Enhancing Linux Kernel Test Result Analysis: Automated Log Clustering in the KernelCI Database

Gabriela Bittencourt   *Islene C. Garcia*

Relatório Técnico - IC-PFG-24-38
Projeto Final de Graduação
2024 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Enhancing Linux Kernel Test Result Analysis: Automated Log Clustering in the KernelCI Database

Gabriela Bittencourt, gbittencourt@riseup.net
Advisor: Islene Calciolari Garcia, islene@unicamp.br

Abstract

The Linux kernel is one of the largest collaborative efforts of software development in the world, powering a large majority of the infrastructure that runs computing workloads of all scales – from embedded systems to HPC clusters. As such, improving the testing ecosystem for the Linux kernel is critical to ensure the longevity of the project. The KernelCI project is a recent initiative that looks to provide a unified testing infrastructure for all the kernel subsystems; this work aims to improve the automatic evaluation and labeling of test results for the kernel in the context of the KernelCI project, through the use of modern clustering and data aggregation techniques; in particular, we propose a frequency-based algorithm for filtering and labeling logs from kernel tests as a way to facilitate their analysis by kernel maintainers, greatly improving the efficiency of the review process.

Key words: Linux Kernel, Automatic Testing, Continuous Integration, Clustering, TF-IDF, K-Means

Contents

1 Introduction

The Linux kernel is the core part of the Linux operating system, responsible for managing hardware resources and providing a foundation for software to interact with the computer's hardware. It acts as a bridge between the system's applications and the underlying physical hardware components, such as the processor, memory, and storage devices. It's one of the largest open source projects in the world with thousands of developers contributing code and millions of lines of code changed for each release [7]. Leading to widespread adoption and customization of the kernel for various devices.

Small incremental changes, also known as patches, add new features, make enhancements, and fix bugs; this year there was approximately 1 patch being merged each 6,2 minutes [6]. A new released version of Linux Kernel comes out once every 10 to 11 weeks and they are time-based rather than feature-based, and once a week several stable and extended stable releases. The Linux kernel is the result of collaborative 24-hour, seven days a week, and 365 days continuous development process from developers from diverse companies or academia and all across the globe [11].

This collaborative effort, coupled with a fast development cycle, enables rapid updates and improvements. Its customizability allows it to support a wide range of hardware and hardware combinations, making it highly versatile and adaptable to diverse computing environments. Additionally, the Linux kernel performs critical tasks such as process management, device management, and file system management. Ensuring efficient and reliable system operation is therefore a significant challenge.

1.1 Kernel testing

Ensuring software is stable without regressions before the release helps avoid debugging and fixing customer-found and user-found bugs after the release; after all, it is a lot more costly, both in time and effort, to debug and fix issues found in production. Hence, testing is very important when it comes to any software [11]. And the Linux Kernel brings many challenges to testing, with several developers continuing to add new features, and fixing bugs, continuous integration and testing is vital to ensure the kernel continues to work on existing hardware as new hardware support and features get added [11]. Additionally, many kernel functions cannot be tested independently of the building or booting process. As a result, testing extends beyond unit tests or test cases typically found in simpler applications. In kernel there are developer tests, integration tests, regression tests, stress tests, booting tests and others.

Users and developers unfamiliar with new code may test it more effectively than the original author by uncovering issues the author did not anticipate. While developer testing is crucial for verifying functionality, it alone cannot identify interactions with other code, features, or hardware configurations. User testing is vital for detecting unintended regressions and ensuring broader compatibility [11].

Each kernel version must be tested across multiple systems to verify that it is capable of building, booting, and behaving correctly, all while producing logs without anomalies [13], ensuring proper operation on diverse hardware platforms. There are several types of tests, each one with different goals:

- Unit test using KUnit are largely used by developers to test small, self-contained parts of the kernel, they are important to guarantee that, at least in an isolated scope, the changes that developers make do not introduce regressions or change the expected behavior of the kernel APIs [4].

- Integration tests (using the kselftest framework) are well-suited for more complex tests of entire kernel subsystems, as these typically expose an interface to userspace. Kselftest is limited in that it cannot directly invoke kernel functions; it can only test kernel functionality exposed to userspace through system calls, devices, filesystems, or similar interfaces. It provides a form of ”system” or ”end-to-end” testing, and all new system calls should include corresponding kselftest cases [5].

- Boot tests in the Linux kernel are essential to ensure that the kernel initializes correctly across various hardware platforms and configurations. These tests validate the kernel's compatibility and stability during the critical boot process. They also help identify hardware or configuration-specific issues early in the development cycle, preventing potential failures. Additionally, boot tests ensure that changes to the kernel do not introduce regressions that could disrupt the boot process.

- Compilation/build tests are designed to verify that the kernel can be successfully built across a variety of configurations, architectures, and toolchains. This ensures that changes to the kernel source code do not introduce build failures or incompatibilities and verifies that dependencies and build scripts are correctly configured.

Even though the developer may test their code locally (through any of the aforementioned methods), the majority of the kernel testing is done on specialized infrastructure, such as build pipelines on the cloud that run cross-compilation and other build-related tests. There are also various test labs provided by silicon vendors that offer access to a wide range of hardware for testing. These setups enable testing the kernel build and boot process on various various hardware platforms and configurations. Additionally, it allows for testing the kernel behavior under heavy workloads, boot-time errors, and faulty hardware n in order to effectively assess reliability and robustness.

2 State of the art

Today, there are numerous testing platforms and frameworks for the Linux kernel, with some subsystems developing their own solutions. These independent testing solutions are designed to evaluate specific subsystems without depending on centralized or general-purpose testing frameworks. Tailored to the unique requirements of each subsystem, they enable focused validation and troubleshooting. Examples include testing various filesystems with the xfstests tool, memory management with mmtests, block devices with blktests, networking with packetdrill, and many others.

Although these independent solutions are tailored to the specific needs of subsystems, ensuring each is rigorously tested to meet its requirements, they can be time-consuming for maintainers to set up and often involve redundant efforts across different subsystems.

While isolated testing is a crucial step in verifying the functionality of specific tools or subsystems, it alone is insufficient to uncover interactions with other code, features, and unintended regressions on configurations and/or hardware, the developer did not anticipate and did not have the opportunity and resources to test [11]. Therefore, broader testing strategies are essential due to the kernel's complexity and the vast array of hardware and configurations it supports.

2.1 Systematic testing projects

There are a few initiatives that attempt to address the problem of duplication of effort in testing the kernel by employing a systematic, kernel-wide approach. One such initiative is the 0-day Project, spearheaded by Intel; by recognizing that performance and scalability are critical to the operating system kernel's success, the project emerged in response to discussions on the Linux Kernel Mailing List about significant performance regressions between kernel versions. The project was designed to conduct patch-by-patch testing on every changeset sent to the kernel mailing lists (including non-merged patches); it performs boot, functional, and performance tests across various platforms in hardware labs, though it is limited to the Intel processor architecture [16].

In a similar fashion, but with the idea of covering unusual edge-cases in the interaction between kernel and userspace, the syzkaller project was created; it is a kernel testing bot designed to detect bugs and vulnerabilities by generating and executing random system calls uncovering subtle bugs in kernel interfaces and subsystems. Syzkaller generates randomized test programs which invoke one or more system calls (or generally interact with the kernel in other ways, such as injecting network packets) that are purposefully faulty, and then run them, trying to see if this causes a crash or an error of some sort (e.g. a kernel panic). It also has the autonomy of collecting kernel code coverage information and deciding whether to try iterating upon the previous semi-random test programs, or start anew. Upon detecting a crash, it collects information about the issue and tries to discover a minimal test case (called a reproducer) that triggers the crash, and then informs the relevant developers about it in the mailing lists [12].

Syzkaller, as a coverage-guided fuzzer, can "trace" the code paths executed when processing a given input, and use that information to try and generate inputs which uncover previously unexecuted code by leveraging software instrumentation facilities. Fuzzers use this technique to achieve high levels of test coverage very efficiently, and indeed, the aforementioned fuzzers have been used to find thousands of severe bugs in all sorts of parts of the Linux kernel, even those considered mature and well-tested [12].

2.2 KernelCI project

KernelCI is a community-driven, open-source test automation system dedicated to upstream Linux kernel development. It provides extensive hardware and configuration testing, ensuring broad compatibility, stability, and long-term maintainability of the Linux kernel. With a focus on open testing practices, KernelCI consolidates existing testing initiatives, improves LTS kernel testing and validation, and expands testing resources and hardware coverage. Its wide scope and comprehensive coverage make it an essential tool in ensuring the quality and reliability of the Linux kernel across diverse platforms [9].

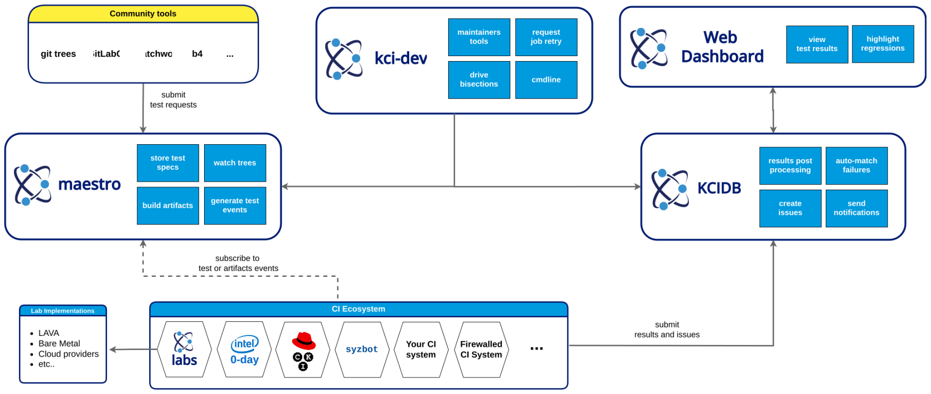


Figure 1: KernelCI architecture [8]

The figure 1 shows the inner details of the architecture of KernelCI and the interactions between its components [8]. The KernelCI project aggregates results from various test sources, acting as an orchestrator for several different test methods. Any continuous integration (CI) ecosystem can upload its results to the KernelCI database (kcidb), enabling even small ecosystems to contribute their data to a larger, more generalized project. This centralized approach allows for comprehensive analyses based on the collected data and encourages small groups to develop their own CI lab projects to contribute to the kernel testing.

A centralized approach minimizes redundant work caused by independent testing and promotes extensive test coverage. Additionally, CI systems with reliable test results and effective triage significantly alliviates the pressure on maintainers by enabling a progressively more automated testing process with broader coverage. However, KernelCI is still a relatively new tool, leaving room for improvements in test triage and the development of analysis tools.

The tests on KernelCI project database provide one of the following possible outcomes:

- PASS (no error was detected);
- MISS (The test was not executed due to some infrastructure error);
- SKIP (The test wasn't executed, probably due to failures in its dependencies, e.g., checkout or build);
- FAIL (The test was executed and failed);
- ERROR (The test started executing but didn't complete) [3].

In this project, we will focus on the status "ERROR", which, as it's possible to see in figure 2 is responsabily of the developers.

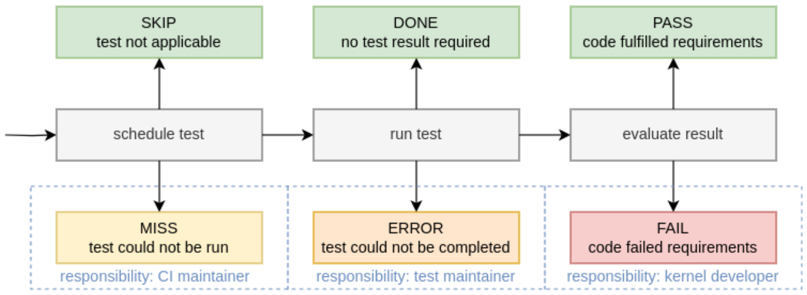


Figure 2: Workflow of test execution and responsability of its status [2].

To ease the analysis for the data, there is a dashboard named Grafana dashboard [9] intended for kernel and test maintainers/developers. It reads data from the KCIDB database and presents it in various formats for analysis. The maintainer is capable to filter: origin of the data (CI lab that generate the result); git tree and branch from kernel; the path of the tests performed (e.g., "kselftest.dt", "baseline", etc.); the platform (used for non-build tests); the configuration and the period data to show [3].

The dashboard presents an overall statistics useful for understanding the last time data arrived in KCIDB and identifying which tree has errors. It also separate errors of building, named: "build" and errors non related to build, named: "tests". Also, detected infrastructure errors are filtered out (except in the tables that show all nodes) [3]. The dashboard displays various data points for each test, including the log document—a file containing all log messages generated by the kernel during building, booting, and other testing processes. This document will serve as the primary source for analysis in this project.

The "Issues" section is designed to facilitate the triage of similar errors by grouping them based on a common root cause. Its primary goal is to consolidate all errors that stem from the same problem into a single issue. Currently, developers must manually create issues and link them to the corresponding errors, a process that is both time-consuming and prone to inefficiencies. This inefficiency not only wastes developers' time but also distorts the perception of the actual number of problems, as the total error count does not accurately reflect the number of underlying issues, leaving developers and maintainers without a clear understanding of the system's true state [1].

3 Objectives

Tools are being developed to automate the process of matching errors with issues by applying regular expressions to log messages. However, this solution still requires maintainers

[1]The information presented here was gathered by the author through participation in the weekly international kernel CI meetings and talks with maintainers of the tool.

to manually create the issues and define the associated regular expressions[2].

This project introduces a new idea: clustering errors of similar nature based on log message information and highlighting shared characteristics within each cluster. This method enables developers and maintainers to instantly identify the number of issues detected based on their selected filters. Additionally, it allows maintainers to address each issue individually, armed with comprehensive data collected during testing. With clusters already organized and triaged, the investigation process becomes significantly more efficient.

- Clustering the data in meaningful groups of tests fails relate to the same issue
- Retrieve information from the cluster that contains the log lines the error

Effective clustering of errors can significantly reduce the time developers and maintainers spend organizing issues. It simplifies the investigation process for identifying possible root causes of errors and provides immediate insight into the number of issues or problems introduced by a specific version. Consequently, this tool may become more appealing to developers and maintainers by centralizing kernel test management and encouraging broader contributions to the process. By minimizing duplicated efforts caused by isolated solutions in various subsystems, it fosters a more streamlined and collaborative workflow.

4 Methodology

To utilize information from log messages, this project focuses on two main steps: vectorizing the log messages, converting each one into a set of values; and clustering them based on their similarities (with or without a previous dimensionality reduction of the features extracted).

4.1 Feature extraction

The first step of this project was to extract useful information from each test failure. As previously mentioned, each test failure is associated with a document containing log messages from the kernel. To achieve this, an information retrieval approach was employed, representing each document as a set of parameters derived from its content. The method used was Term Frequency-Inverse Document Frequency (TF-IDF), a statistical technique widely utilized in text mining and Natural Language Processing. Term Frequency and Inverse Document Frequency are two distinct components that are multiplied together to calculate the final TF-IDF value for each term in a document. This value reflects the term's importance within the document, emphasizing those frequent in a document but uncommon across the collection [17].

The Term Frequency (TF) measures how often a term appears in a document, assuming frequent terms are more significant. In kernel execution, if a routine fails, it logs a problem diagnosis, making it crucial to identify terms and their frequencies. Significant terms are often highlighted in logs, offering insightsinto the issue.

[2]Information was gathered through discussions with the community.

While the Inverse Document Frequency (IDF) measures a term's importance across documents, deeming frequent terms less important while giving more weight to rarer terms. This is crucial for identifying key log messages, as many common messages during building and booting add little value, while rare messages like "kernel panic" are critical for diagnosing errors and understanding their causes.

The TF-IDF is expressed formally by eq. (1) below:

$$\begin{aligned} \text{TF}(t,d) &= \frac{\text{Number of times of term } t \text{ appears in a document } i}{\text{Total number of terms in document } i} \\ \text{IDF}(t) &= \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing the term } t}\right) \end{aligned} \tag{1}$$

Other alternative approaches were initially considered but ultimately dismissed. One critical aspect of log analysis is weighing the rarity of terms across all logs, making the IDF component essential for vectorizing logs in alignment with the project objectives. Because of this requirement, the Bag of Words method, which lacks this weighting mechanism, was discarded.

