Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML '10)

October 3, 2010
Vancouver, BC, Canada

Invited Talk

Summarized by Raja Sambasivan (rajas@andrew.cmu.edu)

QPS,KW-hr,MTBF,∆T,PUE,IOPS,DB/RH, . . . : A Day in the Life of a Datacenter Infrastructure Architect
Kushagra Vaid, Microsoft

Kushagra Vaid, principal datacenter infrastructure architect at Microsoft, presented this talk on challenges in datacenter design. Data centers are complicated, and datacenter design needs to take into account datacenter and server architecture, platform architecture, and reliability analysis.

Challenges in datacenter architecture include finding ways to optimize power distribution and cost efficiency. The metric of interest for the former is power utilization efficiency (PUE), computed as total facility power/IT equipment power. Typical industry averages range between 1.5 and 2.0. One common design choice that affects power distribution effi-

ciency is whether to propagate AC all the way to individual machines' power supply units or to convert to DC at the entry point to the data center. Vaid showed that DC configurations are more efficient at low loads and that AC configs prevail at higher loads, but that at highest efficiency both configurations are within 1–2% of each other. With regard to cost efficiency, Vaid showed how the scale of modularization has increased over time within Microsoft's data centers so as to increase this metric.

There are several challenges in the platform architecture area; for example, determining how to analyze workloads to find their optimal CPU requirements (frequency, number of cores, etc.) and determining whether it is worthwhile to pursue new hardware technologies (e.g., replacing desktop CPUs in datacenters with mobile CPUs or replacing hard drives with SSDs). With regard to the latter, Vaid showed that overall TCO for mobile processors, such as Atom, is 2.5x worse than regular desktop CPUs for both performance per dollar and performance per watt. Future Atom CPUs should either provide much better performance or much lower power in order to be considered feasible alternatives.

For reliability analysis, the main challenges involve determining how MTBF corresponds to environmental operating ranges. For example, Vaid showed how hard drive failure rates increase with temperature.

At the end the talk, Vaid made the case that finding an optimal solution for all of the areas together is essentially a multi-dimensional optimization problem, for which data-mining techniques and machine learning are required. Erik Riedel asked whether Vaid knew the distribution of hard drive failure modes with temperature. Vaid replied that the statistics collected were an aggregate and he did not know the breakdown. Has Microsoft considered releasing datacenter traces to researchers, so that they can investigate techniques for optimization? Microsoft already has released traces of datacenter workloads. Is there anything that keeps current ML tools from being useful for the problems presented by the author? Data formatting is a problem—the logs that contain the information ML tools need aren't in standard formats, making it difficult to use them.

Refereed Paper

Summarized by Raja Sambasivan (rajas@andrew.cmu.edu)

Creating the Knowledge about IT Events

Gilad Barash, Ira Cohen, Eli Mordechai, Carl Staelin, and Rafael Dakar, HP-Labs Israel

Gilad Barash from HP Labs presented research about a tool for answering user queries about IT events. Instead of spending time searching Web forums for answers to questions such as, "Why does my HP printer driver always return error message X?" users can pose their questions to the tool directly; the tool returns a ranked list of possible answers by either directly scraping IT Web forums or looking up the answer in a pre-computed knowledge database. The tool is composed of four distinct components—a search composer, a searcher, a ranker, and a knowledge database. The search composer simply creates progressively more generic search terms from the user input. The searcher queries standard search engines (e.g., Google) using the search terms and stops when a pre-determined number of results have been returned. The ranker is faced with the challenge of creating a better ordering than that returned by the search engine, by using domain and content-specific information. The knowledge database simply stores the results of previously stored queries.

