Log-Based Anomaly Detection For Large-Scale Cloud Systems- Exploratory Data Analysis

Dataset	Description	Source link	Rows	Columns	Duration	Time Period
HDFS	Logs from the Hadoop Distributed File System on a 203-node cluster. These logs capture various distributed file operations and system events.	HDFS Dataset	2000	9	1 day 13 hours	2008-11-09 20:36:15 to 2008-11-11 10:20:17
BGL	Blue Gene/L supercomputer logs from Lawrence Livermore National Lab. These include timestamped hardware	BGL Dataset	2000	13	213 days 15 hours	2005-06-03 15:42:50 to 2006-01-03 07:13:09

	and software events.					
OpenStack	Logs from an OpenStack cloud deployment collected via Logstash. These reflect operations in a private cloud infrastructure.	OpenStack Dataset	2000	11	15 minutes	2017-05-16 00:00:00 to 2017-05-16 00:14:48
Thunderbird	Logs from a Thunderbird email system, including timestamped records of system and application-level events.	Thunderbird Dataset	2000	14	15 minutes	2005-11-09 12:01:01 to 2005-11-09 12:15:32

Zookeeper	Logs from the Zookeeper coordination service, used for maintaining configuration information and distributed synchronization	Zookeeper Dataset	2000	10	12 days 1 hour	2015-07-29 17:41:44 to 2015-08-10 18:12:34

HDFS

Figure 1: Top Logged Components in HDFS LogsFigure 1: Top Logged Components in HDFS Logs

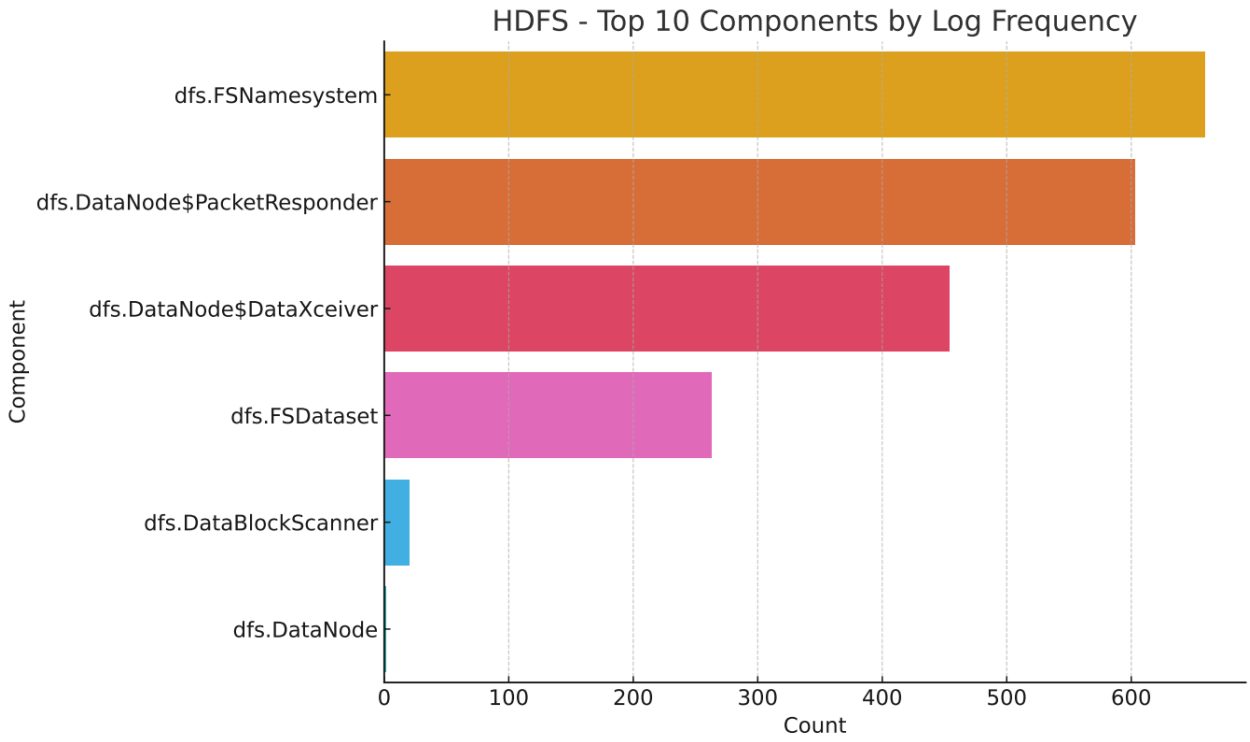

Figure 1: Top Logged Components in HDFS Logs

Figure 1 shows the most frequently logged components in the HDFS dataset.It highlights critical modules like `dfs.FSNamesystem` and `dfs.DataNode$PacketResponder`, indicating their central role in system operations. These components are key points of monitoring for anomaly detection and performance diagnostics.

Insights:

High Activity Components: The most frequently logged components are `dfs.FSNamesystem`, `dfs.DataNode$PacketResponder`, and `dfs.DataNode$DataXceiver`. These are fundamental components of HDFS that deal with namespace management, communication between data blocks, and data transfer, which means these components are always involved in routine operations.

System Focus: Being the focus of these components means they are critical in system operations and monitoring them may be vital to identify anomalies or system optimization.

Decreased Activity Parts: Components like `dfs.DataBlockScanner` and `dfs.DataNode` generate significantly less logging, possibly indicating either less functional activity or fewer instrumentation points.

Conclusion:This chart makes it easier to decide on which parts of the HDFS system are most active or critical, and can inform projects in anomaly detection, performance tuning, or system monitoring.

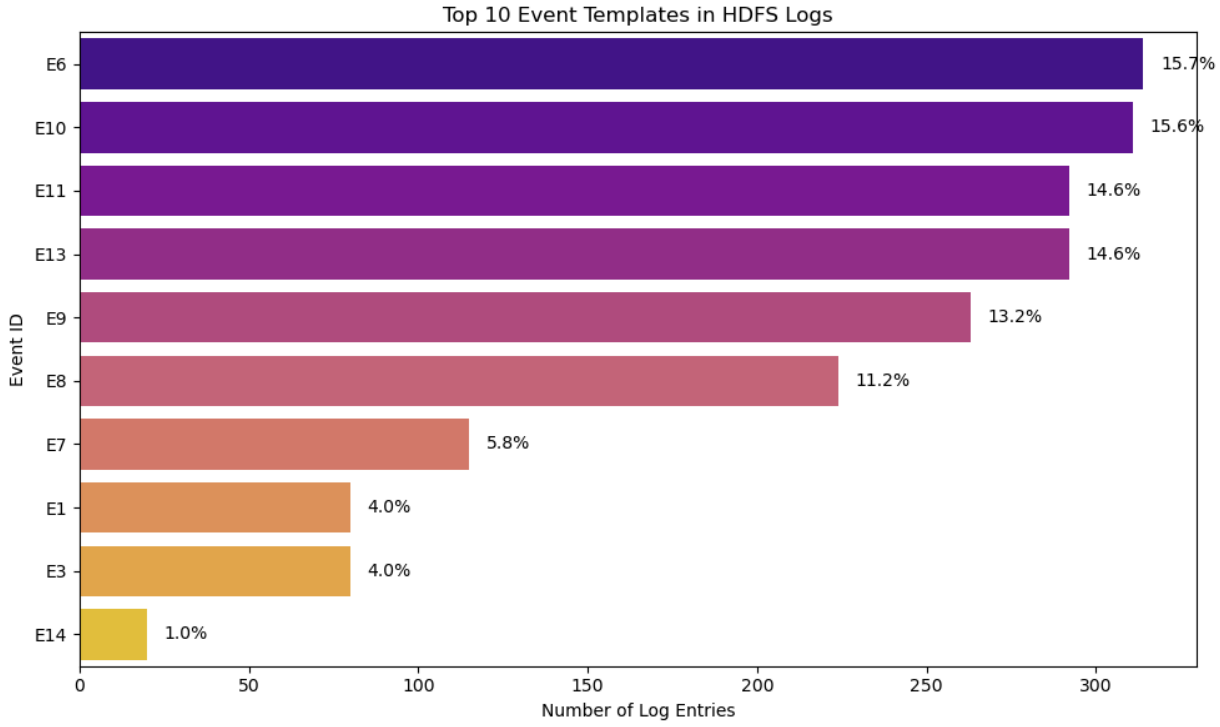

Figure 2: Distribution of Top 10 Event Templates in HDFS

Figure 2 displays the distribution of the top 10 event templates in the HDFS logs.. It reveals a skewed distribution with templates like E6 and E10 dominating, suggesting frequent routine operations. Understanding this distribution aids in filtering noise and focusing anomaly detection on rare patterns.

Insights:

Skewed Distribution: A few event IDs (e.g., E6, E10, E11, E13) account for a disproportionately high percentage of the log data—each contributing between 13% and 16%. This indicates that the generation of log is highly biased toward repeated events.

Frequent System Behaviors: These frequent templates would likely indicate typical operations or routine system messages within HDFS, such as heartbeat messages, replication of a block, or file operations.

Optimization Potential: Having some knowledge of what these typical templates are can guide log compression, filtering, or feature selection for detecting anomalies. Highly frequent uninformative logs, for example, could be omitted to reduce noise.

Rare Events: Event IDs towards the end (like E14) occur infrequently, which could be indicators of rare behavior or potential anomalies to watch out for.

Conclusion: This chart helps determine most common patterns in HDFS logs and identifies where the system keeps most of its logging activity. It proves priceless to guide analysis and understand typical vs. atypical system behavior.

