Characterizing, Modeling, and Generating Workload Spikes for Stateful Services

Peter Bodík, Armando Fox, Michael J. Franklin, Michael I. Jordan, David A. Patterson
EECS Department, UC Berkeley, Berkeley, CA, USA
{bodikp,fox,franklin,jordan,patterson}@cs.berkeley.edu

ABSTRACT

Evaluating the resiliency of stateful Internet services to significant workload spikes and data hotspots requires realistic workload traces that are usually very difficult to obtain. A popular approach is to create a workload model and generate synthetic workload, however, there exists no characterization and model of stateful spikes. In this paper we analyze five workload and data spikes and find that they vary significantly in many important aspects such as steepness, magnitude, duration, and spatial locality. We propose and validate a model of stateful spikes that allows us to synthesize volume and data spikes and could thus be used by both cloud computing users and providers to stress-test their infrastructure.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services

General Terms

Measurement

1. INTRODUCTION

A public-facing Internet-scale service available to tens of millions of users can experience extremely rapid and unexpected shifts in workload patterns. Handling such spikes is extremely challenging. Provisioning for them in advance is analogous to preparing for earthquakes: while designing for a magnitude-9 earthquake is theoretically possible, it is economically infeasible because such an event is very unlikely. Instead, engineers use statistical characterizations of earthquakes—distributions of magnitude, frequency, and phase—to design buildings that can withstand most earthquakes. Similarly, while overprovisioning an Internet service to handle the largest possible spike is infeasible, pay-as-you-go cloud computing offers the option to quickly add capacity to deal with spikes. However, exploiting this feature requires understanding the nature of unexpected events—how fast they ramp up, how they peak, etc.—and testing the service under such scenarios.

Workload modeling and synthesis has always been important in systems R&D for stress-testing new production systems, as well as in academic research because of the difficulty of obtaining real commercial workload traces that might reveal confidential company information. The challenge of understanding and modeling spikes and the need to synthesize realistic "spiky" workloads motivate the present work.

The term "spike," or "volume spike," is commonly used to refer to an unexpected sustained increase in aggregate workload volume. In the case of stateless servers such as Web servers, such spikes can in principle be "absorbed" by a combination of adding more servers and using a combination of L7 switches, DNS, and geo-replication to redirect load to the new servers. Much work has focused on using both reactive and proactive approaches to do this [26].

However, Internet-scale data-intensive sites—social networking (Facebook, Twitter), reference content (Wikipedia), and search engines (Google)—must also deal with *data spikes*: a sudden increase in demand for certain objects, or more generally, a pronounced change in the distribution of object popularity on the site. Even data spikes that are not accompanied by an overall surge in volume may have severe effects on the system internally. For example, consider a partitioned database in which all objects are equally popular, but a sudden event causes 90% of all queries to go to one or two objects even though the total request volume does not increase.

Although volume spikes and data spikes can arise independently, it is becoming increasingly common for the two phenomena to occur together. For example, at the peak of the spike following Michael Jackson's death in 2009, 22% of all tweets on Twitter mentioned Michael Jackson [7] (the largest Twitter spike in 2009), 15% of traffic to Wikipedia was directed at the article about Michael Jackson (see Section 4), and Google initially mistook it for an automated attack [5]. The second and third most significant spikes on Twitter were Kanye West and the Oscars with 14.3% and 13.1% of all tweets [7]. During the inauguration of President Obama, the number of tweets per second was five times higher than on a regular day [3].

Such events are becoming more regular and larger in scale. They pose new challenges beyond those of modeling and stress-testing volume spikes for two reasons. First, as mentioned previously, the effect of a data spike may be severe even if the aggregate workload volume does not change much. Second, unlike stateless systems, simply adding more

SoCC'10, June 10–11, 2010, Indianapolis, Indiana, USA.
Copyright 2010 ACM 978-1-4503-0036-0/10/06 ...$10.00.

machines may not be enough to handle data spikes: data may need to be moved or copied onto those machines. Deciding what to copy requires information not only about the distribution of object popularities during a data spike, but about the locality of accesses: if a group of objects becomes popular rather than a single object, are the popular objects grouped together on storage servers or scattered across the system?