The ranker uses three metrics to rank results: the source rank, the quality of information of each result, and the relevancy of each result. The source rank is simply computed by Web domain—if a search query is about HP printers, results from HP Web forums will be given a higher source rank than those from IBM Web forums. The overall quality of information (QOI) of each result is computed by combining several indicators of QOI—whether or not the relevant forum thread is marked as "question answered" or "not answered," the date the thread was last modified, the number of replies to the original poster, etc. In computing the QOI, the authors of the paper found that an important indicator of this value—whether or not the thread is marked as answered—tends to be noisy. That is, users often forget to update the label of a thread containing a valid answer from "not answered" to "answered." To deal with this problem, the authors developed a method for learning whether a thread contains a valid answer from the other QOI indicators. Finally, the relevancy of a result is computed by simple distance measures (e.g., string-edit distance) that compute the closeness of the search terms to words that appear in the result.

Barash concluded by saying that a prototype of the system has been implemented. An audience member asked how the tool's ranked results compared with just raw results from searching on Google. Barash stated that the ranks his tool yielded were often better than Google's results, because it takes into account domain-specific information, such as QOI scores. A concern raised was whether this conclusion was based on just the snippet of information returned by Google with each result, or whether it was based on looking at the actual documents. Barash stated that they had looked at the actual documents. Another audience member commented that this tool seemed great as long as the nmber of people using it were small compared to those creating data by posting on Web forums. He then asked whether it was possible to extend the tool so that it could feed back information

about the most relevant results to the Web forums it scraped for data. Barash said that such feedback is something that they're thinking about. For example, he said the tool could add tags to "strengthen" specific posts that it has computed are very useful. It could also automatically answer new questions.

Logging Design and Visualization

Synoptic: Summarizing System Logs with Refinement

Sigurd Schneider, Saarland University; Ivan Beschastnikh, University of Washington; Slava Chernyak, Google, Inc.; Michael D. Ernst and Yuriy Brun, University of Washington

Summarized by Raja Sambasivan (rajas@andrew.cmu.edu)

Ivan Beschastnikh, a student at the University of Washington, presented his research on algorithms for generating concise graph-based summaries of system log information. Ivan presented arguments for refining graphs of systems logs as opposed to coarsening. The former is the process of starting with a very coarse-grained graph in which related events are merged into single nodes and iteratively splitting them until a list of invariants is met. The latter is the process of starting out with a fine-grained graph and merging nodes until some invariant is violated. It is easier to satisfy important invariants by refinement than by coarsening, but refinement can create graphs that are too constrained. For such cases, combining refinement with coarsening can yield less constrained graphs that still satisfy all invariants.

The hybrid algorithm presented, called BisimH, starts by creating a graph of system-log events based on invariants specified by the user. Events that can be grouped together without violating the invariants are merged into partitions and depicted as nodes in the graph; edges depict dependencies between partitions.

BisimH then mines the system logs for additional invariants and uses them to generate examples that the current graph allows, but which the invariants do not. It then iteratively re-partitions the graph so as to satisfy the mined invariants. The problem of finding the most general graph that satisfies all of the invariants is NP-hard; as such, BisimH uses heuristics to explore the search space, often resulting in graphs that are too strict or too refined. As such, after each iterative refinement of the graph BisimH uses coarsening algorithms to determine whether a slightly coarser graph would satisfy the same invariants.

Ivan concluded by presenting case studies in which the BisimH algorithm, implemented in Synoptic, is used to understand the behavior of two protocols: the Peterson leader election algorithm and reverse traceroute. For both, he showed that BisimH yielded more accurate summaries than kTail, a popular coarsening algorithm. He also showed that BisimH was faster than coarsening with invariants.

An audience member asked Ivan to clarify the size of the logs used in his study and the processing overhead. He used only one machine for the case studies presented. He said that runtime was exponential in log events, but then changed his mind and said that it probably wasn't exponential.

A Graphical Representation for Identifier Structure in Logs

Ariel Rabkin and Wei Xu, University of California, Berkeley; Avani Wildani, University of California, Santa Cruz; Armando Fox, David Patterson, and Randy Katz, University of California at Berkeley

Summarized by Peter Hornyack (pjh@cs.washington.edu)

Ariel Rabkin presented a new system to uncover flaws in the coverage and consistency of application console logs. This work was motivated by the fact that while log messages are a primary tool for debugging applications, analysis of these logs is often hampered by missing, incorrect, or inconsistent information. The goal of the system is to improve future log analysis by visualizing the log message types and identifier fields in a way that makes common flaws easily visible and facilitates comparison across logs.

