September 2023

Proactive Anomaly Detection in Large-Scale Cloud-Native Databases

Engy Elsayed

Helen Dong

Suzhen Lin

Thomas Chang

Reza Sherkat

Proactive Anomaly Detection in Large-Scale Cloud-Native Databases

ABSTRACT

This disclosure describes techniques to identify anomalous patterns in customer workloads from database logs and to enable timely, corrective action that ensures uninterrupted operation of the database. Examples of anomalies include sudden increases (bursts) in the number of error messages written to a log file. An adaptive behavior norm is defined for each message type. Time instances or periods when the gap between messages of a given type in the database log deviate from the expected behavior norms are detected. A deviation from the behavior norm is a potential indicator of database problems. An anomaly detection tool outputs a ranked list of log statements exhibiting spikes of activity along with their time intervals that a database administrator (DBA) can examine to take corrective action. By automating anomaly detection, the valuable time of DBAs can be spent acting on issues rather than finding them.

KEYWORDS

- Log anomaly
- Burst detection
- Anomaly detection
- Database log
- Database-as-a-service
- Binomial distribution
- Cloud-native database
- Service-level agreement (SLA)

BACKGROUND

Maintaining compliance with an SLA (service-level agreement) in large-scale, database-as-a-service as provided by cloud computing providers requires an accurate understanding of customer workloads and their dynamics. Accurate understanding of workloads helps to identify anomalous patterns that may affect database operations and enables taking appropriate corrective action before the service is impacted. Current techniques of human database administrators (DBAs) analyzing logs to identify anomalies and trigger alarms do not scale to large fleets (e.g., thousands) of cloud databases.

DESCRIPTION

This disclosure describes techniques to automatically identify anomalous patterns in customer workloads from database logs that enable timely, corrective action to ensure uninterrupted operation of the database. An anomaly detection tool attaches to large-scale, cloud-native databases and extracts observable anomalies from database logs. The anomaly detection tool can work either on streaming log files or on static (offline) log files. Examples of anomalies include sudden increases (bursts) in the number of error messages (e.g., from once a day to fifty in an hour) written to the log file of the database, sudden increases in messages that explain unwanted behavior (e.g., inserting duplicate keys), etc.

An adaptive behavior norm is defined for each message type. To enable the behavior norms to be adaptive, no fixed thresholds are selected. Time instances or periods when the gap between messages of a given type in the database log deviate from the expected behavior norms are detected. A deviation from the behavior norm is a potential indicator of database problems and can serve as a signal to DBAs to investigate further. Upon analyzing the database log file,

the anomaly detection tool outputs a ranked list of log statements that exhibit spikes of activity along with their time intervals. The techniques are explained in greater detail below.

Behavior norm

Anomalies can be defined via behavior norms in several ways. For example, a behavior norm for a database can be defined in terms of observable features, e.g., the volume of logs per hour, the number of warnings per day, performance characteristics, etc. The behavior norm can also be based on the content of log statements. An anomaly is defined as a deviation from the behavior norm. The definition of behavior norms can apply across the fleet of databases operated by the cloud database provider. Similarly, the definition of alarms that indicate the formation of various problems can apply across the database fleet.

Burstiness in log statements

The occurrence of log statements is modeled by a binomial probability distribution. The expected probability of appearance of a log statement of a certain type in a given time window is incrementally calculated and used as a basis for calculating the expected number of occurrences of the log statement in the time window along with its standard deviation. Burstiness in a given type of log statement is identified by an appearance of the log statement in a certain time window in numbers greater than expected by a certain number of standard deviations. The threshold number of log statements to declare burstiness can be based on the statement type and can be modified over time.

Detection of anomalies in database logs

A baseline probability of each target event (type of log statement whose burstiness is indicative of a problem) is calculated. The proportion of target events with total events within a

certain time window is calculated. For the purposes of computing the baseline probability, target-events phrases are assessed from all the phrases in the log file. The sum of target events and the sum of the total events is used to compute the baseline probability of each target event.

The bursty state probability can be calculated as the baseline probability multiplied by a sensitivity constant. The sensitivity controls the granularity of detected bursts - a smaller sensitivity favors bursts with larger in-between gaps whereas a larger one identifies bursts that are relatively close. The sequence of observed frequency proportions is used to predict the state of the database, e.g., bursty versus baseline. The state at any given time can be determined by calculating the goodness of fit between the expected probability per state and the observed proportion. Similarly, the difficulty of transitioning from one state to the next can be calculated. In this manner, the total cost of transitioning from the current state to the next can be determined, and, thereby, the optimal state transition sequence (at minimum total cost) can be calculated. The above-described calculation is used for each phrase within a database log file. In particular, the detection of anomalies from burstiness in particular types of log statements can be performed in a two-phase procedure, as described below.

