Demo: Solution Recommender for System Failure Recovery via Log Event Pattern Matching on a Knowledge Graph

Miyuru Dayarathna, Prabhash Akmeemana, Srinath Perera, Malith Jayasinghe
WSO2, Inc.
787, Castro Street
Mountain View, CA, USA 94041
{miyurud,prabhash,srinath,malithj}@wso2.com

ABSTRACT

System anomalies such as network interruptions, operating system halt, disk crash could result in significant financial losses to organizations. In this demonstration we describe a novel log event analysis framework called Solution Recommender which provides a ranked list of solutions to overcome such system errors. The solution recommender gathers log events via a publisher/subscriber mechanism and indexes them inside the WSO2 Data Analytics Server (DAS). Collected information is analyzed using a knowledge graph which conducts log event pattern matching to identify solutions for system failures. We have implemented the proposed approach on WSO2 Log Analyzer for WSO2 API Manager and tested its functionality. In this paper we describe our experience of implementing the log event recommender interface, the first such recommender developed in a log event analyzer system. The insights presented here will assist practitioners with implementing such Log Event analysis solutions for real world scenarios.

CCS CONCEPTS

• **Theory of computation** → **Pattern matching;** • **Applied computing** → **Event-driven architectures;** • **Information systems** → *Data analytics;*

KEYWORDS

Events, Data Visualization, Event Pattern Matching, Log Analysis, Cloud computing, Application Debugging, Systems Administration

ACM Reference format:
Miyuru Dayarathna, Prabhash Akmeemana, Srinath Perera, Malith Jayasinghe. 2017. Demo: Solution Recommender for System Failure Recovery via Log Event Pattern Matching on a Knowledge Graph. In *Proceedings of DEBS '17, Barcelona, Spain, June 19-23, 2017,* 4 pages.
DOI: 10.1145/3093742.3095094

1 INTRODUCTION

Log event stream mining has become a key technology for identifying various cloud system defects. A log is simply a time ordered event sequence. Historically logs have been used for detecting system anomalies (i.e., system defects). In recent times management and mining logs have become a big data problem because of the vast amounts of log data being generated by multiple systems. Log data processing has become an online data processing problem because multiple applications have emerged which uses log events generated by multiple systems to produce real-time alerts. Multiple log analysis solutions have been developed to extract useful information from logs. Some of the famous log analyzers include Splunk, Elasticsearch (i.e., ELK), Microsoft Azure [7], etc. WSO2 Log Analyzer [12] is a generic log analysis framework developed on top of WSO2 Data Analytics Server (DAS) [13].

Currently most of the log analyzers focus on few key areas. First, they index and correlate data from various sources. Second, they allow searching, drilling up/down, pivotting to quickly find information from oceans of data. WSO2 Log Analyzer for WSO2 API Manager is an example for such system [12]. Third, they allow for reporting which can be made available to user as either reports or dashboards. For example, Splunk's log alerting mechanism [11]. These main functionalities are important for identifying the presence of system anomalies. However, none of the current log analyzers provide solution recommendations for system anomalies.

As a solution in this demonstration we present a novel system called *Solution Recommender* which uses log event data produced by severs to identify system errors and provide support for mitigating them. Such solution recommendation system could help reducing the time taken for addressing system maintenance requests made by customers since most of the general issues could be solved through such solution recommendation interface. Our system produces a ranked list of solutions which could be used to mitigate the error scenarios. Currently we implement the proposed solution targeting any logs produced by WSO2 products. However, the same system could be extended to analyse logs from other systems with few modifications. We describe our experience with implementing the Solution Recommender targeting WSO2 API Manager. The WSO2 API Manager is an open source software solution that supports API publishing, lifecycle management, application development, access control, rate limiting, and analytics in single cleanly integrated system.