Since log messages typically lack semantic relationships between lines, lighter approaches were favored over models that encode semantics. Consequently, more advanced models, such as word embeddings (e.g., Word2Vec) and natural language models (e.g., BERT, LLaMA [15]), were deemed unnecessary, at least for this project.

It's worth mentioning that while projects requiring NLP often lean toward LLM (Large Language Models) alternatives [15] to meet their demands, the author chose a simpler and lighter tool. The potential use of LLaMA (Large Language Model Meta AI) combined with RAG (Information Retrieval with Text Generation) was considered and discussed with community members. However, using an LLM would would significantly increase the processing demands, restricting execution to powerful servers. This approach would make it impossible for developers to run the tool locally and would complicate its personalization.

Although there is no unanimous agreement on the matter, it was surmised that a simpler, customizable tool that is lightweight enough to run locally would be more accepted and widely used by the community. Moreover, a TF-IDF-based solution can be combined with other tools and further developed by the community into a powerful test triage tool. Consequently, the LLM-based approach was abandoned. The author believes that such an approach would benefit the LLM more by gathering valuable information than effectively addressing the problem itself, which is better served by a simpler and lighter solution, even if it does not fully resolve the issue.

The complexity analysis for this method is: $O(n \cdot m)$, where n is the number of documents and m the quantity of distinct terms throughout all documents.

4.2 Dimensionality Reduction

The TF-IDF method generates a set of features and corresponding values for each log. One approach explored in this project was dimensionality reduction. Two different methods were tested, each with distinct objectives, alongside clustering applied directly to the TF-IDF distance matrix. For this, cosine distance was used instead of Euclidean distance, as it aligns better with the nature of the TF-IDF calculation.

The first method tested was Singular Value Decomposition (SVD) a mathematical technique used to factorize a matrix into three simpler matrices, it is simple and versatile. Its mathematical decomposition is explain on equation (2), given a A matrix of features.

$$A = U \cdot \Sigma \cdot V^T \tag{2}$$

where U is an $n \times n$ orthogonal matrix, Σ is an $n \times m$ diagonal matrix containing the singular values and V^T is an $m \times m$ orthogonal matrix (right singular vectors transposed). For dimensionality reduction, it is considered the first k rows and columns of the Σ matrix and consequentialy reducing the dimention of the other two.

It's complexity is $O(\min(n, m) \cdot k \cdot \max(n, m))$, where n and m are the number of logs and terms respectively and k is the number of final dimensions is intended.

Another method tested was Non-Negative Matrix Factorization (NMF). This technique is especially valuable in scenarios where interpretability and adherence to non-negativity constraints are critical, such as in text mining. NMF decomposes a given non-negative matrix A into two smaller non-negative matrices W and H, enabling the discovery of latent patterns (k) while maintaining meaningful, non-negative representations, exposed in equation (3).

$$A_{n,m} \approx W_{n,k} \cdot H_{k,m} \tag{3}$$

It's complexity is $O(n \cdot m \cdot k \cdot t)$, where t is the number of iterations to converge

4.3 Clusterization

One of the most popular and straightforward clustering algorithms is k-means. Mathematically, the k-means algorithm approximates a normal mixture model, where the mixture components are estimated through maximum likelihood. In mixture models, cluster membership is represented probabilistically for each data point, based on the means, covariances, and sampling probabilities of the clusters. These models describe data as a combination of distributions (e.g., Gaussian, Poisson), with each distribution representing a sub-population or cluster. The k-means algorithm is a special case of this approach, assuming that all clusters have spherical covariance matrices and equal sampling probabilities. Additionally, the algorithm treats cluster membership for each data point as an independent parameter to be estimated [14].

The k-means algorithm minimizes the sum of variances within clusters, as expressed in equation (4), where n_i represents the number of cases in cluster k, and the variance minimization objective is fundamental to its operation. This makes k-means a **variance-minimization technique** [14].

$$E = \sum_{i=1}^{k} \sum_{j=1}^{n_i} ||x_{ij} - c_i||^2 \tag{4}$$

In this application, cosine distance—a widely used approach in text mining—will be employed, as defined in equation (5). Additionally, the silhouette score will be used to determine the optimal number of clusters, as shown in equation (6).

$$dCos = \frac{c \cdot x}{||c|| \cdot ||x||} \tag{5}$$

where $dCos$ is the cosine distance between poins c and x.

$$\begin{aligned} S(i) &= \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \\ \text{AverageSilhouette} &= \text{mean}\{S(i)\} \end{aligned} \tag{6}$$

where $S(i)$ is the silhouette coefficient of the data point i; $a(i)$ is the average distance between i and all the other data points in the cluster to which i belongs; and $b(i)$ is the average distance from i to all clusters to which i does not belong.

One of the most popular and straightforward clustering algorithms is K-Means. The primary goal of K-Means is to minimize the within-cluster variance, defined as the sum of squared distances between data points and their cluster centroid. A centroid represents the mean position of all data points within a cluster. In this application, cosine similarity is used as the distance metric.

This method is especially effective for large datasets, where computational efficiency is crucial. It's complexity is a combination of K-Means with cosine distance: $O(k \cdot n \cdot m \cdot t)$, where k is the number of clusters n number of documents, m dimentionality, t iterations for convergion; with the silhouette $O(k \cdot n^2 \cdot m)$, added and multiplied by the maximum number of clusters searched: $O(k_{\max} \cdot (k \cdot n \cdot m \cdot t + k \cdot n^2 \cdot m))$

Another tested clustering algorithm was Density-Based Spatial Clustering of Applications with Noise (DBSCAN). It is a density-based clustering algorithm. It groups points into clusters based on their proximity and density, while identifying outliers as noise. Unlike partition-based algorithms like K-Means, DBSCAN doesn't require you to specify the number of clusters beforehand, and it works well with non-globular cluster shapes and datasets containing noise [10]. It's complexity is $O(n^2)$ In this project, we employed HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), that enhances traditional DBSCAN by performing clustering across varying epsilon values and integrating the results to identify clusters with the highest stability. This capability makes HDBSCAN more versatile than DBSCAN, as it can detect clusters with varying densities and is less sensitive to parameter selection [1].

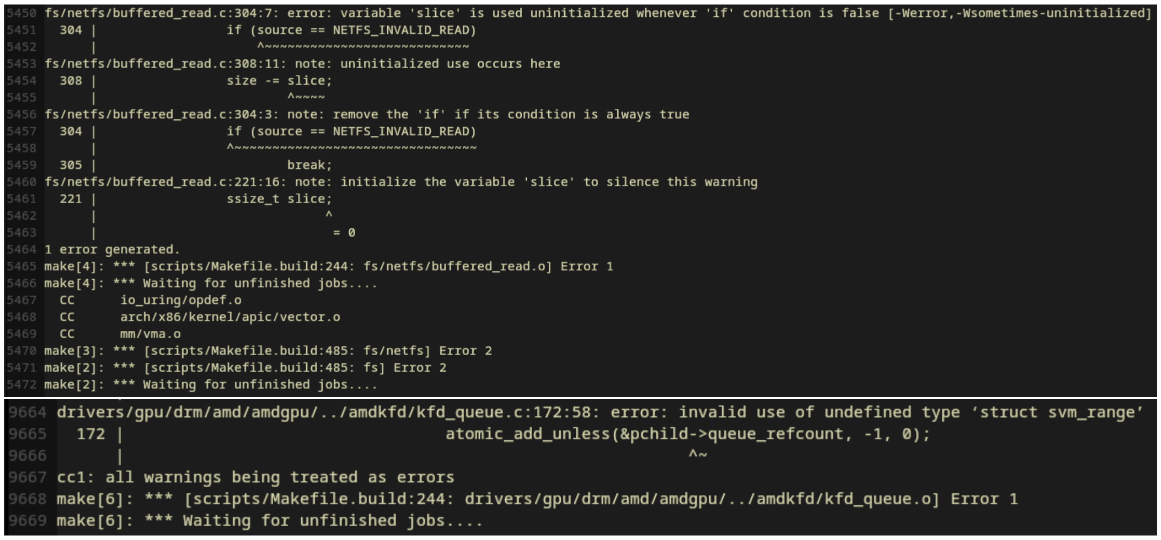
4.4 Choice of dataset

With the program's logic theory developed, a dataset must be selected for testing. Test fails already associated with an issue by maintainers were chosen for this purpose. However, it is important to note that such cases are limited because maintainers, who must manually identify issues, often do not populate the kcidb with this information. Doing so would require additional effort solely to support the development of the clustering feature. Instead, they tend to address these issues directly without adding them to the database.

The selected data for this project consists of logs generated by test fails linked to the issues:

- build issue: ”netdev errors”
- build issue: ”struct svm_range error”
- boot issue: ”usb_kill_anchored_urbs panic during boot”
- boot issue: ”RIP: 0010:usercopy_abort+0x74/0x76 kernel panic”

At the beginning of this research project, a fifth issue (group of errors) was considered: ”kern :crit : acpi LNXTHERM:00: Resources present before probing”. However, during the early stages of the project, the log files from the errors related to this issue were deleted from the maestro’s database, making it impossible to include this data in the follow-up tests. Consequently, the author decided to exclude this data and proceed the research using only the four groups mentioned above, as well as downloading the data from the groups to prevent further data loss.



```
fs/netfs/buffered_read.c:304:7: error: variable 'slice' is used uninitialized whenever 'if' condition is false [-Werror,-Wsometimes-uninitialized]
  304 |                 if (source == NETFS_INVALID_READ)
      |                     ^~~~~~~~~~~~~~~~~~~~~~~~~~~~
fs/netfs/buffered_read.c:308:11: note: uninitialized use occurs here
  308 |                 size -= slice;
      |                         ^~~~~
fs/netfs/buffered_read.c:304:3: note: remove the 'if' if its condition is always true
  304 |                 if (source == NETFS_INVALID_READ)
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  305 |                         break;
fs/netfs/buffered_read.c:221:16: note: initialize the variable 'slice' to silence this warning
  221 |                 ssize_t slice;
      |                              ^
      |                               = 0
1 error generated.
make[4]: *** [scripts/Makefile.build:244: fs/netfs/buffered_read.o] Error 1
make[4]: *** Waiting for unfinished jobs....
  CC      io_uring/opdef.o
  CC      arch/x86/kernel/apic/vector.o
  CC      mm/vma.o
make[3]: *** [scripts/Makefile.build:485: fs/netfs] Error 2
make[2]: *** [scripts/Makefile.build:485: fs] Error 2
make[2]: *** Waiting for unfinished jobs....
```

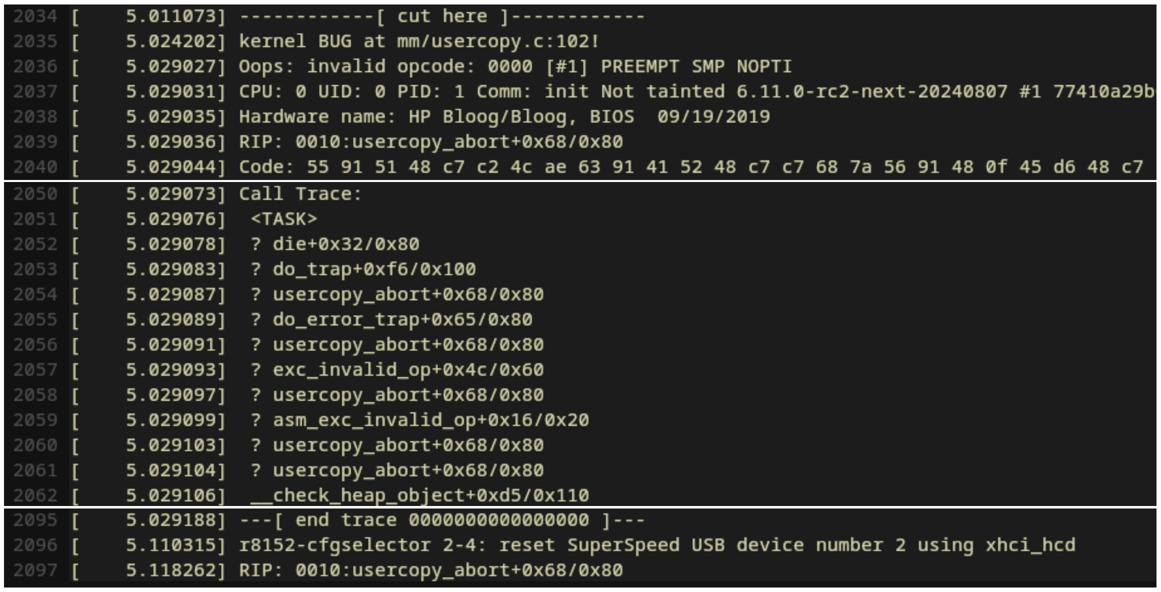
```
drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c:172:58: error: invalid use of undefined type ‘struct svm_range’
  172 |                                       atomic_add_unless(&pchild->queue_refcount, -1, 0);
      |                                                                         ^~
cc1: all warnings being treated as errors
make[6]: *** [scripts/Makefile.build:244: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.o] Error 1
make[6]: *** Waiting for unfinished jobs....
```

Figure 3: Log messages with the build errors: ”netdev” and ”struct svm range”.

Figure 3 show some log lines that may be helpfull to identify the source of the errors for the two build issues presented[3]. For the first issue: ”netdev error” it is possible to see that it is a compilation error throught the messages: `make[4]:  *** [scripts/Makefile.build:244:`

[3]Logs downloaded from Grafana KernelCI through ”netdev errors” issue link and ”struct svm_range error” issue link. Nowadays it is not possible to download these logs from this database because they expired, but they are on the gitlab page of this project.

`fs/netfs/buffered_read.o] Error 1`, also it shows that is a problem in `fs/netfs/buffered_read.o`, in the lines above the log says that a variable may be being used without inialization: `fs/netfs/buffered_read.c:304:7: error: variable 'slice' is used uninitialized whenever 'if' condition is false [-Werror,-Wsometimes-uninitialized]`. These are not the only lines in the log that provide information about the issue, but they are very straightforward. In the same figure 3, it is possible to make a similar analysis regarding the "struct svm range".

