Paper: The Mystery Machine: End-to-end Performance Analysis of Large-scale Internet Services
Name: Xueqing Tian

The authors of this paper develop performance analysis tools for measuring and uncovering performance insights about complex, heterogeneous distributed systems. The paper can be divided into these sub-topics: constructing a model of request execution from pre-existing component logs, generating a large number of potential hypotheses, rejecting hypotheses contradicted by the empirical observations, validating potential performance improvements, leveraging the variation in component behavior and measuring the impact of optimizing individual components. They apply these tools to the Facebook pipeline to validate their methodology with a detailed study of the factors that affect the end-to-end latency of such requests and suggest a scheduling optimization for improving Facebook request latency.

The strength of this paper comes in the following parts. First, the paper provides two detailed case studies of performance optimization based on results from The Mystery Machine. The observation of wide variance in the balance across individual requests leading to the demonstration that end-to-end latency would be improved by having servers produce elements of the response as needed, which would improve the end-to-end latency of requests with no slack while not substantially degrading the latency of requests that currently have considerable slack. Second, the paper combines the advantages of relying on only a minimum common content for component log messages with the benefits of interaction with other components, leading to the unification of output from diverse component logs into a unified tracing system called UberTrace. However, there are some weaknesses which should be illustrated further. First of all, it is said in figure 6 that "JavaScript execution is a factor in performance measurement because there is a global barrier in the browser before the JavaScript engine begins running the executable components of the page". But complexity would be added when we consider the initial request processing before the server begins to transmit any data as a critical process, thus the client rarely accounts for more than half of the critical path. The authors did not discuss this this high variance situation at length in the section of critical path, which should be a valuable yard stick. Secondly, in the section of requirement for log messages in UberTrace, the author mentions that the pair of event name and task name consists a unique request identifier, which means each event-task tuple should be unique. However, this assumption is not valid for all execution environments, it holds only at Facebook given how requests are processed and other similar Internet service pipelines. There is no guarantee for no cycles that would cause a tuple to appear multiple times.

As seen in this paper, the technique of using logs for analysis has been applied to error diagnosis for a long time. Output-Deterministic Replay (ODR) is a software which only replay system that reliably reproduces failures and provides low-overhead multiprocessor recording. The key observation behind ODR is that a high-fidelity replay execution, though sufficient, is not necessary for replay-debugging. Instead, it suffices to produce any execution that exhibits the same output, even if that execution differs from the original. This observation permits ODR to relax its fidelity guarantees and, in so doing, enables it to all-together avoid the problem of reproducing and hence recording datar-ace outcomes. To infer data-race outcomes, ODR uses a technique called Deterministic Run Inference which leverages the original output to search the space of executions for one that exhibits the same outputs as the original.