The log event storage and indexing has been conducted via WSO2 DAS. The main contributions of this paper are as follows,

- *Log event collection, indexing, and visualization* : We developed a log event collection and indexing infrastructure based on DAS.
- *Graph based solution recommendation* : We generate a knowledge graph of solutions which is used for online solution recommendation.
- *Solution recommendation interface* : We develop a solution recommendation interface where users could contribute to the knowledge base via adding solutions as well as use the solution recommender for identifying solutions for system issues.

This paper is organized as follows. First, we provide a brief summary of the related work on log analysis in Section 2. Next, we describe the methodology in Section 3. The system implementation is described in Section 4. Next, we provide the details of the demonstration in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

There have been multiple work done on log event analysis from both industry and academia. Alspaugh *et al.* presented a collection of recommendations to uplift the quality and the quantity of user behavior data collected from interactive data analysis systems [2]. Furthermore, they presented an in-depth study of over 200K log analysis queries from Splunk data analytics platform. They created state machine based descriptions of typical log analysis pipelines. Furthermore, they conducted a cluster analysis of the most common transformation types and Apache Splunk specific constructs [1]. However, their system does not focus on solution recommendation.

Basak *et al.* developed a user-friendly log viewer where they hid unimportant messages from log files [3]. Such unimportant messages do not impact the end user for the purposes such as problem forecasting and troubleshooting. Through use of machine learning techniques they have been able to reduce the size of the log files significantly (30% to 55%) with low error rates (7% to 20%). They also incorporated user feedback to further reduce the error. Similar to them our Solution Recommender utilizes user feedback to improve the quality of its recommendations.

Mahmood *et al.* addressed the issue of storing large amounts of streaming logs with NoSQL. They investigated the degree to which state-of-the-art NoSQL database could achieve high performance persistence and analysing large scale data logs from real world applications using MongoDB [6]. They have compared the pros/cons of the use of such NoSQL database compared to the use of a relational database. They observed that relaxing the consistency does not provide substantial performance enhancement in persisting large scale logs for any system.

Heikkinen *et al.* developed a log analysis system called LOGDIG [5] which is based on a state machine. Different