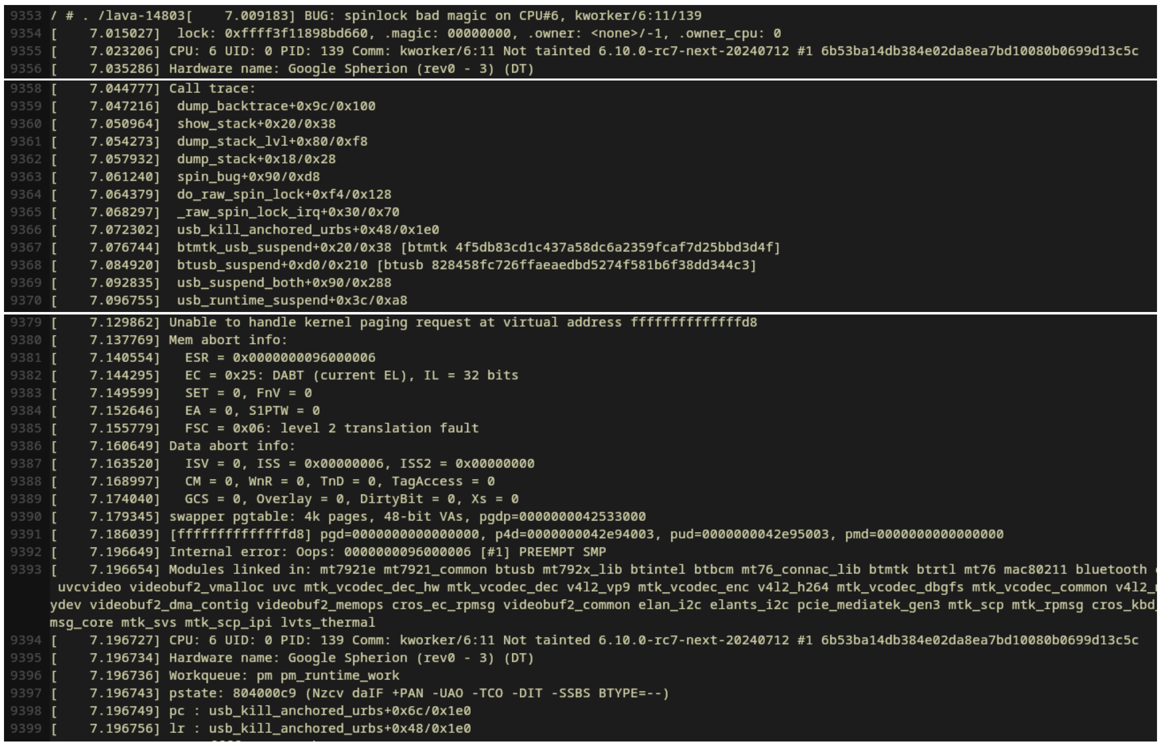


```
[    5.011073] ------------[ cut here ]------------
[    5.024202] kernel BUG at mm/usercopy.c:102!
[    5.029027] Oops: invalid opcode: 0000 [#1] PREEMPT SMP NOPTI
[    5.029031] CPU: 0 UID: 0 PID: 1 Comm: init Not tainted 6.11.0-rc2-next-20240807 #1 77410a29b
[    5.029035] Hardware name: HP Bloog/Bloog, BIOS  09/19/2019
[    5.029036] RIP: 0010:usercopy_abort+0x68/0x80
[    5.029044] Code: 55 91 51 48 c7 c2 4c ae 63 91 41 52 48 c7 c7 68 7a 56 91 48 0f 45 d6 48 c7
[    5.029073] Call Trace:
[    5.029076]  <TASK>
[    5.029078]  ? die+0x32/0x80
[    5.029083]  ? do_trap+0xf6/0x100
[    5.029087]  ? usercopy_abort+0x68/0x80
[    5.029089]  ? do_error_trap+0x65/0x80
[    5.029091]  ? usercopy_abort+0x68/0x80
[    5.029093]  ? exc_invalid_op+0x4c/0x60
[    5.029097]  ? usercopy_abort+0x68/0x80
[    5.029099]  ? asm_exc_invalid_op+0x16/0x20
[    5.029103]  ? usercopy_abort+0x68/0x80
[    5.029104]  ? usercopy_abort+0x68/0x80
[    5.029106]  __check_heap_object+0xd5/0x110
[    5.029188] ---[ end trace 0000000000000000 ]---
[    5.110315] r8152-cfgselector 2-4: reset SuperSpeed USB device number 2 using xhci_hcd
[    5.118262] RIP: 0010:usercopy_abort+0x68/0x80
```

Figure 4: Log messages with the boot error: "RIP: 0010:usercopy_abort+0x74/0x76 kernel panic".

For booting errors, the log messages tend to vary more compared to the build error log messages. The figure 4 displays a portion of the log messages related to the issue "RIP: 0010:usercopy_abort+0x74/0x76 kernel panic". These lines reveal that a traceback occurred, making them a good starting point for finding useful information. Notably, the message `usercopy_abort` appears multiple times within the traceback section (separated by `[ cut here ]` and `[ end trace ]` messages.

Regarding the "usb_kill_anchored_urbs panic during boot" issue on figure 5 there is a bigger diversity of information to analyze. There is a traceback evident on line 9358 on the figure and in the lines surronding the `Call trace:` there are the informations on a `BUG: spinlock`, on line 9353; and the hardware booting the kernel, on line 9356. Furthermore, the log contains details about an unhandled paging request and `pc` and `lr` values on the moment of the traceback.



```
/ # . /lava-14803[    7.009183] BUG: spinlock bad magic on CPU#6, kworker/6:11/139
[    7.015027]  lock: 0xffff3f11898bd660, .magic: 00000000, .owner: <none>/-1, .owner_cpu: 0
[    7.023206] CPU: 6 UID: 0 PID: 139 Comm: kworker/6:11 Not tainted 6.10.0-rc7-next-20240712 #1 6b53ba14db384e02da8ea7bd10080b0699d13c5c
[    7.035286] Hardware name: Google Spherion (rev0 - 3) (DT)
```

```
[    7.044777] Call trace:
[    7.047216]  dump_backtrace+0x9c/0x100
[    7.050964]  show_stack+0x20/0x38
[    7.054273]  dump_stack_lvl+0x80/0xf8
[    7.057932]  dump_stack+0x18/0x28
[    7.061240]  spin_bug+0x90/0xd8
[    7.064379]  do_raw_spin_lock+0xf4/0x128
[    7.068297]  _raw_spin_lock_irq+0x30/0x70
[    7.072302]  usb_kill_anchored_urbs+0x48/0x1e0
[    7.076744]  btmtk_usb_suspend+0x20/0x38 [btmtk 4f5db83cd1c437a58dc6a2359fcaf7d25bbd3d4f]
[    7.084920]  btusb_suspend+0xd0/0x210 [btusb 828458fc726ffaeaedbd5274f581b6f38dd344c3]
[    7.092835]  usb_suspend_both+0x90/0x288
[    7.096755]  usb_runtime_suspend+0x3c/0xa8
```

```
[    7.129862] Unable to handle kernel paging request at virtual address ffffffffffffffd8
[    7.137769] Mem abort info:
[    7.140554]   ESR = 0x0000000096000006
[    7.144295]   EC = 0x25: DABT (current EL), IL = 32 bits
[    7.149599]   SET = 0, FnV = 0
[    7.152646]   EA = 0, S1PTW = 0
[    7.155779]   FSC = 0x06: level 2 translation fault
[    7.160649] Data abort info:
[    7.163520]   ISV = 0, ISS = 0x00000006, ISS2 = 0x00000000
[    7.168997]   CM = 0, WnR = 0, TnD = 0, TagAccess = 0
[    7.174040]   GCS = 0, Overlay = 0, DirtyBit = 0, Xs = 0
[    7.179345] swapper pgtable: 4k pages, 48-bit VAs, pgdp=0000000042533000
[    7.186039] [ffffffffffffffd8] pgd=0000000000000000, p4d=0000000042e94003, pud=0000000042e95003, pmd=0000000000000000
[    7.196649] Internal error: Oops: 0000000096000006 [#1] PREEMPT SMP
[    7.196654] Modules linked in: mt7921e mt7921_common btusb mt792x_lib btintel btbcm mt76_connac_lib btmtk btrtl mt76 mac80211 bluetooth
 uvcvideo videobuf2_vmalloc uvc mtk_vcodec_dec_hw mtk_vcodec_dec v4l2_vp9 mtk_vcodec_enc v4l2_h264 mtk_vcodec_dbgfs mtk_vcodec_common v4l2_
ydev videobuf2_dma_contig videobuf2_memops cros_ec_rpmsg videobuf2_common elan_i2c elants_i2c pcie_mediatek_gen3 mtk_scp mtk_rpmsg cros_kbd_
msg_core mtk_svs mtk_scp_ipi lvts_thermal
[    7.196727] CPU: 6 UID: 0 PID: 139 Comm: kworker/6:11 Not tainted 6.10.0-rc7-next-20240712 #1 6b53ba14db384e02da8ea7bd10080b0699d13c5c
[    7.196734] Hardware name: Google Spherion (rev0 - 3) (DT)
[    7.196736] Workqueue: pm pm_runtime_work
[    7.196743] pstate: 804000c9 (Nzcv daIF +PAN -UAO -TCO -DIT -SSBS BTYPE=--)
[    7.196749] pc : usb_kill_anchored_urbs+0x6c/0x1e0
[    7.196756] lr : usb_kill_anchored_urbs+0x48/0x1e0
```

Figure 5: Log messages with the boot error: "usb_kill_anchored_urbs panic during boot".

4.5 Implementation

Log messages follow many standards and structures, so understanding them can significantly aid in preprocessing the data(on the scope on this project the data were cleaned). An effective preprocessing of the data can substantially increase the program efficiency. In the dataset used for this project, certain lines were removed, along with all numerical values, as shown on the code 1.

Minimal cleaning was performed on the data, as the dataset was very limited, and excessive cleaning could risk overfitting. It is worth mentioning that the TF-IDF algorithm is more effective when applied to larger datasets.

Listing 1: Preprocessing content from log

```
# remove waiting lines of the compilation
log = "\n".join(l for l in log.splitlines() if '.........' not in l)
# remove numbers
log = re.sub(r'\d+', '', log)
# remove empty lines
log = "\n".join(l.strip() for l in log.splitlines() if l.strip())
```

For the TF-IDF implementation, the term was defined as the whole line as shown on the code 2. For the dimensionality reduction and clusterization it was used known python libraries.

Listing 2: TF-IDF function

```
from sklearn.feature_extraction.text import TfidfVectorizer
# tokenize the whole line
TfidfVectorizer(token_pattern=r"(?u).*\n")
# find sparse matrix
tfidf_matrix = vectorizer.fit_transform(logs)
```

5 Results and discussions

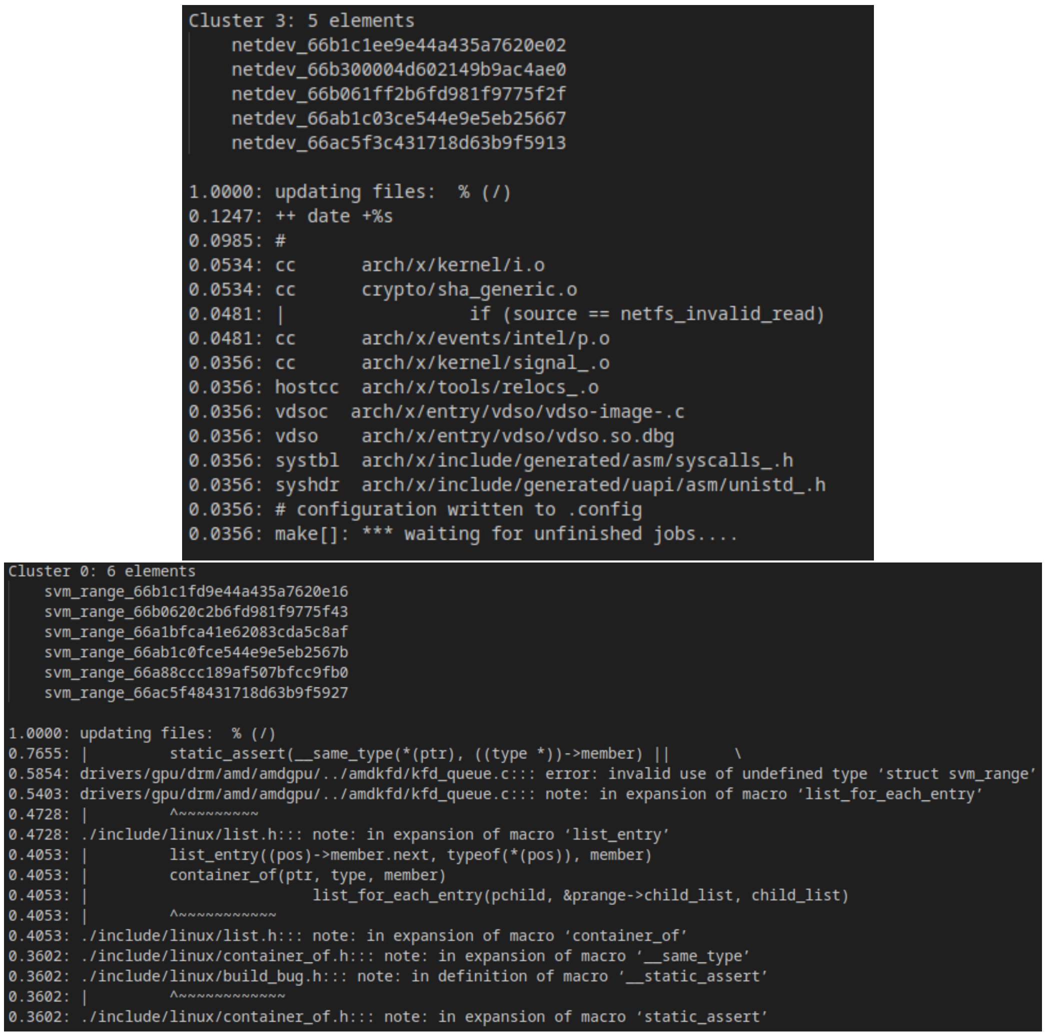
The results of the KMeans clustering with cosine distance on the TF-IDF matrix and no dimensionality reduce algorithm applied are shown on figure 6 and figure 7[4]. In general, it can be concluded that even with the simplest clustering method (KMeans without dimensionality reduction), the clustering results are reasonably satisfactory. Five clusters were produced instead of the expected four. All 'netdev' elements are grouped in cluster 3, all 'svm_range' elements are in cluster 0, all 'usb_kill' elements are in cluster 2, while 'usercopy_abort' elements are distributed between clusters 1 and 4.

While clustering alone can assist maintainers and developers in examining sets of errors associated with the same issue, providing more detailed information from the logs can further enhance their understanding. Thus, figure 6 and figure 7 also displays the most valuable terms in each cluster—terms that are both frequent within the cluster and infrequent outside of it.

Analyzing figure 6, in the 'netdev' cluster, the algorithm provides 15 lines. Based solely on the information in these lines, it is possible to conclude that the error occurred during compilation (`make[]: *** waiting for unfinished jobs....`), and that `| if (source == netfs_invalid_read)` is a promising direction for further investigation. Although the critical line (line 5450 of figure 3, which directly indicates the problem) is not included, the provided lines may still offer valuable guidance to maintainers and developers.

For the 'struct svm_range' issue, the error is directly indicated by the third most valuable term presented in the cluster, the line `drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type 'struct svm_range'`. Additionally, it is noticeable that several lines immediately following an important line, such as `| ^~~~~~~~~~` were included. This observation could be considered in future work by adding a preprocessing step that assigns higher value to the lines immediately preceding those containing `| ^~~~~~~~~~`.

[4]The results for other setups are provided in the appendices.