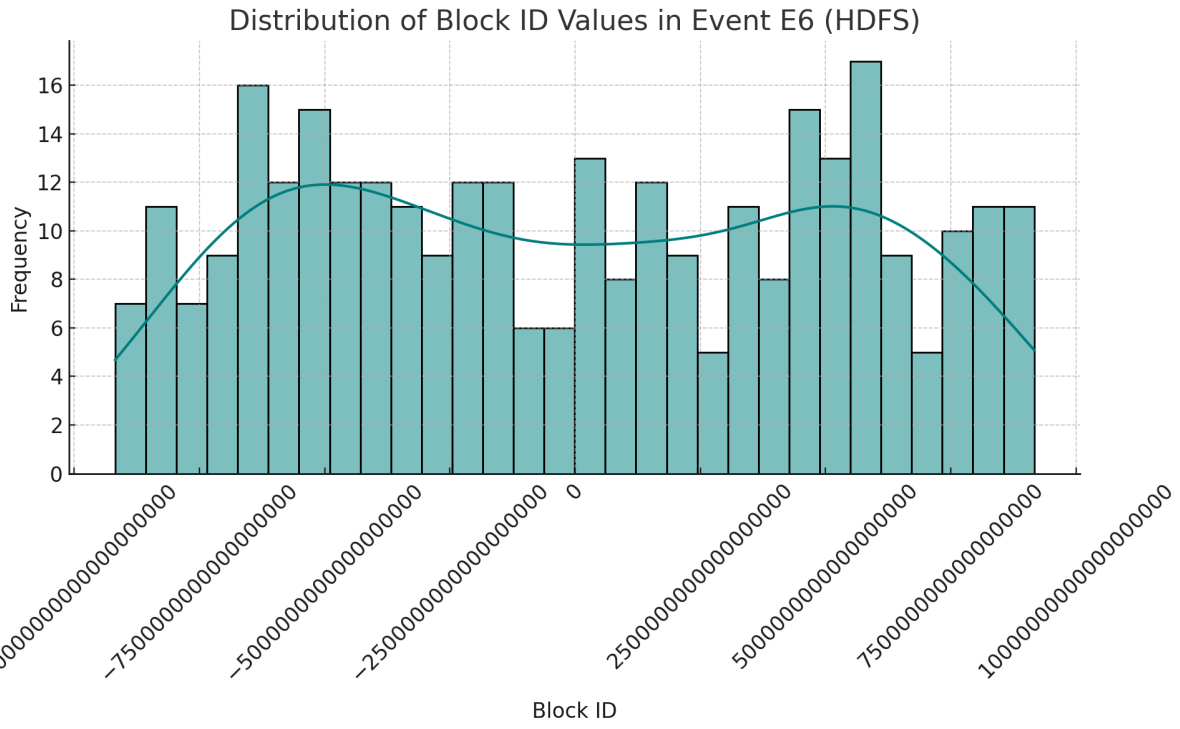

Figure 3: Distribution of Block ID Values in Event E6 (HDFS)

Figure 3 presents the distribution of Block ID values for Event E6 in the HDFS logs. It verifies that block allocation follows a fairly uniform pattern, with a few peaks suggesting

heavy block activity. This serves as a baseline to detect irregular block usage patterns later.

This diagram depicts the distribution of block ID values extracted from log messages associated with Event ID E6 in the HDFS dataset. Event ID E6 corresponds to actions where a block has been stored and the NameNode updates the block map. The block IDs are large identifiers, which were parsed from the textual contents of each log entry. The horizontal axis represents the full range of block ID values while the vertical axis shows how many times each block ID Value range appears in the logs. The histogram shows a fairly uniform distribution with little variation, with a smooth line that shows the overall trend.
From this diagram, we see that block allocation is approximately uniformly distributed with very little extreme clustering or gaps following a natural pattern. This tells us that when the system assigns block IDs during regular operating conditions, it tends to do so with regular frequency. The rare nearby peaks may represent brief spike activity, such as when large files are being written or replicated. This distribution can be used as a reference model to flag strange patterns in future logs.

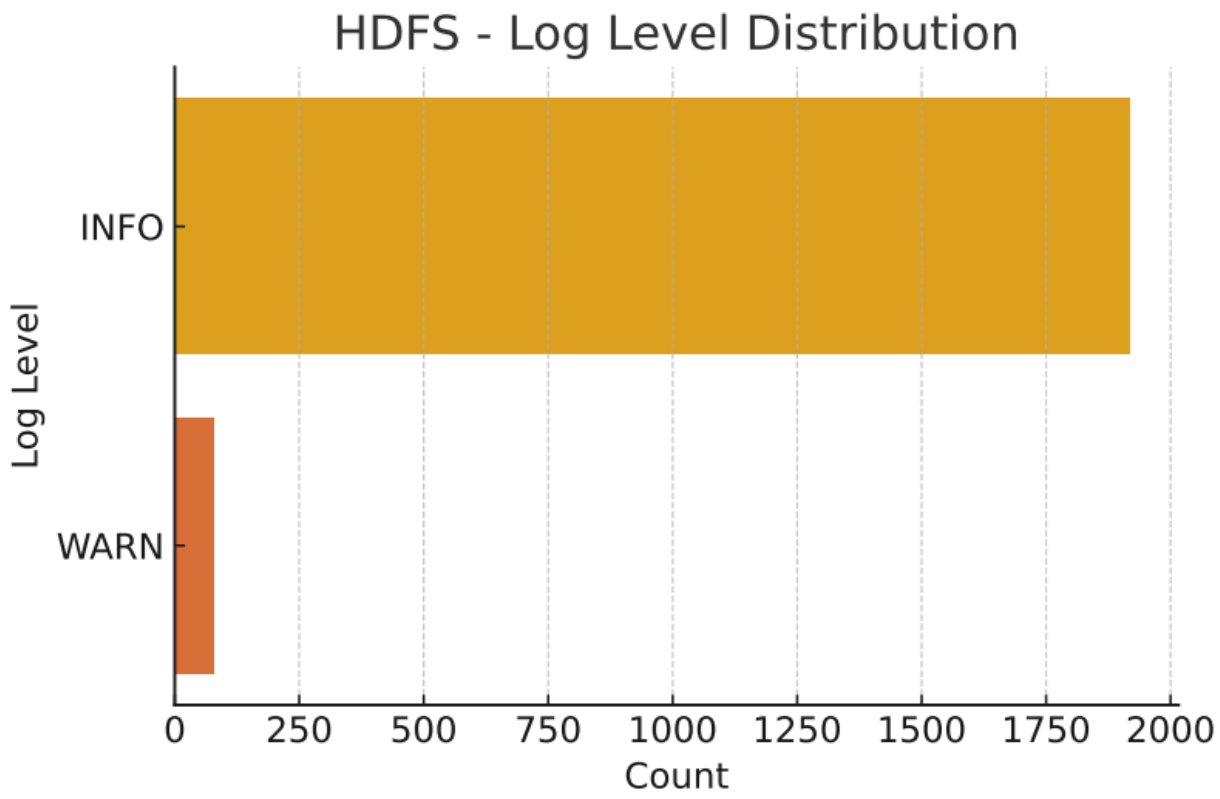

Figure 4: Log Severity Level Distribution in HDFS

Figure 4 presents the number of logs per severity level (INFO, WARN) in the HDFS dataset. It shows that most logs are INFO-level, indicating normal operations. The near absence of WARN logs suggests system stability but also highlights the importance of reviewing the rare warnings.

Insights:

INFO Dominates: The overwhelming majority of log messages are `INFO` level, which means that the HDFS system produces an enormous quantity of normal operational messages. Such logs will typically capture normal activity such as file reads/writes, block reporting, and system heartbeats.

Few WARN Logs: Practically no logs are of the `WARN` type, indicating warning-level issues are highly unusual. This may indicate robust system performance with less than normal recoverable errors or potential misconfigurations.

Operational Health: The low level of warning messages is typically a plus—it shows the system operates in normal limits most of the time.

Conclusion:This distribution provides an extremely quick overview of the system's logging health. While `INFO` logs are important for observability, too many of them are noisy. The extremely rare `WARN` logs require close examination since they can be the harbingers of hidden issues.

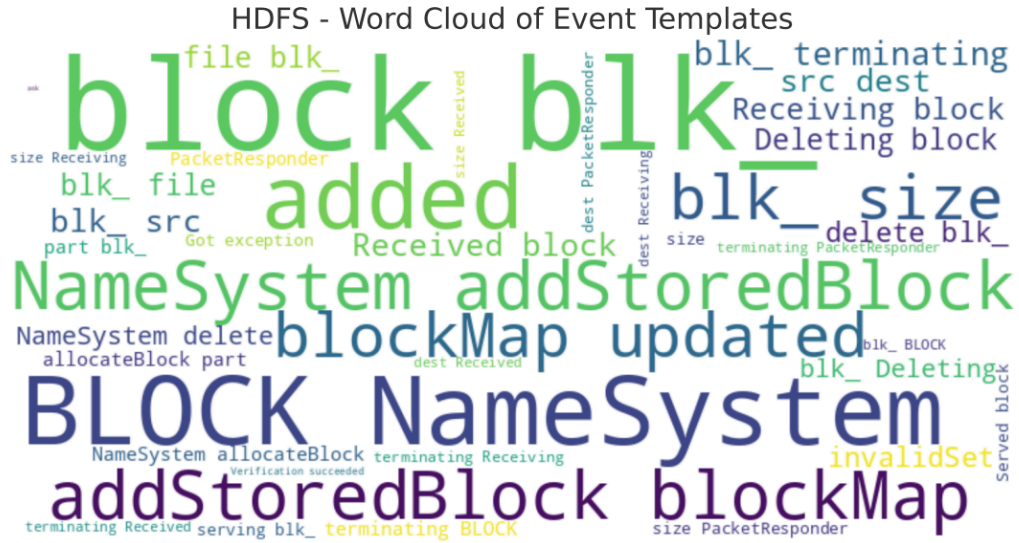

Figure 5: Word Cloud of HDFS Event Templates

Figure 5 depicts the most frequent words used in HDFS log templates. Dominant terms like "block", "NameSystem", and "addStoredBlock" show the system's strong focus on block-level operations. It provides a snapshot of recurring actions and components.

Insights:

Salient terms include "block", "blk_ ", "addStoredBlock", "NameSystem", and "blockMap", which indicates that block activity and the HDFS NameNode (NameSystem) are central to HDFS logging.

Generic verbs ("added", "updated", "delete", "allocateBlock", and "Received") suggest that the logs frequently capture events of block addition, modification, and replication.

Jargon terms ("PacketResponder", "src", "dest", and "terminating") suggest detailed logging of data packet processing and communications protocols within the system.

Conclusion:The data cloud offers a quick, visual understanding of what HDFS is primarily logging. It helps analysts identify major themes and operational focuses, such as block-level data processing and NameNode responsibilities, that are crucial for anomaly detection and system debugging.