The system analyzes application logs offline and visualizes several aspects of them in the form of a graph. The nodes in the graph represent either message types or identifiers such as transaction IDs. Edges in the graph indicate the identifiers that appeared in messages of certain types; other information, such as the relative frequency of message types, is also visualized in the graph. Rabkin presented several example visualizations and pointed out the flaws they reveal; for example, missing identifier errors are easily seen as messages in the graph without edges to any identifiers. Another example showed that inconsistency in the identifiers used across multiple message types appears in the graph as an identifier connected to only a single message. Finally, Rabkin showed how the system can be used to identify logging errors by visually comparing the resulting graphs of logs from production systems.

One audience member noted that the example graphs were for the most part planar, and wondered if the authors had produced any graphs that were too messy to visualize easily. Rabkin replied that for most programs, the set of identifiers and message types is not that large and that manageable graphs usually result, and also emphasized that some of the example graphs were from large real systems, such

as production Hadoop clusters. Another audience member asked how difficult it is to find the corresponding bug in the code when a flaw is observed in the visualization. Rabkin answered that the process is usually straightforward, since the origin of each message type is usually easy to find in the code, and noted that the time spent fixing these logging bugs pays for itself by enabling better error detection and debugging using the logs in the future.

SIP CLF: A Common Log Format (CLF) for the Session Initiation Protocol (SIP)

Vijay K. Gurbani, Bell Laboratories/Alcatel-Lucent; Eric Burger, Georgetown University; Carol Davids and Tricha Anjali, Illinois Institute of Technology

Summarized by Peter Hornyack (pjh@cs.washington.edu)

Vijay Gurbani presented work on the development of a common log format for SIP. Most enterprises have SIP servers and clients for IP telephony and other uses, but these are often obtained from multiple vendors, each of which uses a different log format today. The authors argue that a CLF for SIP is needed to allow trend analysis and anomaly detection across equipment from multiple vendors, and to encourage the development of third-party tools for troubleshooting SIP. The success of the HTTP CLF in these respects and some recent publications on the complexity of SIP parsing were presented as support for a SIP CLF.

Gurbani presented some background information on SIP, then described the HTTP CLF and pointed out the myriad differences between the SIP and HTTP protocols that make defining a CLF for SIP more challenging than for HTTP. For example, unlike HTTP, SIP is not a linear protocol with exactly one reply for every request. The need for multiple responses per request and the potential for long delays between requests and responses in SIP increase the complexity of the state that must be recorded in a SIP CLF. Gurbani presented the work done to create a canonical record format, with an emphasis on its extensibility and the ease with which it can be parsed. He then showed some complex SIP flows and demonstrated how simple grep commands could be used to perform useful queries on logs in the canonical format. The IETF is currently in the process of standardizing the SIP CLF proposed by the authors.

Many audience members wondered why there isn't already a CLF for SIP, despite it being in use for many years. Gurbani replied that this is not unexpected, since the primary focus of the IETF has been to stabilize the protocol (SIP) itself and reduce ambiguities in the specification, and accoutrements such as logging are being looked at now. Furthermore, unlike the dominance of Apache in HTTP which fostered an HTTP CLF, there isn't an open source equivalent in SIP to do the same for a SIP CLF. Have the authors investigated how logs in the proposed format could be transformed into graphs or other useful structures for visualizing the log contents? So far they have been concerned strictly with getting the syntactic specification of the SIP CLF finished in the IETF, and uses such as visualization tools, graphs, correlation engines, and others will be much easier to develop with a canonical format in place.

Applications of Log Analysis

Summarized by Peter Hornyack (pjh@cs.washington.edu)

Experience Mining Google's Production Console Logs

Wei Xu, University of California at Berkeley; Ling Huang, Intel Labs Berkeley; Armando Fox, David Patterson, and Michael Jordan, University of California at Berkeley

Wei Xu presented early findings from his group's investigation of console logs from Google production systems, beginning with some of the challenges in using console logs in large systems: they are usually stored with just best-effort retention, log messages are often generated ad hoc using custom logging libraries, and new message types are constantly introduced. The data set mined by the authors was produced by systems consisting of thousands of nodes, with five orders of magnitude more messages and many more message types than they used in their previous work.

