usenix ;login:
FEBRUARY 2013 VOL. 38, NO. 1
ELECTRONIC SUPPLEMENT

Conference Reports

2012 Workshop on Managing Systems Automatically and Dynamically (MAD '12)

Hollywood, CA
October 7, 2012

Keynote

Summarized by Saad Alaboodi (ssalaboo@uwaterloo.ca)

To Err Is Human, to Log Divine: Expediting Production Failure Diagnosis with Better Logging

Ding Yuan, University of California, San Diego; University of Illinois at Urbana-Champaign; University of Toronto

Ding Yuan began his keynote speech by posing a set of questions important to developing practical solutions to improve the diagnosability and reliability of large systems. For example, how much do log messages really help in debugging? Are they good enough? What are the opportunities for improving log qualities? Can we automatically improve log messages? To further augment his questions, Yuan shared his experiences in the area with some statistics about software failures analyzed from error logs on common systems, and their direct relationship with software design. This was accompanied with a survey about several diagnosis tools developed to accommodate efficient error log automation and analysis.

Yuan then highlighted the common problem of debugging software failures in production environments. In his view, the problem is due to privacy issues of applications data, difficulty of failures reproduction, and difficulty of recreating an environment because it is either prohibited or expensive. Log diagnosis is a common measure used to address such challenges. Even though 99% of users plan to or actually do aggregate log messages, the frustrating reality of impractical error logs continues to exist. To demonstrate a typical scenario, Yuan showed a snapshot taken from an Apache error log about a simple error—inability to read a particular file. The example doesn't simply lay down the exact problem; it is considered an "afterthought" of the actual problem. Such a scenario leads developers to ask for more information (e.g., DNS messages, system configuration, etc.). Thus, the quality of such error logs remain low when it comes to their use.

To improve the quality of error logging and diagnosis for a particular system, Yuan put forth three important steps: (1) understand error log messages, (2) enhance existing error logs, and (3) insert new quality error logs. Several tools are built in this regard: LogEnhancer and SherLog for error logging enhancement, and ErrLog for log insertion. The remaining part of the talk, however, focused on the first two steps.

Yuan stressed that to understand error logging, and more importantly to identify log messages that are not an afterthought, the study of developers' patches modifying software is essential. Out of 9067 logs surveyed, Yuan and his team found that modifications to log message could be broken down into four categories: 26% of modifications on verbosity, 27% on variables, 45% on text, and 2% on location. Each of these categories was further classified into subcategories. For example, verbosity can be panic/fatal messages, errors/warnings, etc.; the team also found that the dependency between verbosity and variables directly affects the availability and reliability of error logs. Usually, the modifications to log text can be fixed easily. Furthermore, developers rarely move the location of logging code. To enhance consistency of verbosity in particular, a simple tool called "CP-Miner" was developed. This tool basically detects similar code snippets.

Yuan then briefly described the LogEnhancer tool. The idea of this tool is to augment existing error logs with useful information whereby diagnosis can be enhanced. This tool basically collects causally related variable values, which can be a complex task. Yuan showed a snapshot of variables in typical program code, demonstrating the main idea of this tool. To record variable values during runtime and avoid fragmentation, the check value conditions written by code developers are used. Another technique is to reconstruct the values from other expressions in the code. To avoid overhead, only a few relevant parameters are logged, which amounted to less than 1% in software evaluated by Yuan and his team. Yuan also emphasized that, because parameters are collected from error messages rather than data, this technique doesn't raise privacy concerns. Yuan mentioned that overall evaluation of such automated tools matched 90–95% of manual logging performed in the past 5–10 years on the selected code.

In terms of future work, Yuan demonstrated the importance of quality error logs in current systems and presented an example using aviation and pilot systems. Complex systems involve complex interactions in the code, where correlating log events become more challenging. Therefore, the focus will be on making logs natural and effective so system users can understand actual system behavior.