Thus, for modeling Internet-scale workload spikes in full generality, we must be able to model a volume spike, a data spike, and their superposition. To this end, we make three contributions:

1. An analysis of five spikes from four real workload traces, showing that the range of volume and data spike behaviors is more varied than one might expect and does not necessarily match existing assumptions about object popularity;

2. A methodology for capturing both volume and data spikes with a simple seven-parameter model based on a sound statistical foundation;

3. A closed-loop workload generator that uses these models to synthesize realistic workloads that include volume and data spikes.

Contribution 1: analysis of five spikes from four real Web server traces. We identify several important spike characteristics that suggest that volume spikes and data spikes are more varied than one might expect. The five spikes we analyzed represent a wide range of behaviors, including data spikes with little volume change, data spikes accompanied by volume spikes, and different distributions of object popularity and locality during data spikes. The spike durations vary from hours to days, and the workload volume increases (peak of the spike) vary from insignificant to factors of 5 or more. We also find that, in accordance with intuition, the onset shapes and peak values of volume spikes for "anticipated" events such as holidays or the Oscars differ substantially from those for unexpected events such as the death of Michael Jackson or the Balloon Boy hoax. This suggests that "stress testing for spikes" (especially for stateful services) is more subtle and involves more degrees of freedom than previously thought.

Contribution 2: a methodology for capturing volume and data spikes with a simple yet statistically sound model. We model the spikes in three steps. First, we model normal workload without spikes based on workload volume and baseline object popularity. Second, we add a workload volume spike with a particular steepness, duration, and magnitude. Finally, we select a set of hot objects with a particular spatial locality and increase their popularity. A particular novel aspect of our methodology is the identification and representation of *popularity distribution change* and *data locality* during a data spike:

(1) How does the distribution of object popularities change after spike onset and throughout the spike? What is the shape of this distribution—do the top N objects increase equally in popularity, or is there a significant tail?

(2) Given a simple representation of how data objects are mapped to storage nodes, what locality is present in accesses to groups of popular objects during the spike? That is, when a group of objects increases in popularity, do accesses to the group tend to cluster on one or two servers, or do they require touching multiple servers across the installation?

Contribution 3: synthesis of realistic workloads from the model. We built a closed-loop workload generator that uses our models to synthesize realistic volume and data spikes. We show that by properly setting the model parameter values, we can generate workload with any desired characteristics. In particular, we show that the total workload volume, workload volume seen by the hottest object, and workload variance generated by the model match those characteristics in the real traces we analyzed.

Our goal is not to create an intricate model that imitates every detail of the real traces, but a simple yet statistically sound model that is experimentally practical and captures the important characteristics of workload and data spikes. Note that the goal of the model is not to predict the *occurrence* of spikes, but to model the changes in workload volume and popularity of individual objects when spikes occur. Recalling the comparison to earthquakes, while we may not be able to predict individual earthquakes, it is useful to characterize general properties of earthquakes so as to provision appropriately.

2. RELATED WORK

Web server workloads have been thoroughly studied in many papers, such as [10, 15]. Most of these papers, however, analyze workload for stateless Web servers in context of caching, prefetching, or content distribution networks. While some papers study surges, spikes, or flash crowds [25, 19], they concentrate only on the increase in workload volume to the Web server.

Most of the workload generators [4, 13, 2, 1, 6] used to evaluate computer systems or Web sites do not support either volume or data spikes. Httperf [2] provides a simple tool for generating http workload for evaluating Web server performance. Httperf, however, generates requests only at a fixed rate specified by the user. Rubis [6] is a Web application benchmark with a workload generator that uses a fixed number of clients and thus cannot generate a volume spike. Faban [1] is a flexible workload-generation framework in which volume and data spikes could be implemented, but are not supported out of the box.

