

Combining Structured and Unstructured Data for Web Application Performance Analysis

Jeff Martin

`jmartin@cs.duke.edu`

December 11, 2007

Abstract

System performance analysis is typically conducted by collecting structured data in the form of metrics or by examining unstructured data in the form of logs. Tools currently available either analyze the structured data or the unstructured data, but not both. In some settings, this is not enough to make meaningful statements about the performance of the system. In this work, we explore alternatives for querying system performance in a web application environment. PHP applications execute under the same process as the web server itself. This makes characterizing the performance of PHP applications more difficult because tools like UNIX Top only show process-level metrics. However, logs generated by the web server can also contain valuable information about the the web applications. Because these logs are sources of largely unstructured information, we use information retrieval techniques to query them. By combining the unstructured data from the logs with structured metrics from Top, we can assign resource usage estimates to the web applications.

1 Introduction

Performance analysis of a system typically involves a system administrator querying the system in some way to answer questions related to the utilization of resources available to the system. Often, tools are available to facilitate the administrator's requests by collecting statistics about the system while it is running. For example, in a Windows environment, a system administrator might open Task Manager to query the CPU and memory usage of a particular application. Or perhaps the administrator wishes

to know which application is responsible for high CPU utilization. Task manager can also provide this information by sorting the running applications by amount of CPU usage. UNIX-based systems have a similar program called Top that also provides resource utilization statistics.

Another approach taken by administrators is to analyze logs generated by the target applications. These logs can contain markers tagged with specific times. One can determine the amount of processing time needed for a particular task by examining its corresponding start and stop markers. This information will not satisfy the what-is-going-on-right-now class of queries, but instead can provide a history of information about the applications.

Performance analyses can be conducted with respect to a variety of resources (e.g. CPU, memory, network, disk). In this work, we will focus on CPU analysis of applications in a web-based environment under the assumption that analyses on other resources can be conducted using similar methods and similar tools.

1.1 Moving to the Web-Based Environment

These tools and approaches are well-suited to the desktop environment where each application of interest executes in process separate from other applications. Tools like Task Manager and Top are able to accurately report resource usage statistics for applications in this environment. But in a web-based environment, applications share the executing process of the web server effectively hiding them from Top. The typical tools and approaches described above cannot be used in these situations effectively.

Top, for instance, is unable to provide specific metrics of CPU utilization for a single web application in isolation. This is because all of the web applications execute in the same process as the web server and so their CPU utilization is reported as the CPU utilized by the web server. The web server does indeed use some of the CPU for accessing files from disk. Many web servers also do processing on the URL and other parts of the HTTP requests which also consumes CPU. However, for web applications, the large majority of the CPU usage is the result of processing needed by the web applications themselves. Top does provide the CPU utilization of the entire web server process which is of some use.

Log analysis can yield more detailed information, but is often disconnected from direct CPU performance. Events in the log tagged with times can yield the amount of processing time needed for an operation, but it cannot tell us how much CPU was actually being used during that time in relation to other applications. For instance, one could write a marker to the log signaling the start of processing a request and another to signal the end of processing, but this information only gives us the length of time used to process the request. What it does not tell us is whether or not the CPU was devoted to this request entirely or if the CPU shared execution with other requests which is more often the case. Again, this information is insufficient by itself to break down the CPU usage of each web application.

1.2 Combining Methods

Therefore, we turn to combining these methods to give us the complete picture. The data provided by Top can be considered as structured data. If sampled over time, the CPU utilization statistics of the web server process can be stored in a DBMS. By having a history of CPU utilization statistics available to us, we can begin to answer questions such as the following. For a given period of time, what was the highest CPU utilization recorded? Or the reverse: Over what periods of time did the web server sustain a CPU utilization of at least a given threshold. We can use this information to identify periods of time characterized by high CPU utilization.

Once we have identified these periods, if we can show that a particular application is significantly more active than the others, we can say that applica-

tion is largely responsible for the observed CPU utilization. To determine application activity, we turn to analyzing information contained in logs. Since logs are largely collections of unstructured data, we use specialized techniques to analyze them effectively. Thus, we combine structured data from Top with unstructured data from logs to make statements about the performance of web applications individually.

We do the analysis in this manner under the following assumptions.

- Application activity has a direct relationship with overall CPU utilization
- CPU utilization of each application (in isolation) is equivalent given equivalent activity

This framework allows to make informative statements about the performance of the system. Namely, a collection of active applications will result in a higher CPU utilization than a collection of inactive applications. Also, if two applications have the same activity level, then they should have the same CPU utilization. Finally, we can say that the most active application is the greatest contributor to CPU utilization in a given time period.

