Software Monitoring, Logging, and AIOps

Arnav Gupta

April 22, 2024

Contents

1 DevOps

Software operations: people and management processes associated with IT service management to deliver the right set of services at the right quality and at competitive costs for customers

Tradition system admin as manual, automated operations use repeated scripts (manually-coded rules).

DevOps combines software development and operations to shorten the delivery cycle. DevOps aims to improve collaboration between development and operations. DevOps is a culture that aims to break development-operations barrier with continuous integration and continuous feedback. DevOps engineers monitor software systems to understand state and runtime behaviour.

Monitoring is critical for ensuring quality of delivered services and bridges the gap between development and operations.

Challenges:

- observing a system perturbs it
- increasing volume and complexity of monitoring data

2 Logging

Inserting log statements into code can help debugging, anomaly detection, system comprehension, auditing, and more. May be the only way because debeggers are not always available or applicable (production, distributed).

DevLogs can be:

- performance metrics
- timestamp
- level/severity
- thread name
- method name
- message

More logs are better since allows diagnosis, distributed systems require them, and legal compliance. However there are performance overheads. Logging everything is the same as logging nothing (since unusable).

Log messages should tell what happened and why:

- who was involved
- what happened
- where/when/why/how did it happen

Log types are:

- **AAA (authentication, authorization, access)**: successful and failed authentication or authoorization decisions, and system/data/application/remote access
- **change events**: system or application changes, data changes, and application and component installation and changes
- **badness/threats**: invalid inputs and other likely application abuses, and other security issues known to affect the application
- **resource issues**: exhausted resources, exceeded capacities, connectivity issues/problems, reached limits
- **mixed availability issues**: startups and shutdown of systems/applications/modules/components faults and errors (availability), backup successes and failures

Log structure can be unstructured (require later processing, difficult to find info).

Logs contain data aside from context. Can be consumed in different formats to add structure.

Old fashioned file logs are robust but don't scale since risk of losing logs, killing machine, and difficulty to prase.

Modern log architectures are:

- syslog (old) which sends logs to an aggregator that stores files on an individual machine
- aggregate into a single engine, providing a single interface to log data, so search language is more expressive than grep or regex, and can give analytics

Simple analytics involve checking for exceptions, overtime, performance from real users. Can check unique values, counts, and sanity check. See behaviour overtime and traffic patterns overtime.

2.1 Library

Logging library disables certain log statements while enabling others.

Logger hierarchy is maintained using a dot structure. Get any logger with `LogManager.getLogger`.

For log level hierarchy, if log level is defined in the config use it, otherwise use parent's level.

Logger name and assigned logger config must match exactly.

Appender allows printing logs to multiple destinations forwarded down the hierarchy.

Ideally, logs should be put into a formatted datastore. Many logs are formatted according to a pattern, so this requires regexp to do any kind of analysis.

2.2 Analysis

Filtering, normalization, and correlation can be done with:

- **raw log data**: start with this, first input into the process

- **filter**: look for log messages that are important, unimportant ones can be dropped to reduce load on the overall system
  - keeping or discarding log data based on importance
  - parsing the raw log message into a common format for easier analysis (log parsing)
- **normalization**: take the raw log data and map its various elements (ex. source and destination IP) to a common format
  - assume that important log data is known
  - parse log messages into common format
- **correlation**: often leads to groups of individually unimportant events to be flagged
  - tie together several similar or dissimilar events in a single piece of knowledge that something much larger is happening
- **action**: generally what is done after a correlation has occurred

Steps for normalization:

1. get documentation for product(s) being used
2. read documentation for descriptions of what the raw log data looks like and what each field is
3. come up with the proper parsing expression to normalize data, like regex
4. test parsing logic on sample raw log data
5. deploy parsing logic

Rule correlation deals with crafting a rule which is able to model a certain behaviour.

AIOps is Artificial Intelligence for IT operations.