Similar to our notion of data spikes is the concept of *temporal stability* of workload patterns in a Web server described in [23]. The authors analyze Web server traces from a large commercial web site mostly in the context of caching. They define temporal stability as the overlap of the top N most popular pages during two days and find that the stability is reasonably high on the scale of days. While the idea of popularity of the top N pages is similar to data spikes, we are interested in changes on time scale of minutes, not days or weeks. Also, the authors do not provide a model or a workload generator to simulate such changes.

The authors of [22] propose a methodology and a workload generator to introduce burstiness to a stateless, Web server workload. The authors characterize burstiness using the *index of dispersion*—a metric used in network engineering. The workload-generating clients are in one of two states—high or low—with transitions between the two states governed by a simple Markov process. The clients in high state generate requests at high frequency and vice versa, thus introducing burstiness into the workload. However, the proposed model does not increase popularity of individual objects and one cannot easily change the steepness and magnitude of the bursts.

The papers [18, 20] characterize workload for storage systems at disk-level, but do not analyze volume or data spikes. Finally, many researchers in the networking community have focused on characterizing and modeling *self-similarity* in network traffic [16] (burstiness over a wide range of time scales) or anomalies in network traffic volume [21]. All of these efforts, however, concentrate on workload volume and not on data popularity.

3. METHODOLOGY

Although obtaining real workload traces of public Internet sites is extremely difficult, we were able to obtain four traces and some partial data from a fifth source. Since our goal is to characterize data spikes as well as volume spikes, for each trace we identify the fundamental underlying *object* served by the site and extract per-object access information from the logs. For example, for a site such as Wikipedia or a photo sharing site, the object of interest can be represented by a static URL; for a site that serves dynamic content identified by embedding parameters in a common base URL, such as `http://www.site.com?merchant_id=99`, the object would be identified by the value of the relevant embedded parameter(s). Because we want to characterize sustained workload spikes lasting at least several minutes, we aggregate the traffic into five-minute intervals. We are also forced to aggregate because of the low workload volume in some of the datasets.

In the rest of the paper, *workload volume* represents the total workload rate during a five-minute interval. *Object popularity* represents the fraction of workload directed at a particular object. A *volume spike* is a large sustained increase in workload volume in a short period of time, while a *data spike* is a significant shift in popularity of individual objects. We provide no formal definition of a volume and data spikes and identify the spikes manually by visual inspection. *Hotspots* or *hot objects* refer to objects in the system whose workload increases significantly. We provide a formal definition in Section 4.3.

3.1 Datasets

Our datasets are described below and summarized in Table 1. The spikes are summarized in Table 2.

World Cup 1998. Web server logs from week four of the World Cup 1998 [10]. The logs contain time of individual requests along with the file that was accessed. We use the path of the file as the name of the object.

UC Berkeley EECS Website. Web server logs from the UC Berkeley, EECS Department website hosting faculty, student, class web pages, technical reports and other content. The logs contain time of individual requests along with the file that was accessed. We use the path of the file as the name of the object.

Ebates.com. Web server logs from Ebates.com used in [14]. The logs contain the time of individual requests along with the full URL. We use the value of the *merchant_id* URL parameter as the object. The merchants are represented as arbitrary integers.

Wikipedia.org. Wikipedia.org periodically publishes the hourly number of hits to individual Wikipedia articles [8]. We use the individual articles as objects. While Wikipedia is one of the most popular Web sites, the disadvantage of this dataset is that it only contains hourly aggregates of workload instead of individual requests. We have only two days of data surrounding the Michael Jackson spike.

Partial sources of data from Twitter. We also use data from [3], which contains plots of workload at Twitter during the inauguration of President Obama, and from [7], which contains hourly popularity of the top 20 trends on Twitter in 2009. Because both of these are incomplete sources of data, we do not use them in our full analysis. However, they still demonstrate important properties of workload spikes.

4. ANALYSIS OF REAL SPIKES

In this section we characterize the important aspects of a workload spike. Since the change in workload volume and change in data popularity could occur independently, we analyze them separately. We first analyze the change in workload volume (Section 4.1) and object popularity during a normal period before a spike (Section 4.2). Next we define data hotspots (Section 4.3), analyze change in their popularity during a spike (Section 4.4) and examine their spatial locality (Section 4.5). In Section 5 we propose quantitative metrics that capture the phenomena we measured.

