CHINA LINUX KERNEL
中国Linux内核开发者大会

vivo

Decoding Kernel Callstacks with MCP Tools

Gavin Guo

1 November 2025, Shenzhen, China

igalia

Wechat

Slide

Agenda

- The Problem: Kernel Debugging Challenges
- AI Agent for Kernel Stack Analyzer
  - What's MCP (Model Context Protocol)?
  - What's An Embedding Model?
  - Git Log Vector Database Construction
  - The Kernel Error Classifier
  - The General Kernel Calltrace Analyzer AI Agent
  - The Final Analysis from AI Agents
- Current Upstream Trend
- Future Directions

The Problem: Kernel Debugging Challenges

The Problem: Kernel Debugging Challenges

[CVE 2025-37958]

BUG: **unable to handle page fault for address**: ffffea60001db008
CPU: 0 UID: 0 PID: 2199114 Comm: tee Not tainted 6.14.0+ #4 NONE
Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.16.3-debian-1.16.3-2 04/01/2014
RIP: 0010:**split_huge_pmd_locked**+0x3b5/0x2b60
Call Trace:
<TASK>
try_to_migrate_one+0x28c/0x3730
rmap_walk_anon+0x4f6/0x770
unmap_folio+0x196/0x1f0
split_huge_page_to_list_to_order+0x9f6/0x1560
deferred_split_scan+0xac5/0x12a0
shrinker_debugfs_scan_write+0x376/0x470
full_proxy_write+0x15c/0x220
vfs_write+0x2fc/0xcb0
ksys_write+0x146/0x250
do_syscall_64+0x6a/0x120
entry_SYSCALL_64_after_hwframe+0x76/0x7e

How to debug the kernel page fault?

[PATCH v3] mm/huge_memory: fix dereferencing invalid pmd migration entry

The Problem: Kernel Debugging Challenges

- Why This Matters?
  - Manual callstack analysis is ***time-consuming*** and **tedious**
    - Engineers spend hours/days tracing through thousands of lines of code
  - Repetitive work correlating RIP addresses with root causes
  - Impact on stable kernel maintenance and upstream development

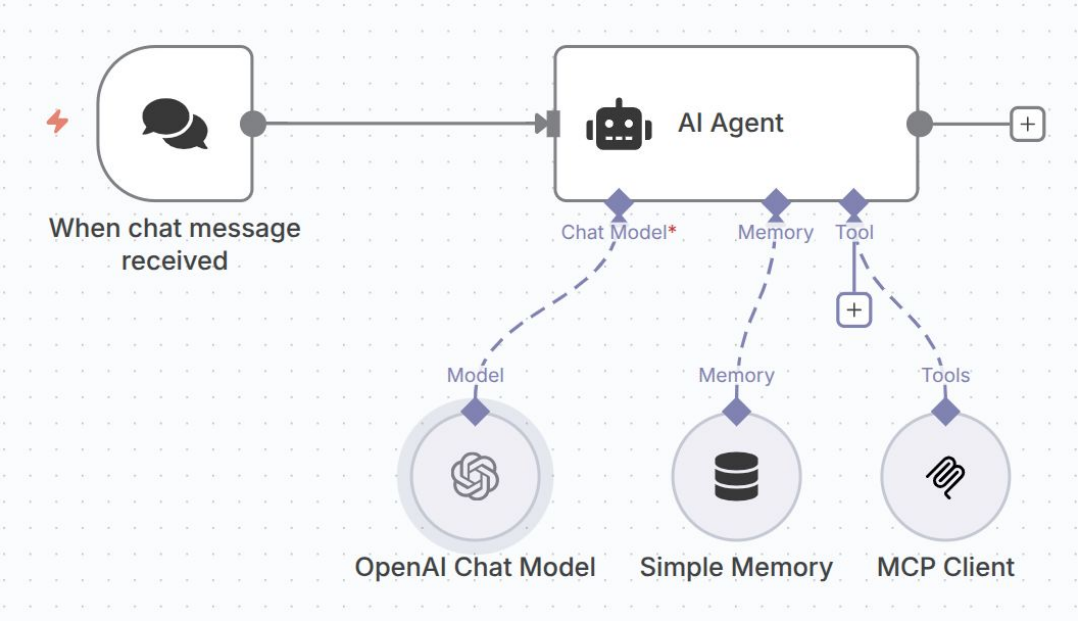
How to Debug the kernel calltrace?

- Use the addr2line to find the source code with given address
- Looking into the function to figure out what happen
- Search git log to find existing fixes
- **Could we use AI to have a preliminary analysis?**

Kernel Error Analysis AI Agents Design

What's an AI agent?

- Anthropic: "AI agents are programs where LLM outputs control the workflow."
- Anthropic: "Building Effective Agents"
- Three components of an AI agent
  - LLM
  - Memory
  - Tools



Kernel Error Analysis AI Agents Design

- **Objective**:
  - Develop an AI agent framework for kernel error analysis, specifically to assist in diagnosing and resolving issues within kernel calltraces. Finally, the AI agent will try to find any **existing fix** in the git log vector database.
- **User Input:**
  - Users provide kernel calltraces as input to **chatbox** to walk through the workflow.
- **Error Categorization:**
  - A central AI agent **classifies** the calltrace and **routes** it to specialized sub-agents for in-depth processing.
- **Illustrative Examples:**
  - For demonstration purposes, the design incorporates two sample agents:
    - A **general** kernel error analyzer agent for general issue handling.
    - A **deadlock** kernel error analyzer agent focused on detecting and resolving deadlock scenarios.