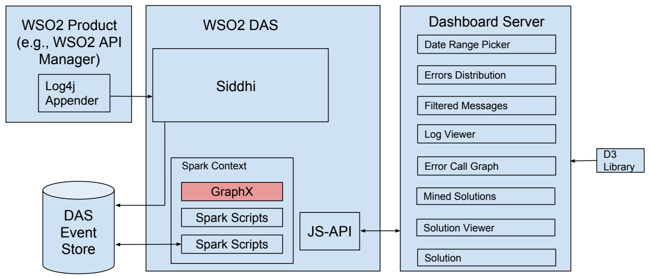


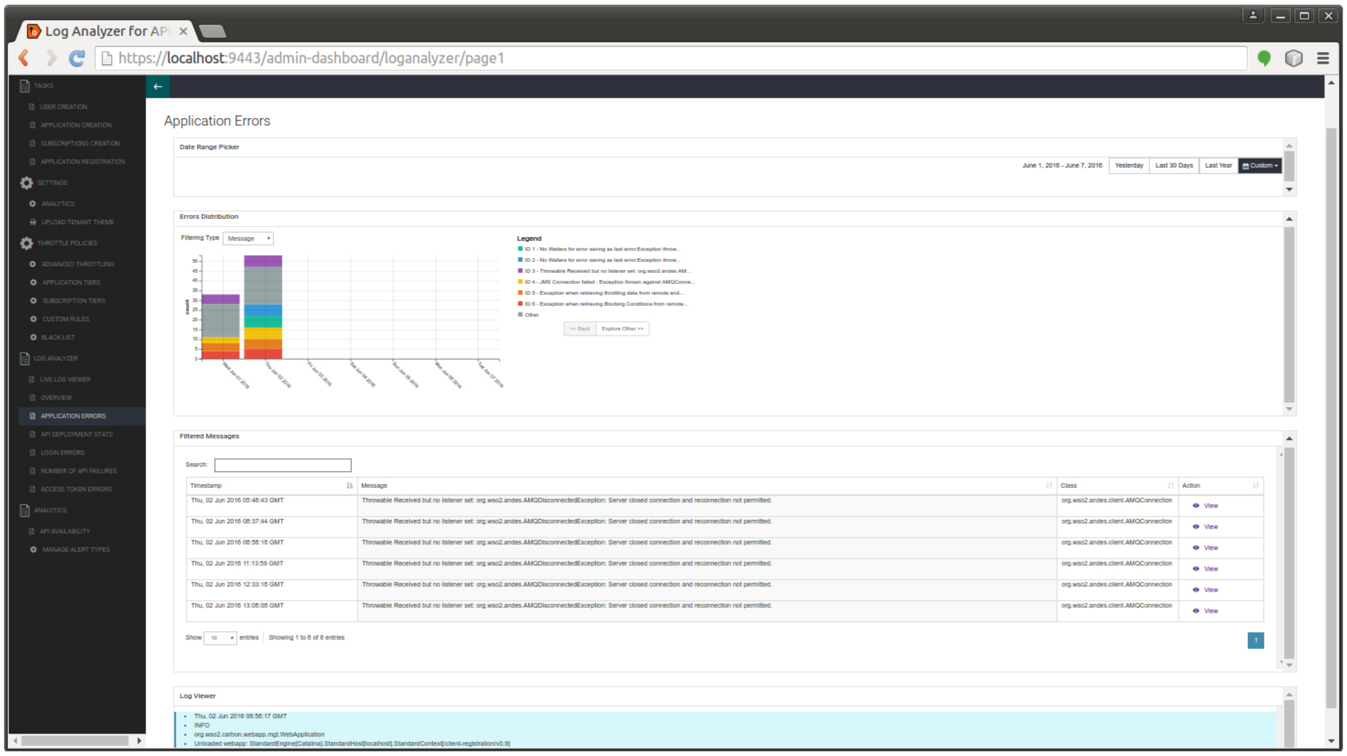
Figure 1: System Architecture

from other systems log events in LOGDIG do not directly trigger transitions but special functions are used for searching the event state. Rabkin *et al.* described a methodology for characterizing and visualizing logs [9]. They presented an abstract graphical representation of console logs called identifier graph and a visualization. However, these works only focused on searching through/visualizing the logs, but not solution recommendation like we do.

GraphJet is a graph-based system for generating content recommendations at Twitter [10]. Similar to our Solution Recommender, it uses a graph structure for content recommendation. A graph database could be used for storing and running the pattern matching queries [4][8]. However, we consider this option as a future work.

3 METHODOLOGY

The methodology we follow in Solution Recommender is as follows. First, we collect log events from the system under surveillance (WSO2 API Manager in this example). The gathered log events are stored and indexed in a NoSQL event store. Second, the gathered information is used to populate multiple sub event tables which are used for presenting information gathered from log analysis at different levels of granularities. For example, we allow the user to input a date range and allow them to observe the appearance of different categories of system anomalies on a timeline. Then we allow them to drill-down further by clicking on one category. Through that mechanism they can easily identify on which dates and times did such anomalies occurred. Next, they can further drill-down and visit the actual log lines where the error (Exception scenario) occur. Through this drill-down mechanism we allow the users to identify system issues quite easily and fast. Third, we allow the users to enter solution recommendations via the same dashboard if the system cannot find a solution for a given anomaly scenario. This is useful when there is no recommendations entered to the system. Finally, during the runtime we construct a knowledge graph (we name it as Solutions Graph) from the historical information gathered from previous recommendations. When user browses through some sequence of log events the system constructs an error/warning event sequence and it uses that sequence to conduct pattern matching on top of the solutions graph.