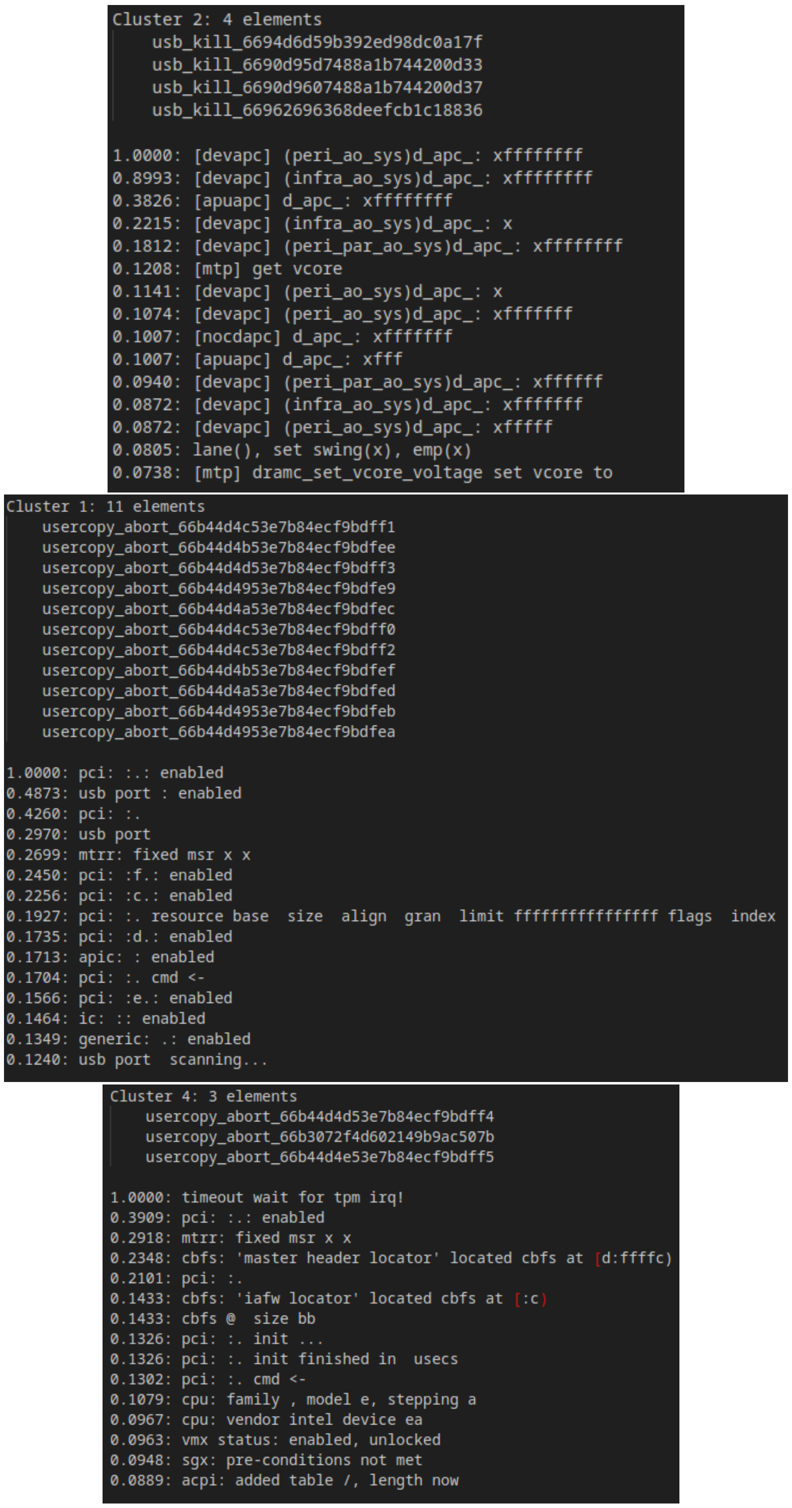
```
Cluster 3: 5 elements
    netdev_66b1c1ee9e44a435a7620e02
    netdev_66b300004d602149b9ac4ae0
    netdev_66b061ff2b6fd981f9775f2f
    netdev_66ab1c03ce544e9e5eb25667
    netdev_66ac5f3c431718d63b9f5913

1.0000: updating files:  % (/)
0.1247: ++ date +%s
0.0985: #
0.0534: cc      arch/x/kernel/i.o
0.0534: cc      crypto/sha_generic.o
0.0481: |                 if (source == netfs_invalid_read)
0.0481: cc      arch/x/events/intel/p.o
0.0356: cc      arch/x/kernel/signal_.o
0.0356: hostcc  arch/x/tools/relocs_.o
0.0356: vdsoc  arch/x/entry/vdso/vdso-image-.c
0.0356: vdso    arch/x/entry/vdso/vdso.so.dbg
0.0356: systbl  arch/x/include/generated/asm/syscalls_.h
0.0356: syshdr  arch/x/include/generated/uapi/asm/unistd_.h
0.0356: # configuration written to .config
0.0356: make[]: *** waiting for unfinished jobs....
```

```
Cluster 0: 6 elements
    svm_range_66b1c1fd9e44a435a7620e16
    svm_range_66b0620c2b6fd981f9775f43
    svm_range_66a1bfca41e62083cda5c8af
    svm_range_66ab1c0fce544e9e5eb2567b
    svm_range_66a88ccc189af507bfcc9fb0
    svm_range_66ac5f48431718d63b9f5927

1.0000: updating files:  % (/)
0.7655: |         static_assert(__same_type(*(ptr), ((type *))->member) ||       \
0.5854: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type ‘struct svm_range’
0.5403: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: note: in expansion of macro ‘list_for_each_entry’
0.4728: |         ^~~~~~~~~~
0.4728: ./include/linux/list.h::: note: in expansion of macro ‘list_entry’
0.4053: |         list_entry((pos)->member.next, typeof(*(pos)), member)
0.4053: |         container_of(ptr, type, member)
0.4053: |                         list_for_each_entry(pchild, &prange->child_list, child_list)
0.4053: |         ^~~~~~~~~~~~
0.4053: ./include/linux/list.h::: note: in expansion of macro ‘container_of’
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘__same_type’
0.3602: ./include/linux/build_bug.h::: note: in definition of macro ‘__static_assert’
0.3602: |         ^~~~~~~~~~~~~
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘static_assert’
```

Figure 6: Clusters of ”netdev” and ”struct svm_range” errors and 15 of most scored terms, using KMeans with cosine distance matrix.

The results from the boot errors, in figure 7, were less obvious and direct compared to the build errors analyzed cases. Although the clustering was relatively accurate, the information provided by the valuable terms could have been better focused to aid in identifying the underlying issues.

This may be explored through more focused preprocessing of the data, which may include cleaning noisy parts, adjusting the weighting in the TF-IDF function to give more importance to rarer messages, or prioritizing lines containing specific keywords such as ’BUG’ or ’kernel panic’.



```
Cluster 2: 4 elements
    usb_kill_6694d6d59b392ed98dc0a17f
    usb_kill_6690d95d7488a1b744200d33
    usb_kill_6690d9607488a1b744200d37
    usb_kill_66962696368deefcb1c18836

1.0000: [devapc] (peri_ao_sys)d_apc_: xffffffff
0.8993: [devapc] (infra_ao_sys)d_apc_: xffffffff
0.3826: [apuapc] d_apc_: xffffffff
0.2215: [devapc] (infra_ao_sys)d_apc_: x
0.1812: [devapc] (peri_par_ao_sys)d_apc_: xffffffff
0.1208: [mtp] get vcore
0.1141: [devapc] (peri_ao_sys)d_apc_: x
0.1074: [devapc] (peri_ao_sys)d_apc_: xfffffff
0.1007: [nocdapc] d_apc_: xfffffff
0.1007: [apuapc] d_apc_: xfff
0.0940: [devapc] (peri_par_ao_sys)d_apc_: xffffff
0.0872: [devapc] (infra_ao_sys)d_apc_: xfffffff
0.0872: [devapc] (peri_ao_sys)d_apc_: xfffff
0.0805: lane(), set swing(x), emp(x)
0.0738: [mtp] dramc_set_vcore_voltage set vcore to
```

```
Cluster 1: 11 elements
    usercopy_abort_66b44d4c53e7b84ecf9bdff1
    usercopy_abort_66b44d4b53e7b84ecf9bdfee
    usercopy_abort_66b44d4d53e7b84ecf9bdff3
    usercopy_abort_66b44d4953e7b84ecf9bdfe9
    usercopy_abort_66b44d4a53e7b84ecf9bdfec
    usercopy_abort_66b44d4c53e7b84ecf9bdff0
    usercopy_abort_66b44d4c53e7b84ecf9bdff2
    usercopy_abort_66b44d4b53e7b84ecf9bdfef
    usercopy_abort_66b44d4a53e7b84ecf9bdfed
    usercopy_abort_66b44d4953e7b84ecf9bdfeb
    usercopy_abort_66b44d4953e7b84ecf9bdfea

1.0000: pci: :.: enabled
0.4873: usb port : enabled
0.4260: pci: :.
0.2970: usb port
0.2699: mtrr: fixed msr x x
0.2450: pci: :f.: enabled
0.2256: pci: :c.: enabled
0.1927: pci: :. resource base  size  align  gran  limit ffffffffffffffff flags  index
0.1735: pci: :d.: enabled
0.1713: apic: : enabled
0.1704: pci: :. cmd <-
0.1566: pci: :e.: enabled
0.1464: ic: :: enabled
0.1349: generic: .: enabled
0.1240: usb port  scanning...
```

```
Cluster 4: 3 elements
    usercopy_abort_66b44d4d53e7b84ecf9bdff4
    usercopy_abort_66b3072f4d602149b9ac507b
    usercopy_abort_66b44d4e53e7b84ecf9bdff5

1.0000: timeout wait for tpm irq!
0.3909: pci: :.: enabled
0.2918: mtrr: fixed msr x x
0.2348: cbfs: 'master header locator' located cbfs at [d:ffffc)
0.2101: pci: :.
0.1433: cbfs: 'iafw locator' located cbfs at [:c)
0.1433: cbfs @  size bb
0.1326: pci: :. init ...
0.1326: pci: :. init finished in  usecs
0.1302: pci: :. cmd <-
0.1079: cpu: family , model e, stepping a
0.0967: cpu: vendor intel device ea
0.0963: vmx status: enabled, unlocked
0.0948: sgx: pre-conditions not met
0.0889: acpi: added table /, length now
```

Figure 7: Clusters of ”usb_kill_anchored_urbs panic during boot” and ”RIP: 0010:usercopy_abort+0x74/0x76 kernel panic” errors and 15 of most scored terms, using KMeans with cosine distance matrix.

After exploring various algorithms, it became evident that simpler alternatives often provided comparable or even superior performance. This insight highlighted the value of prioritizing a deep understanding of log structures and refining pre-processing techniques, which ultimately emerged as a more effective and reliable strategy compared to the application of overly complex data processing methods.

6 Conclusions and Future Works

In conclusion, this project demonstrated that it is possible to significantly reduce labor-intensive tasks with a a straightforward yet effective approach. A valuable next step would involve deepening the understanding of log structures and experimenting with alternative pre-processing methods. The flexibility of the TF-IDF algorithm presents opportunities for customization; for instance, increasing the weight of the IDF component could enhance the value assigned to rare terms, as these are more likely to carry critical information compared to frequently occurring terms. Additionally, investigating common patterns and terms within log messages, particularly those associated with errors or issues, could provide further insights and improve the effectiveness of future implementations. TF-IDF algorithm can also be used for anomaly detector, a scope not approached by this project.

An alternative approach worth exploring would involve applying an initial rough clustering technique to partition the data into broad groups. This preliminary step could facilitate a more focused and refined analysis within each cluster, allowing for tailored processing and potentially uncovering patterns or insights that might be overlooked in a more uniform approach.

7 Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Islene Garcia, for her guidance throughout my academic journey. She has supported me on various projects during my graduation, inspiring me to engage with numerous free software initiatives and become involved in their communities. Additionally, I would like to thank my friend and project advisor, Helen Koike, who introduced me to the Linux Kernel project six years ago and to KernelCI earlier this year. Her mentorship has been instrumental in deepening my understanding of the kernel and its community. I am truly grateful for her encouragement and the insightful conversations we had. I would also like to extend my thanks to the KernelCI community for welcoming me to their weekly meetings, listening to my project proposal, and offering valuable suggestions and references.

I also want to express my deep appreciation to the members of LKCamp (the Linux Kernel development group at UNICAMP). My fellow LKCampers made my graduation a truly special and enriching experience. The weekends spent in the IC labs tackling new kernel challenges, the inspiring individuals who have been part of the group over the years, the coffee breaks, the fun jokes, and the exciting conversations at the bar were all essential to my growth as a kernel developer. A special thanks to my friends Helen Koike and André Almeida, who supported me from the beginning of my kernel development journey, encouraging me through internship projects, and to Vinícius Peixoto, with whom I've shared great ideas for the future of LKCamp and who helped me during times of demotivation and doubt in the final stages of my graduation.

Last but not least, I would like to express my gratitude to the many friends I made during my graduation. They have become an essential part of my support network and have played a key role in both my personal and professional growth, in special I thank Pedro Orlando and Arathi. I am also deeply thankful to my family, especially my mother and siblings, who have been my first and strongest source of support and encouragement.

Afterall, maybe the true clustering are the relationship we build along the journey.

References

[1] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. "Density-based clustering based on hierarchical density estimates". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172.

[2] CKI Project community. *Reporting KCIDB results*. URL: `https://cki-project.org/docs/test-maintainers/status-meaning/` (visited on 12/07/2024).

[3] Grafana KernelCI Project community. *Grafana KernelCI dashboads*. URL: `https://grafana.kernelci.org/d/maestro-home/` (visited on 12/07/2024).

[4] Linux Kernel community. *KUnit - Linux Kernel Unit Testing*. URL: `https://www.kernel.org/doc/html/v6.13-rc1/dev-tools/kunit/index.html` (visited on 12/07/2024).

[5] Linux Kernel community. *Linux Kernel Selftests*. URL: `https://www.kernel.org/doc/html/v6.13-rc1/dev-tools/kselftest.html` (visited on 12/07/2024).

[6] The Kernel development community. *Development statistics for the –version kernel*. URL: `https://lwn.net/Articles/956765/`(v. 6.7), `https://lwn.net/Articles/964106/`(v. 6.8), `https://lwn.net/Articles/972605/`(v. 6.9), `https://lwn.net/Articles/981559/`(v. 6.10), `https://lwn.net/Articles/989528/`(v. 6.11), `https://lwn.net/SubscriberLink/997959/311c3b27c8561938/`(v. 6.12), last accessed on 2024.11.25.

[7] The Kernel development community. *Linux Kernel Teaching [online]*. URL: `https://linux-kernel-labs.github.io/` (visited on 11/30/2024).

[8] The KernelCI development community. *KernelCI Architecture*. URL: `https://docs.kernelci.org/architecture/` (visited on 11/22/2024).

[9] The KernelCI development community. *KernelCI Documentation*. URL: `https://github.com/kernelci/kcidb/tree/main/doc` (visited on 11/22/2024).

[10] Dingsheng Deng. "DBSCAN Clustering Algorithm Based on Density". In: *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*. 2020, pp. 949–953. DOI: `10.1109/IFEEA51475.2020.00199`.

[11] Linux Foundation. *A Beginner's Guide to Linux Kernel Development [online]*. URL: https://trainingportal.linuxfoundation.org/courses/a-beginners-guide-to-linux-kernel-development-lfd103 (visited on 11/30/2024).

[12] Mark Johnston. *Kernel Fuzzing with syskaller*. Ed. by FreeBSD Foundation. Available on: https://freebsdfoundation.org/wp-content/uploads/2021/01/Kernel-Fuzzing.pdf (last accessed on 12/07/2024). FreeBSD Journal, 2020.

[13] Ondrej Klinovský and Adam Rambousek. "Detect Anomalies in Linux Kernel Logs". Available on: https://is.muni.cz/th/ug8ef/detect-anomalies-in-linux-kernel-logs.pdf (last accessed on 12/07/2024). MA thesis. Masaryk University, 2022.

[14] Laurence Morissette and Sylvain Chartier. "The k-means clustering technique: General considerations and implementation in Mathematica". In: *Tutorials in Quantitative Methods for Psychology* 9.1 (2013), pp. 15–24.

[15] Alina Petukhova, Joao P Matos-Carvalho, and Nuno Fachada. "Text clustering with LLM embeddings". In: *International Journal of Cognitive Computing in Engineering* 6 (2024), pp. 100–108. ISSN: 2666-3074. DOI: https://doi.org/10.1016/j.ijcce.2024.11.004. URL: https://www.sciencedirect.com/science/article/pii/S2666307424000482.