Kernel Error Analysis AI Agents Design

N8N agentic workflow

INPUT:

kernel error

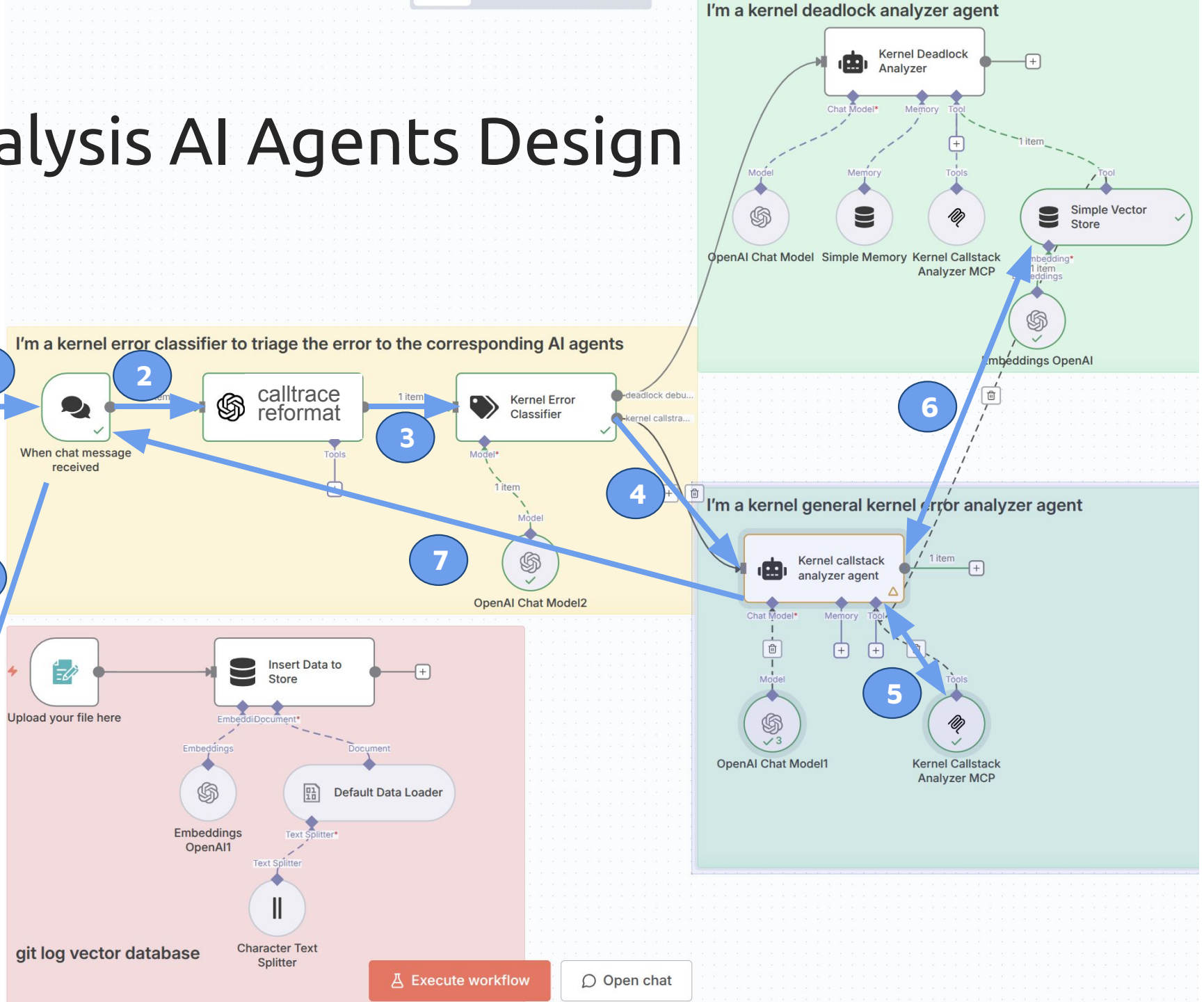
calltrace

OUTPUT:

Error

Resolution

Summary

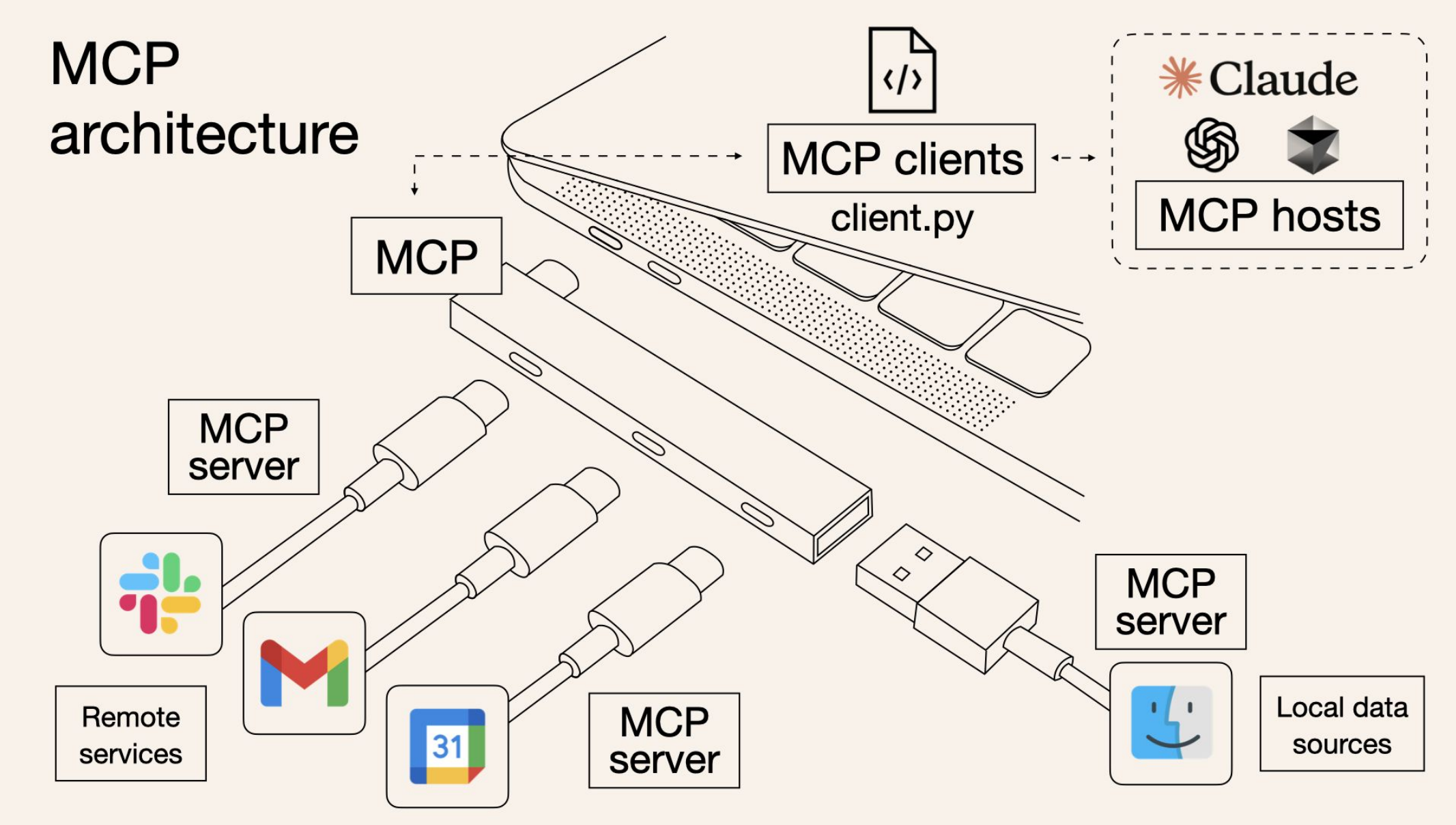


What’s MCP (Model Context Protocol)?

What's MCP (Model Context Protocol)?

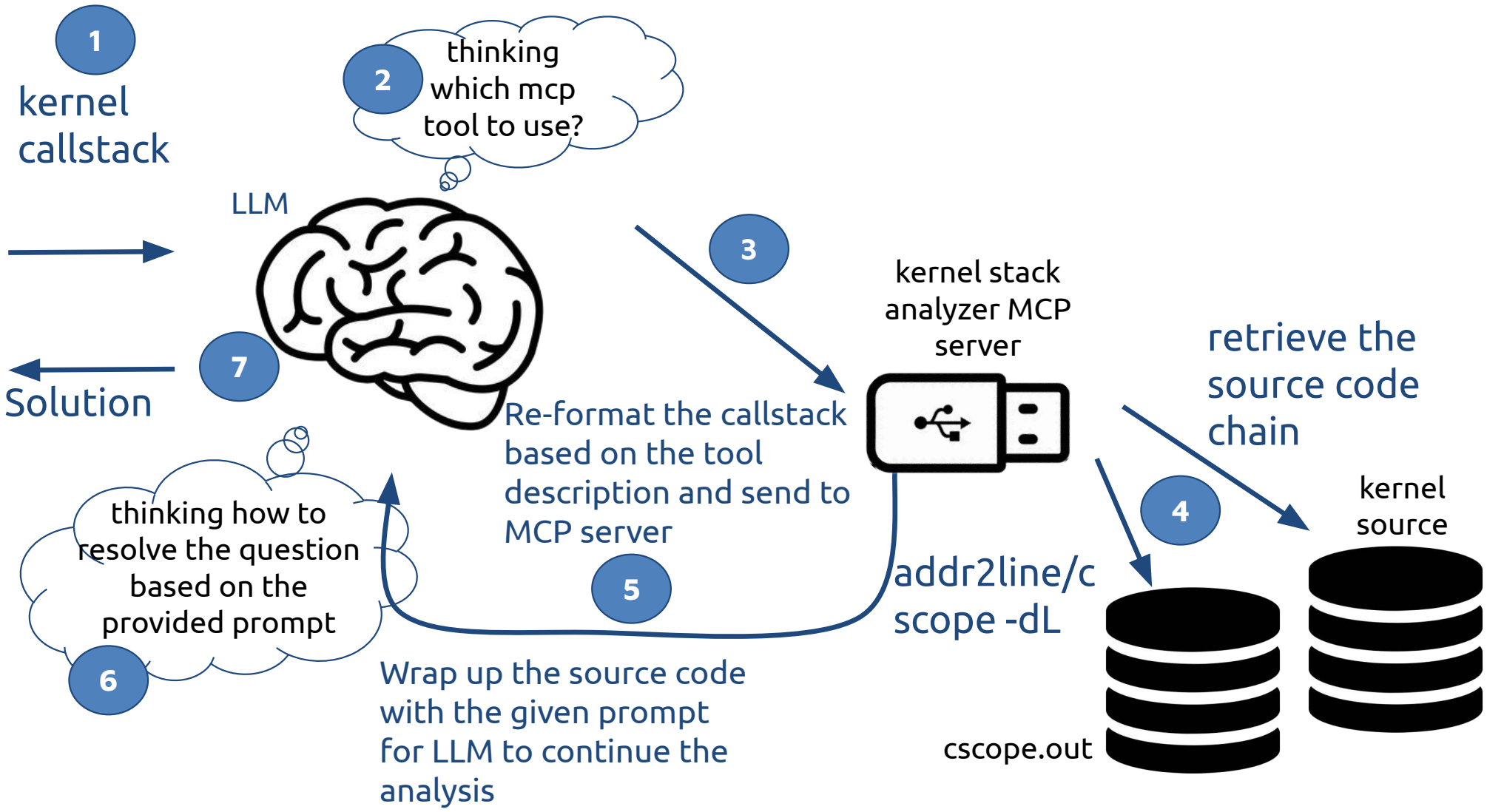
- **What is MCP?**
  - Protocol for connecting AI models with external tools and data sources
  - Bridge between traditional system tools and AI-assisted workflows
- **Why MCP for Kernel Debugging?**
  - Structured way to provide kernel context–such as source code, and git log–to LLMs
  - Standardized interface to leverage AI-powered debugging tools
  - See Github: Kernel Stack Analyzer MCP Tool
- **A temporary Kernel Callstack MCP server to experiment during CLK**
  - http://clk.gavinguo.cc/mcp