Jon Stearley asked whether any values were captured from RAM. Yuan said that values were taken from error messages in the code. The reconstruction of values raised several questions. For example, when Yuan was asked about where to stop when recreating values back from their previous variables, he replied that no limit is usually set to the analysis; however, only related values are recorded, and redundancy is removed. Also, the tool stops at clear code boundaries. Greg Bronevetsky raised the point that some function pointers

cannot be recreated. Yuan commented that values closer to the error message are more valuable to the error anyway. Wei Xu asked about the case of multithreading, and how it could be captured. Yuan acknowledged that it is hard to check for such scenarios; thus, it was not included as it could be misleading where the number of values would be exponential. Finally, Saad Alaboodi asked about whether cyber attacks targeting error logs, such as injected error logs, were detected or considered. Yuan replied that although security attacks were not considered, some tools might be used to check for consistency of values based on the original code, which might address some general security issues.

Recommendation Systems

Summarized by Devesh Tiwari (devesh.dtiwari@ncsu.edu)

Towards a Data Analysis Recommendation System

Sara Alspaugh, University of California, Berkeley; Archana Ganapathi, Splunk, Inc.

Sara presented her experience with collecting and analyzing large data sets at Splunk. The amount of data generated by such companies is growing rapidly, but analyzing these data sets in a meaningful way is very challenging for three main reasons: (1) it often requires too much effort, in many cases manual effort; (2) it requires the knowledge of the domain the data is coming from; and (3) it is easy to overlook interesting patterns. Sara talked about her experience in meeting some of these challenges and how to move toward automated data analysis and a recommendation tool.

Sara specifically focused on data collected at Splunk, where data sets are queried and indexed using a MapReduce-like architecture. Data sets also come from Splunk customers such as Oracle, US Airways, etc. At Splunk, a pipeline of queries is performed to analyze data, and intermediate results are stored in a matrix-like structure suitable for performing pipe queries. One interesting insight from analyzing these data and queries is that customers are usually querying the most recent data; up to 80% of queries look into the data generated within the past six hours.

This study also discovered that the length of searches are mostly with 20 terms; however, Sara could not provide an intuitive reason behind why ad hoc searches were longer than scheduled searches. A surprising finding was that 30% of the people were searching with just the "search" function, and this is surprising because this is equal to doing **grep** with such an expensive service. Finally, she introduced the ongoing work ART, an Analysis Recommendation Tool, where ART will be guided by user feedback to find patterns, and find correlation across different data sets instead of querying each data set in isolation.

Sara answered the questions about how to do this analysis maintaining anonymity, without looking at the actual fields but instead looking at the hashes. Another suggestion was to apply decision trees for finding correlation among data sets. Sara replied that this was in-progress using similar techniques such as PCA and other machine learning algorithms. Another important clarification was that the time to process queries was much smaller than the timeline shown on the result graphs, making the observations valid.

Mojave: A Recommendation System for Software Upgrades

Rekha Bachwani, Rutgers University; Olivier Crameri, EPFL; Ricardo Bianchini, Rutgers University; Willy Zwaenepoel, EPFL

Rekha Bachwani proposed a solution for preventing software upgrade failure. As applying patches is becoming more and more common, software upgrades (patches) may and do themselves fail (there's a 5–10% failure rate). Worse, these upgrades themselves become the root cause of future software bugs. One current technique is to delay the upgrade until it has "matured," which is inefficient in many cases.

Rekha presented a solution to this problem with Mojave, a system in which users (who apply software upgrades) and developers (who write patches) collaborate to reduce the failure of software upgrades in different environments. Mojave provides accurate recommendations to the user whether to apply a patch or not, what is the likelihood of failure, and so on. The system makes an observation that two systems that fail on patches may have the same environment settings before applying the patches; hence, this can be "learned" to avoid future failures. Therefore, the developer learns the system settings and outcome of applying patches from willing users/early adopters and uses a machine learning-based regression model to predict the likelihood of failure if this patch is applied by a new user. The tool was developed and tested under an academic setting (approx. 80 machines) and shows promising results (with 96–100% prediction accuracy, preventing failures for most new users).

During Q&A, Rekha pointed out that their machine-learning algorithms account for higher order dependencies for predicting failures. Another interesting question was why would users collaborate? Rekha replied that having an incentive system for users to provide full information about the system on which an upgrade failed will help everyone because developers cannot simulate all the possible configurations (even using techniques such as symbolic execution). Rekha also noted that in the future such systems can be automated to recommend settings to users so that the upgrade will not fail. She also mentioned that the threshold knob can be tuned to achieve more pessimistic or optimistic recommendations. Finally, she pointed out that this is similar to cooperative bug isolation approach, with two main differences: no overhead during production runs and short periods of data collection.