Application Errors

Legend
- 🟩 ID 1 - No Waiters for error saving as last error:Exception throw...
- 🟦 ID 2 - No Waiters for error saving as last error:Exception throw...
- 🟪 ID 3 - Throwable Received but no listener set: org.wso2.andes.AM...
- 🟨 ID 4 - JMS Connection failed : Exception thrown against AMQConne...
- 🟧 ID 5 - Exception when retrieving throttling data from remote end...
- 🟥 ID 6 - Exception when retrieving Blocking Conditions from remote...
- ⬜ Other

Timestamp	Message	Class	Action
Thu, 02 Jun 2016 05:48:43 GMT	Throwable Received but no listener set: org.wso2.andes.AMQDisconnectedException: Server closed connection and reconnection not permitted.	org.wso2.andes.client.AMQConnection	View
Thu, 02 Jun 2016 08:37:44 GMT	Throwable Received but no listener set: org.wso2.andes.AMQDisconnectedException: Server closed connection and reconnection not permitted.	org.wso2.andes.client.AMQConnection	View
Thu, 02 Jun 2016 08:58:18 GMT	Throwable Received but no listener set: org.wso2.andes.AMQDisconnectedException: Server closed connection and reconnection not permitted.	org.wso2.andes.client.AMQConnection	View
Thu, 02 Jun 2016 11:13:59 GMT	Throwable Received but no listener set: org.wso2.andes.AMQDisconnectedException: Server closed connection and reconnection not permitted.	org.wso2.andes.client.AMQConnection	View
Thu, 02 Jun 2016 12:33:18 GMT	Throwable Received but no listener set: org.wso2.andes.AMQDisconnectedException: Server closed connection and reconnection not permitted.	org.wso2.andes.client.AMQConnection	View
Thu, 02 Jun 2016 13:08:08 GMT	Throwable Received but no listener set: org.wso2.andes.AMQDisconnectedException: Server closed connection and reconnection not permitted.	org.wso2.andes.client.AMQConnection	View

Show 10 entries | Showing 1 to 6 of 6 entries

Log Viewer

```
• Thu, 02 Jun 2016 08:58:17 GMT
• INFO
• org.wso2.carbon.webapp.mgt.WebApplication
• Unloaded webapp: StandardEngine[Catalina].StandardHost[localhost].StandardContext[/client-registration/v0.9]
```

Figure 2: The drill-down structure of solution recommender.

Solution recommendations are entered in the user interface only for the exception scenario currently displayed in the user interface. However, the solution recommender could show not only solution recommendations made on the current exception scenario but also from the matching patterns observed from the previous exception scenarios. We assign a confidence score for the solutions recommended by our system for any exception scenario. Solutions which have been recommended for a particular exception scenario carries 100% confidence score because they have been made by the users particularly targeting such scenarios. However, other solutions which were mined from pattern matching on the solutions graph also gets ranked in the UI with 100% if their patterns exactly matches with the exception pattern. If the current exception pattern matches (say the length of the current exception pattern is n vertices) with the solutions graph partially (say m vertices), we assign $(m * 100/n)$ percentage as the confidence score.

4 SOLUTION RECOMMENDER ARCHITECTURE

The system architecture of the Solution Recommender is shown in Figure 1. Any WSO2 middleware product can be used as the log event source. However, the initial implementation of the Solution Recommender described in this paper has been conducted targeting WSO2 API-Manager.

The log events published by the Log Publisher modules are received by the Siddhi event processing library which then get stored in DAS Event Store. The stored data are processed by a set of Spark scripts which constructs the information to be presented in multiple dashboard server gadgets.