kudos to http://www.icangyu.com/uploadfile/2018/0428/20180428121655556.jpg



https://norahsakal.com/blog/mcp-vs-api-model-context-protocol-explained/

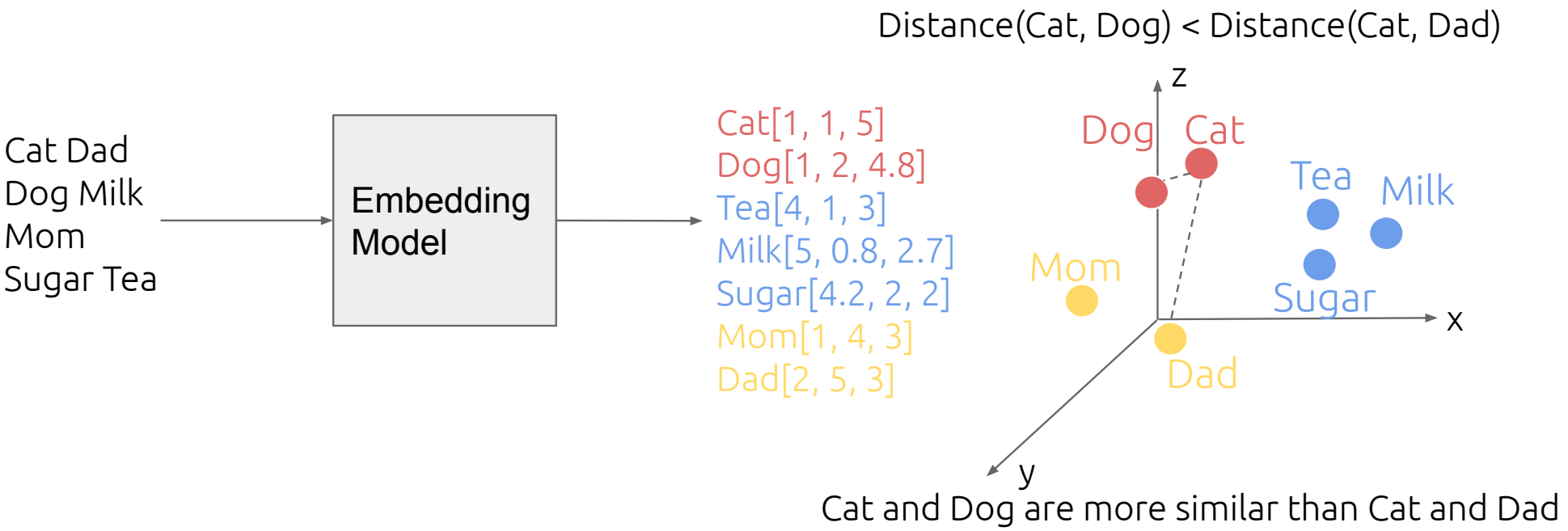
The Kernel Stack Analyzer MCP Tool



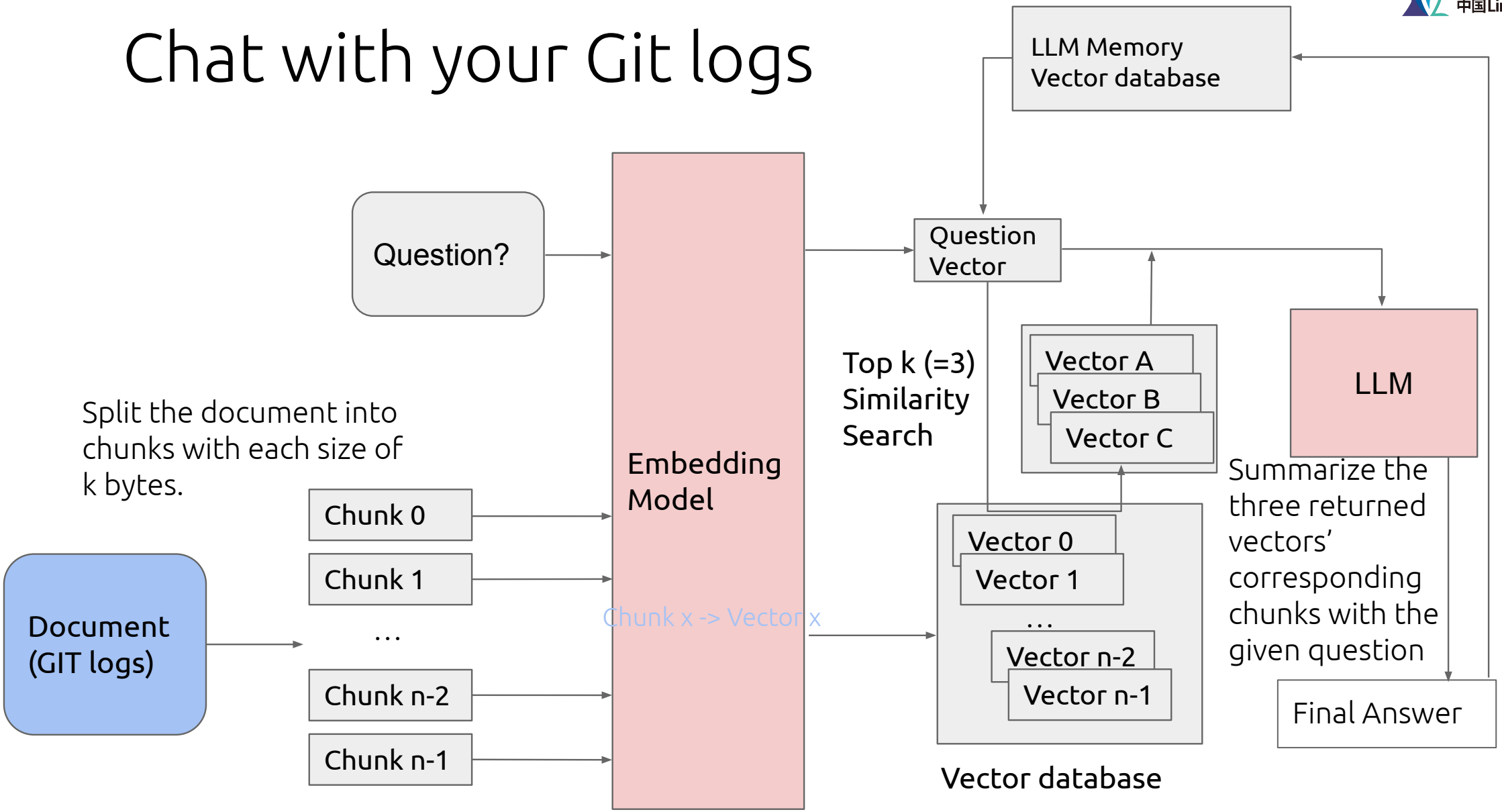
What’s an Embedding Model?

