

Log Analysis

Computer-system logs provide a glimpse into the states of a running system. Both infrastructure and applications generate short messages, in response to events, that are collected in a system-specific logs. The content and format of logs can vary widely from one system to another and even among components within a system. As the content of the logs is varied, so are their uses. Indeed, while a printer log might be used for troubleshooting, a Web-server log is used to study traffic patterns to maximize advertising revenue.

Objectives:

- Mastering ingestion of a stream of data (Using the latest version of Kafka and Kraft)
- Monitoring server activities for both infrastructure and applications
- Using generative AI to interact with indexed data in a near real time.

Overview

This section provides an overview of some of the most common applications of log analysis, describes some of the logs that might be analyzed and the methods of analyzing them, and elucidates some of the lingering challenges. Log analysis is a rich field of research, we do intend to provide a clear understanding of why log analysis is both vital and difficult.

Data

We provide 20 days of continuous integration logs from Mozilla. Each file represent more than 100 000 builds' job.

1. Debugging

Many logs are intended to facilitate debugging. The simplest and most common use for a debug log is to grep for a specific message. If a server operator believes that a program crashed because of a network failure, then he or she might try to find a "connection dropped" message in the server logs. In many cases, it is difficult to figure out what to search for, as there is no well-defined mapping between log messages and observed symptoms. When a Web service suddenly becomes slow, the operator is unlikely to see an obvious error message saying, "ERROR: The service latency increased by 10% because bug X, on line Y, was triggered." Instead, users often perform a search for severity keywords such as "error" or "failure." Such severity levels are often used inaccurately, however, because a developer rarely has complete knowledge of how the code will ultimately be used.

Another debugging use could be used in Continuous integration practices. We can analyze builds outcome to investigate failures of Builds and to provide feedback to the upstream development phases. On several open-source systems, monitoring was able to analyze billions of lines of logs, accurately detect anomalies that were often overlooked by human eyes, and visualize the results in a single-page decision-tree diagram.

Machine-learning techniques, especially anomaly detection, are commonly used to discover "interesting" log messages. Machine-learning tools usually require input data as numerical feature vectors. It is nontrivial to convert free-text log messages into meaningful features.

Challenges remain in statistical anomaly detection. Even if some messages are abnormal in a statistical sense, there may be no further evidence on whether these messages are the cause, the symptom, or simply innocuous. Also, statistical methods rely heavily on log quality, especially whether "important" events are logged. The methods themselves do not define what could be "important."

Static program analysis can help discover the root cause of a specific message by analyzing paths in the program that could lead to the message. Static analysis can also reveal ways to improve log quality by finding divergence points, from which program execution might enter an error path; such points are excellent candidates for logging instrumentation. Static analysis techniques are usually limited by the size and complexity of the target system. It takes hours to analyze a relatively simple program such as Apache Web Server. Heuristics and domain knowledge of the target system usually make such analyses more effective.

2. Performance

Log analysis can help optimize or debug system performance. Understanding a system's performance is often related to understanding how the resources in that system are used. Some logs are the same as in the case of debugging, such as logging lock operations to debug a bottleneck. Some logs track the use of individual resources, producing a time series. Resource-usage statistics often come in the form of cumulative use per time period (e.g., b bits transmitted in the last minute). One might use bandwidth data to characterize network or disk performance, page swaps to characterize memory effectiveness, or CPU utilization to characterize load-balancing quality.

As in the debugging case, performance logs must be interpreted in context. Two types of contexts are especially useful in performance analysis: the environment in which the performance number occurs and the workload of the system.

Performance problems are often caused by interactions between components, and to reveal these interactions you may have to synthesize information from heterogeneous logs generated by multiple sources. Synthesis can be challenging. In addition to heterogeneous log formats, components in distributed systems may disagree on the exact time, making the precise ordering of events across multiple components impossible to reconstruct. Also, an event that is benign to one component (e.g., a log flushing to disk) might cause serious problems for another (e.g., because of the I/O resource contention). As the component causing the problem is unlikely to log the event, it may be hard to capture this root cause. These are just a few of the difficulties that emerge.

One approach to solving this problem is to compute influence, which infers relationships between components or groups of components by looking for surprising behavior that is correlated in time.⁸ For example, bursty disk writes might correlate in time with client communication errors; a sufficiently strong correlation suggests some shared influence between these two parts of the system.

A salient challenge in this area is the risk of influencing the measurements by the act of measuring. Extensive logging that consumes resources can complicate the task of accounting for how those resources are used in the first place.