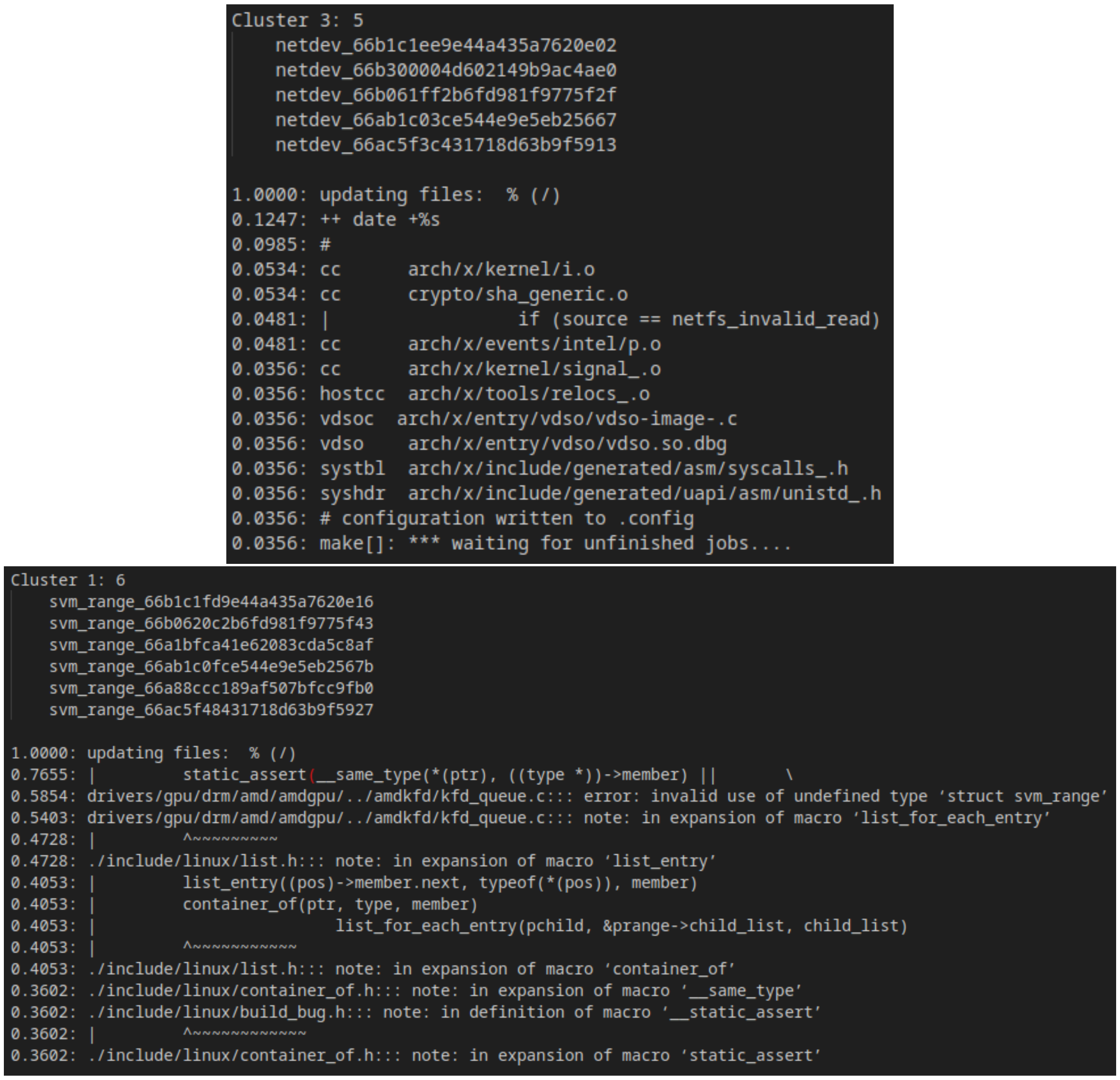
[16] Intel 0-day Project. *Linux Kernel Performance*. URL: https://www.intel.com/content/www/us/en/developer/topic-technology/open/linux-kernel-performance/overview.html (visited on 12/07/2024).

[17] Anand Rajaraman. "Mining of massive datasets". In: Cambridge University Press, 2011. Chap. Data Mining, pp. 1–17. ISBN: 978-1-108-47634-8. DOI: 10.1017/9781108684163.

Appendices

A KMeans Clustering results

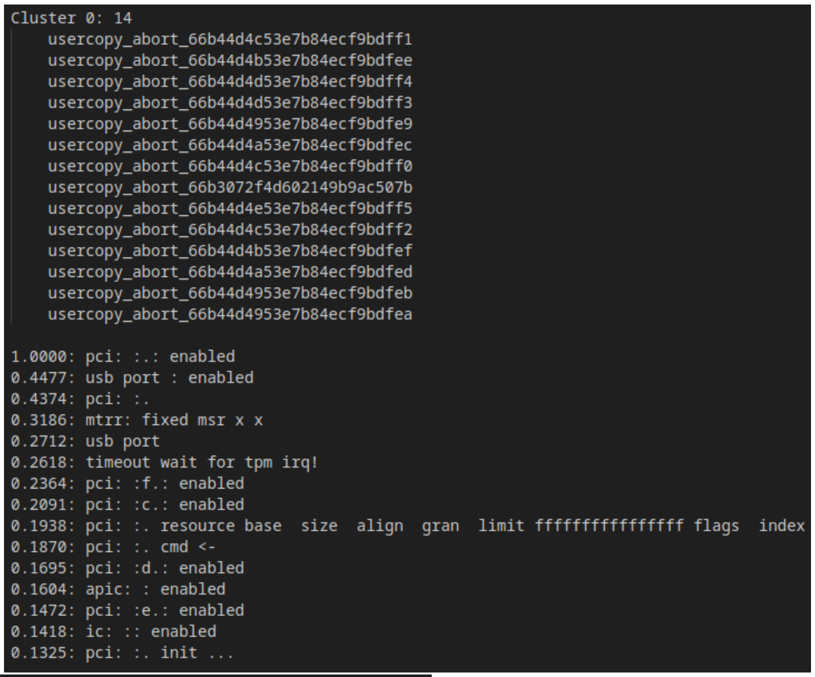
A.1 SVD - dimentionality reduction to 31 components



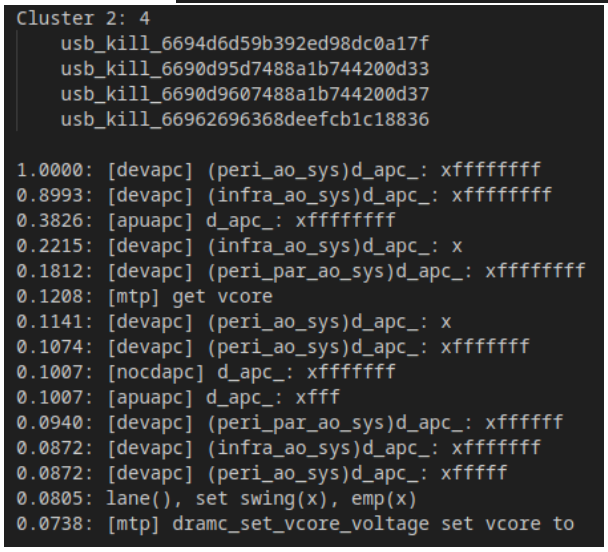
```
Cluster 3: 5
    netdev_66b1c1ee9e44a435a7620e02
    netdev_66b300004d602149b9ac4ae0
    netdev_66b061ff2b6fd981f9775f2f
    netdev_66ab1c03ce544e9e5eb25667
    netdev_66ac5f3c431718d63b9f5913

1.0000: updating files:  % (/)
0.1247: ++ date +%s
0.0985: #
0.0534: cc      arch/x/kernel/i.o
0.0534: cc      crypto/sha_generic.o
0.0481: |                 if (source == netfs_invalid_read)
0.0481: cc      arch/x/events/intel/p.o
0.0356: cc      arch/x/kernel/signal_.o
0.0356: hostcc  arch/x/tools/relocs_.o
0.0356: vdsoc  arch/x/entry/vdso/vdso-image-.c
0.0356: vdso    arch/x/entry/vdso/vdso.so.dbg
0.0356: systbl  arch/x/include/generated/asm/syscalls_.h
0.0356: syshdr  arch/x/include/generated/uapi/asm/unistd_.h
0.0356: # configuration written to .config
0.0356: make[]: *** waiting for unfinished jobs....
```

```
Cluster 1: 6
    svm_range_66b1c1fd9e44a435a7620e16
    svm_range_66b0620c2b6fd981f9775f43
    svm_range_66a1bfca41e62083cda5c8af
    svm_range_66ab1c0fce544e9e5eb2567b
    svm_range_66a88ccc189af507bfcc9fb0
    svm_range_66ac5f48431718d63b9f5927

1.0000: updating files:  % (/)
0.7655: |         static_assert(__same_type(*(ptr), ((type *))->member) ||       \
0.5854: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type 'struct svm_range'
0.5403: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: note: in expansion of macro 'list_for_each_entry'
0.4728: |         ^~~~~~~~~~
0.4728: ./include/linux/list.h::: note: in expansion of macro 'list_entry'
0.4053: |         list_entry((pos)->member.next, typeof(*(pos)), member)
0.4053: |         container_of(ptr, type, member)
0.4053: |                         list_for_each_entry(pchild, &prange->child_list, child_list)
0.4053: |         ^~~~~~~~~~~~
0.4053: ./include/linux/list.h::: note: in expansion of macro 'container_of'
0.3602: ./include/linux/container_of.h::: note: in expansion of macro '__same_type'
0.3602: ./include/linux/build_bug.h::: note: in definition of macro '__static_assert'
0.3602: |         ^~~~~~~~~~~~~
0.3602: ./include/linux/container_of.h::: note: in expansion of macro 'static_assert'
```

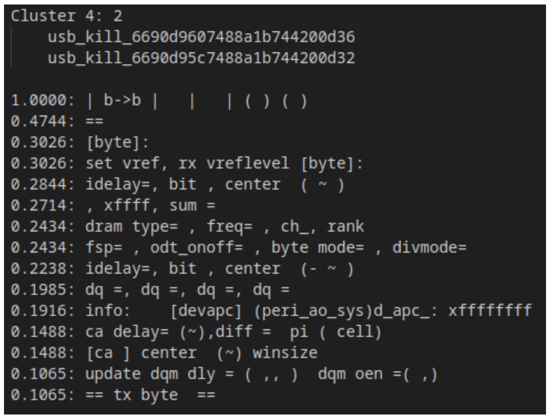
Figure 8: Clusters of "netdev" and "struct svm range" errors and 15 of most scored terms.



```
Cluster 0: 14
    usercopy_abort_66b44d4c53e7b84ecf9bdff1
    usercopy_abort_66b44d4b53e7b84ecf9bdfee
    usercopy_abort_66b44d4d53e7b84ecf9bdff4
    usercopy_abort_66b44d4d53e7b84ecf9bdff3
    usercopy_abort_66b44d4953e7b84ecf9bdfe9
    usercopy_abort_66b44d4a53e7b84ecf9bdfec
    usercopy_abort_66b44d4c53e7b84ecf9bdff0
    usercopy_abort_66b3072f4d602149b9ac507b
    usercopy_abort_66b44d4e53e7b84ecf9bdff5
    usercopy_abort_66b44d4c53e7b84ecf9bdff2
    usercopy_abort_66b44d4b53e7b84ecf9bdfef
    usercopy_abort_66b44d4a53e7b84ecf9bdfed
    usercopy_abort_66b44d4953e7b84ecf9bdfeb
    usercopy_abort_66b44d4953e7b84ecf9bdfea

1.0000: pci: :.: enabled
0.4477: usb port : enabled
0.4374: pci: :.
0.3186: mtrr: fixed msr x x
0.2712: usb port
0.2618: timeout wait for tpm irq!
0.2364: pci: :f.: enabled
0.2091: pci: :c.: enabled
0.1938: pci: :. resource base  size  align  gran  limit ffffffffffffffff flags  index
0.1870: pci: :. cmd <-
0.1695: pci: :d.: enabled
0.1604: apic: : enabled
0.1472: pci: :e.: enabled
0.1418: ic: :: enabled
0.1325: pci: :. init ...
```



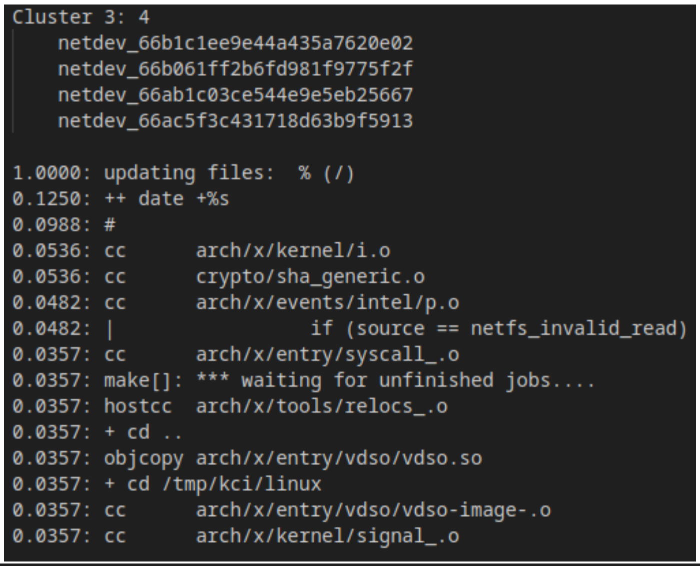
```
Cluster 2: 4
    usb_kill_6694d6d59b392ed98dc0a17f
    usb_kill_6690d95d7488a1b744200d33
    usb_kill_6690d9607488a1b744200d37
    usb_kill_66962696368deefcb1c18836

1.0000: [devapc] (peri_ao_sys)d_apc_: xffffffff
0.8993: [devapc] (infra_ao_sys)d_apc_: xffffffff
0.3826: [apuapc] d_apc_: xffffffff
0.2215: [devapc] (infra_ao_sys)d_apc_: x
0.1812: [devapc] (peri_par_ao_sys)d_apc_: xffffffff
0.1208: [mtp] get vcore
0.1141: [devapc] (peri_ao_sys)d_apc_: x
0.1074: [devapc] (peri_ao_sys)d_apc_: xfffffff
0.1007: [nocdapc] d_apc_: xfffffff
0.1007: [apuapc] d_apc_: xfff
0.0940: [devapc] (peri_par_ao_sys)d_apc_: xffffff
0.0872: [devapc] (infra_ao_sys)d_apc_: xfffffff
0.0872: [devapc] (peri_ao_sys)d_apc_: xfffff
0.0805: lane(), set swing(x), emp(x)
0.0738: [mtp] dramc_set_vcore_voltage set vcore to
```



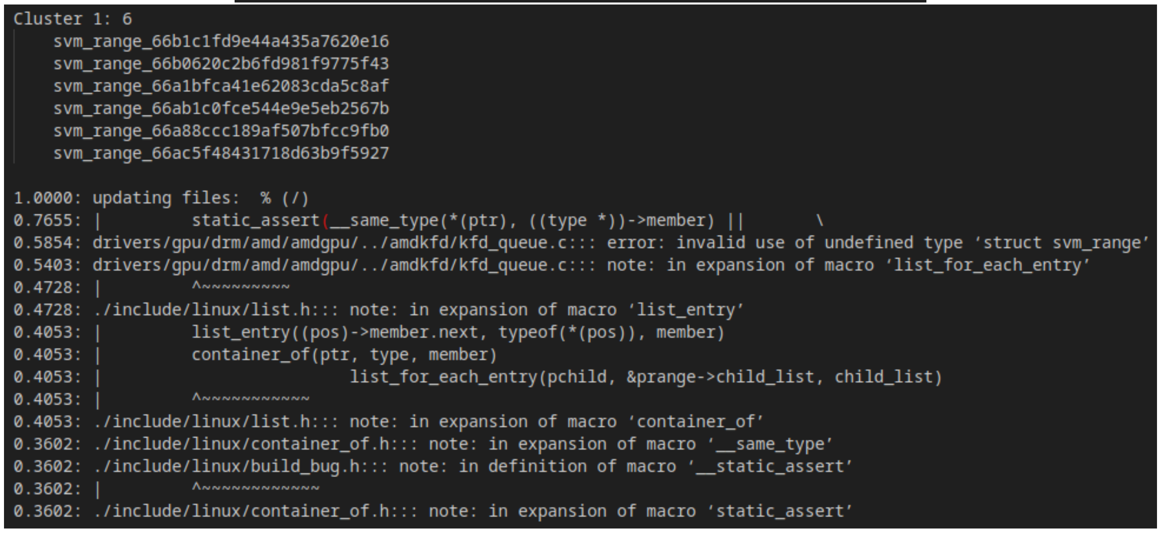
```
Cluster 4: 2
    usb_kill_6690d9607488a1b744200d36
    usb_kill_6690d95c7488a1b744200d32

1.0000: | b->b |   |   | ( ) ( )
0.4744: ==
0.3026: [byte]:
0.3026: set vref, rx vreflevel [byte]:
0.2844: idelay=, bit , center  ( ~ )
0.2714: , xffff, sum =
0.2434: dram type= , freq= , ch_, rank
0.2434: fsp= , odt_onoff= , byte mode= , divmode=
0.2238: idelay=, bit , center  (- ~ )
0.1985: dq =, dq =, dq =, dq =
0.1916: info:    [devapc] (peri_ao_sys)d_apc_: xffffffff
0.1488: ca delay= (~),diff =  pi ( cell)
0.1488: [ca ] center  (~) winsize
0.1065: update dqm dly = ( ,, )  dqm oen =( ,)
0.1065: == tx byte  ==
```

Figure 9: Clusters of ”usb_kill_anchored_urbs panic during boot” and ”RIP: 0010:usercopy_abort+0x74/0x76 kernel panic” errors and 15 of most scored terms.