What's An Embedding Model?

Embedding aims to represent the words in the dense vector, while making sure the similar words close to each other in the embedding vector space.



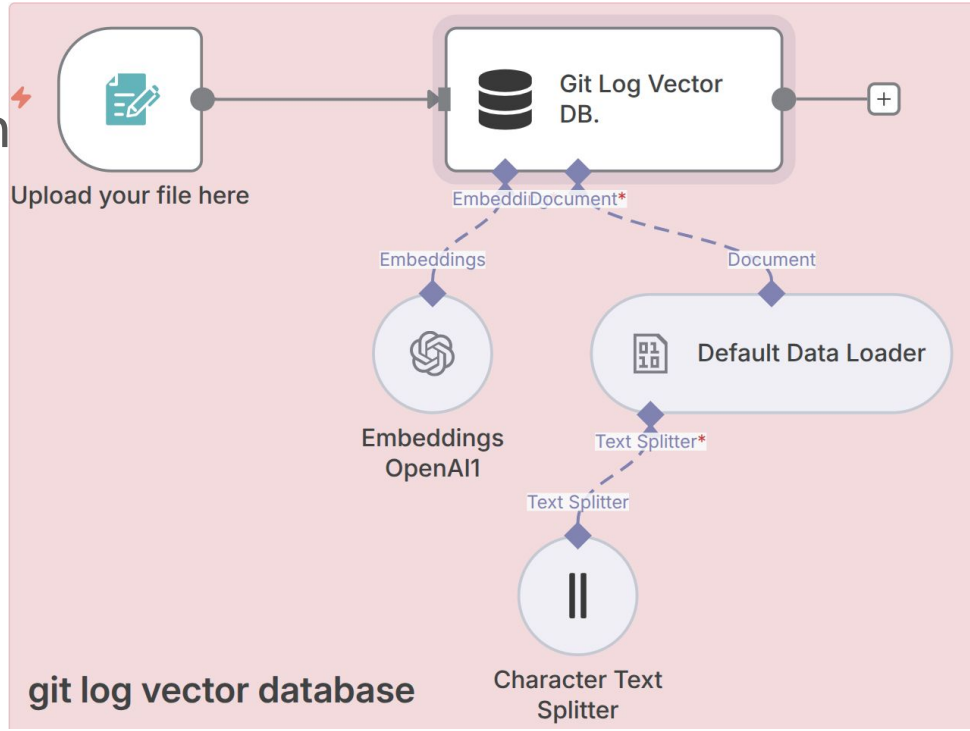
Chat with your Git logs



Git Log Vector Database Construction

Git Log Vector Database Construction

- **Capture Relevant Git Commits:**
  - Gather all commits associated with the project or feature.
- **Reformat to Markdown (.md):**
  - Convert commit data into a structured Markdown file for easy processing.
- **Slice into Chunks:**
  - Divide the .md file using '---' separators to create manageable sections.
- **Generate Embeddings:**
  - Feed each chunk into an embedding model to produce vector representations.
- **Store in Vector Database:**
  - Save the generated vectors to a vector DB for efficient retrieval and querying.



Git Log Vector Database (Commit template)

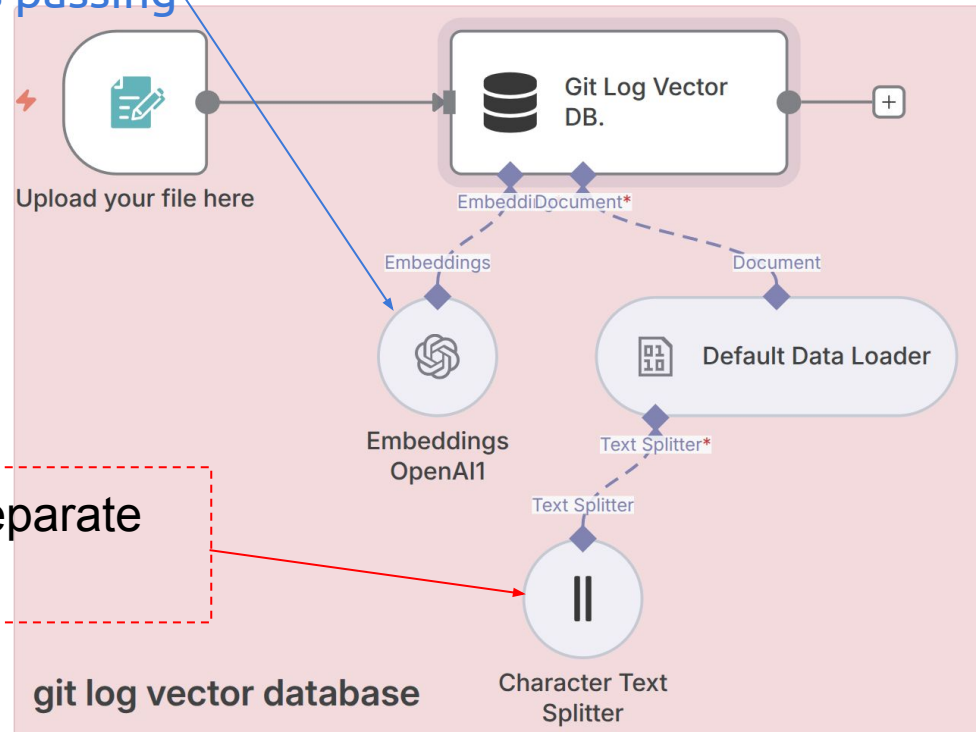
```
---
## Commit 17
### Metadata
- **Commit Hash:** `b960818d51b3...`
- **Author:** Gavin Guo <gavinguo@igalia.com>
- **Date:** Fri Apr 25 18:38:59 2025 +0800
- **Subject:** mm/huge_memory: remove useless folio pointers passing

### Commit Message
[Commit message]
### Diff
```diff
<Code Block>
```
Full Content
<Complete Git show content>