BGL

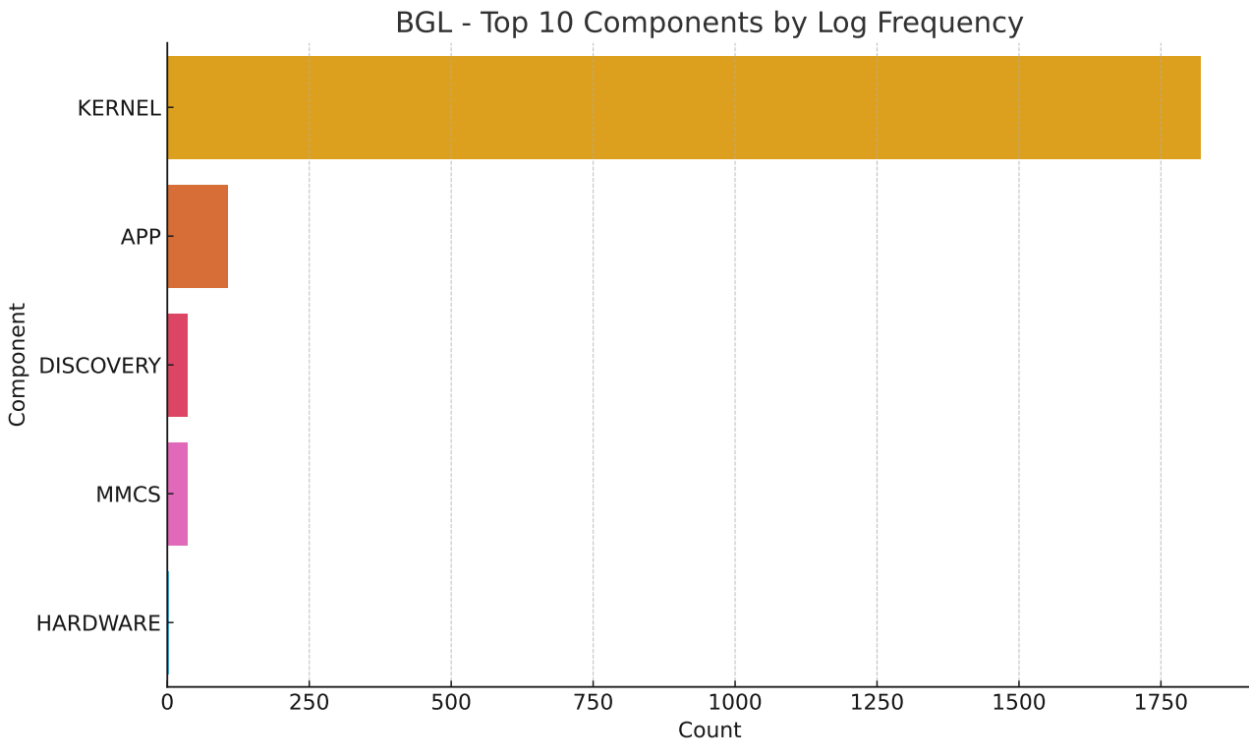

Figure 6: Top Logged Components in BGL Logs

Figure 6 ranks BGL components by log frequency, with "KERNEL" overwhelmingly dominant. This indicates that system-level events are extensively logged at the kernel layer. It highlights the kernel as the most informative component for anomaly detection.

Insights:

KERNEL outranks all others by a substantial margin in terms of log frequency: most of the operational or system-level activities are logged at this level. This is characteristic of systems logging, where the kernel does a core job such as process management, memory handling, and device interaction.

Other components like "APP", "DISCOVERY", and "MMCS" end up contributing very few logs for reasons that could be less activity or fewer points of the logs.

The preponderance of "KERNEL" suggests that concentrating on this component for performance monitoring and fault detection would yield the best results as it is the richest source of operational data.

A relatively smaller number of "HARDWARE" logs signify either less instrumentation or basically a steady hardware environment.

Conclusion:Based on this distribution, logging in the BGL system essentially occurs only on the kernel level, emphasizing the importance of

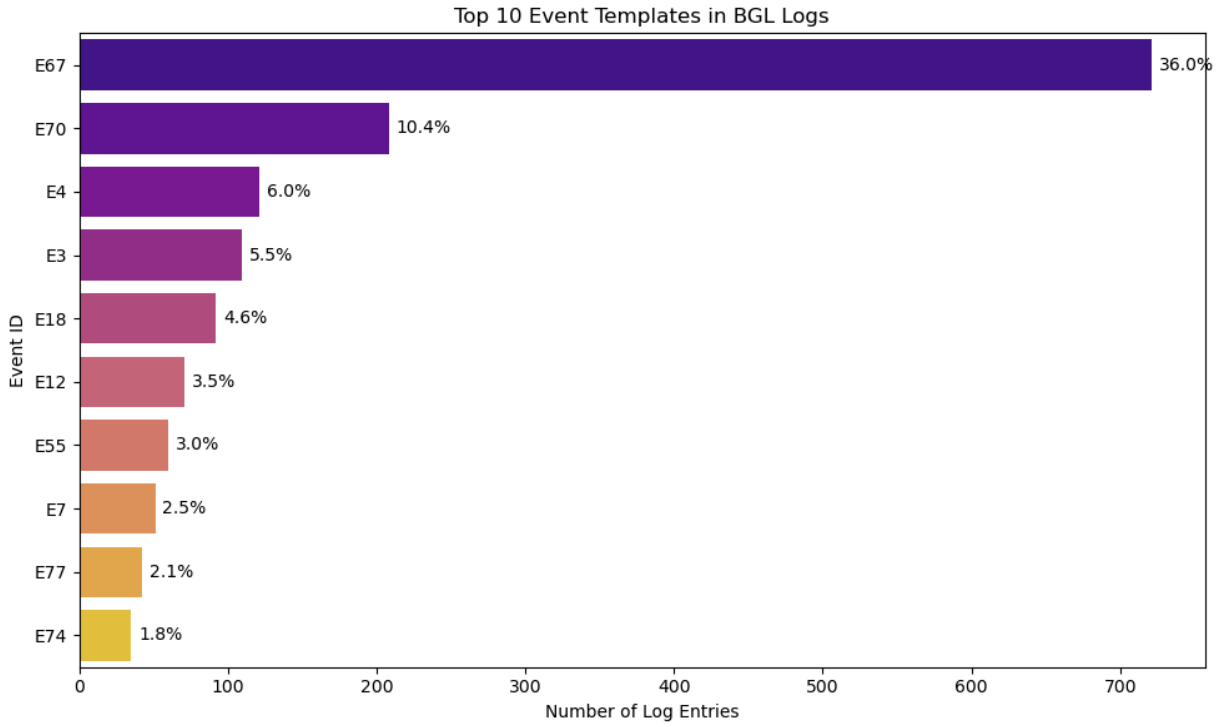

Figure 7: Top 10 Event Templates in BGL

Figure 7 shows the top event templates by log percentage, revealing that E67 alone accounts for over a third of the logs. This extreme skew indicates that logs are highly redundant, and anomaly detection may benefit from focusing on less frequent templates.

Insights:

E67 goes beyond the 35% of the total logs to say that this event template is far more frequent than any other. It might be an event template typical of common or repetitive system behavior.

Second ranks with about a tenth of the logs is E70.

The rest of the events (E4 to E74) constitute less than 10% for each, with a decreasing numerical order in frequencies.

This way of the distribution fairly favors a few templates as the main sources of logging activity: those could be heartbeats, routine hardware checks, or system-level updates repeated at a mundane rate.

Conclusion:This diagram shows a very imbalanced distribution of events in BGL logs, where one or two patterns dominate them. This type of information is handy either to filter redundancy, to reduce dataset size, or to focus anomaly detection on less frequently occurring patterns.

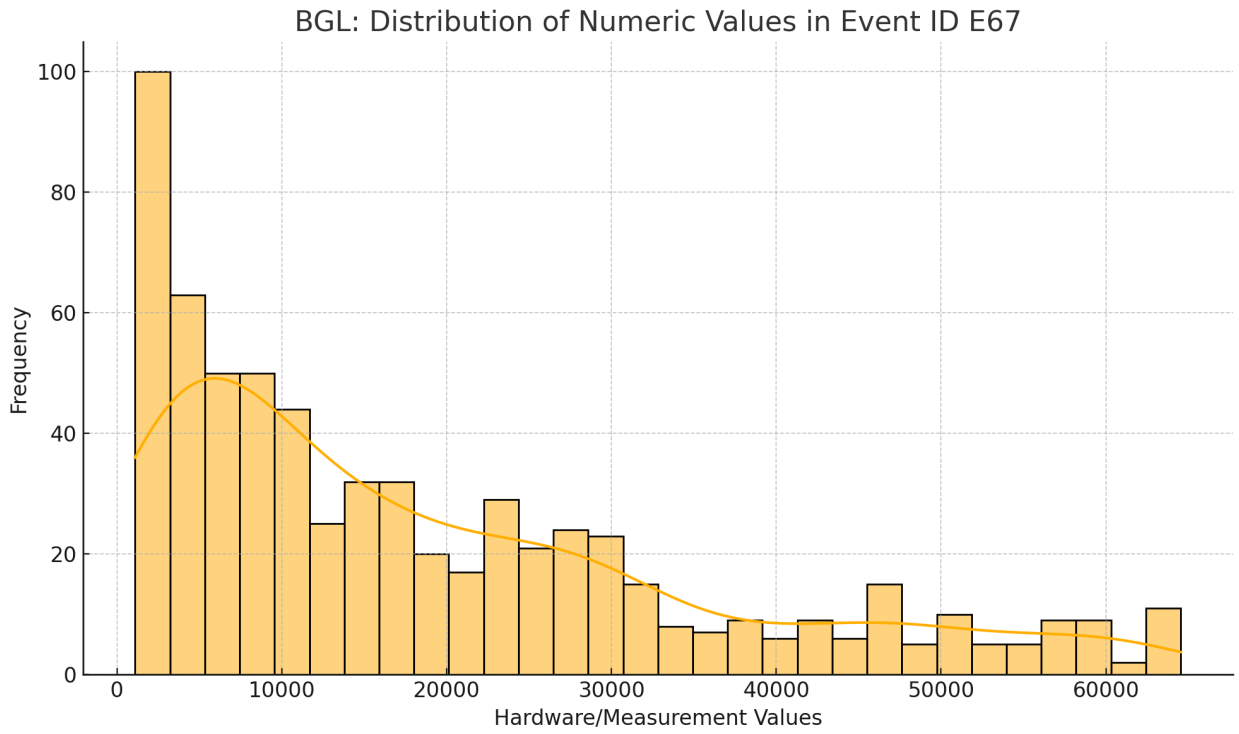

Figure 8: Value Distribution for Event ID E67 (BGL)

Figure 8 plots numeric values extracted from Event E67 messages. A right-skewed distribution suggests that smaller values occur far more often, which is typical for system counters or resource metrics. This pattern can be used to spot abnormal spikes.

Insights:
The lower horizontal axis (x-axis) shows the range of these numeric values and the vertical axis shows the frequency of each range. The histogram shows a right-skewed distribution; smaller numeric values are far more common than the larger numeric values. The smoothed trend line highlights the shape of this distribution.
From this figure we observe that the system is producing a high number of events with smaller numeric values, with the lower values clearly occurring more frequently than higher values. This distribution shape is common for metrics with sensor data, hardware

utilization statistics, or resource counters metrics. It may help in identifying outliers or changes in system behavior, when comparing with logs from periods of abnormal operation.