In a first, parameter-free phase, statements in a database log are read and split into time windows. Once the time-windowed fraction of the log file is passed through the parameter-free phase, bursty log statements are identified and passed to a second phase, referred to as the automaton. The second phase finds the start time and the end time of a spike (or burst) and its associated magnitude.

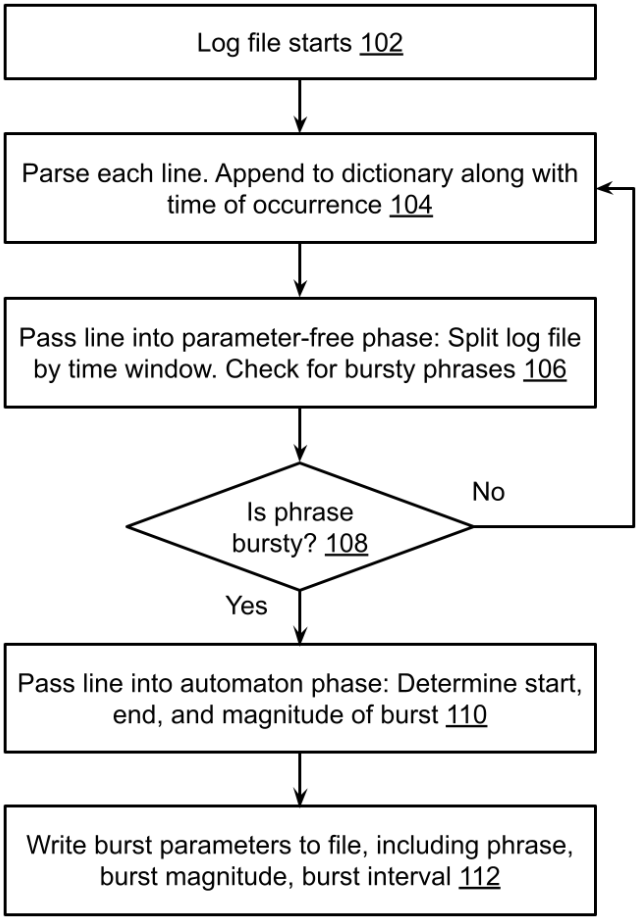

Fig. 1: Anomaly detection in large-scale cloud-native databases

Fig. 1 illustrates anomaly detection in large-scale cloud-native databases. A server builds the log file and starts writing to it (102). A line of the log file is parsed as it is generated and, along with the time of occurrence, is appended to a dictionary (104). Essentially, as the log is parsed, when a log statement not already in the dictionary is encountered, a new key-value pair is added to the dictionary. By doing so, every log statement in the database log is accounted for.

The line is passed into the parameter-free phase in which the log file is split by time window, checking for bursty phrases (106). For example, a time window can be two minutes long, and phrases can be checked for burstiness using four windows of data. With every new window accumulated, the expected probability can be recalculated as follows: if, prior to

accumulating the new window, there exists data for previous windows (numbering, e.g., four), the oldest window is discarded, and the new window takes its place. With the updated expected probability, the expected value and standard deviation are re-computed, thereby re-determining the threshold for comparison.

If a phrase is bursty (108), it is sent to the automaton phase to determine the start, end, and magnitude of the burst (110). The results, which include burst parameters such as phrase, burst magnitude (number of phrases), start time of burst, end time of burst, etc., are written to a file (112). A database administrator can examine the file to rapidly hone into potential problems with the database.

The first phase filters phrases and passes potentially concerning log statements to the second phase. The second phase identifies bursts and provides granularity and time intervals per phrase. The described two-phase approach to identifying bursts of anomalies benefits from the scalability of the first phase and the accuracy of the second phase, resulting in an overall performance improvement. For example, the only lines that are examined using the automaton are the ones that have been identified as bursty from the first phase. Effectively, the two-phase procedure comprises a probe-and-process approach that efficiently obtains the levels and the intervals of the bursts while ensuring that no important bursts are missed. A DBA can examine the ranked list of potentially problematic log statements generated by the tool to investigate anomalies.