Managing the Cloud

Summarized by Saad Alaboodi (Ssalaboo@uwaterloo.ca)

Vayu: Learning to Control the Cloud

Ira Cohen, Ohad Assulin, Eli Mordechai, Yaniv Sayers, and Ruth Bernstein, HP Software

Ira Cohen presented Vayu, a system intended for managing cloud applications from performance, availability, and capacity standpoints. Cohen started the presentation with a video that demonstrated the essential, challenging issues of cost and performance when moving into the cloud. Ultimately, this product aims at achieving an optimization goal with respect to availability and performance, cost, and resources. To achieve this, Cohen said, we need to understand the application behavior under load, which is the premise on which this product relies. Vayu learns cloud-based application behavior and analyzes the tradeoff between those optimization goals mentioned earlier in accordance with customer needs.

As a simple case study, Cohen analyzed the application load of a cloud-using online flower retail store during the month of February. This month has three distinct periods of application usage loads: regular days, weekends, and Valentine's Day. Vayu compared different cloud providers by virtually running their capabilities based on different demand scenarios serving the flower store loads. A virtualized sandbox is used for such runs. Upon completion, the business owner can then perform cloud-sizing analysis based on summarized comparisons with respect to cost, performance, and resources. These generated dashboards allow flexible manipulation of the results, whereby the user can filter results per user, transaction, day, and so forth. Additionally, the user can change some of the parameters to see their effects on calculated results. Finally, Vayu can implement the desired action rules chosen by the user.

In terms of the underlying technology, Cohen explained that Vayu consists of four major components: demand feeding into application monitors, performance problem detection and characterization, action learning, and recommendation. These components iterate to build the Vayu knowledge base. Cohen provided a top-level description of some of these components and mentioned that the performance detection and characterization component addresses both normal and anomalous behavior of the application. For normal behavior, the system detects seasons, which are important to make it adaptive. This is implemented using a heuristic-based algorithm. For anomalous behavior, the employed technique keeps tracks of all metrics exceeding their respective baselines, groups them, then computes the corresponding probability of significance. For the action-learning component, the K-nearest neighbor algorithm is used to address the classification problem.

Greg Bronevetsky asked about the assumptions used in identifying normal behavior. Cohen simply answered that no particular assumptions were made except relying on predefined metrics of interest, for example, the speed of deviation from an expected behavior. Marc Chiarini asked about the capability of analyzing multiple, more complicated seasons. Cohen acknowledged the capability of Vayu handling multiple seasons without mentioning any specifics. Finally, as a sweet ending, Cohen played a short birthday clip made for him, and shared some chocolate with the audience for this occasion!

A Framework for Thermal and Performance Management

Davide Basilio Bartolini, Politecnico di Milano; Filippo Sironi, Massachusetts Institute of Technology; Martina Maggio, Lund University; Riccardo Cattaneo and Donatella Sciuto, Politecnico di Milano; Marco Domenico Santambrogio, Politecnico di Milano and Massachusetts Institute of Technology

Davide Bartolini started the presentation with a brief introduction of power and its importance to key operational metrics in server farms today, particularly the power consumed by processors, one of the most power-hungry components in computing systems. These metrics include total cost ownership (TCO); reliability and efficiency, e.g., mean time to failure (MTTF); and power leakage. Bartolini then moved to highlight the proposed framework, Dynamic Performance and Temperature Manager (DPTM), for combined thermal and performance management. The framework aims to provide self-adaptive, cost-effective heat dissipation solutions that fit chip multi-processors (CMPs). The framework basically harnesses idle-cycle injection to control processor absorbed power and thus temperature. Being adaptive, this framework avoids the problems of impaired SLAs and QoS in current methods.

Bartolini explained how their proposal extends the Dimetrodon framework with respect to thermal and performance control. Within the DPTM framework itself, a simple heuristic is devised to couple the thermal and performance-aware policies, thus allowing for stronger performance control and finer thermal control altogether.

The framework was implemented as an extension of FreeBSD 7.2. The measurement of temperature is performed by reading the appropriate model-specific register (MSR) for each core. Additionally, the measurement of throughput is gathered through the port of the heart rate monitor (HRM). The evaluation covered two parts: (1) a comparison experiment showing that the thermal-aware policy outperforms the Dimetrodon framework, and (2) a complete evaluation of the DPTM framework, showing the successful achievement of SLA and QoS throughput goals. Bartolini said that future work may consider improving the thermal model to account

for thermal interactions among multiple cores, and the idle-cycle injection technique itself.