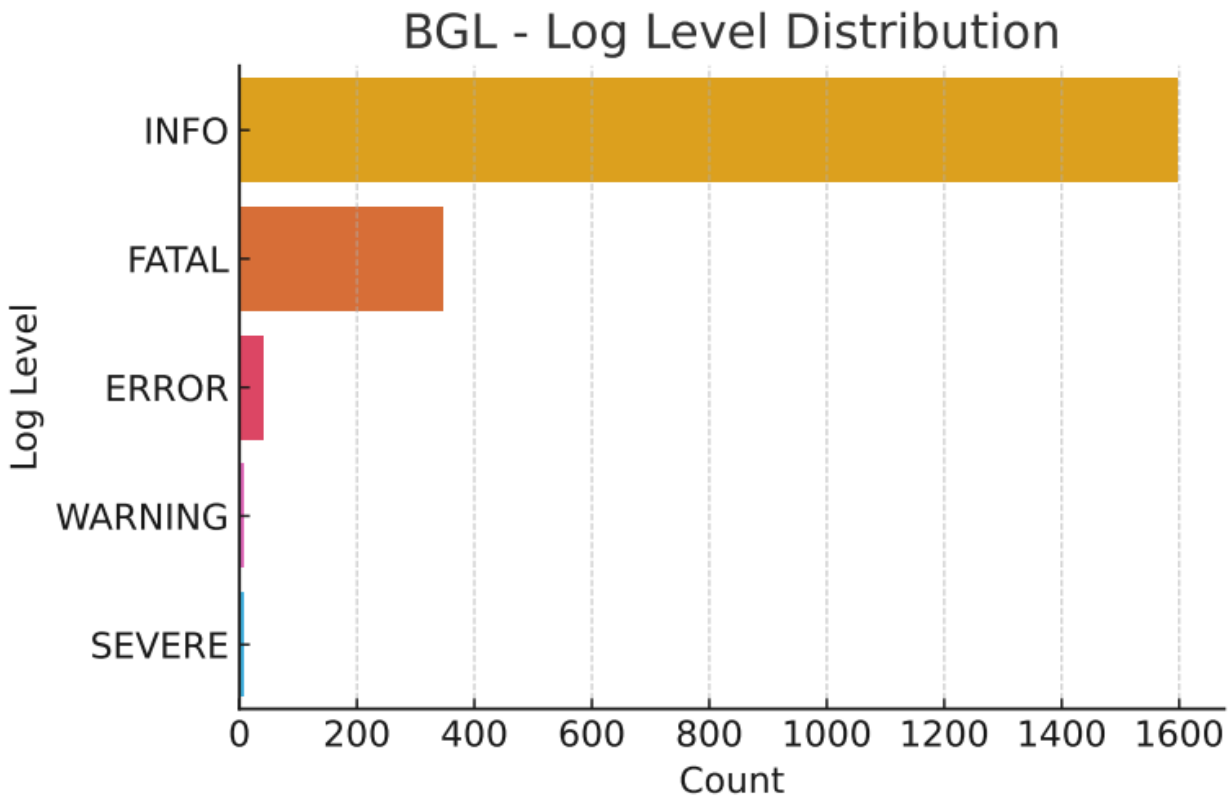

Figure 9: Log Severity Level Distribution in BGL

Figure 9 categorizes BGL logs by severity, showing most messages as INFO with a notable number of FATAL logs. The presence of severe errors highlights the need to monitor critical system failures while filtering routine messages.

Insights:

To a significant extent there are more INFO logs than any other kind of log to indicate that most of the messages relate to normal operational activity and system state reporting.

There are next more FATAL logs. This is interesting considering that FATAL logs normally indicate critical error conditions with the system, that could halt process or result in substantive fault.

ERROR, WARNING and SEVERE levels are rare. This suggests serious and recoverable errors have been logged, but compared to INFO operations and fatal errors these are seldom events for BGL.

FATAL and SEVERE levels detected in our data set suggests that the logging framework is capable of logging a number of critical issues and problems of differing severity levels, and the logs in BGL would need to be interpreted consistently during analysis to derive relevance.

Conclusion: What does this distribution mean? It is evident that most of the content in the slice of BGL logs are informative messages, but there is a sufficient volume of critical logs to merit consideration when monitoring system reliability, and performing anomaly/pathway detection.

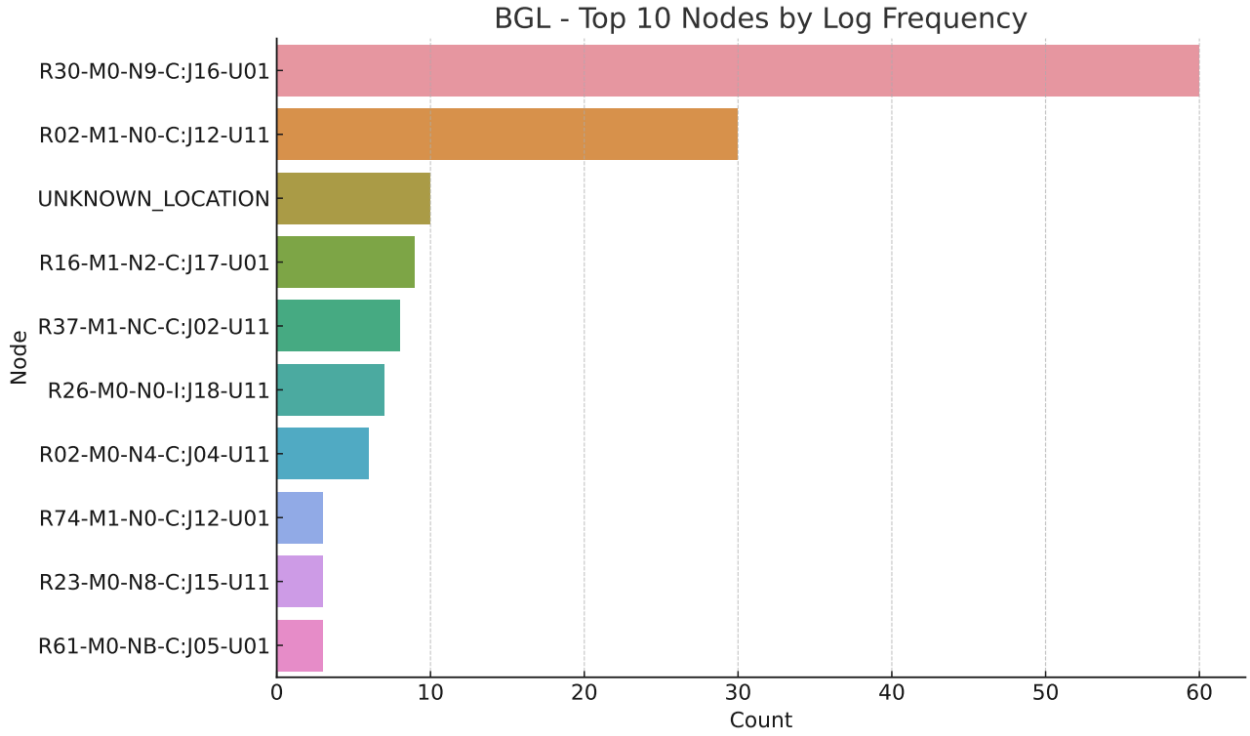

Figure 10: Top Logging Nodes in BGL

Figure 10 ranks BGL system nodes by log count. Nodes like R30-M0-N9-C:J16-U01 log the most, pointing to higher activity or issues. It helps localize hotspots for performance monitoring and troubleshooting.

Insights:

Node R30-M0-N9-C\:J16-U01 logged the most messages, which indicates that this node is working hard or seeing many activitylog-worthy events on the node.

Node R02-M1-N0-C\:J12-U11 comes next- it also had a high log volume and same activity as above.

The entry UNKNOWN_LOCATION shows up with a significant volume, which could indicate that these events are being improperly tagged or untagged log source by the system.

The other nodes are lower frequency and trending downward gradually; cumulatively, these nodes are moderate activity log users.

More logs in the specific nodes could point to uneven system load, more instrumentation in these nodes or isolated issues that need to be addressed.

Conclusion: Whatever the cause, this chart identifies which nodes in the BGL system were the most active, or were logging events most frequently. As a result, this will help focus on monitoring, or troubleshoot or addressed node-specific problems, or the overall system activity.

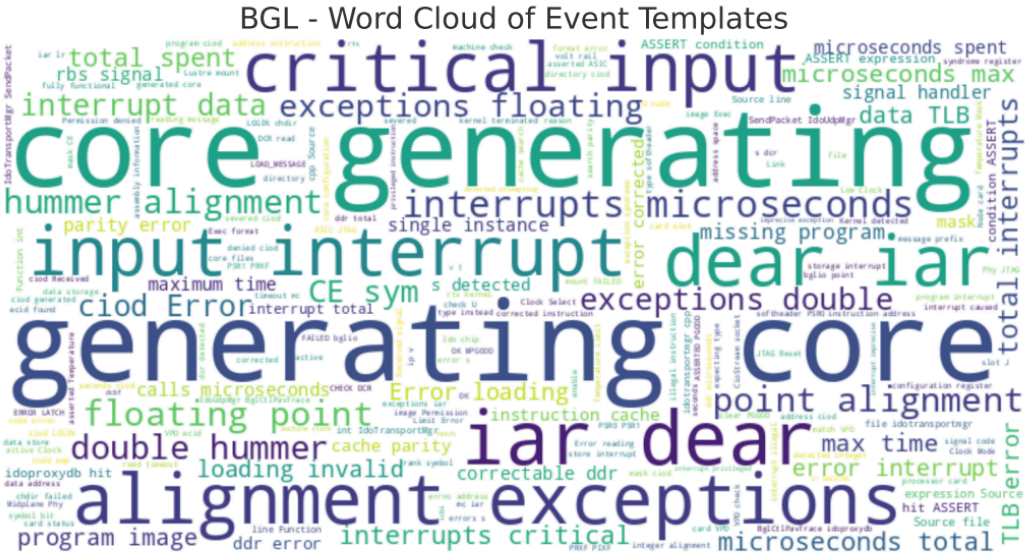

Figure 11: Word Cloud of BGL Event Templates

Figure 11 displays frequently occurring terms in BGL logs, such as "interrupt", "core", and "alignment". It indicates a strong focus on low-level system and hardware events, offering direction for anomaly investigation.