One problem that the authors focused on is using global system state, rather than local node state, to detect problems in the system. By applying machine learning techniques for anomaly detection, the authors were able to find features in the log messages that correlated with system alarms. The authors also used sequence-based detection to identify problems on single nodes in the system. Finally, Xu presented techniques that the authors used for removing sensitive information from data gathered on production systems, a process termed "log sanitization." The authors' evaluation of their log mining techniques demonstrates that the techniques scale and apply to the extremely large production data set.

Analyzing Web Logs to Detect User-Visible Failures

Wanchun Li, Georgia Institute of Technology; Ian Gorton, Pacific Northwest National Laboratory

Most Web applications suffer from unreliability, which causes downtime and transaction errors that directly impact users. Wanchun Li pointed out that early failure detection can mitigate later failures, but detection itself is difficult to

perform in complex Web apps, and existing automated techniques are often ineffective.

Li presented a system that detects failures that users experience when using Web applications. The authors' approach to detecting these failures is based on the principle that when users experience failures, they will respond to the failure in a way that breaks from the usual navigation paths in a Web app. The system models a Web app as a graph with pages as nodes and users' navigation as edges, then uses a trained Markov model to estimate the probability of a given navigation path. If the computed probability of a navigation path is less than some threshold, then the system raises a failure alarm, indicating that the anomalous path may have been the result of the user experiencing some failure. The authors evaluated the system using an access log of HTTP requests from NASA's Web site, and found that at the optimal balance point between detection rate and false-positive rate, the system correctly detected 71% of failures with 26% false positives.

One audience member asked if the system would have to construct and train a completely new model when the Web application is modified. Li replied that the graph is constructed incrementally and can adapt to changes in the set of pages and navigation paths. Since even the least popular pages on large Web sites are visited regularly, would the system classify these visits to the least popular pages as failures? With sufficient training data, the visits to even the least popular pages and the typical navigation paths to them would be captured by the model, and only navigations that don't follow some typical path would be marked as failures.

Industry Track—Experiences

Summarized by Ivan Beschastnikh (ivan@cs.washington.edu)

Optimizing Data Analysis with a Semi-structured Time Series Database

Ledion Bitincka, Archana Ganapathi, Stephen Sorkin, and Steve Zhang, Splunk Inc.

The workshop's industry track featured two papers, both of which focused on Splunk, a platform for collecting, searching, and analyzing time series data from many sources with possibly varying formats. Archana Ganapathi presented the first paper, and gave an overview of how Splunk works.

Archana quoted Joe Hellerstein's statement that we are in the "industrial revolution of data." Managing the growing explosion of data is a key challenge, and one of the most difficult aspects of big data is enabling efficient analysis. After all, what's the point of storing it all if there is no means of extracting valuable insights? The Splunk platform enables one to search, report, monitor, and analyze streaming and historical data from a variety of sources—for example, in an IT infrastructure: logs, configurations, messages, traps and alerts, scripts, custom code, and more. As Archana stated, "If a machine can generate it, Splunk can eat it."

Splunk's design uses three tiers: (1) forwarders collect data and send it to the indexers; (2) indexers denormalize the data, attach keywords, and of course index the data; (3) a search head provides a single query point to which users can then submit queries. These queries are converted into MapReduce jobs which run across the indexers. Because co-temporality is crucial to many queries, Splunk uses a modified MapReduce hashing function to map data from the same time-window onto the same machine.

One of Splunk's important features is its streaming indexing system, which enables real-time search results. However, possibly the most attractive feature in Splunk is its advanced query language. Expressions in this language eventually compile down to MapReduce jobs, but the user is relieved from thinking about the map and reduce functions—the conversion is entirely transparent. Because Splunk does not use data schemas, the query language supports searches over heterogeneous and constantly evolving formats. Combine operators, which are essentially UNIX pipes, are used to string multiple expressions into complex programs. Archana gave detailed examples of uses of this query language for outlier detection, clustering, and data munging (combining data from multiple sources and with different formats).