```

This is an example of reformated commit message, generated from the script, used to create a vector by embeddings.

The characters '---' is used to separate different commits.



The Kernel Error Classifier

The Kernel Error Classifier

- Reformat detected kernel calltraces to ensure correct analysis by the MCP tool.
- Error Classification Paths
  a. **Deadlock Debugger**: The classifier analyze the log to determine if it indicates a deadlock issue; if so, triage to this specialized path for debugging.
  b. **General Kernel Calltrace Analyzer**: Default path for inputs, providing broad analysis of the calltrace to identify root causes.

The Kernel Error Classifier

INPUT

```
BUG: unable to handle page fault for address: ffffea60001db008
   CPU: 0 UID: 0 PID: 2199114 Comm: tee Not tainted 6.14.0+ #4
NONE
   Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS
1.16.3-debian-1.16.3-2 04/01/2014
   RIP: 0010:split_huge_pmd_locked+0x3b5/0x2b60
   Call Trace:
   <TASK>
   try_to_migrate_one+0x28c/0x3730
   rmap_walk_anon+0x4f6/0x770
   unmap_folio+0x196/0x1f0
   split_huge_page_to_list_to_order+0x9f6/0x1560
   deferred_split_scan+0xac5/0x12a0
   shrinker_debugfs_scan_write+0x376/0x470
   full_proxy_write+0x15c/0x220
   vfs_write+0x2fc/0xcb0
   ksys_write+0x146/0x250
   do_syscall_64+0x6a/0x120
   entry_SYSCALL_64_after_hwframe+0x76/0x7e
```

Calltrace Reformat

OUTPUT

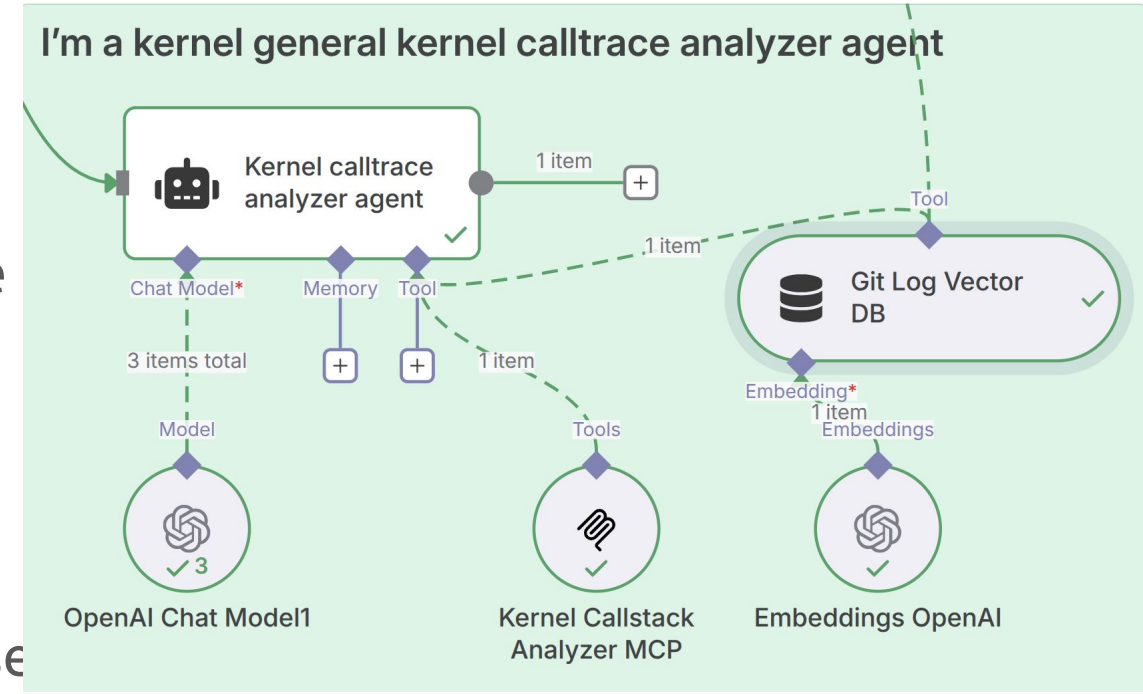
```
BUG: unable to handle page fault for address: ffffea60001db008
   CPU: 0 UID: 0 PID: 2199114 Comm: tee Not tainted 6.14.0+ #4
NONE
   Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS
1.16.3-debian-1.16.3-2 04/01/2014
   RIP: 0010:split_huge_pmd_locked+0x3b5/0x2b60
   Call Trace:
   <TASK>
   [1234.123][ T1] try_to_migrate_one+0x28c/0x3730
   [1234.123][ T1] rmap_walk_anon+0x4f6/0x770
   [1234.123][ T1] unmap_folio+0x196/0x1f0
   [1234.123][ T1] split_huge_page_to_list_to_order+0x9f6/0x1560
   [1234.123][ T1] deferred_split_scan+0xac5/0x12a0
   [1234.123][ T1] shrinker_debugfs_scan_write+0x376/0x470
   [1234.123][ T1] full_proxy_write+0x15c/0x220
   [1234.123][ T1] vfs_write+0x2fc/0xcb0
   [1234.123][ T1] ksys_write+0x146/0x250
   [1234.123][ T1] do_syscall_64+0x6a/0x120
   [1234.123][ T1] entry_SYSCALL_64_after_hwframe+0x76/0x7e
```