Wei Xu asked about the capability of the proposed controller to work on different applications. Bartolini replied that the effect of different applications is captured and controlled using different parameters in the thermal policy that represent different workloads. Greg Bronevetsky asked whether the monitoring is required for CPU only or for the whole chip; he also asked whether the power used for cache was considered in this framework. Bartolini replied that injecting idle cycles saves power regardless. As for the cache question, Bartolini acknowledged that it wasn't considered in this work.

Transparent System Call-Based Performance Debugging for Cloud Computing

Nikhil Khadke, Michael P. Kasick, Soila P. Kavulya, Jiaqi Tan, and Priya Narasimhan, Carnegie Mellon University

Soila Kavulya started by stating that automated problem diagnosis in distributed environments is hard for many reasons but, most importantly, because system interactions are complex, systems are large scale, and systems are in production environments. One common example of large scale computing that demonstrates this problem is the MapReduce framework. Currently, most diagnosis systems focus on various forms of instrumentation and signatures that lack a proper representation of a system's state. This limitation is because such methods provide information relevant to the application or node of interest as opposed to the actual system state. To address these issues, Kavulya introduced their method, which principally relies on using a low level abstraction of the problem: a small set of system calls. The primary system calls of interest are network related (e.g., accept(), connect(), socket()), and file system related (e.g., access(), stat()). The target system of the proposed method is MapReduce frameworks, such as the open source Hadoop.

Kavulya's justifications for using system calls were that they (1) capture interaction between application and the OS, and (2) represent rich sources of both statistical data (e.g., disk access times) and semantic data (programs/files accessed). Kavulya mentioned that the main goals of the proposal are application transparency, minimized false positive rate, and coverage of both network- and disk-related problems. Their method assumes fault-free behavior of MapReduce nodes, identical hardware configuration of nodes, and synchronized time among the nodes. Their method also uses the UNIX tool strace to attain system call instrumentation. This method omits the data sent among network nodes because of incurred overhead and variability involved in tracing such calls.

Kavulya described the experimental setup to test the proposed method: running two MapReduce workload scenarios on a cluster of five identical machines running Hadoop 0.20.1. This was followed by a brief summary of both statistical-based and semantic-based system call diagnosis, along with a conclusion that diagnosis using system calls is effective. Future work may consider new ways to cope with heterogeneous systems, reduce instrumentation overhead, and distinguish between application-level and infrastructure-level problems.

Questions mainly focused on the coverage of the diagnosis. Marvin Theimer asked whether the method can combine multiple methods of diagnosis, e.g., high and low levels. Kavulya carefully answered no. Ajay Gulati asked about the coverage of memory calls and whether swapping can be involved in the diagnosis. Kavulya replied that although coverage can be extended to cover such a requirement, system calls remain unable to capture everything. Marc Chiarini asked about the diagnosis of cascading failures, as false negatives usually tend to demonstrate cascading behavior. Kavulya replied that false negatives are basically linked to thresholds and variances in addition to different instrumentations, where larger scope for such scenarios can be captured. Finally, Greg Bronevetsky asked about the idea of dealing with heterogeneity in general. Kavulya said that two ideas are currently in mind: using regression and searching for specific key scenarios among the different nodes.

Tracing

Summarized by Davide Basilio Bartolini (bartolini@elet.polimi.it)

Monitoring the Dynamics of Network Traffic by Recursive Multi-Dimensional Aggregation

Midori Kato, Keio University; Kenjiro Cho, IIJ/Keio University; Michio Honda, NEC Europe Ltd.; Hideyuki Tokuda, Keio University

Midori Kato presented Agurim, a technique for efficient and flexible multidimensional flow aggregation. Mining clusters in packet traffic is a promising way of capturing varying characteristics in a data stream; however, since packets are identified by a five-tuple (i.e., <{source, destination} IP, {source, destination} port, protocol>), the resulting space is huge and traditional techniques a re not applicable for on-the-fly clustering. Moreover, state-of-the-art aggregation flow visualization tools miss interactivity, failing to provide easily understandable and valuable information.