Insights:

Typical words such as generating, core, input, dear, iar, interrupt, alignment, and exceptions indicate that the system frequently stores messages pertaining to processor cores, input/output activities, alignment faults, and interrupts

Usage of technical terms such as floating point, cache, TLB, parity error, and microseconds implies detailed system timing, hardware-level operations, and memory or instruction issues logging

Terms such as invalid, loading, and critical imply the presence of error status or significant system states being monitored

This graph assists in determining dominating operational problems and the nature of errors or activities most frequently recorded in the BGL log data

Conclusion:This graph is a snapshot of salient terms in BGL event templates that can inform further log analysis by highlighting key subjects and error types that dominate the system's narrative of operation.

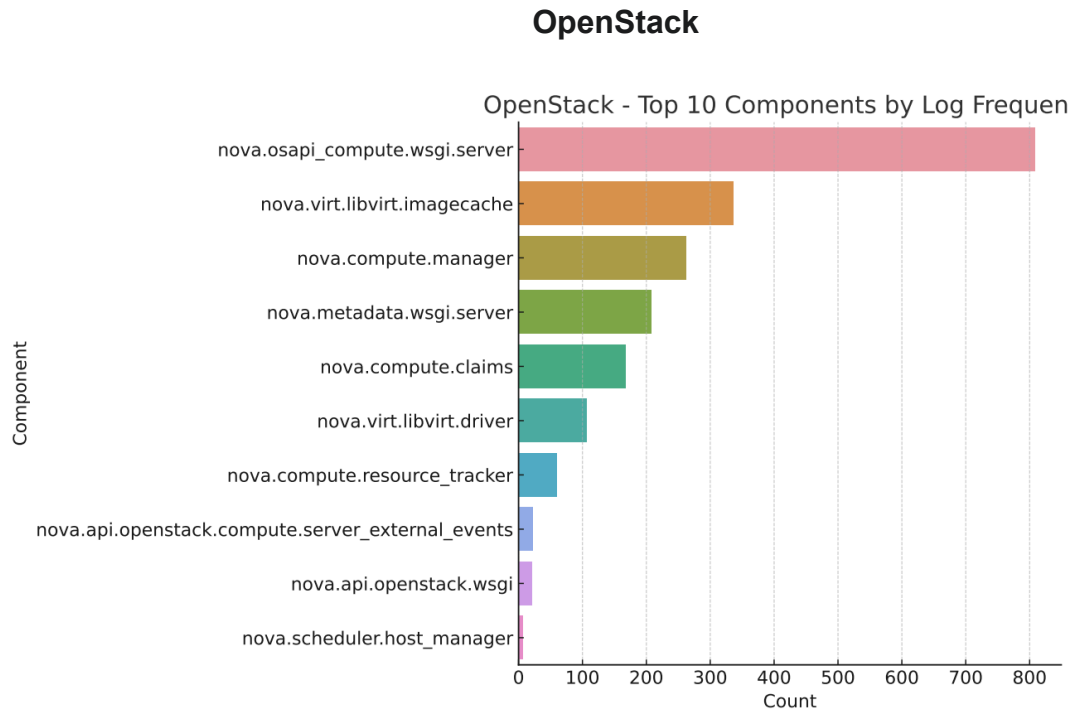

Figure 12: Top Logged Components in OpenStack Logs

Figure 12 shows which OpenStack components generate the most logs, with `nova.osapi_compute.wsgi.server` leading. The data indicates logging is focused on compute services and VM management, suggesting key areas for monitoring.

Insights:

nova.osapi_compute.wsgi.server records the most messages, indicating that it is at the heart of compute web service operations

nova.virt.libvirt.imagecache and nova.compute.manager both record high frequency logs, indicating active involvement in image caching and virtual machine management

Modules such as nova.metadata.wsgi.server and nova.compute.claims also exhibit high frequency of logging, reflecting their involvement in metadata handling and resource claiming

Lower-level components such as nova.scheduler.host_manager and nova.api.openstack.wsgi can be engaged in less frequent or lower-level API operations and orchestration

The graph suggests that most of the logs are targeting the compute and virtualization layers of OpenStack, specifically with regards to API servers and resource management

Conclusion:This distribution shows which OpenStack components are most busy writing logs, providing a clear picture of system hotspots and where to focus for performance debugging, troubleshooting, or anomaly detection.

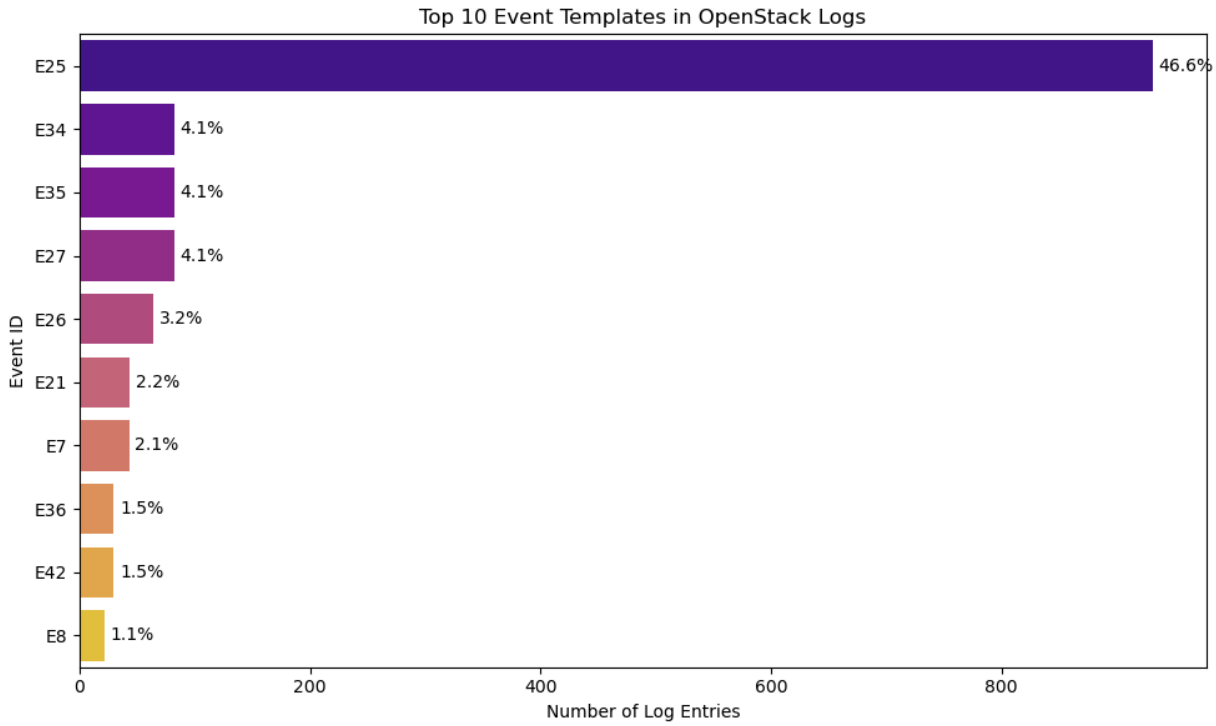

Figure 13: Event Template Distribution in OpenStack Logs

Figure 13 shows log percentages for event templates in OpenStack, with E25 dominating at over 40%. This reveals repetitive system activity and highlights potential for log compression and focused anomaly checks on rare templates.

Insights:

Event ID E25 is predominant in log distribution with more than 40 percent of the total events being logged. This indicates that one event type is extremely repetitive and either core to the functioning of OpenStack or monitoring

Event IDs E34, E35, and E27 come next with much smaller percentages, revealing an extreme decline in frequency after E25

The rest of the event IDs all have a small and similar percentage of logs, each only accounting for a few percent of the total

This bias suggests that the system has a strong dependence on a number of event templates, which are probably repetitive system actions like API calls, resource checks, or VM state changes

Conclusion:This graph helps to identify what types of events are most frequent in OpenStack's log. Being aware of what E25 is could be key information about system health and efficiency, while less frequent events might provide more anomaly or diagnostic insight.

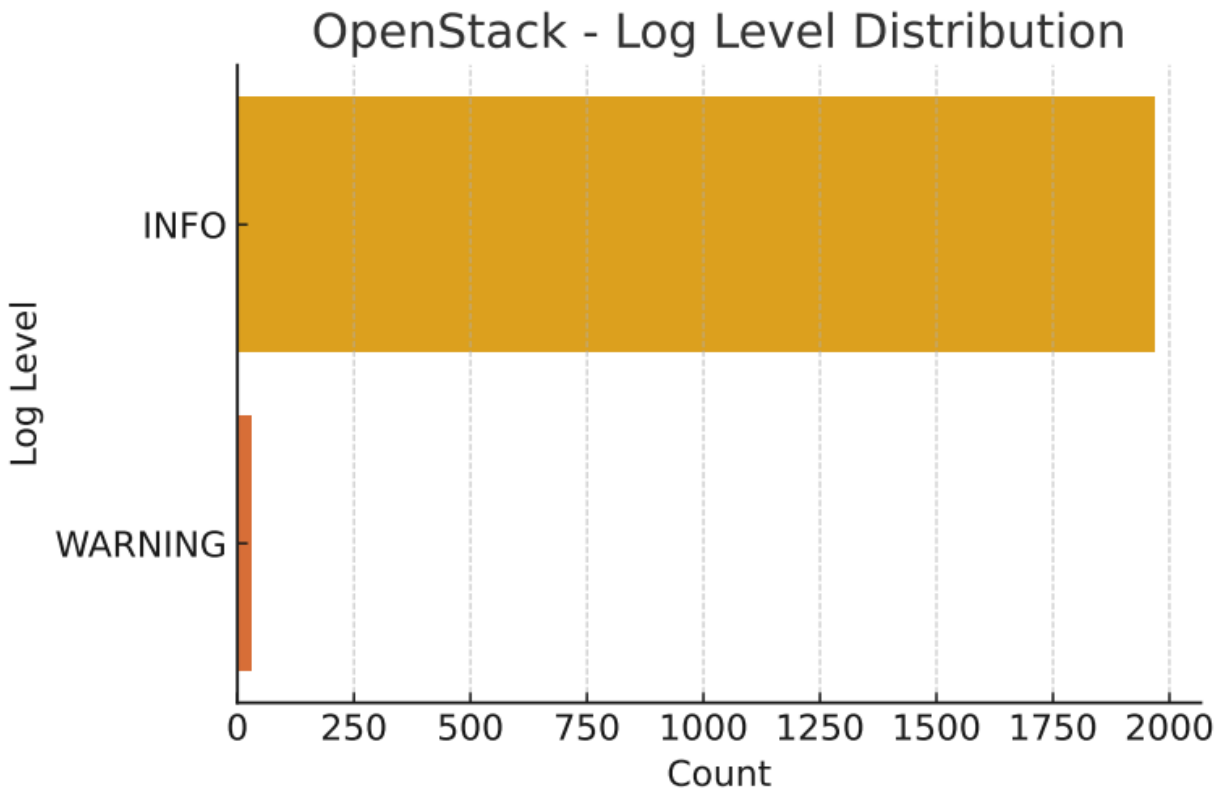

Figure 14: Log Severity Level Distribution in OpenStack

Figure 14 shows that INFO-level logs overwhelmingly dominate, while WARNING-level entries are sparse. The system is verbose in tracking normal behavior but less informative on borderline issues, suggesting a need for improved warning-level logging.