A.2 NMF - dimentionality reduction to 31 components

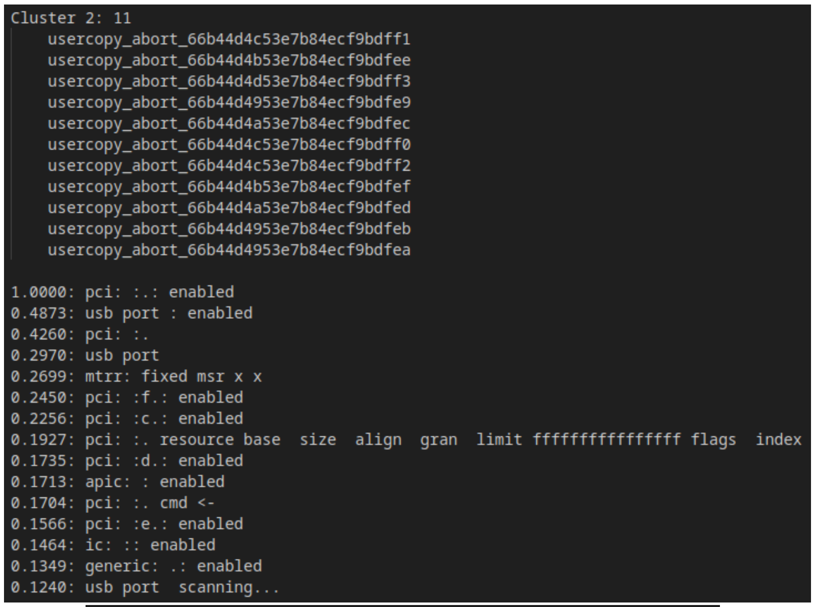


```
Cluster 3: 4
    netdev_66b1c1ee9e44a435a7620e02
    netdev_66b061ff2b6fd981f9775f2f
    netdev_66ab1c03ce544e9e5eb25667
    netdev_66ac5f3c431718d63b9f5913

1.0000: updating files:  % (/)
0.1250: ++ date +%s
0.0988: #
0.0536: cc      arch/x/kernel/i.o
0.0536: cc      crypto/sha_generic.o
0.0482: cc      arch/x/events/intel/p.o
0.0482: |                 if (source == netfs_invalid_read)
0.0357: cc      arch/x/entry/syscall_.o
0.0357: make[]: *** waiting for unfinished jobs....
0.0357: hostcc  arch/x/tools/relocs_.o
0.0357: + cd ..
0.0357: objcopy arch/x/entry/vdso/vdso.so
0.0357: + cd /tmp/kci/linux
0.0357: cc      arch/x/entry/vdso/vdso-image-.o
0.0357: cc      arch/x/kernel/signal_.o
```

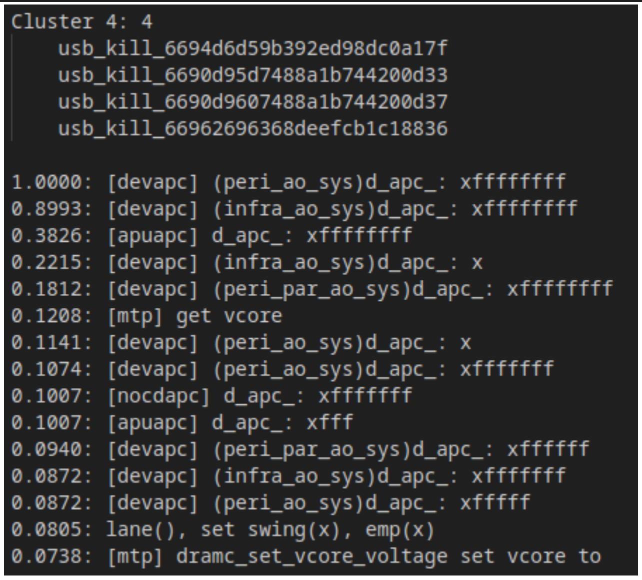


```
Cluster 1: 6
    svm_range_66b1c1fd9e44a435a7620e16
    svm_range_66b0620c2b6fd981f9775f43
    svm_range_66a1bfca41e62083cda5c8af
    svm_range_66ab1c0fce544e9e5eb2567b
    svm_range_66a88ccc189af507bfcc9fb0
    svm_range_66ac5f48431718d63b9f5927

1.0000: updating files:  % (/)
0.7655: |         static_assert(__same_type(*(ptr), ((type *))->member) ||       \
0.5854: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type ‘struct svm_range’
0.5403: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: note: in expansion of macro ‘list_for_each_entry’
0.4728: |         ^~~~~~~~~~
0.4728: ./include/linux/list.h::: note: in expansion of macro ‘list_entry’
0.4053: |         list_entry((pos)->member.next, typeof(*(pos)), member)
0.4053: |         container_of(ptr, type, member)
0.4053: |                         list_for_each_entry(pchild, &prange->child_list, child_list)
0.4053: |         ^~~~~~~~~~~~
0.4053: ./include/linux/list.h::: note: in expansion of macro ‘container_of’
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘__same_type’
0.3602: ./include/linux/build_bug.h::: note: in definition of macro ‘__static_assert’
0.3602: |         ^~~~~~~~~~~~~
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘static_assert’
```

Figure 10: Clusters of ”netdev” and ”struct svm range” errors and 15 of most scored terms.

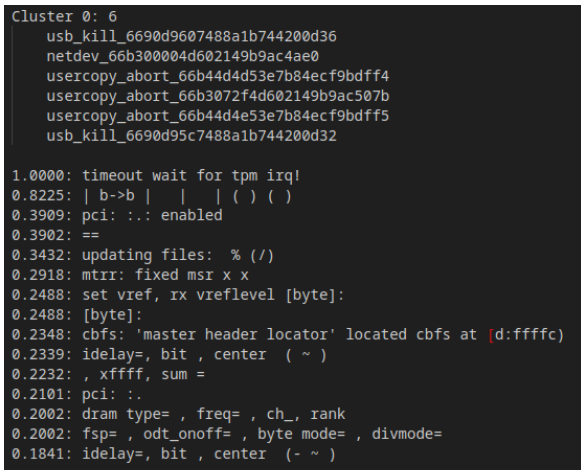


```
Cluster 2: 11
    usercopy_abort_66b44d4c53e7b84ecf9bdff1
    usercopy_abort_66b44d4b53e7b84ecf9bdfee
    usercopy_abort_66b44d4d53e7b84ecf9bdff3
    usercopy_abort_66b44d4953e7b84ecf9bdfe9
    usercopy_abort_66b44d4a53e7b84ecf9bdfec
    usercopy_abort_66b44d4c53e7b84ecf9bdff0
    usercopy_abort_66b44d4c53e7b84ecf9bdff2
    usercopy_abort_66b44d4b53e7b84ecf9bdfef
    usercopy_abort_66b44d4a53e7b84ecf9bdfed
    usercopy_abort_66b44d4953e7b84ecf9bdfeb
    usercopy_abort_66b44d4953e7b84ecf9bdfea

1.0000: pci: :.: enabled
0.4873: usb port : enabled
0.4260: pci: :.
0.2970: usb port
0.2699: mtrr: fixed msr x x
0.2450: pci: :f.: enabled
0.2256: pci: :c.: enabled
0.1927: pci: :. resource base  size  align  gran  limit ffffffffffffffff flags  index
0.1735: pci: :d.: enabled
0.1713: apic: : enabled
0.1704: pci: :. cmd <-
0.1566: pci: :e.: enabled
0.1464: ic: :: enabled
0.1349: generic: .: enabled
0.1240: usb port  scanning...
```



```
Cluster 4: 4
    usb_kill_6694d6d59b392ed98dc0a17f
    usb_kill_6690d95d7488a1b744200d33
    usb_kill_6690d9607488a1b744200d37
    usb_kill_66962696368deefcb1c18836

1.0000: [devapc] (peri_ao_sys)d_apc_: xffffffff
0.8993: [devapc] (infra_ao_sys)d_apc_: xffffffff
0.3826: [apuapc] d_apc_: xffffffff
0.2215: [devapc] (infra_ao_sys)d_apc_: x
0.1812: [devapc] (peri_par_ao_sys)d_apc_: xffffffff
0.1208: [mtp] get vcore
0.1141: [devapc] (peri_ao_sys)d_apc_: x
0.1074: [devapc] (peri_ao_sys)d_apc_: xfffffff
0.1007: [nocdapc] d_apc_: xfffffff
0.1007: [apuapc] d_apc_: xfff
0.0940: [devapc] (peri_par_ao_sys)d_apc_: xffffff
0.0872: [devapc] (infra_ao_sys)d_apc_: xfffffff
0.0872: [devapc] (peri_ao_sys)d_apc_: xfffff
0.0805: lane(), set swing(x), emp(x)
0.0738: [mtp] dramc_set_vcore_voltage set vcore to
```

Figure 11: Clusters of "usb_kill_anchored_urbs panic during boot" and "RIP: 0010:usercopy_abort+0x74/0x76 kernel panic" errors and 15 of most scored terms.

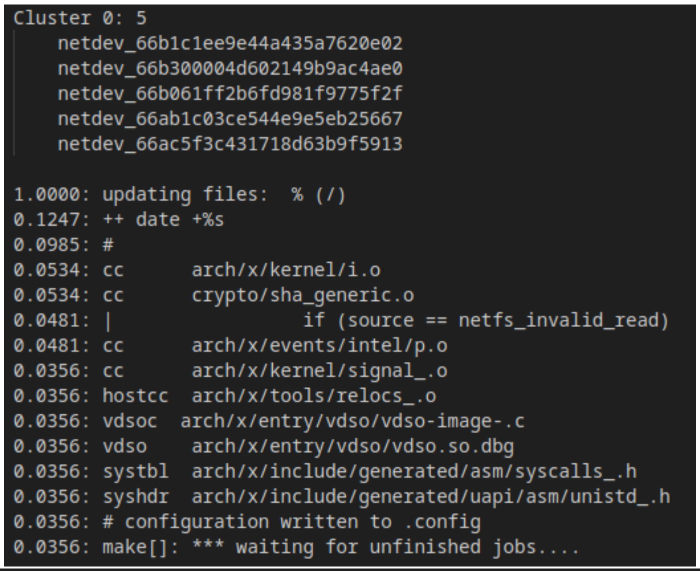


```
Cluster 0: 6
    usb_kill_6690d9607488a1b744200d36
    netdev_66b300004d602149b9ac4ae0
    usercopy_abort_66b44d4d53e7b84ecf9bdff4
    usercopy_abort_66b3072f4d602149b9ac507b
    usercopy_abort_66b44d4e53e7b84ecf9bdff5
    usb_kill_6690d95c7488a1b744200d32

1.0000: timeout wait for tpm irq!
0.8225: | b->b |   |   | ( ) ( )
0.3909: pci: :.: enabled
0.3902: ==
0.3432: updating files:  % (/)
0.2918: mtrr: fixed msr x x
0.2488: set vref, rx vreflevel [byte]:
0.2488: [byte]:
0.2348: cbfs: 'master header locator' located cbfs at [d:ffffc)
0.2339: idelay=, bit , center  ( ~ )
0.2232: , xffff, sum =
0.2101: pci: :.
0.2002: dram type= , freq= , ch_, rank
0.2002: fsp= , odt_onoff= , byte mode= , divmode=
0.1841: idelay=, bit , center  (- ~ )
```

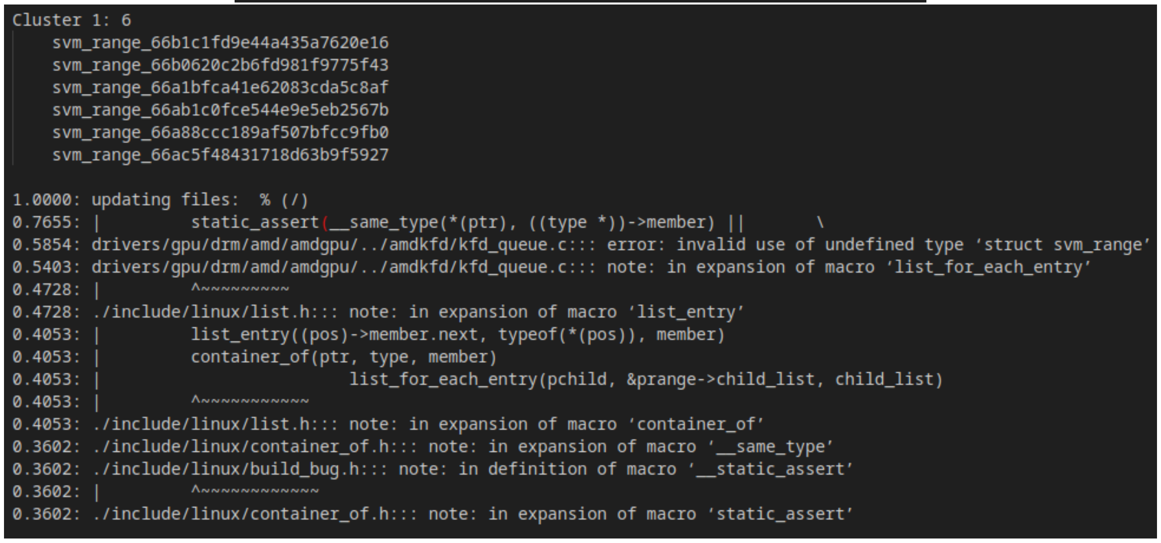
Figure 12: Cluster mixed.