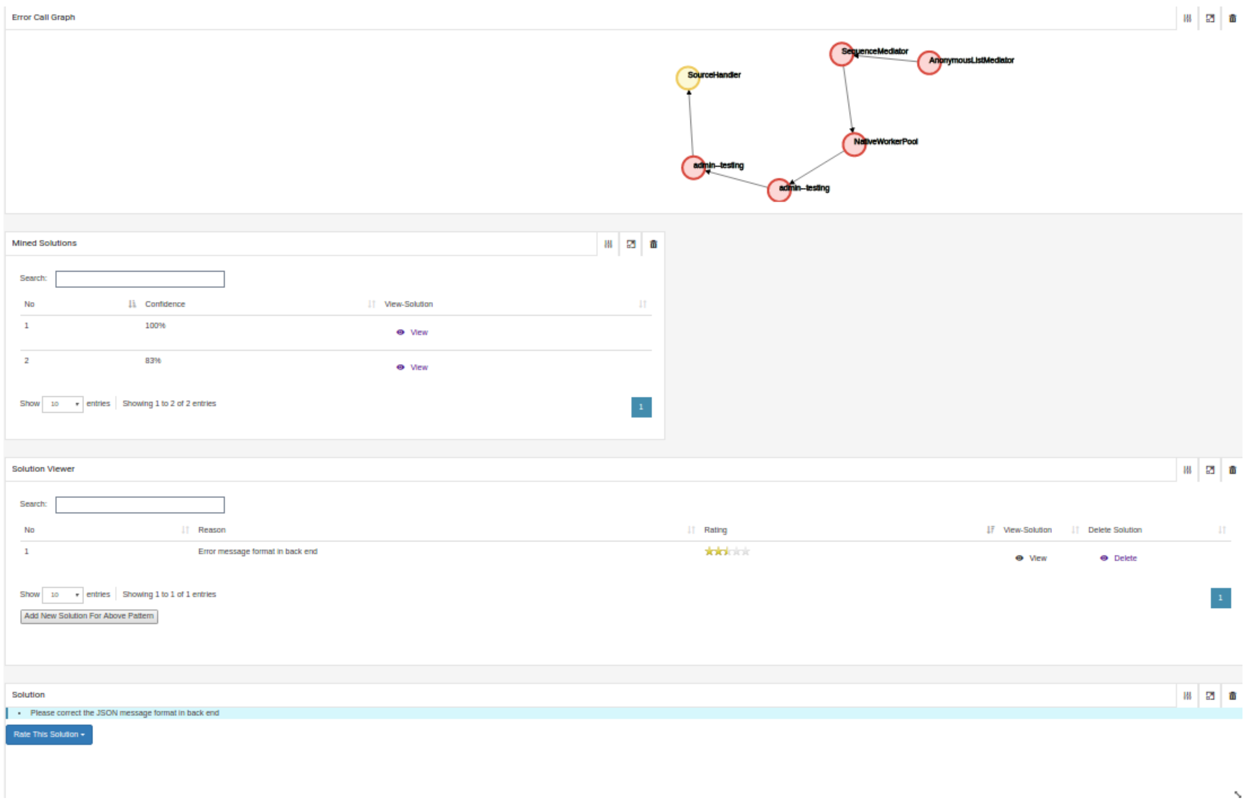
The Dashboard Server consists of four different UI components (i.e., gadgets) which display the processed information. The visualizations made on these gadgets have been rendered through a visualization library called D3.

5 DEMONSTRATION OVERVIEW

Log event indexing. Solution Recommender supports collection and indexing process of log events from WSO2 servers. We first demonstrate how log events get published from the WSO2 servers by simulating a scenario of message processor error occurrence in WSO2 API Manager. The indexed log events will be shown to the audience via DAS data explorer.

Drilling-down a specific exception scenario. We next demonstrate the drilling-down functionality provided by Solution Recommender for identifying a specific exception scenario which has occurred in the configured WSO2 server. We will select the appropriate date range from data range picker which includes the date and time of the message processing error we simulated in the previous step. Next, we will demonstrate how the specific exception scenario will be pointed out using the Errors Distribution and Filtered Messages gadgets (as shown in Figure 2).

Visualizing recommended solutions. Finally, we demonstrate how the Error Call Graph is used for



No	Confidence	View-Solution
1	100%	View
2	83%	View

Showing 1 to 2 of 2 entries

No	Reason	Rating	View-Solution	Delete Solution
1	Error message format in back end	★★☆☆☆	View	Delete

Showing 1 to 1 of 1 entries

- Please correct the JSON message format in back end

Figure 3: Presentation of solution recommendations.

visualizing the exception scenario. We will also describe how the mined solutions be viewed (via Mined Solutions) and how users could update the solutions via the solution viewer (as shown in Figure 3).