1.3 Characterizing Application Activity

Logs are composed of mostly unstructured data and as such, one must create a framework in which the log can be analyzed based on assumptions about the log's content. For instance, we will assume that a log is a collection of ordered events with each event contained one line of the log. In this framework, not all events will be of use to us, so we will define an *interesting event* as one that meets the following criteria. An interesting event must have a timestamp and must refer to one of the web applications. Furthermore, we define an *interesting log* to be one that contains at least one interesting event.

This reference can be defined in a number of ways. The event could simply contain the name of the application directly. Or the event could mention a file structure that is unique to an application. Essentially, the event must mention a token or collection of tokens that is completely specific to one of the applications.

We can quantify the activity of an application over a period of time by counting the number of interesting events among the logs for that period of time. Naively, one could say that the application with the most interesting events recorded could for a given period of time is the most active. However, this information alone can lead to slightly skewed results as one application could contribute more events to the log on average than another application arbitrarily. For example, a user could click a button in application *A* resulting in 2 events being sent to the log. In contrast, a similar button click could contribute 4 events to the log for application *B*. Therefore, we must normalize application activity in a way that makes applications' event counts more comparable.

1.4 Normalizing by Requests

A *request* to an application is any input to the application that initiates processing. In the web environment, this is typically the result of clicking on a link or button. In the HTTP protocol itself, these actions send GET and POST requests from the web browser to the web server. All processing in a web-based environment is initiated by these requests so they become a good indicator of application activity levels. If we assume that these requests are responsible for the load generated by the web application, then the number of requests in a given time period will be more informative than the number of events in terms of determining CPU usage.

Therefore, we can translate the event counts for an application into request counts to provide us with more useful information for comparison. If we can determine the average number of events-per-request for an application, we can scale its event counts to estimate the number of requests that application receives. Then, we can determine the most active application by choosing the application that has the largest number of requests relative to the other applications.

2 Related Work

The combination of analyzing structured and unstructured data together falls into the realm of database management systems and information retrieval (DBMS+IR). Researchers have taken standard queries written in SQL and transformed them

into keywords used to retrieve relevant unstructured information[2]. They also demonstrated that this could be done with high accuracy and with reasonable overhead. In contrast, EROCS[1] is a system designed to process a document containing unstructured data and return relevant system entities from a structured system even when the entity is not explicitly mentioned in the document. Further, they can perform the relationship determination with minimal access to the database.

Both of these works explore the feasibility of translating queries from one domain to the other. Namely, they either translate a query on structured data to one on unstructured data or vice versa. This work performs a similar translation from a query on structured data to a query on unstructured data. We first ask the Top history for periods of high CPU utilization by querying structured data. Then we use that information to query the unstructured data to determine application activity levels for that time period.

3 Experimental Method

To test our model, we conducted four experiments on a web application system. We installed 4 PHP applications

- DokuWiki - an openly editable collection of articles similar to Wikipedia
- Gallery - a repository for photos and images similar to Flickr
- WebCalendar - an online calendar application similar to Outlook
- phpbb - a discussion forum system

on an Apache web server and simulated user traffic to these applications to stimulate CPU and log activity. The CPU activity was recorded by sampling top and inserting the results in to a MySQL database. The logs were cleared before each experiment and then copied at the experiment's conclusion to ensure each experiment uses a log clear of clutter from other simulations. In each of the four experiments, one application was selected to have a high level of usage while the others would be accessed at much lower levels. We can granularize application usage by varying the delay between requests submitted to the applications

in our load simulation framework. Thus, an application with a higher usage level received more requests per second than a less used application. We should expect our analyses to show that the application selected to have high usage would be quantifiably more active than the other applications using the methods presented in this work.

4 Results

In our first experiment, we selected the *Wiki* application to have the highest usage level relative to the others. Shown in Figure 1 is a graph depicting requests over time for the experiment. To clarify the results, the requests for each application have been smoothed using a moving average. We see that in-

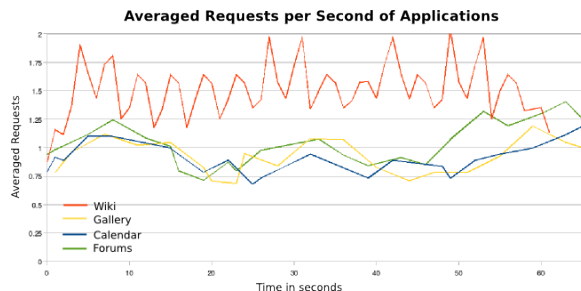


Figure 1: Wiki Experiment Requests

deed the *Wiki* application is more active than the others because its curve shows a higher number of requests over time. But how can one quantify that the application is significantly more active than the others? We can do this by continuing to smooth out the curves all the way down to a single value. Essentially, this is the average requests-per-second over a given period of time which we can use to represent an application’s observed activity level. To facilitate direct comparison of the averaged values, we can normalize them such that the average requests-per-second for all of the applications sum to 100%. The results of this averaging and normalization are presented in Table 1. Figures 2, 3, 4 and tables 2, 3, 4 show results for the remaining three experiments.