The General Kernel Calltrace Analyzer AI Agent

The General Kernel Calltrace Analyzer AI Agent

- **Callstack Analysis:**
  - AI agent reads and analyzes the callstack based on user-provided prompts.
- **Calltrace Identification:**
  - AI agent detects the calltrace and invokes the MCP tool to fetch detailed source code information.
- **History Preservation:**
  - No history is stored in memory.
- **Fix Retrieval:**
  - AI agent queries the Git log vector database to identify and suggest potential fixes.

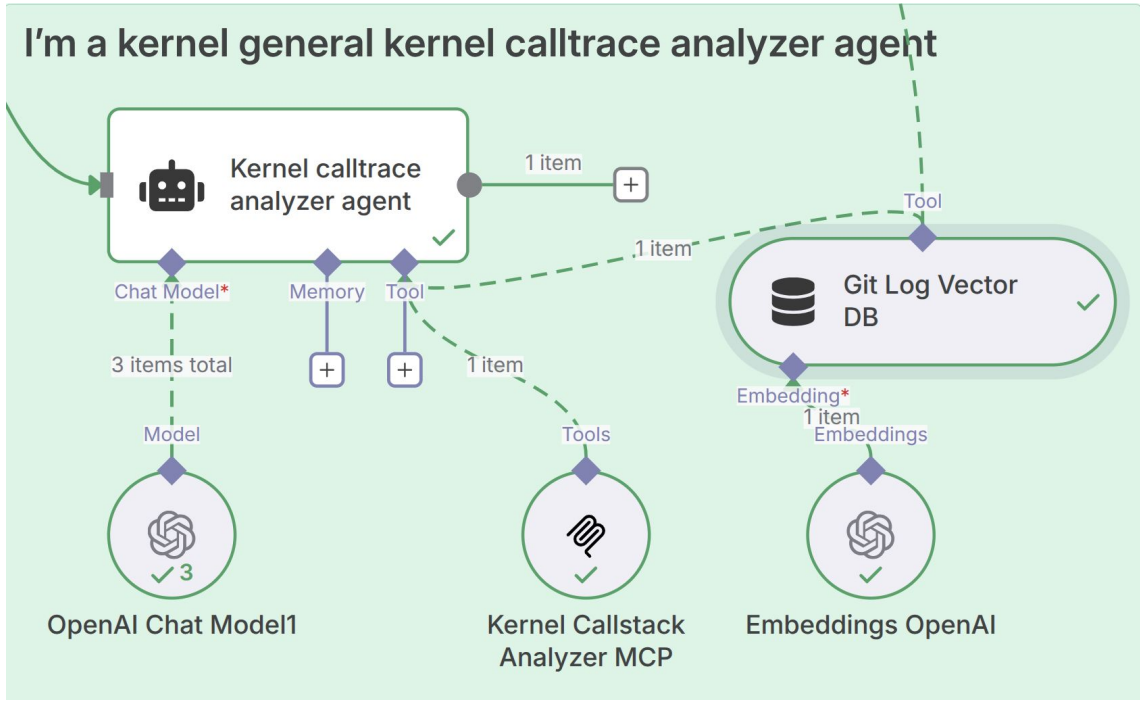
[General Kernel Calltrace Analyzer Prompt]



The General Kernel Calltrace Analyzer AI Agent

- **Query Callstack MCP**
  - LLM identifies the calltrace and invokes MCP tool to retrieve required source code.
  - Please read MCP response for complete data.
- **Query Git Log Vector DB**
  - LLM prepares the best key words to query vector DB for best possible fixes retrieval.
  - Vector DB will reply with four most related fixes. Vector DB response

General Kernel Calltrace Analyzer Prompt



The Final Analysis from AI Agents

The Final Analysis From the AI Agent

- **Please see detailed analysis from the AI agent in [final analysis]**
- **1. Issue Identification and reformat processing**

> **AI Agent Analysis**
>
> *"I found 1 kernel calltrace in the log you gave. (The single BUG/pf report that starts with "BUG: unable to handle page fault..." and the following Call Trace.)"*

- **2. MCP Tool Invocation**
  - The agent invokes the Kernel Callstack Analyzer MCP Tool to normalize the raw callstack, which pinpoints the faulting function as split_huge_pmd_locked.

The Final Analysis From the AI Agent

- **3. Vector DB Search & Knowledge Retrieval**

AI Agent Analysis

*"Relevant upstream fix(s) found (from the **vector store**)*

*- **Commit**: be6e843fc51a584672dfd9c4a6a24c8cb81d5fb7*

*- **Title**: mm/huge_memory: fix dereferencing invalid pmd migration entry*

*- **Summary**: Detect PMD migration entries and avoid applying pmd_folio()/pmd_folio-based equality checks on migration entries. The patch adds or uses an is_pmd_migration_entry() check so that pmd_folio() is not used on migration entries and an early return avoids dereferencing a migration entry.*

Why the commit is the plausible fix

- The crash is precisely caused by incorrectly treating a PMD migration entry as a normal PMD/folio and dereferencing it. The commit directly addresses that: detect migration entries and avoid pmd_folio-based dereferences when the PMD is a migration entry. The stack trace and code location match the commit's described scenario."