Insights:

INFO-level logs comprise the vast majority of entries and indicate that the majority of OpenStack logs are created to log normal operations, system states, or common activity

WARNING-level logs are relatively fewer in comparison to the others, which indicates either the system does not frequently experience warning conditions or does not log non-critical problems extensively

This asymmetry implies that even though the logging mechanism is chatty in recording normal behaviour, it yields relatively low levels of visibility for possible early warnings or recoverable faults

The low frequency of WARNING logs can indicate either a fairly stable system or low logging of issues that are not severe enough to be considered errors.

Conclusion: This chart assists in grasping OpenStack log reliability and verboseness. It shows a system crafted for observability through verbose INFO messages but one that could be improved with more informative warning-level logging to identify issues preemptively.

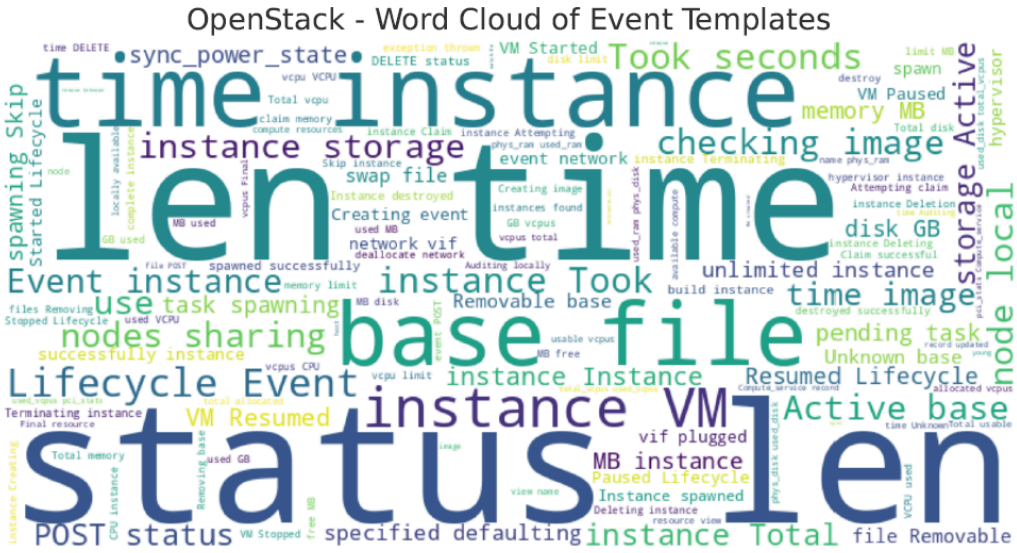

Figure 15: Word Cloud of OpenStack Event Templates

Figure 15 captures the frequent vocabulary in OpenStack logs. Terms like "instance", "file", and "vm" highlight the focus on resource allocation and virtual machine lifecycle events, guiding areas for anomaly monitoring.

Insights:

High-frequency terms like instance, time, file, status, and len indicate that OpenStack logs focus especially on instance management, time tracking, and file operation management

Keywords such as VM, Resumed, Paused, Active, and lifecycle indicate virtual machine state and state transitions as a shared theme in the logging activity

Keywords such as storage, memory, vcpu, MB, and GB suggest that there is a strong emphasis on resource consumption and monitoring within the cloud ecosystem

Terminology such as syncing, spawning, image, and POST indicates operational activities such as VM creation, API calls, and data syncing are being logged on a regular basis

This word cloud exposes OpenStack's working vocabulary and common events that prevail in its log templates

Conclusion:This diagram helps in finding important objects and operations that are continuously tracked by OpenStack. It helps to learn system behavior, template parsing, and areas of possible interest for anomaly or performance analysis.

Thunderbird

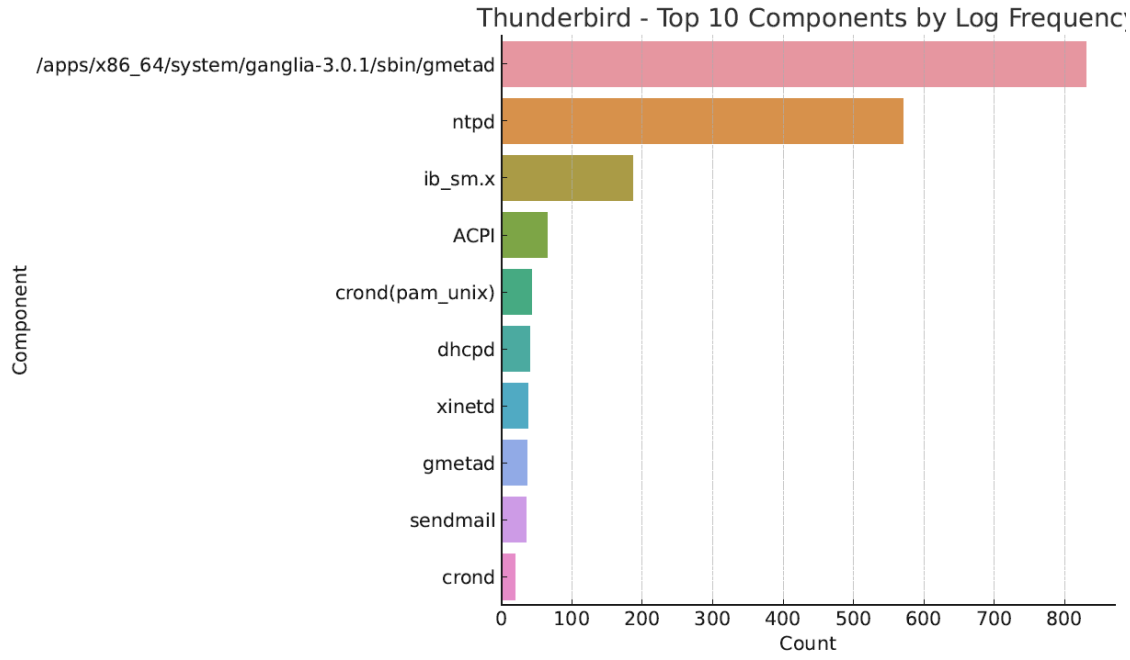

Figure 16: Top Logged Components in Thunderbird Logs

Figure 16 ranks Thunderbird components by number of logs, led by Ganglia's monitoring daemon (`gmetad`). It shows the system prioritizes infrastructure and synchronization services, which are key for reliability monitoring.

Insights:

The /apps/x86_64/system/ganglia-3.0.1/sbin/gmetad module logs the highest number of entries, indicating that Ganglia's monitoring daemon is very active on this system

ntpd, the network time protocol daemon, also reports a high number of logs, indicating a lot of synchronization or time activity in the environment

ib_sm.x, which is expected to be utilized in InfiniBand subnet management, is trailed by a moderate volume of logs, indicating a function within network communication or infrastructure management

Other factors like ACPI, crond(pam_unix), dhcpd, xinetd, gmetad, sendmail, and crond all have fewer logs but are all assisting the system in monitoring and controlling services overall

The logged component diversity indicates a wide variety of system services being logged, such as scheduling, email, network booting, and power management

Conclusion: This graph shows which services are most highly logged into Thunderbird. It demonstrates a system with strong emphasis on monitoring, synchronization, and infrastructure-level event detection, which would be useful for system administrators or for anomaly detection initiatives.

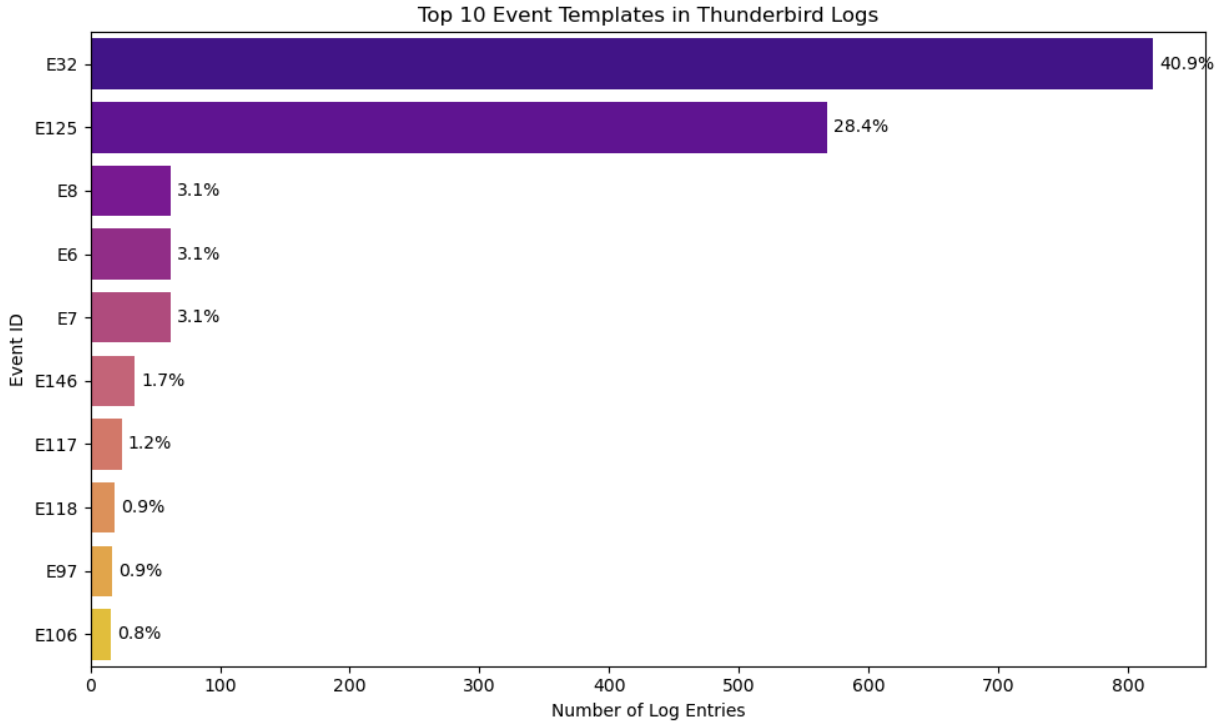

Figure 17: Event Template Distribution in Thunderbird

Figure 17 shows how heavily logs are dominated by templates like E32 and E125. This indicates a repetitive structure and helps isolate meaningful anomalies by filtering recurring messages.