5 Conclusion and Future Work

Experimentally, the methods we describe work reasonably well. For each test we ran, the results sup-

Application	Percent Activity
Wiki	35
Gallery	22
Calendar	21
Forums	22

Table 1: Wiki Experiment Activity

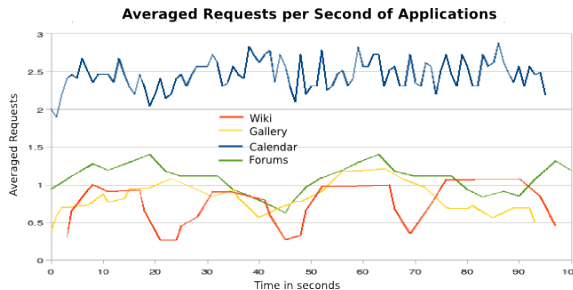


Figure 2: Calendar Experiment Requests

Application	Percent Activity
Wiki	14
Gallery	16
Calendar	49
Forums	21

Table 2: Calendar Experiment Activity

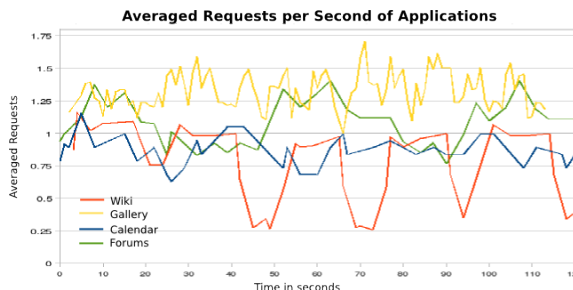


Figure 3: Gallery Experiment Requests

ported our model of application activity in general. However, the degree to which the results supported the model varied from experiment to experiment. The results for the *Forums* experiment showed a strong support for our model, but the results for the *Gallery* experiment were slightly more ambiguous. This suggests that the assumptions under which our framework for analysis was built may not be entirely

Application	Percent Activity
Wiki	19
Gallery	33
Calendar	21
Forums	27

Table 3: Gallery Experiment Activity

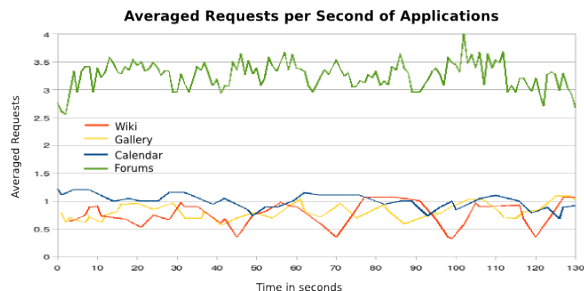


Figure 4: Forums Experiment Requests

Application	Percent Activity
Wiki	13
Gallery	14
Calendar	17
Forums	56

Table 4: Forums Experiment Activity

valid.

It is unlikely that application activity is not directly related to CPU usage. Such a counterexample would involve an application whose repeated usage actually decreased CPU utilization. So we examine the second assumption where we assumed each application utilized the CPU similarly given similar activity. It is likely this assumption is only partially true given all situations. For instance, the results show the *Forums* application can process on average more requests per second than the *Gallery* application under the same usage level. While the evidence is not conclusive, it does suggest the CPU impact of the applications is similar but not identical given identical usage levels. In future work, we could explore ways to measure the CPU impact of each application in isolation and scale event counts similarly to the procedure for scaling by requests.

References

- [1] Venkatesan T. Chakaravarthy, Himanshu Gupta, Prasan Roy, and Mukesh Mohania. Efficiently linking text documents with relevant structured information. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 667–678. VLDB Endowment, 2006.
- [2] Prasan Roy, Mukesh Mohania, Bhuvan Bamba, and Shree Raman. Towards automatic association of relevant unstructured content with structured query results. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 405–412, New York, NY, USA, 2005. ACM.