3. Security

Logs are also used for security applications, such as detecting breaches or misbehavior, and for performing postmortem inspection of security incidents. Depending on the system and the threat model, logs of nearly any kind might be amenable to security analysis: logs related to firewalls, login sessions, resource utilization, system calls, network flows, etc. Intrusion detection often requires reconstructing sessions from logs.

Security applications face the distinguishing challenge of an adversary. To avoid the notice of a log-analysis tool, an adversary will try to behave in such a way that the logs generated during the attack look—exactly or approximately—the same as the logs generated during correct operation.

4. Prediction

Log data can be used to predict and provision for the future. Predictive models help automate or provide insights for resource provisioning, capacity planning, workload management, scheduling, and configuration optimization. From a business point of view, predictive models can guide marketing strategy, ad placement, or inventory management.

Some analytical models are built and honed for a specific system. Experts manually identify dependencies and relevant metrics, quantify the relationships between components, and devise a prediction strategy.

Predictive models often provide a range of values rather than a single number; this range sometimes represents a confidence interval, meaning the true value is likely to lie within that interval. Whether or not to act on a prediction is a decision that must weigh the confidence against the costs (i.e., whether acting on a low-confidence prediction is better than doing nothing). Acting on a prediction may depend on whether the log granularity matches the decision-making granularity. For example, per-query resource-utilization logs do not help with task-level scheduling decisions, as there is insufficient insight into the parallelism and lower-level resource-utilization metrics.

5. Reporting and Profiling

Another use for log analysis is to profile resource utilization, workload, or user behavior. Logs that record characteristics of tasks from a cluster's workload can be used to profile resource utilization at a large data center. The same data might be leveraged to understand inter-arrival times between jobs in a workload, as well as diurnal patterns.

In addition to system management, profiling is used for business analytics. For example, Web-server logs characterize visitors to a Web site, which can yield customer demographics or conversion and drop-off statistics. Web-log analysis techniques range from simple statistics that capture page popularity trends to sophisticated time-series methods that describe access patterns across multiple user sessions. These insights inform marketing initiatives, content hosting, and resource provisioning.

A variety of statistical techniques have been used for profiling and reporting on log data. Clustering algorithms such as k-means and hierarchical clustering group similar events. Markov chains have been used for pattern mining where temporal ordering is essential.

Many profiling and alerting techniques require hints in the form of expert knowledge. For example, the k-means clustering algorithm requires the user either to specify the number of clusters (k) or to provide example events that serve as seed cluster centers. Other techniques require heuristics for merging or partitioning clusters. Most techniques rely on mathematical representations of events, and

the results of the analysis are presented in similar terms. It may then be necessary to map these mathematical representations back into the original domain, though this can be difficult without understanding the log semantics.

Classifying log events is often challenging. To categorize system performance, for example, you may profile CPU utilization and memory consumption. Suppose you have a performance profile for high CPU utilization and low memory consumption, and a separate profile of events with low CPU utilization and high memory consumption; when an event arrives containing low CPU utilization and low memory consumption, it is unclear to which of the two profiles (or both) it should belong. If there are enough such events, the best choice might be to include a third profile. There is no universally applicable rule for how to handle events that straddle multiple profiles or how to create such profiles in the first place.

6. Logging infrastructures

A logging infrastructure is essential for supporting the variety of applications described here. It requires at least two features: log generation and log storage.

Most general-purpose logs are **unstructured text**. Developers use `printf` and string concatenations to generate messages because these primitives are well understood and ubiquitous. This kind of logging has drawbacks, however. First, serializing variables into text is expensive (almost 80 percent of the total cost of printing a message). Second, the analysis needs to parse the text message, which may be complicated and expensive.

Conclusion

The applications and examples in this article demonstrate the degree to which system management has become log-centric. Whether used for debugging problems or provisioning resources, logs contain a wealth of information that can pinpoint, or at least implicate, solutions.

Although log-analysis techniques have made much progress recently, several challenges remain. First, as systems become increasingly composed of many, often distributed, components, using a single log file to monitor events from different parts of the system is difficult. In some scenarios, logs from entirely different systems must be cross-correlated for analysis.

Second, the logging process itself requires additional management. Controlling the verbosity of logging is important, especially in the event of spikes or potential adversarial behavior, to manage overhead and facilitate analysis.

A third challenge is that although various analytical and statistical modeling techniques can mine large quantities of log data, they do not always provide actionable insights. For example, statistical techniques could reveal an anomaly in the workload or that the system's CPU utilization is high but not explain what to do about it. The interpretation of the information is subjective, and whether the information is actionable or not depends on many factors. It is important to investigate techniques that trade off efficiency, accuracy, and actionability.