Name: Prashanth Mallyampatti
Unity ID: pmallya
Student ID: 200250501

CSC724- Advanced Distributed Systems
Paper Review

PerfScope: Practical Online Server Performance Bug Inference in Production Cloud Computing Infrastructures

Summary:

The paper addresses one of the major problems in distributed systems- bug detection, inference and categorization. Diagnosis of performance bugs that occurred inside the cloud infrastructure is very difficult to reproduce outside and hence many offline tools cannot be applied. Unexpected bugs accounts for the major bug count as do not generate any error message and are unknown by the developer. Performance anomalies caused by software hang, slow-down, hardware faults, kernel bugs are some of the commonly known ones.

PerfScope is limited to software bugs such as software-hang, slow-down diagnosis. PerfScope collects recent system call traces of window size 5 minutes. This is done offline for each function call and frequent call episodes are generated using Finite State Machine (FSM) based matching algorithm. PerfScope eliminates the functions such as system libraries to get a subset of user defined functions. The online bug inference scheme extracts the execution units by detecting a change in the thread or CPU ID indicating a context change. To identify the abnormal units, hierarchical clustering is applied and system call appearance vector is extracted, Euclidean distances for abnormality detection. These abnormal units are then mapped to a bug-related functions and rank them based on abnormality degree. Also, PerfScope is capable of inferring the system call path, which can be further be used to understand the bug source and propagation.

PerfScope has been tested on various open source server systems. PerfScope has identified real performance bugs and handled correctly. An external trigger was used to run PerfScope when a performance anomaly is detected. PerfScope then runs the most recent 1.5 minutes of system call trace and apply the above proposed system design for bug inference. However, the entire scope of the anomaly cannot be generated but provides a small subset to look at. Some of the top ranked bug-related functions identified by PerfScope are getState(), Reader.performIO(), etc; which when contributed via multiple functions PerfScope was able to identify them. The paper also discusses Bug Inference case studies on Cassandra, HDFS and Lighttpd-1 systems which gives a clear idea on how PerfScope has inferred the bug. In detail overheads are discussed for various distributed systems which is noteworthy.

Strong Points:

1) PerfScope is an online bug inference tool, which finds many advantages over offline tools.
2) It doesn't require any production runs, application source code and system instrumentation thus can be applied to any environments without any major restrictions.
3) Overheads are comparatively low, due to analysis of system calls rather than raw system metrics.
4) System call episodes better indicate the nature of the bug, as they are derived from the raw system metrics and thus are more meaningful.
5) Since signature extraction is done offline, it doesn't affect the online bug inference time.

Weak Points:

1) Without access of source code and kernel stack, deeper problems might not be inferred.
2) PerfScope assumes stability in function signatures with changing workloads, inputs, which is highly impossible in real time cloud infrastructures.
3) The "time gap" to split long execution threads is very vague, and long time gap might not always indicate function switch, such as IO, slow disk reads. Also the experimental results of this proposed "time gap" has not been discussed.
4) PerfScope evaluation presented in the paper is limited to single node server, thus the behavior of it in multi node cluster is into question.

5) The paper says they had occurred no case of two functions having same frequent episode set, but it has a good chance of occurring in large scale multi node clusters.