B HDBSCAN Clustering results

B.1 No dimentionality reduction

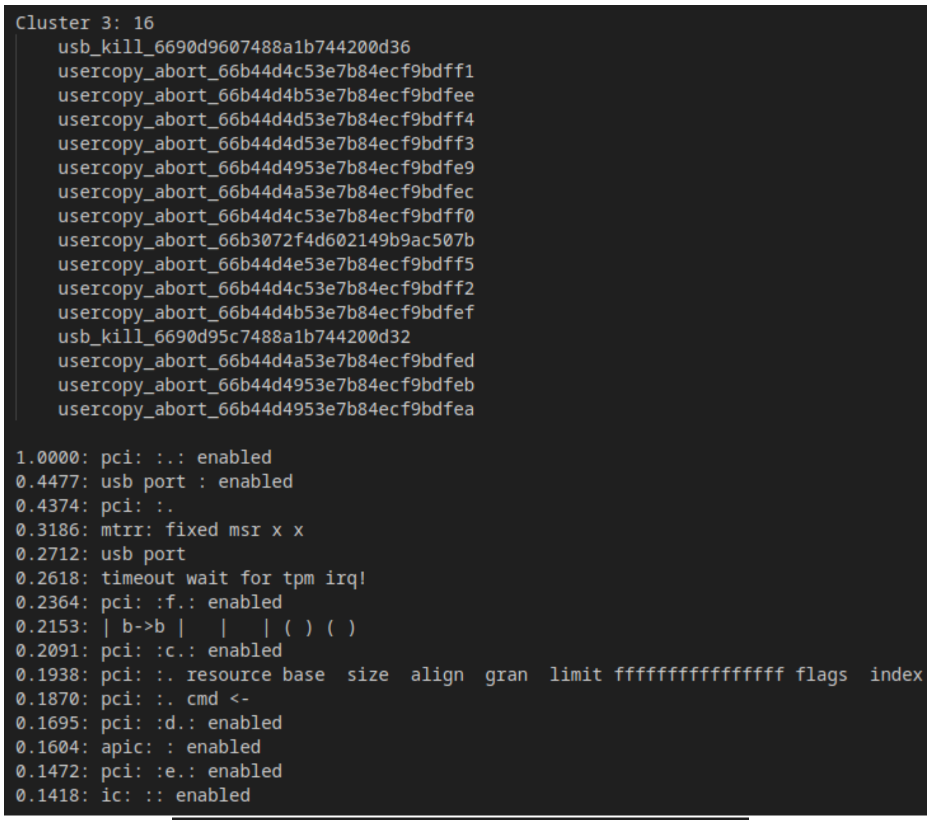


```
Cluster 0: 5
    netdev_66b1c1ee9e44a435a7620e02
    netdev_66b300004d602149b9ac4ae0
    netdev_66b061ff2b6fd981f9775f2f
    netdev_66ab1c03ce544e9e5eb25667
    netdev_66ac5f3c431718d63b9f5913

1.0000: updating files:  % (/)
0.1247: ++ date +%s
0.0985: #
0.0534: cc      arch/x/kernel/i.o
0.0534: cc      crypto/sha_generic.o
0.0481: |                 if (source == netfs_invalid_read)
0.0481: cc      arch/x/events/intel/p.o
0.0356: cc      arch/x/kernel/signal_.o
0.0356: hostcc  arch/x/tools/relocs_.o
0.0356: vdsoc  arch/x/entry/vdso/vdso-image-.c
0.0356: vdso    arch/x/entry/vdso/vdso.so.dbg
0.0356: systbl  arch/x/include/generated/asm/syscalls_.h
0.0356: syshdr  arch/x/include/generated/uapi/asm/unistd_.h
0.0356: # configuration written to .config
0.0356: make[]: *** waiting for unfinished jobs....
```

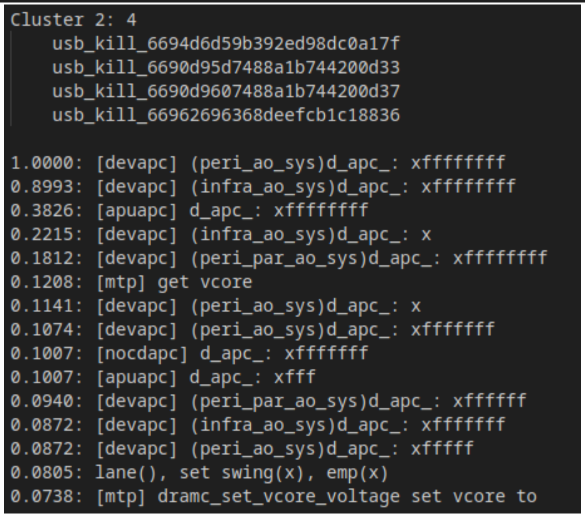


```
Cluster 1: 6
    svm_range_66b1c1fd9e44a435a7620e16
    svm_range_66b0620c2b6fd981f9775f43
    svm_range_66a1bfca41e62083cda5c8af
    svm_range_66ab1c0fce544e9e5eb2567b
    svm_range_66a88ccc189af507bfcc9fb0
    svm_range_66ac5f48431718d63b9f5927

1.0000: updating files:  % (/)
0.7655: |         static_assert(__same_type(*(ptr), ((type *))->member) ||       \
0.5854: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type ‘struct svm_range’
0.5403: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: note: in expansion of macro ‘list_for_each_entry’
0.4728: |         ^~~~~~~~~~
0.4728: ./include/linux/list.h::: note: in expansion of macro ‘list_entry’
0.4053: |         list_entry((pos)->member.next, typeof(*(pos)), member)
0.4053: |         container_of(ptr, type, member)
0.4053: |                         list_for_each_entry(pchild, &prange->child_list, child_list)
0.4053: |         ^~~~~~~~~~~~
0.4053: ./include/linux/list.h::: note: in expansion of macro ‘container_of’
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘__same_type’
0.3602: ./include/linux/build_bug.h::: note: in definition of macro ‘__static_assert’
0.3602: |         ^~~~~~~~~~~~~
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘static_assert’
```

Figure 13: Clusters of "netdev" and "struct svm range" errors and 15 of most scored terms.



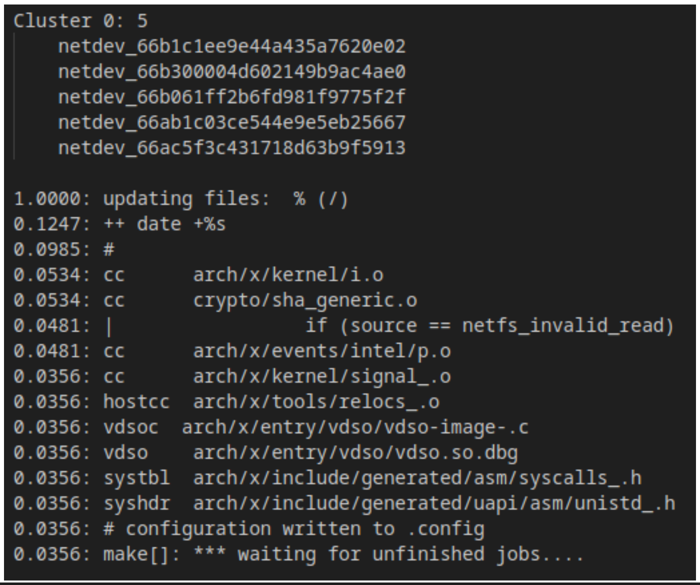
```
Cluster 3: 16
    usb_kill_6690d9607488a1b744200d36
    usercopy_abort_66b44d4c53e7b84ecf9bdff1
    usercopy_abort_66b44d4b53e7b84ecf9bdfee
    usercopy_abort_66b44d4d53e7b84ecf9bdff4
    usercopy_abort_66b44d4d53e7b84ecf9bdff3
    usercopy_abort_66b44d4953e7b84ecf9bdfe9
    usercopy_abort_66b44d4a53e7b84ecf9bdfec
    usercopy_abort_66b44d4c53e7b84ecf9bdff0
    usercopy_abort_66b3072f4d602149b9ac507b
    usercopy_abort_66b44d4e53e7b84ecf9bdff5
    usercopy_abort_66b44d4c53e7b84ecf9bdff2
    usercopy_abort_66b44d4b53e7b84ecf9bdfef
    usb_kill_6690d95c7488a1b744200d32
    usercopy_abort_66b44d4a53e7b84ecf9bdfed
    usercopy_abort_66b44d4953e7b84ecf9bdfeb
    usercopy_abort_66b44d4953e7b84ecf9bdfea

1.0000: pci: :.: enabled
0.4477: usb port : enabled
0.4374: pci: :.
0.3186: mtrr: fixed msr x x
0.2712: usb port
0.2618: timeout wait for tpm irq!
0.2364: pci: :f.: enabled
0.2153: | b->b |   |   | ( ) ( )
0.2091: pci: :c.: enabled
0.1938: pci: :. resource base  size  align  gran  limit ffffffffffffffff flags  index
0.1870: pci: :. cmd <-
0.1695: pci: :d.: enabled
0.1604: apic: : enabled
0.1472: pci: :e.: enabled
0.1418: ic: :: enabled
```



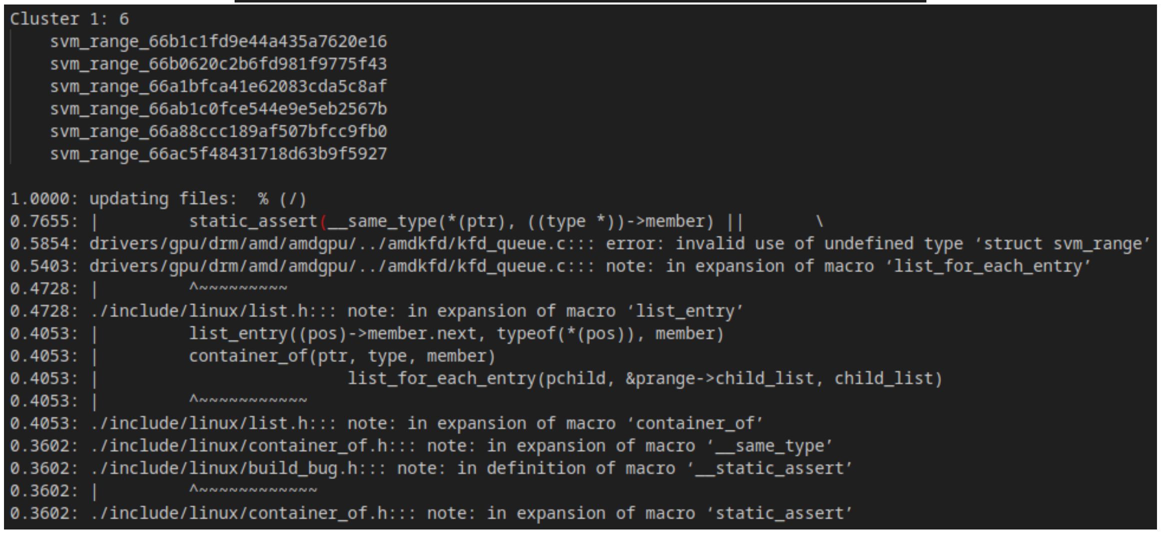
```
Cluster 2: 4
    usb_kill_6694d6d59b392ed98dc0a17f
    usb_kill_6690d95d7488a1b744200d33
    usb_kill_6690d9607488a1b744200d37
    usb_kill_66962696368deefcb1c18836

1.0000: [devapc] (peri_ao_sys)d_apc_: xffffffff
0.8993: [devapc] (infra_ao_sys)d_apc_: xffffffff
0.3826: [apuapc] d_apc_: xffffffff
0.2215: [devapc] (infra_ao_sys)d_apc_: x
0.1812: [devapc] (peri_par_ao_sys)d_apc_: xffffffff
0.1208: [mtp] get vcore
0.1141: [devapc] (peri_ao_sys)d_apc_: x
0.1074: [devapc] (peri_ao_sys)d_apc_: xfffffff
0.1007: [nocdapc] d_apc_: xfffffff
0.1007: [apuapc] d_apc_: xfff
0.0940: [devapc] (peri_par_ao_sys)d_apc_: xffffff
0.0872: [devapc] (infra_ao_sys)d_apc_: xfffffff
0.0872: [devapc] (peri_ao_sys)d_apc_: xfffff
0.0805: lane(), set swing(x), emp(x)
0.0738: [mtp] dramc_set_vcore_voltage set vcore to
```

Figure 14: Clusters of "usb_kill_anchored_urbs panic during boot" and "RIP: 0010:usercopy_abort+0x74/0x76 kernel panic" errors and 15 of most scored terms.

B.2 SVD - dimentionality reduction to 31 components

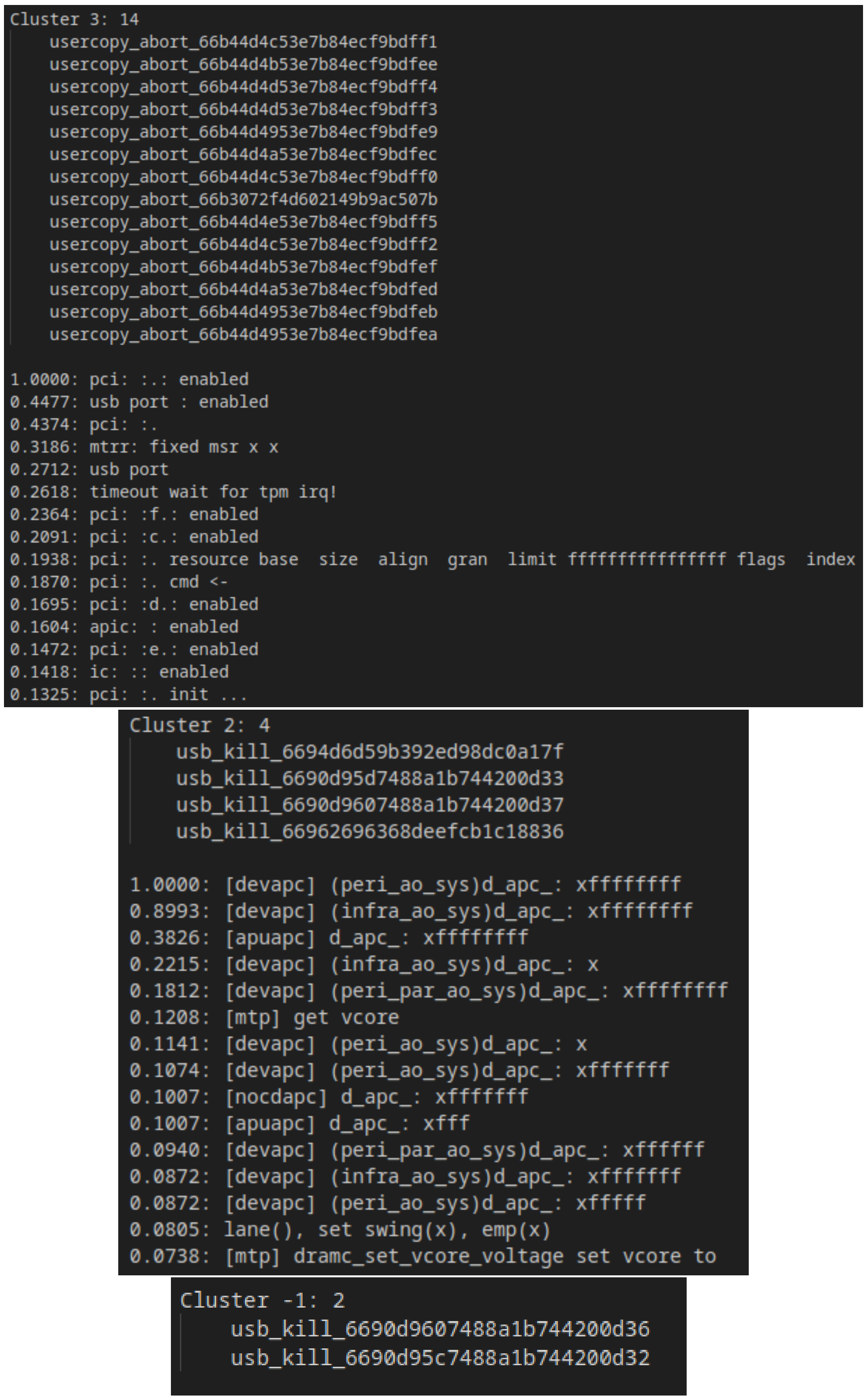


```
Cluster 0: 5
    netdev_66b1c1ee9e44a435a7620e02
    netdev_66b300004d602149b9ac4ae0
    netdev_66b061ff2b6fd981f9775f2f
    netdev_66ab1c03ce544e9e5eb25667
    netdev_66ac5f3c431718d63b9f5913

1.0000: updating files:  % (/)
0.1247: ++ date +%s
0.0985: #
0.0534: cc      arch/x/kernel/i.o
0.0534: cc      crypto/sha_generic.o
0.0481: |                 if (source == netfs_invalid_read)
0.0481: cc      arch/x/events/intel/p.o
0.0356: cc      arch/x/kernel/signal_.o
0.0356: hostcc  arch/x/tools/relocs_.o
0.0356: vdsoc  arch/x/entry/vdso/vdso-image-.c
0.0356: vdso    arch/x/entry/vdso/vdso.so.dbg
0.0356: systbl  arch/x/include/generated/asm/syscalls_.h
0.0356: syshdr  arch/x/include/generated/uapi/asm/unistd_.h
0.0356: # configuration written to .config
0.0356: make[]: *** waiting for unfinished jobs....
```



```
Cluster 1: 6
    svm_range_66b1c1fd9e44a435a7620e16
    svm_range_66b0620c2b6fd981f9775f43
    svm_range_66a1bfca41e62083cda5c8af
    svm_range_66ab1c0fce544e9e5eb2567b
    svm_range_66a88ccc189af507bfcc9fb0
    svm_range_66ac5f48431718d63b9f5927

1.0000: updating files:  % (/)
0.7655: |         static_assert(__same_type(*(ptr), ((type *))->member) ||       \
0.5854: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type ‘struct svm_range’
0.5403: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: note: in expansion of macro ‘list_for_each_entry’
0.4728: |         ^~~~~~~~~~
0.4728: ./include/linux/list.h::: note: in expansion of macro ‘list_entry’
0.4053: |         list_entry((pos)->member.next, typeof(*(pos)), member)
0.4053: |         container_of(ptr, type, member)
0.4053: |                         list_for_each_entry(pchild, &prange->child_list, child_list)
0.4053: |         ^~~~~~~~~~~~
0.4053: ./include/linux/list.h::: note: in expansion of macro ‘container_of’
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘__same_type’
0.3602: ./include/linux/build_bug.h::: note: in definition of macro ‘__static_assert’
0.3602: |         ^~~~~~~~~~~~~
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘static_assert’
```

Figure 15: Clusters of ”netdev” and ”struct svm range” errors and 15 of most scored terms.