In the Q&A, a few of the audience members wanted to gain a more detailed understanding of how Splunk works. For example, the presentation did not describe how the various features of the language were realized in the MapReduce framework. These questions were taken offline.

Bridging the Gaps: Joining Information Sources with Splunk

Jon Stearley, Sophia Corwell, and Ken Lord, Sandia National Laboratories

The second presentation in the workshop's industry track was presented by Jon Stearley, who discussed his experiences with using Splunk to make sense of supercomputer logs. Jon set the stage by describing Sisyphus, a tool he developed to collate unstructured logs from across many systems. The many features of Splunk, however, convinced Jon to try the new system. In particular, he found Splunk to be robust in dealing with logs lacking a well-defined schema, and that Splunk's built-in support for collaborative log exploration made it easy to involve many people in the process and to capture and share knowledge.

During the bulk of his presentation, Jon focused on a few Splunk features. One of these is Splunk's ability to treat unstructured input logs as relational entities. This allows

one to, for example, compose queries that join logs across log fields. Another feature is Splunk's ability to associate event types with those messages that satisfy at least one query pattern in a set (or some other constraint) and to write queries over event types. Yet another feature is subqueries, which provide a powerful composition mechanism. On top of a complex log analysis feature set, Splunk makes it easy to document, save, and share queries. This captures communal log knowledge and allows more people to participate in log analysis tasks. As a summary of what Splunk is capable of, Jon emphasized that Splunk "takes care of all the ugly preprocessing" that log analysis typically involves.

In the Q&A an attendee asked how Splunk deals with logs that have unsynchronized clocks. Jon hadn't dealt with such logs before and therefore couldn't comment. Is Splunk difficult to learn? Splunk is simple, and many features of its query language have UNIX command-line analogs. How well does Splunk integrate external analysis tools? It is straightforward to plug other tools into Splunk.

Panel Discussion

Summarized by Ivan Beschastnikh (ivan@cs.washington.edu)

John Hammond, University of Texas at Austin; Jon Stearley, Sandia National Laboratories; Eric Fitzgerald, Microsoft

The SLAML panel discussion revolved around many outstanding issues in the log mining and analysis research community. It also touched on how these issues relate to the challenges faced by industry.

The first theme to generate interesting discussion was log completeness. Eric Fitzgerald pointed to the incompleteness and a lack of standardization of existing logging formats as a major problem for log analysis and tool interoperability. Eric described and advocated that the community pay attention to the Common Event Expression (CEE) effort. This effort aims to define a standard for event interoperability across a wide range of logging sources and establishes which events must be raised when, and what fields an event of a particular type must include.

The CEE proposal generated heated debate. Ledion Bitincka from Splunk responded that log mining software, such as Splunk, offers the only practical solution to unifying the multitude of existing logging formats. Moreover, the log line itself cannot accurately report on what caused it to appear; a mining tool to explore log patterns is therefore essential. Eric's response was that Splunk requires significant user expertise—the language may be simple to learn, but in an unstructured log one must still know how to recognize an error (e.g., HTTP code 404). Another retort to CEE came from Jon Stearley, who asked how a developer might be incentivized to switch and follow the CEE standard. Eric replied that developers today re-invent logging, including timestamp generation, setting delimiters, escaping multi-word messages, etc. CEE would help with all this, and in addition developers may be incentivized to use CEE if customers prefer software that generates logs in this format, as this would be an indicator of software quality and log interoperability.

Greg Bronevetsky asked what makes a good log and which is better, an informational or an activity-based logging strategy. Eric replied that activity-based logs, in which the log captures a change in state, are the most useful for security applications. Others in the audience thought that an informational logging strategy can also play an important role for debugging insidious errors that may span multiple components.