Example results

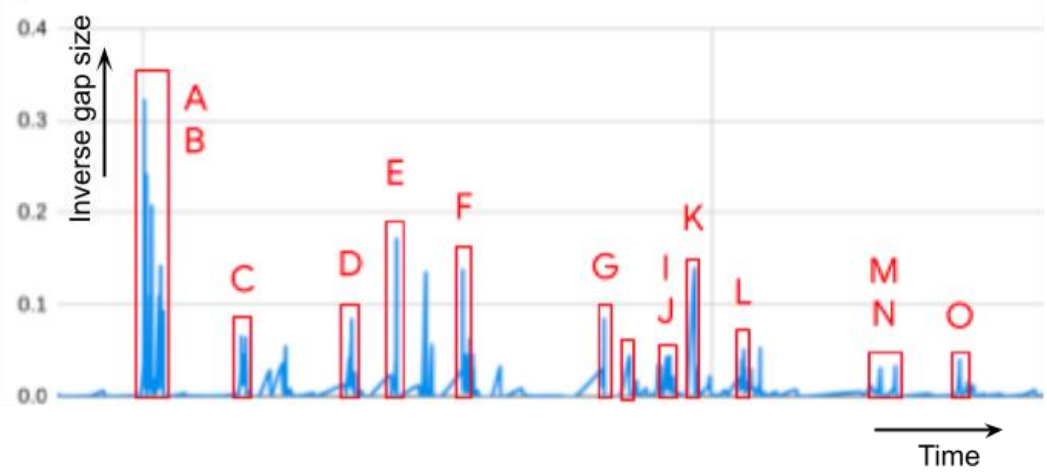

Fig. 2: An example of detecting anomalies in database operations by testing for burstiness of a particular log statement in log files

Fig. 2 illustrates an example of detecting anomalies in database operations by testing for burstiness of a particular log statement in log files. The graph visually depicts the occurrence times of the phrase against the inverse of the time gap between one occurrence and the next. The peaks in the graph represent smaller gaps between the two successive occurrences. The red boxes mark regions that the anomaly detector identified as bursty. The highlighted and labeled peaks are detected even as a burst is taking place, thereby enabling rapid action by the DBA.

Example user (DBA) journey for static (offline) log files

DBAs can run the anomaly detection tool with a database log by providing a path to the log file as an argument to the tool, or the tool can be made native to the database. The DBA can:

- run anomaly detection script with the desired log file;
- run the parameter-free and automaton phases interactively, such that for all or selected frequent log statements, the user can choose to run just the automaton phase;
- review the ranked list of bursty phrases with their time intervals and magnitudes; and
- investigate concerning log statements.

Example user (DBA) journey for streaming log files

For streaming log files, the anomaly detection tool is made native to the database, e.g., run as a function call within the database application. A typical user journey is:

- The DBA starts up their server, with the function call to the tool being included in the server startup file.
- The tool parses the log file that is actively being written to (line by line), running the parameter-free and automaton phases every new window, each window being, e.g., two minutes long.
- A stream of bursts is produced. The intensity, (start, end) times, and other parameters of the bursts are recorded.
- The DBA can investigate concerning bursts, based on burst magnitude or phrases they include.

In this manner, by examining anomaly patterns for burstiness and for the magnitude of burstiness, databases can be proactively tested for likely malfunction and can be operated without interruption. By automating anomaly detection of logs, DBAs can spend their time and expertise acting on issues rather than finding them. The described anomaly detection tool enables processing of log files with substantial (e.g., millions) numbers of lines within minutes, outputting anomalies in a detailed yet concise manner. Rare but significant events that are hard to find can be detected automatically. The techniques can reduce human error and are scalable to cloud-based fleets comprising thousands of databases. Furthermore, the techniques can scale with the logs produced and are not customized to specific anomalies or message types.

CONCLUSION

This disclosure describes techniques to identify anomalous patterns in customer workloads from database logs and to enable timely, corrective action that ensures uninterrupted operation of the database. Examples of anomalies include sudden increases (bursts) in the number of error messages written to a log file. An adaptive behavior norm is defined for each message type. Time instances or periods when the gap between messages of a given type in the database log deviate from the expected behavior norms are detected. A deviation from the behavior norm is a potential indicator of database problems. An anomaly detection tool outputs a ranked list of log statements exhibiting spikes of activity along with their time intervals that a database administrator (DBA) can examine to take corrective action. By automating anomaly detection, the valuable time of DBAs can be spent acting on issues rather than finding them.

REFERENCES

1. Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu. “Parameter-free bursty events detection in text streams.” In *Proceedings of the 31st International Conference on Very Large Databases*, pp. 181-192. 2005.
2. Jon Kleinberg, “Bursty and hierarchical structure in streams.” In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 91-101. 2002.