Insights:

Event ID E32 is responsible for over 40 percent of the logs, which implies that a single event type is responsible for the majority of system activity or monitoring.

Then there is Event ID E125, which also comprises a big chunk of the dataset because it takes part in recurring or routine activity

The other event IDs, from E8 to E106, account for small sections of the logs, each less than 5 percent

This skewed distribution suggests that Thunderbird logs are highly redundant with a strong dependence on only a few template types

The co-occurrence of a pair of event IDs indicates concentration on routine system-level activity, for example, health checks or regular metric gathering, while the remainder that are less frequent may indicate more specialized events or exceptions

Conclusion: This graph highlights the frequency of certain log patterns within Thunderbird's log traffic. It assists in identifying which event templates would be most useful to filter, compress, or analyze for anomaly detection.

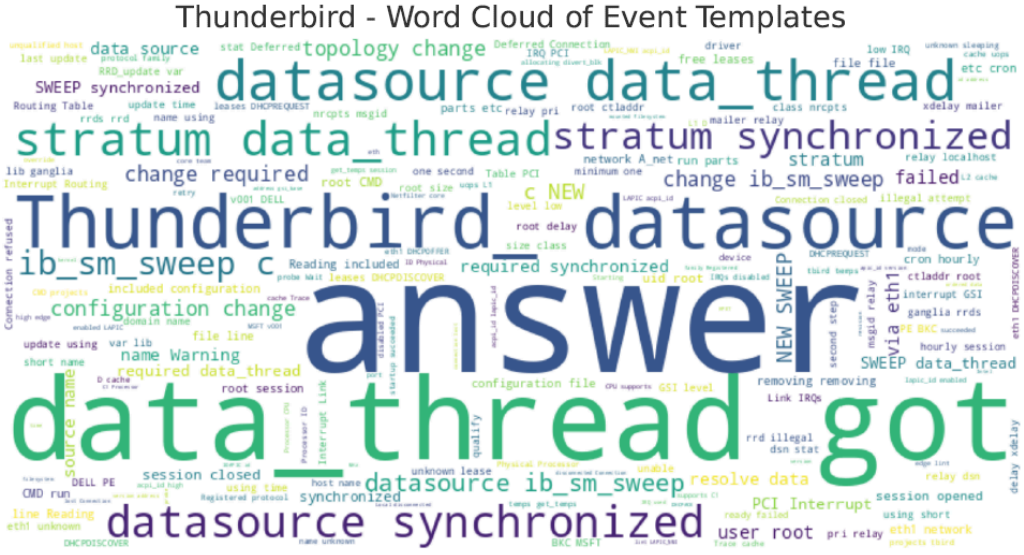

Figure 18: Word Cloud of Thunderbird Log Templates
Figure 18 showcases key terms in Thunderbird logs such as "data", "thread", and "configuration". It reflects a strong focus on synchronization, system responses, and service monitoring.

Insights:

Words like answer, got, data, thread, datasource, and thunderbird stand out, indicating that they are at the center of most log messages and represent key operational events or entities

The presence of the words synchronized, ib_sm_sweep, and stratum shows that synchronization activity, thread activity, and network activity are logged from time to time

Key words such as configuration change, resolved, CMD, and warning indicate that configuration events, execution of commands, and warning messages are typical subjects

The presence of juxtaposed system-level terminology such as cron, DHCPREQUEST, root, and relay also indicates periodic logging of task scheduling, network services, and user-level activity

This word cloud is a good indication that Thunderbird logs strongly concentrate on data threads being monitored, synchronization states, network events, and service responses

Conclusion: This diagram provides an overview of major subjects in Thunderbird logs, identifying operational themes and frequent appearing patterns beneficial for log analysis, parsing techniques, or anomaly detection.

Zookeeper

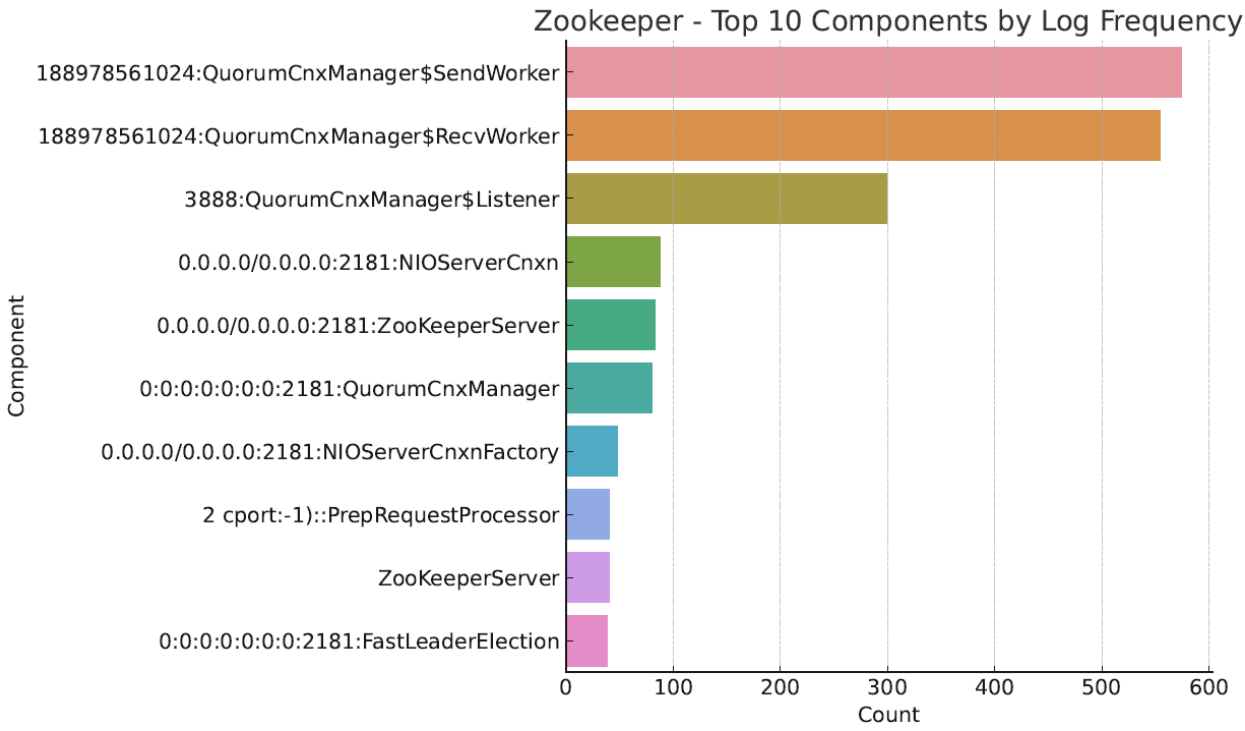

Figure 19: Top Logged Components in Zookeeper Logs

Figure 19 ranks the most active components in Zookeeper logs, with `SendWorker` and `RecvWorker` dominating. The focus is on inter-node messaging and quorum coordination, central to Zookeeper's function..

Insights:

QuorumCnxManager\$SendWorker and QuorumCnxManager\$RecvWorker top the list with the highest log frequencies, indicating that much of the activity is comprised of sending and receiving messages between quorum members

The Listener on port 3888 also produces a high volume of logs, indicating active involvement in client or peer communication

The other classes such as NIOServerCnxn, ZooKeeperServer, and QuorumCnxManager capture the essential coordination and network roles of Zookeeper

PrepRequestProcessor and FastLeaderElection components are less common but crucial steps in Zookeeper request processing and leadership election processes

The frequent recording of IP-bound objects with port 2181 emphasizes the significance of network I/O activity in Zookeeper's functioning

Conclusion: This graph indicates that the majority of the log activity within Zookeeper is communication and coordination among nodes. It illustrates a distributed system in which messaging between nodes and leadership management are at the center of system activity and therefore under extreme examination.

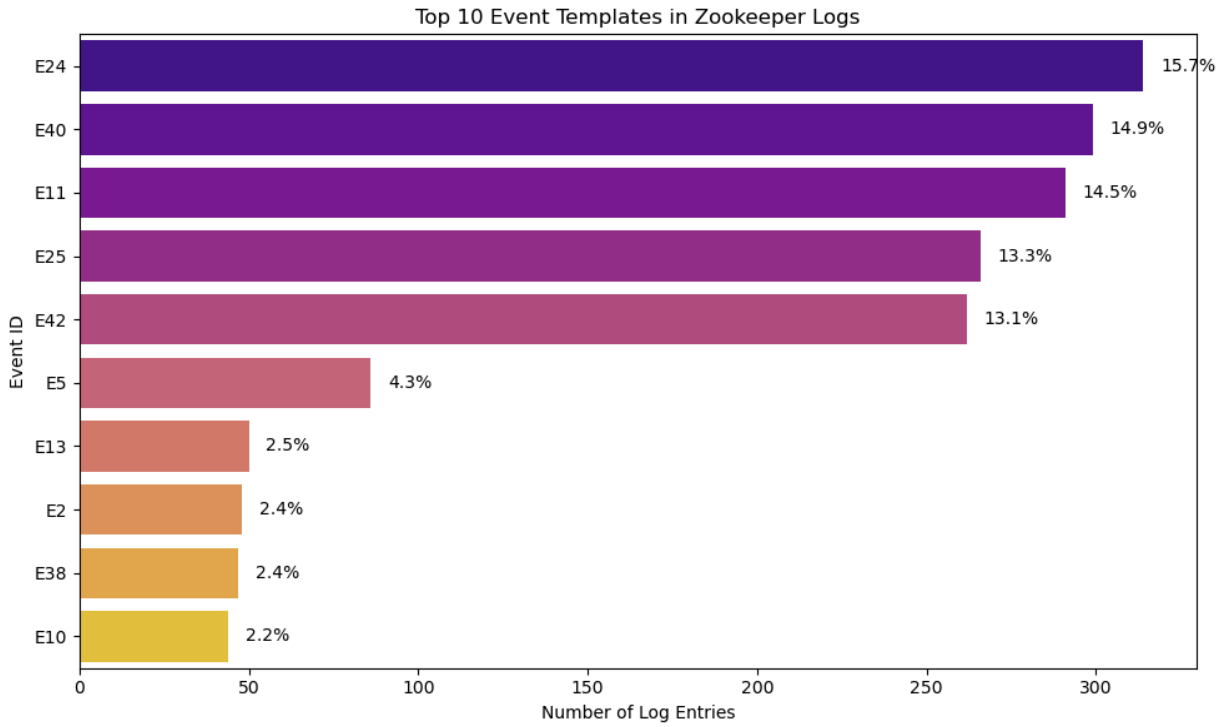

Figure 20: Event Template Distribution in Zookeeper

Figure 20 presents the proportion of logs per event ID. A few event templates like E24 and E40 dominate, pointing to routine operations such as request processing and heartbeats. This helps focus attention on less frequent, potentially anomalous events.

