Technical Disclosure Commons

Defensive Publications Series

January 2020

USING SEQUENTIAL ANALYSIS AND SEARCH SPACE MAPPING MACHINE LEARNING TECHNIQUES FOR TROUBLESHOOTING A SOFTWARE DEFINED NETWORK OR A NETWORK FUNCTION VIRTUALIZATION BASED NETWORK

Sarthak Sharma

Lovepreet Singh

Recommended Citation

Sharma, Sarthak and Singh, Lovepreet, "USING SEQUENTIAL ANALYSIS AND SEARCH SPACE MAPPING MACHINE LEARNING TECHNIQUES FOR TROUBLESHOOTING A SOFTWARE DEFINED NETWORK OR A NETWORK FUNCTION VIRTUALIZATION BASED NETWORK", Technical Disclosure Commons, (January 21, 2020)
https://www.tdcommons.org/dpubs_series/2879

USING SEQUENTIAL ANALYSIS AND SEARCH SPACE MAPPING MACHINE LEARNING TECHNIQUES FOR TROUBLESHOOTING A SOFTWARE DEFINED NETWORK OR A NETWORK FUNCTION VIRTUALIZATION BASED NETWORK

AUTHORS:

Sarthak Sharma
Lovepreet Singh

ABSTRACT

Proposed herein is a technique to utilize a sequential analysis to map the probability of a sequence of commands executed via Command Line Interface (CLI) in order to determine a root cause of a problem in a network. This technique may provide different paths that can be analyzed in order to debug an issue. The sequential analysis may be performed in tandem with an interpreter model that may read text generated by each command, verify whether there is any problem in a sub-segment, and, upon determining a problem, may enable automatic remediation to the problem.

DETAILED DESCRIPTION

Presently, much human effort may be wasted on troubleshooting network-related issues. Often, the people that spend time trying to rectify network-related issues are senior technical professionals, who could be spending their time on other demanding tasks, such as exploring newer technologies to use in a product, designing new systems, re-architecting older systems to modern design patterns, etc. rather than troubleshooting network-related issues. In addition, troubleshooting network-related issues typically involves time-consuming communications between customer and a network provider for resolving such issues.

Many of the tasks involved in troubleshooting network-related issues can be automated by leveraging the human effort spent and knowledge obtained in prior troubleshooting cases. Machine learning offers an approach to leverage an existing knowledge base by translating such troubleshooting knowledge to models that may make decisions regarding new issues such that skilled professionals may only be tasked with network troubleshooting as may be needed.

Network architectures such as Software Defined Networking (SDN) architectures and/or Network Function Virtualization (NFV) based architectures may provide a global view of a network/system that may be used to debug failures. Having a global view of a network/system may provide an advantage of being able to correlate information gathered from individual nodes, which may provide salient debugging capabilities.

This proposal involves a technique in which human expertise gained over time may be encoded into a sequence of commands whose outputs could be leveraged to determine a root cause of a problem in a network/system. This technique may provide different paths that can be analyzed in order to debug an issue.

For example, if more Linux® HugePages than default are allocated for a system, then a next set of commands to execute may be "show system memory" and "virsh dumpxml" in order to look deeper into the state of a system. Based on an output of these commands, further commands may be executed to pinpoint an exact issue that may be affecting the system.

The sequential analysis may be performed in tandem with an interpreter model that may read text generated by each command and may verify whether there is any problem in a sub-segment. A graph data structure may be utilized to implement the technique in which a search algorithm may visit different paths based on encoded steps. In case a fault is not detected on a particular path, the search algorithm can return to a parent node at which it branched off in order to explore other paths.

Implementing such a search algorithm for global debugging may include creating a graph and processing text associated with commands input via a CLI. Graph traversal can be performed in order to identify a root cause of a given problem. In some implementations, automatic remediation of a problem may be provided.

For creating a graph, a tree may be created manually by a user or network administrator based on an existing knowledge base that identifies system debugging operations, processes, etc. that may be performed for debugging issues of a network/system. The knowledge base may be based on human technical knowledge of which commands may be executed after a certain condition in a set of command outputs has been observed. Commands executed on the CLI will produce an output text that can be parsed into data structures for easier programmability.

Graph traversal can be performed in order to identify a root cause of a problem. The graph may consist of nodes for each concept related to observed issues. Figure 1, below, illustrates an example graph that may be utilized for an example search algorithm.