The Final Analysis From the AI Agent

- **4. Bug Comprehension**

AI Agent Analysis

*"- The problematic area (from the **MCP-supplied source**) previously did something along these lines:*

```
if (pmd_trans_huge(*pmd) || pmd_devmap(*pmd) || is_pmd_migration_entry(*pmd)) {
    if (folio && folio != pmd_folio(*pmd)) return;
    __split_huge_pmd_locked(...);
}
```

*The bug stems from applying pmd_folio() to a PMD that may be a migration entry (i.e., not a normal PMD pointing to a folio). A migration entry uses a **swap/migration-encoded** PMD value, and pmd_folio() or other PMD->folio decoding paths can dereference data that is not a folio pointer — leading to an invalid kernel address dereference (page fault in kernel context)."*

The Final Analysis From the AI Agent

- **5. Automated Solution Delivery**

AI Agent Analysis

*"**1. Patch or upgrade:***

- Apply the upstream patch (commit be6e843f…) or upgrade to a kernel release that includes this fix (the commit message includes references to the fix and the syzkaller reproducer). [...]

2. Verify:

- Re-run the workload or syzkaller job that triggered the BUG. The crash should not repeat once the check is in place.

3. If you cannot patch:

- As a temporary mitigation, avoid triggering deferred huge page splits via the shrinker debugfs interface or do not enable the specific code paths that schedule deferred splits. This is only a stopgap and not recommended for production."

Current Upstream AI Trend

Current Upstream AI Status

- Sasha Levin
  - AI for Kernel Engineers - Sasha Levin, NVIDIA
  - [ANNOUNCE] AUTOSEL: Modern AI-powered Linux Kernel Stable Backport Classifier
- Chris Mason
  - https://github.com/facebookexperimental/semcode

Future Directions

Future Directions

- **Develop specialized agents for different kernel subsystems**
  - Each subsystem requires tailored prompts for accurate analysis.
  - For complex issues like deadlocks, multiple specialized agents could analyze individual callstacks, while a primary agent synthesizes their findings to identify the root cause.
- **Integrate the agentic framework with Kernel CI systems like Syzkaller and the 0-day test robot**
  - This proactive integration allows for automated bug detection and analysis at the source, significantly enhancing system reliability.
- **Enable real-time integration with performance analysis tools**
  - This would allow the agent to analyze performance metrics and traces as they occur, providing the ability to prevent critical system failures in a timely manner.

Thanks! Q&A?

The Reformat Prompt

You are an assistant who is helping transfer the input message from users line by line to detect the calltrace, if it's not a calltrace, just print it. If it's a part of calltrace, please reformat the calltrace for the sequential LLMs to address the root cause of kernel errors by following the examples:

Warning: this is just an example, you need to convert the symbol based on what you read.

From:

```
 __schedule+0x1755/0x4f50
 schedule+0x158/0x330
 schedule_preempt_disabled+0x15/0x30
 __mutex_lock+0x75f/0xeb0
 hugetlb_wp+0xf88/0x3440
 hugetlb_fault+0x14c8/0x2c30
 trace_clock_x86_tsc+0x20/0x20
 do_user_addr_fault+0x61d/0x1490
 exc_page_fault+0x64/0x100
 asm_exc_page_fault+0x26/0x30
RIP: 0010:__put_user_4+0xd/0x20
 copy_process+0x1f4a/0x3d60
 kernel_clone+0x210/0x8f0
 __x64_sys_clone+0x18d/0x1f0
 do_syscall_64+0x6a/0x120
 entry_SYSCALL_64_after_hwframe+0x76/0x7e
```

To:

```
[1234.123][ T1] __schedule+0x1755/0x4f50
[1234.123][ T1] schedule+0x158/0x330
[1234.123][ T1] schedule_preempt_disabled+0x15/0x30
[1234.123][ T1] __mutex_lock+0x75f/0xeb0
[1234.123][ T1] hugetlb_wp+0xf88/0x3440
[1234.123][ T1] hugetlb_fault+0x14c8/0x2c30
[1234.123][ T1] trace_clock_x86_tsc+0x20/0x20
[1234.123][ T1] do_user_addr_fault+0x61d/0x1490
[1234.123][ T1] exc_page_fault+0x64/0x100
[1234.123][ T1] asm_exc_page_fault+0x26/0x30
RIP: 0010:__put_user_4+0xd/0x20
[1234.123][ T1] copy_process+0x1f4a/0x3d60
[1234.123][ T1] kernel_clone+0x210/0x8f0
[1234.123][ T1] __x64_sys_clone+0x18d/0x1f0
[1234.123][ T1] do_syscall_64+0x6a/0x120
[1234.123][ T1] entry_SYSCALL_64_after_hwframe+0x76/0x7e
```

Please bear in mind that your job is to reformat the calltrace in the text and don't answer any questions. Also, please keep other text from users unmodified and intact. Thanks!

Here is the input from user: {{ $json.chatInput }}

The General Kernel Calltrace Analyzer Prompt

CHINA LINUX KERNEL
中国Linux内核开发者大会

Your role
You are a seasoned Linux kernel panic debugger. Your mission is to help the user analyze the callstack messages and figure out the what's the context of callstack execution.

Debugging Steps
Extract the callstacks from the error message
You will be given a calltrace from the user. At first, when you receive the error message. Please identify how many kernel error calltraces are in the message.

Then, you would like to extract the calltraces and send them to the MCP tool one by one. The function of the callstack **MCP tool** is to extract the related source code of the function symbols for you to proceed to analyze and figure out what's running in the calltrace.

Analyze the calltrace
When you receive the calltrace and the corresponding source code, you would like to read the source code and provide your insight. When you have several source code with the calltraces, please read the specific calltrace calling flow carefully. Eventually, you need to provide the analysis result based on combining different calltraces.

Find the possible fix commits in the vector database tool
We have an attached **vector storage tool** including all the **possible git commits fix** in the linux kernel git tree. After analyzing the calltrace, you should have a good understanding of the problem. Then, please ask **a good question** to the vector database tool. The tool will come back with the most-related commits with the **similarity search**. With the commits, please summarzie the plausible fix commit with your previous analysis for user's reference.

This is calltrace log from user for you to analyze:
{{ $json.choices[0].message.content }}