Insights:

Event IDs E24, E40, and E11 are most frequent, with each of them over 13 percent of total logs, meaning that these templates represent very common activities

Event IDs E25 and E42 also have substantial parts in common, which points to their frequent appearance in system operations or monitoring

The rest of the events E5 through E10 represent diminishing frequency, with each taking up 2 to 5 percent of the logs

This distribution mirrors a moderately skewed logging pattern in which a small number of templates are dominant and the others fall into specialized or less common actions

This repetition might include routine operations like heartbeat messages, session events, or request processing being logged at detailed levels

Conclusion: This graph displays the most frequent log patterns in Zookeeper, helpful for directing efforts towards common event types when parsing logs or picking out outlying templates that could represent anomalies or out-of-the-norm functionality.

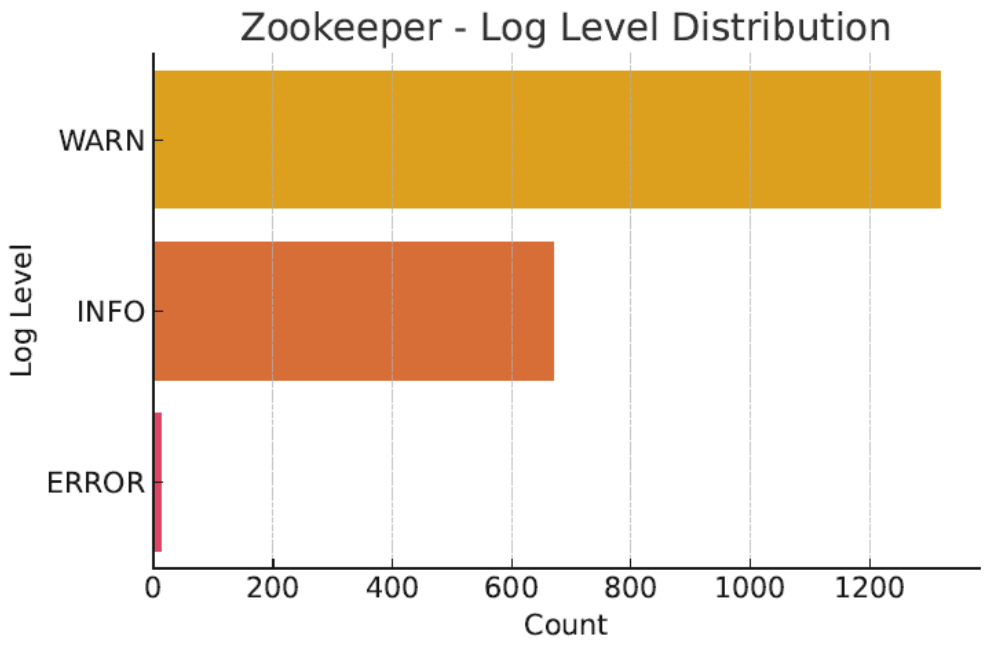

Figure 21: Log Severity Level Distribution in Zookeeper
Figure 21 shows that WARN-level logs outnumber INFO logs, indicating that the system logs potential issues more than normal operations. This setup is helpful for preemptive debugging and proactive maintenance.

Insights:

WARN-level logs are most common, meaning the system logs normally things that aren't necessarily critical but might require a look or warn of future problems.

INFO logs come next in terms of frequency, revealing how a large proportion of logging is still concerned with daily operations and system status messages

ERROR logs are very few, which indicates that critical failure or unrecoverable issues do not happen often in the given time period

The existence of WARN logs is a sign that Zookeeper is busy capturing and logging non-critical irregularities, which may prove useful during proactive maintenance or debugging

That there are fewer INFO than WARN implies that the logging level can be configured to record more potential trouble occurrences and less ordinary operating activity.

Conclusion: This distribution gives an insight into the priority that Zookeeper assigns to various categories of log messages. The system appears to be actively disclosing warnings compared to informational logs, which will assist in the early identification of anomalies without appreciably increasing the rate of critical errors.

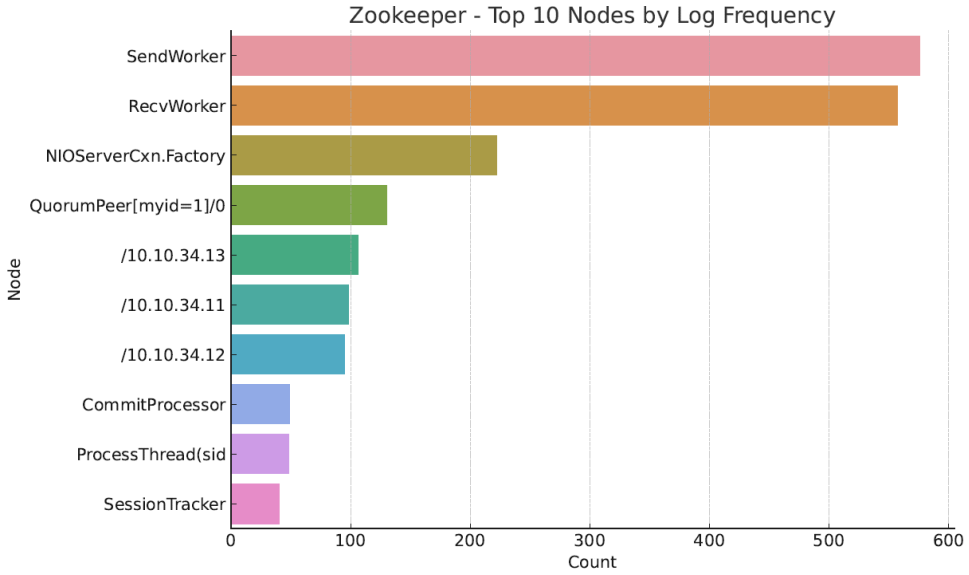

Figure 22: Top Logging Nodes in Zookeeper

Figure 22 ranks Zookeeper nodes by log count. `SendWorker`, `RecvWorker`, and other network-related components show the highest activity, highlighting coordination-heavy processes in distributed environments.

Insights:

SendWorker and RecvWorker lead in log frequency, suggesting that message sending and receiving are the most logged operations among the nodes

NIOServerCxn.Factory follows, which signifies the logging of server-side network connection handling activities

QuorumPeer[myid=1]/0 and IP addresses /10.10.34.13, /10.10.34.11, and /10.10.34.12 have moderate values, showing these peer or client nodes are currently involved in Zookeeper activities

The remaining nodes such as CommitProcessor, ProcessThread(sid, and SessionTracker produce less log but play critical roles in transaction coordination and session management

The graph demonstrates a logging emphasis on peer coordination activity, connection management, and inter-node communication in the distributed Zookeeper setup

Conclusion: This graph assists in determining which areas of the Zookeeper system are the most active or most significant in regards to logging. It provides insight into operational hotspots and can direct attention for system monitoring or anomaly detection initiatives.

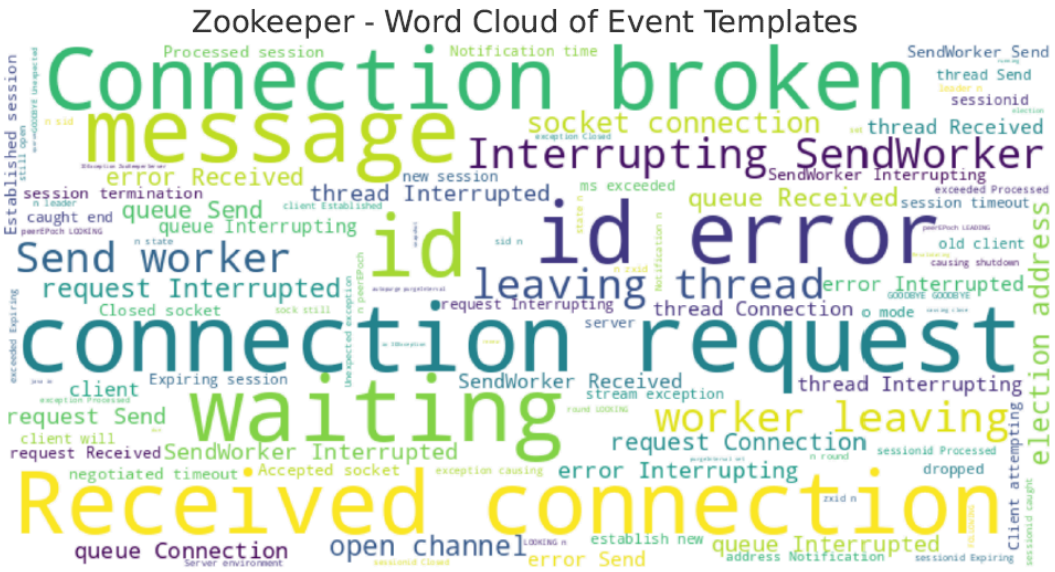

Figure 23: Word Cloud of Zookeeper Event Templates

Figure 23 shows common words like "connection", "message", and "thread" in Zookeeper logs. The vocabulary emphasizes connection handling, distributed communication, and fault logging—key themes in system behavior.

Insights:

Typical terms like connection, request, received, message, waiting, id, error, and broken imply a strong emphasis on connection management, data transfer, and error handling within the Zookeeper system SendWorker, Interrupting, queue, thread, and worker are names that depict thread-level operations and node-to-node messaging with emphasis on Zookeeper's distributed and multithreaded nature

The fact that words like error, broken, interrupted, and leaving are repeated suggests that Zookeeper logs a lot of transitions, failures, or interruptions in its session or messaging flows.

Other terminologies like accepted, processed, established, and timeout are also seen, suggesting that the logs record both successful transactions and error states at different stages

Conclusion: This graph summarizes the vocabulary of Zookeeper's log templates, reflecting an emphasis on connection stability, thread coordination, and message handling. It is used to identify the most common operating concern areas and provides a point of departure for the selection of relevant log patterns in downstream analysis or anomaly detection.