6 CONCLUSION

This work demonstrated Solution Recommender, a system which uses the log event data produced by servers to identify system errors and provide support for fixing them. Solution Recommender advances over the state-of-the art Log Analyzers via providing a graph based solution recommendation engine for fixing system issues. It also provides a drill-down based visualization interface for solution recommendation. We demonstrated the Solution Recommender's functionality via taking a real world scenario of identifying solutions for WSO2 API Manager's message processing error scenario.

REFERENCES

[1] S. Alspaugh, Beidi Chen, Jessica Lin, Archana Ganapathi, Marti A. Hearst, and Randy Katz. 2014. Analyzing Log Analysis: An Empirical Study of User Log Mining. In *Proceedings of the 28th USENIX Conference on Large Installation System Administration (LISA'14)*. USENIX Association, Berkeley, CA, USA, 53–68. http://dl.acm.org/citation.cfm?id=2717491.2717495

[2] Sara Alspaugh, Archana Ganapathi, Marti A Hearst, and Randy Katz. 2014. Better logging to improve interactive data analysis tools. In *KDD Workshop on Interactive Data Exploration and Analytics (IDEA)*.

[3] Jayanta Basak and P. C. Nagesh. 2016. A User-Friendly Log Viewer for Storage Systems. *Trans. Storage* 12, 3, Article 17 (May 2016), 18 pages. DOI:http://dx.doi.org/10.1145/2846101

[4] M. Dayarathna and T. Suzumura. 2014. Towards Scalable Distributed Graph Database Engine for Hybrid Clouds. In *2014 5th International Workshop on Data-Intensive Computing in the Clouds*. 1–8. DOI:http://dx.doi.org/10.1109/DataCloud.2014.9

[5] Esa Heikkinen and Timo Hämäläinen. 2015. LOGDIG log file analyzer for mining expected behavior from log files. In *SPLST*.

[6] Khalid Mahmood, Tore Risch, and Minpeng Zhu. 2014. Utilizing a NoSQL Data Store for Scalable Log Analysis. In *Proceedings of the 19th International Database Engineering &#38; Applications Symposium (IDEAS '15)*. ACM, New York, NY, USA, 49–55. DOI:http://dx.doi.org/10.1145/2790755.2790772

[7] Microsoft 2017. *Log Analytics Documentation*. Microsoft. https://azure.microsoft.com/en-us/services/log-analytics/.

[8] Neo4j. 2017. Neo4j - The World's Leading Graph Database. URL: http://www.neo4j.org/. (May 2017).

[9] Ariel Rabkin, Wei Xu, Avani Wildani, Armando Fox, David Patterson, and Randy Katz. 2010. A Graphical Representation for Identifier Structure in Logs. In *Proceedings of the 2010 Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML'10)*. USENIX Association, Berkeley, CA, USA, 3–3. http://dl.acm.org/citation.cfm?id=1928991.1928996

[10] Aneesh Sharma, Jerry Jiang, Praveen Bommannavar, Brian Larson, and Jimmy Lin. 2016. GraphJet: Real-time Content Recommendations at Twitter. *Proc. VLDB Endow.* 9, 13 (Sept. 2016), 1281–1292. DOI:http://dx.doi.org/10.14778/3007263.3007267

[11] Splunk Inc. 2017. *Getting started with alerts - Splunk Documentation*. Splunk Inc. http://docs.splunk.com/Documentation/Splunk/6.5.2/Alert/Aboutalerts.

[12] WSO2 Inc. 2017. *Analyzing Logs with the Log Analyzer*. WSO2 Inc. https://docs.wso2.com/display/AM200/Analyzing+Logs+with+the+Log+Analyzer.

[13] WSO2 Inc. 2017. Smart Analytics. URL: http://wso2.com/smart-analytics/. (May 2017).