Jon asked whether anyone knew of a good survey paper in the field of log analysis. No one could name such a paper, and Jon suggested that this field is ripe for survey papers. Alice Zheng added that a survey paper focusing on diagnosis would be particularly welcomed. Greg mentioned a forthcoming survey paper by Felix Salfner et al. covering online failure detection methods (http://portal.acm.org/citation.cfm?id=1670680). Wei Xu also noted that his dissertation includes a survey in Chapter 7 (http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-112.pdf).

Raja Sambasivan asked the practitioners to imagine what they might want to find in logs if they contained perfect data. John Hammond responded with a description of supercomputing applications. Many of these were diagnostic—having observed some event, one wants to know what and who might have caused it. If there is uncertainty, a list of possibilities ordered by their likelihood would be useful. Alice wondered whether this is feasible if the data does not contain enough labels. To this, John said that one can label things by hand, as they have done with their own datasets. Ideally, however, such labeled datasets would be widely available to the community to experiment with.

This discussion was broadened by Adam Oliner, who pointed out that the log analysis field, if it is to be scientifically rigorous, needs a freely accessible log repository and a set of common metrics. Adam mentioned that he and Jon Stearley have made available a large, tagged, unfiltered dataset and that they encourage the community to make use of it (http://cfdr.usenix.org/data.html#hpc4). More generally, Adam pointed to the USENIX Common Failure Data Repository (CFDR—http://cfdr.usenix.org/) as an example of how logs can be made available to the broader research community. Greg proposed the idea that the SLAML community can organize around a few logs a year so that analyses reported for the same data source may be compared.

During the discussion of common analysis metrics, user studies were pointed out as a rigorous means to evaluate graphical log representations. To this Ari asked, what sorts of user studies in particular would the SLAML community trust and find useful? Adam responded that this is not something the community has a standard for. Wanchun Li noted that usability studies in security research face a similar issue. Everyone knows that evaluating usability is often an important aspect of the research, but there is little progress on establishing common usability metrics.

Another challenge touched on by a few participants is that the meaning of logged messages may change over time. This can, for example, make log priority levels meaningless (what used to be an error message is now an informational note). This is in part because there is no incentive to remove a log line after the error is fixed. Greg proposed fault injection as a potential solution. With fault injection one can see which messages correlate, and this reveals information about the underlying dependency graph. After all, Greg asked, isn't this the only thing we can find out, namely that certain events correlate, while others do not? Alice questioned whether fault injection can be a complete solution. In particular, she asked about how one translates information gained from fault injection in a testing environment into a production environment. Greg admitted that this is a limitation, but also emphasized that by performing fault injection across different configurations one can often glean important properties of the system, such as its scalability. Raja also thought fault injection to be impractical because it is difficult to trust results gathered in an artificial setting. He pointed out that interesting real-world bugs always seem to be much more involved, and reproducing them in fault injection studies is a research study in itself.

The mention of many limiting features of system logs led Ivan Beschastnikh to ask whether the community should instead consider bridging log analysis with program analysis, as program source can offer logs analysis key contextual clues. Ari Rabkin mentioned relevant work by Ben Liblit on cooperative BUG isolation, which leverages large numbers of execution observations (execution logs) for debugging. Ari also indicated that Ivan's proposal is impractical because program analysis rarely scales to large software and that system software analysis is especially difficult, as it may involve tracing complex execution (e.g., across multiple bash scripts). Alice pointed out that instrumenting real code has an overhead and it's difficult to tell which pieces are important to instrument and which ones do not add much more value.

Alice mentioned that a key challenge in applying machine learning to logs is the lack of labels. She suggested that crowd sourcing (e.g., via Mechanical Turk) can be leveraged to label existing logs. For example, Google improves its search by instrumenting their pages and monitoring the click-through behavior of its many users. Vijay Gurbani noted that the lack of labeled logs is primarily an issue with computer science education—students are taught how to program, but not how to properly organize their program's log output nor how to label and then study the output to understand their programs. Also in response, Wei noted that labeling is especially difficult, because the same log message may mean different things to different people (e.g., a developer versus a system administrator). Mitchell Blank raised the related logging incentives challenge—developers will always opt for an easier way, so one must provide an incentive for them to produce meaningful and easy-to-analyze logs.