The proposed algorithm, Agurim, employs two-stage flow aggregation in order to provide both coarse- and fine-grained aggregated flows. This way, an operator can quickly identify anomalies looking at the coarse-grained data and then use the fine-grained information for in-depth analysis. The first aggregation stage is focused on efficiency, while the second stage achieves higher flexibility, further elaborating the

output of the first stage by leveraging R-tree data structures to manage multidimensional data. During the talk, she also gave a brief demo showing seven aggregated flows from one week of recorded traffic and demonstrating the interactivity of the approach by highlighting aggregated data over different time ranges.

Greg Bronevetsky (the workshop chair) asked for more information regarding possible use cases and examples of application for Agurim. Midori answered that it can be employed to detect anomalies (e.g., security violations and attacks) by analyzing the aggregate data. Greg asked for a more concrete example, and Midori offered a brief demo showing how a security violation should be visible to an expert operator examining the aggregated data provided by Agurim.

A State-Machine Approach to Disambiguating Supercomputer Event Logs

Jon Stearley, Robert Ballance, and Lara Bauman, Sandia National Laboratories

Jon Stearley presented a state-machine (SM) based approach to help analyze supercomputer event logs in order to identify real failures. Initially, he showed how supercomputer logs are difficult to interpret, using some real-world examples from a Cray supercomputer. For instance, messages can be found saying something like "FAILURE: process X exited normally." He also showed that, analyzing a log from a Cray supercomputer, 70,000 different events at seven different points in time were somehow correlated with failures, but only one of these was a real failure, the others being programmed reboots or other non-critical events.

Jon illustrated how it is possible to build an SM for log disambiguation. The states represent the current status of operation of the supercomputer (e.g., UP, HARD_DOWN, ...), and the transitions are triggered by rules based on the log events (e.g., events indicating that heartbeat signals from a certain node in a cluster stopped). The transition rules are synthesized by using domain knowledge from the supercomputer administrators. Jon showed how it is possible, with such an SM, to automatically analyze a log from a Cray supercomputer in order to highlight real failures over non-relevant messages. He claimed that the results of the analysis were validated by the supercomputer administrators. The evaluation of the proposed mechanism was based on a Splunk-based implementation.

Marc Chiarini (Harvard) asked if multiple events could be used in the transition rules, which could be beneficial in case correlations exist among different events. Jon answered that this is possible and more states could be added to the SM to support this. Ioan Stefanovici, (U Toronto) pointed out that the structure of log messages may not be well documented. Jon replied that it is crucial to leverage administrators' expertise in such cases. Greg Bronevetsky asked whether this approach would be applicable in environments lacking Splunk support. Jon answered that porting the work out of Splunk would not be conceptually complex, but it would imply some practical difficulties, especially in the management of different name spaces.

Uncertainty in Aggregate Estimates from Sampled Distributed Traces

Nate Coehlo, Arif Merchant, and Murray Stokely, Google, Inc.

Arif Merchant concluded the Tracing session with a talk dealing with how to determine the uncertainty of estimated metrics in distributed traces. The basic problem, he said, is that tracing logs from distributed systems sometimes lack some samples that would have been interesting for analysis. An example of such a system is Dapper, an always-on system for distributed tracing and performance analysis employed at Google. Dapper samples fractions of the remote procedure calls (RPCs) traffic, but may miss some relevant samples during runtime.

When samples are missing, metrics can be estimated based on aggregation of available data on the metric of interest. When these quantities are estimated, however, a measure of uncertainty in the estimate is needed; Arif presented a method for finding unbiased estimates of linear statistics over RPCs that quantifies uncertainty. The method presented by Arif is based upon hypothesis testing, and he presented it using disk accesses as a sample metric. Moreover, he presented a case study showing how the proposed technique can be useful in optimizing bin packing and cross-datacenter reads in a distributed storage environment.

Marc Chiarini (Harvard) asked whether the temporal granularity of the samples can affect the reliability of the hypothesis test-based technique. Arif answered that it depends on how frequent an event of interest is; if an event is rare, too much information may be missing due to a too-long sampling period. Arif pointed out that this technique is devised for frequent events. Greg Bronevetsky asked whether the proposed technique would be applicable to loops profiling (e.g., using gprof); Arif replied that it might be possible, but only if RPCs were present in the loops, since the proposed technique only considers RPCs.