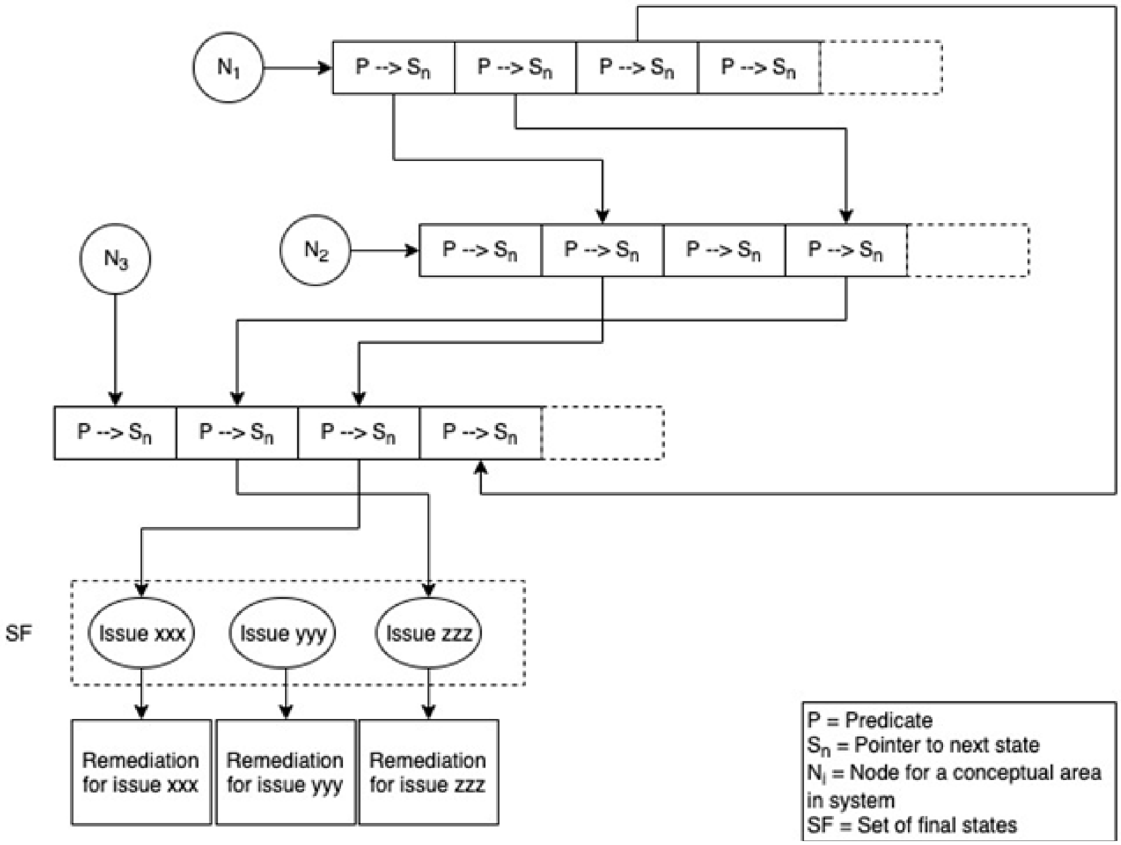
N1
P --> Sn
N2
N3
SF
Issue xxx
Issue yyy
Issue zzz
Remediation for issue xxx
Remediation for issue yyy
Remediation for issue zzz
P = Predicate
Sn = Pointer to next state
Ni = Node for a conceptual area in system
SF = Set of final states

Figure 1

For the graph illustrated in Figure 1, let (Ni) denote a node related to a concept (i) in which each node (Ni) will have a list (Li) of known issues related to concept (i). Each element of a list will be a set of pre-conditions or predicates (P) for an issue and can indicate a next step (Sn) to which to proceed if a precondition matches. In one example, P may be a condition on outputs from a set of commands and Sn may be a similar list element in another node. A final state belongs to a set of final states (SF) in which each state can denote an exact issue in the network/system.

During operation, the search algorithm can be initiated from a concept node based on either human input or processing of logs. The search algorithm can iterate over the list of elements and, if any set of pre-conditions matches, the search algorithm can proceed to the next state. If a final state is reached on a particular path, an exact issue is identified.

Otherwise, if a final state is not reached, the search algorithm can backtrack to a nearest ancestor having unexplored elements to attempt to match another element in the list. These steps can be executed until a final state is reached or the whole graph is traversed. If a final state is not reached, human input may be provided.

Thus, detecting an exact issue involves constructing a graph with symptoms of faults, as noted above. A path for a known issue will always end in a final state from a set of final states and each final state will have a sequence of steps that correspond to the final state. The steps may be followed to facilitate automatic remediation of the issue represented by the final state. Upon performing a particular remediation, the same path of nodes can be followed again to determine whether or not the problem is resolved. When a problem is resolved, the system may notify a customer regarding the resolution.

In addition to using machine learning techniques to identify a root cause of an issue in a network/system, machine learning may also be used to detect probable failures in a network/system. For example, critical statistics of a network/system (e.g., pnic statistics, available memory, CPU usage, etc.) may be collected that may provide insight into whether there may be an anomaly in the network/system. Once a certain discrepancy or anomaly is observed, a constructed graph can be traversed using techniques as discussed above in order to identify an issue causing the anomaly.

Unsupervised learning may be utilized to detect anomalies in collected statistics. A model can be trained over statistics observed in a controlled environment to learn normal behavior of a network/system. The trained model may then be used to detect an anomaly in network/system behavior.

In one implementation Natural Language Processing (NLP) based log monitoring may be performed (assuming logs are in natural language). In such an implementation, system logs can be actively monitored for error or warning level logs. Detecting such error or warning logs can trigger a process to analyze the log messages. Based on a trained model, each such log message may be mapped to a concept from a pre-defined set of concepts. An assigned concept can then be used to select a start node in a constructed graph (e.g., as shown in Figure 1) and the graph can be searched for an exact issue based on the log message using the graph traversal techniques discussed herein.

In summary, this proposal provides various machine learning techniques involving a search algorithm that may be utilized to search a constructed graph to determine a root cause of a problem in a network/system and/or to detect probable failures in a network/system.