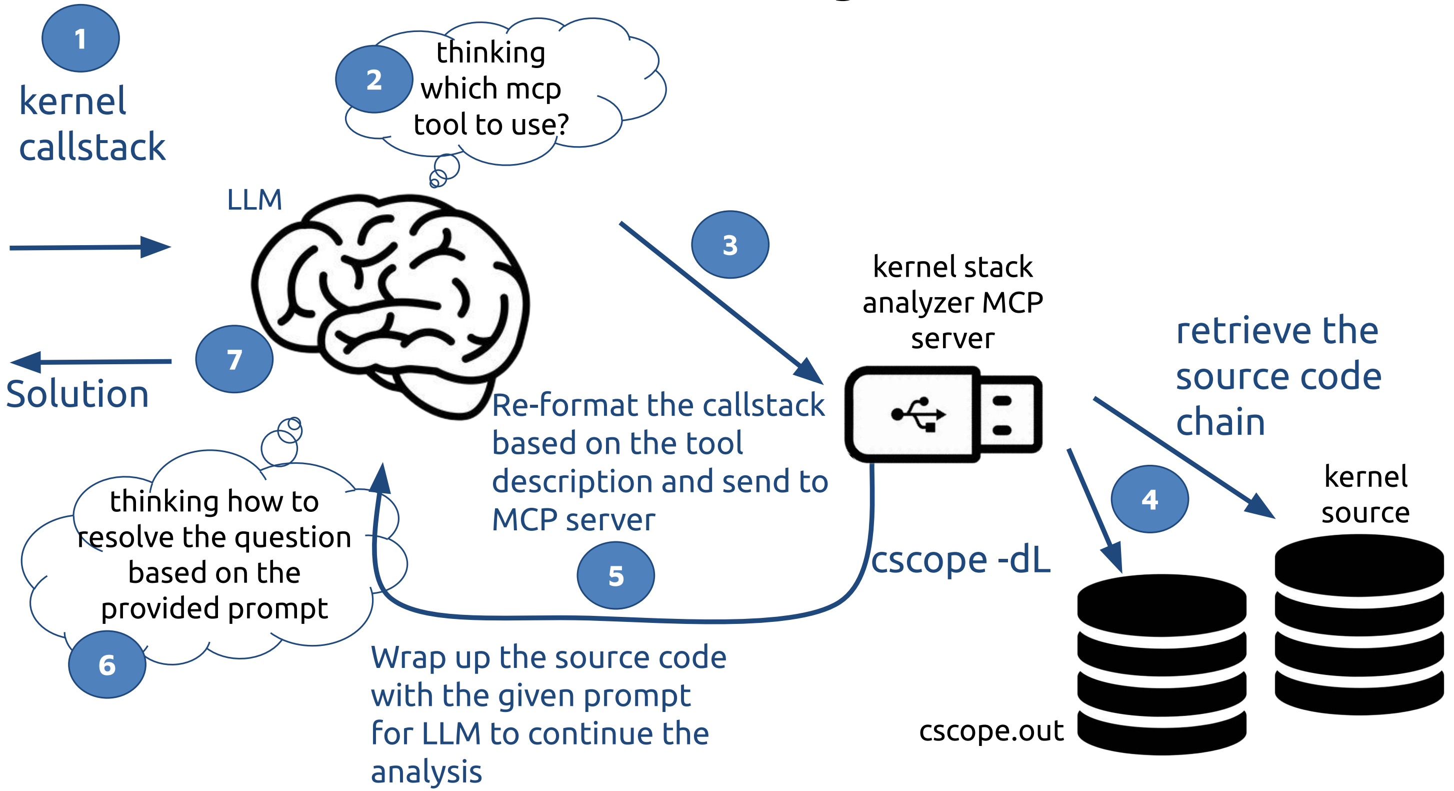
N8N agentic workflow

INPUT:
kernel error
calltrace

OUTPUT:
Error
Resolution
Summary



Kernel CallStack Analyzer MCP Server


1
kernel callstack
LLM
2
thinking which mcp tool to use?
3
Re-format the callstack based on the tool description and send to MCP server
kernel stack analyzer MCP server
retrieve the source code chain
4
kernel source
cscope -dL
cscope.out
5
Wrap up the source code with the given prompt for LLM to continue the analysis
6
thinking how to resolve the question based on the provided prompt
7
Solution

Discussion

- **Memory**
  - Should stackcollapse-tracecmd.pl go upstream kernel (tools/perf/scripts/perl/)?
  - Should both scripts be merged into trace-cmd?
- **Latency**
  - Is it useful to be merged into *trace-cmd* to output the designated outlier's latency *FlameGraph*?
  - How to design an approach to print out the outlier from thousands of callstacks?
  - [stackcollapse] Add the stackcollapse-tracecmd.py to generate latency
- **Kernel log**
  - Is there interest in integrating kern-log *FlameGraph* for the kernel CI?
  - [stackcollapse] Add the stackcollapse-kernel.pl to parse dmesg/kern.log
- **Kdump**
  - Is crash utility a suitable place to be merged?
  - [stackcollapse] Add the support of the "bt" command in the Crash utility
- **ScatterPlot + FlameGraph**
- **Kernel Error Analysis AI Agents Design**

igalia

Join us!

https://www.igalia.com/jobs