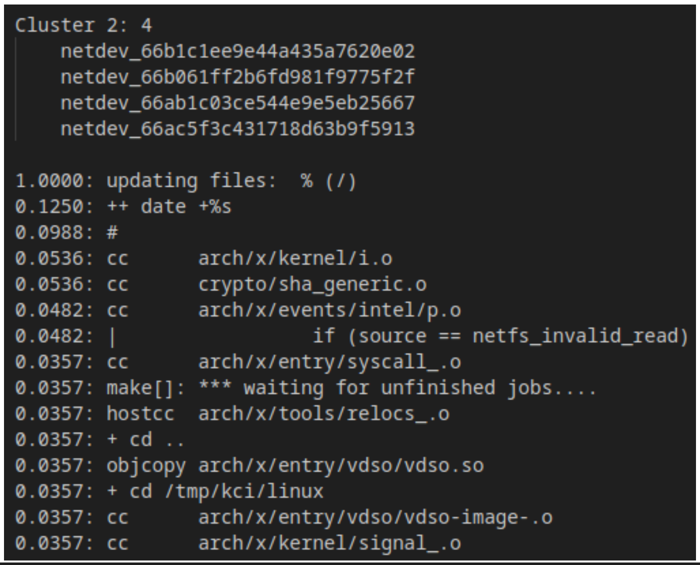
```
Cluster 3: 14
    usercopy_abort_66b44d4c53e7b84ecf9bdff1
    usercopy_abort_66b44d4b53e7b84ecf9bdfee
    usercopy_abort_66b44d4d53e7b84ecf9bdff4
    usercopy_abort_66b44d4d53e7b84ecf9bdff3
    usercopy_abort_66b44d4953e7b84ecf9bdfe9
    usercopy_abort_66b44d4a53e7b84ecf9bdfec
    usercopy_abort_66b44d4c53e7b84ecf9bdff0
    usercopy_abort_66b3072f4d602149b9ac507b
    usercopy_abort_66b44d4e53e7b84ecf9bdff5
    usercopy_abort_66b44d4c53e7b84ecf9bdff2
    usercopy_abort_66b44d4b53e7b84ecf9bdfef
    usercopy_abort_66b44d4a53e7b84ecf9bdfed
    usercopy_abort_66b44d4953e7b84ecf9bdfeb
    usercopy_abort_66b44d4953e7b84ecf9bdfea

1.0000: pci: :.: enabled
0.4477: usb port : enabled
0.4374: pci: :.
0.3186: mtrr: fixed msr x x
0.2712: usb port
0.2618: timeout wait for tpm irq!
0.2364: pci: :f.: enabled
0.2091: pci: :c.: enabled
0.1938: pci: :. resource base  size  align  gran  limit ffffffffffffffff flags  index
0.1870: pci: :. cmd <-
0.1695: pci: :d.: enabled
0.1604: apic: : enabled
0.1472: pci: :e.: enabled
0.1418: ic: :: enabled
0.1325: pci: :. init ...
```

```
Cluster 2: 4
    usb_kill_6694d6d59b392ed98dc0a17f
    usb_kill_6690d95d7488a1b744200d33
    usb_kill_6690d9607488a1b744200d37
    usb_kill_66962696368deefcb1c18836

1.0000: [devapc] (peri_ao_sys)d_apc_: xffffffff
0.8993: [devapc] (infra_ao_sys)d_apc_: xffffffff
0.3826: [apuapc] d_apc_: xffffffff
0.2215: [devapc] (infra_ao_sys)d_apc_: x
0.1812: [devapc] (peri_par_ao_sys)d_apc_: xffffffff
0.1208: [mtp] get vcore
0.1141: [devapc] (peri_ao_sys)d_apc_: x
0.1074: [devapc] (peri_ao_sys)d_apc_: xfffffff
0.1007: [nocdapc] d_apc_: xfffffff
0.1007: [apuapc] d_apc_: xfff
0.0940: [devapc] (peri_par_ao_sys)d_apc_: xffffff
0.0872: [devapc] (infra_ao_sys)d_apc_: xfffffff
0.0872: [devapc] (peri_ao_sys)d_apc_: xfffff
0.0805: lane(), set swing(x), emp(x)
0.0738: [mtp] dramc_set_vcore_voltage set vcore to
```

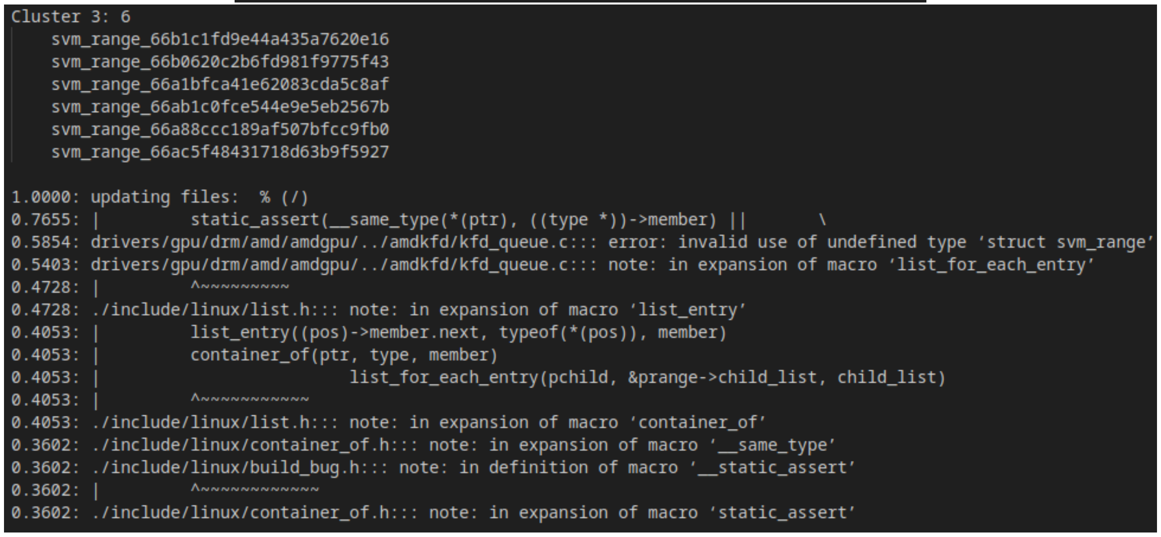
```
Cluster -1: 2
    usb_kill_6690d9607488a1b744200d36
    usb_kill_6690d95c7488a1b744200d32
```

Figure 16: Clusters of "usb_kill_anchored_urbs panic during boot" and "RIP: 0010:usercopy_abort+0x74/0x76 kernel panic" errors and 15 of most scored terms.

B.3 NMF - dimentionality reduction to 31 components

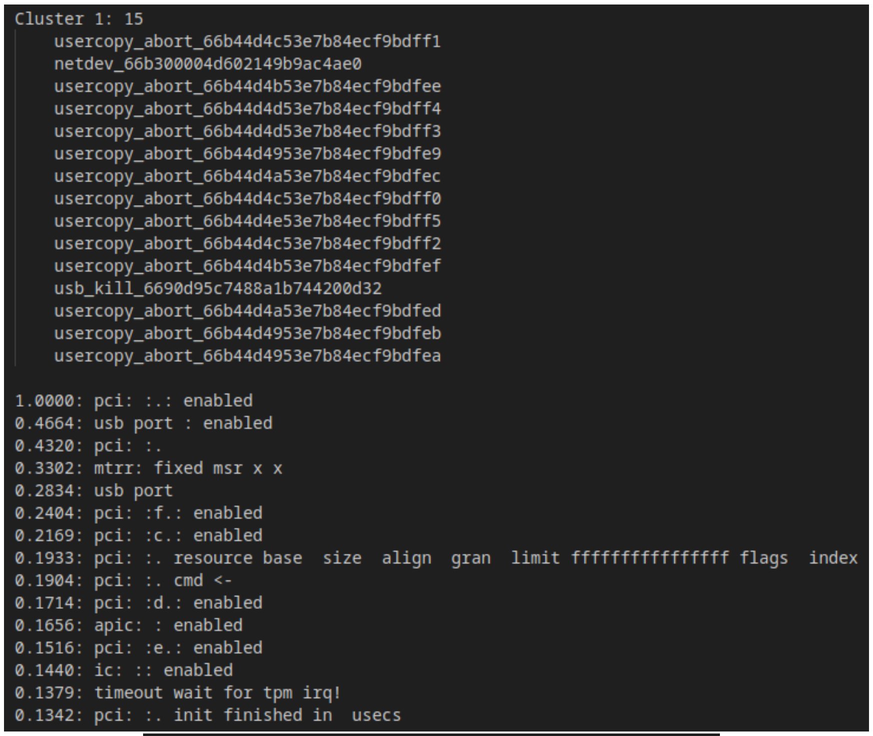


```
Cluster 2: 4
    netdev_66b1c1ee9e44a435a7620e02
    netdev_66b061ff2b6fd981f9775f2f
    netdev_66ab1c03ce544e9e5eb25667
    netdev_66ac5f3c431718d63b9f5913

1.0000: updating files:  % (/)
0.1250: ++ date +%s
0.0988: #
0.0536: cc      arch/x/kernel/i.o
0.0536: cc      crypto/sha_generic.o
0.0482: cc      arch/x/events/intel/p.o
0.0482: |                 if (source == netfs_invalid_read)
0.0357: cc      arch/x/entry/syscall_.o
0.0357: make[]: *** waiting for unfinished jobs....
0.0357: hostcc  arch/x/tools/relocs_.o
0.0357: + cd ..
0.0357: objcopy arch/x/entry/vdso/vdso.so
0.0357: + cd /tmp/kci/linux
0.0357: cc      arch/x/entry/vdso/vdso-image-.o
0.0357: cc      arch/x/kernel/signal_.o
```

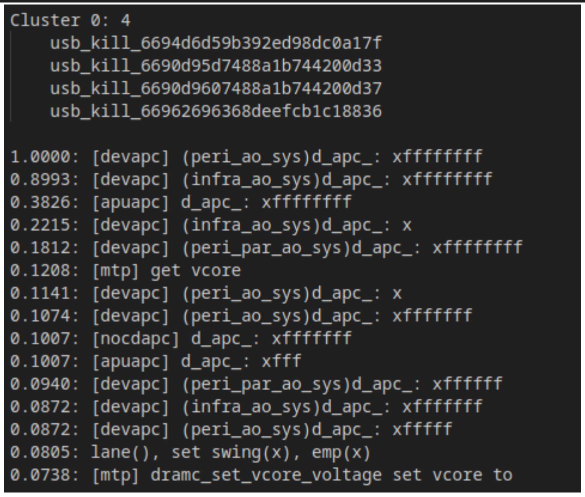


```
Cluster 3: 6
    svm_range_66b1c1fd9e44a435a7620e16
    svm_range_66b0620c2b6fd981f9775f43
    svm_range_66a1bfca41e62083cda5c8af
    svm_range_66ab1c0fce544e9e5eb2567b
    svm_range_66a88ccc189af507bfcc9fb0
    svm_range_66ac5f48431718d63b9f5927

1.0000: updating files:  % (/)
0.7655: |         static_assert(__same_type(*(ptr), ((type *))->member) ||       \
0.5854: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: error: invalid use of undefined type ‘struct svm_range’
0.5403: drivers/gpu/drm/amd/amdgpu/../amdkfd/kfd_queue.c::: note: in expansion of macro ‘list_for_each_entry’
0.4728: |         ^~~~~~~~~~
0.4728: ./include/linux/list.h::: note: in expansion of macro ‘list_entry’
0.4053: |         list_entry((pos)->member.next, typeof(*(pos)), member)
0.4053: |         container_of(ptr, type, member)
0.4053: |                         list_for_each_entry(pchild, &prange->child_list, child_list)
0.4053: |         ^~~~~~~~~~~~
0.4053: ./include/linux/list.h::: note: in expansion of macro ‘container_of’
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘__same_type’
0.3602: ./include/linux/build_bug.h::: note: in definition of macro ‘__static_assert’
0.3602: |         ^~~~~~~~~~~~~
0.3602: ./include/linux/container_of.h::: note: in expansion of macro ‘static_assert’
```

Figure 17: Clusters of ”netdev” and ”struct svm range” errors and 15 of most scored terms.

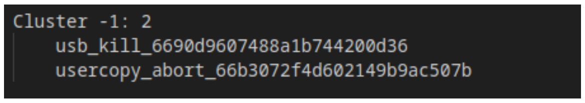


```
Cluster 1: 15
    usercopy_abort_66b44d4c53e7b84ecf9bdff1
    netdev_66b300004d602149b9ac4ae0
    usercopy_abort_66b44d4b53e7b84ecf9bdfee
    usercopy_abort_66b44d4d53e7b84ecf9bdff4
    usercopy_abort_66b44d4d53e7b84ecf9bdff3
    usercopy_abort_66b44d4953e7b84ecf9bdfe9
    usercopy_abort_66b44d4a53e7b84ecf9bdfec
    usercopy_abort_66b44d4c53e7b84ecf9bdff0
    usercopy_abort_66b44d4e53e7b84ecf9bdff5
    usercopy_abort_66b44d4c53e7b84ecf9bdff2
    usercopy_abort_66b44d4b53e7b84ecf9bdfef
    usb_kill_6690d95c7488a1b744200d32
    usercopy_abort_66b44d4a53e7b84ecf9bdfed
    usercopy_abort_66b44d4953e7b84ecf9bdfeb
    usercopy_abort_66b44d4953e7b84ecf9bdfea

1.0000: pci: :.: enabled
0.4664: usb port : enabled
0.4320: pci: :.
0.3302: mtrr: fixed msr x x
0.2834: usb port
0.2404: pci: :f.: enabled
0.2169: pci: :c.: enabled
0.1933: pci: :. resource base  size  align  gran  limit ffffffffffffffff flags  index
0.1904: pci: :. cmd <-
0.1714: pci: :d.: enabled
0.1656: apic: : enabled
0.1516: pci: :e.: enabled
0.1440: ic: :: enabled
0.1379: timeout wait for tpm irq!
0.1342: pci: :. init finished in  usecs
```



```
Cluster 0: 4
    usb_kill_6694d6d59b392ed98dc0a17f
    usb_kill_6690d95d7488a1b744200d33
    usb_kill_6690d9607488a1b744200d37
    usb_kill_66962696368deefcb1c18836

1.0000: [devapc] (peri_ao_sys)d_apc_: xffffffff
0.8993: [devapc] (infra_ao_sys)d_apc_: xffffffff
0.3826: [apuapc] d_apc_: xffffffff
0.2215: [devapc] (infra_ao_sys)d_apc_: x
0.1812: [devapc] (peri_par_ao_sys)d_apc_: xffffffff
0.1208: [mtp] get vcore
0.1141: [devapc] (peri_ao_sys)d_apc_: x
0.1074: [devapc] (peri_ao_sys)d_apc_: xfffffff
0.1007: [nocdapc] d_apc_: xfffffff
0.1007: [apuapc] d_apc_: xfff
0.0940: [devapc] (peri_par_ao_sys)d_apc_: xffffff
0.0872: [devapc] (infra_ao_sys)d_apc_: xfffffff
0.0872: [devapc] (peri_ao_sys)d_apc_: xffffff
0.0805: lane(), set swing(x), emp(x)
0.0738: [mtp] dramc_set_vcore_voltage set vcore to
```

Figure 18: Clusters of "usb_kill_anchored_urbs panic during boot" and "RIP: 0010:usercopy_abort+0x74/0x76 kernel panic" errors and 15 of most scored terms.



```
Cluster -1: 2
    usb_kill_6690d9607488a1b744200d36
    usercopy_abort_66b3072f4d602149b9ac507b
```

Figure 19: Cluster mixed.