4.1 Steepness of volume spikes

Figure 1 shows the increase in workload volume for each spike normalized to the beginning of the spikes.[1] We observe that the workload increase during the spikes varies significantly. The workload increased by a factor of almost four in the EECS-photos and WorldCup spikes, while it stayed almost flat in the Ebates, Above-the-Clouds, and Michael Jackson spikes. The steepness also varies; during the EECS-image and WorldCup spikes, the workload reached its peak in 20 and 60 minutes, respectively.

4.2 Static object popularity

Much has been written about objects' popularities in the World Wide Web following a Zipf law [27, 23], which states that the popularity of the ith most popular object is proportional to $i^{-\alpha}$, where α is the power-law parameter. Power-law distributions are linear when object popularities are plotted against their ranks on log-log scale. Figure 2 shows the object popularity distribution for our four datasets along with a line fit to the data.[2]

If the data were from a power-law distribution, the dashed lines would coincide with the black curves. We observe that this occurs in none of the four datasets: either the most popular objects are not as popular as the power law predicts, or the tail of the distribution falls off much faster than the power law predicts. The EECS data come closest, followed by the Wikipedia articles, while the Ebates and WorldCup datasets show the most significant differences. (The flat region in the EECS dataset corresponds to a set of faculty photos periodically displayed on a digital board; their popularities are thus almost identical.)

Table 1 summarizes the fraction of traffic that corresponds to the most popular 1% and 10% of objects in all four datasets and shows significant differences among the datasets.

[1] We do not have raw workload data for the Twitter spikes.

[2] We do not have popularity data for all the topics on Twitter.

dataset	granularity	objects	# objects	% traffic to top 1% objects	% traffic to top 10% objects
World Cup 1998	requests	files	29,475	88%	99%
EECS website	requests	files	338,276	83%	93%
Ebates.com	requests	merchant_id	877	38%	82%
Wikipedia	hourly	articles	3,160,492	32%	67%
Twitter [3]	aggr. workload	N/A	N/A	N/A	N/A
Twitter [7]	top popularity	topics	N/A	N/A	N/A

Table 1: List of datasets and their various statistics

dataset	name of spike	description
WorldCup	WorldCup	spike at the beginning of a soccer match
EECS	Above-the-Clouds	Above the Clouds [11] paper is mentioned on Slashdot.org
EECS	EECS-photos	spike in traffic to about 50 photos on a student's page
Ebates	Ebates spike	change in popularity of merchants in ad campaign
Wikipedia	Michael Jackson	spike after the death of Michael Jackson
Twitter [3]	inauguration	inauguration of President Obama
Twitter [7]	top 20 trends	top 20 trends on Twitter according to trendistic.com

Table 2: List of spikes

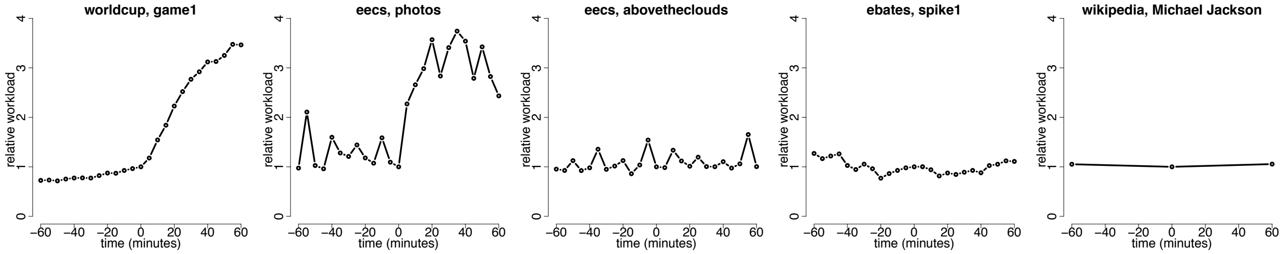

Figure 1: Workload volume profile of spikes normalized to the volume at start of spike at time 0.

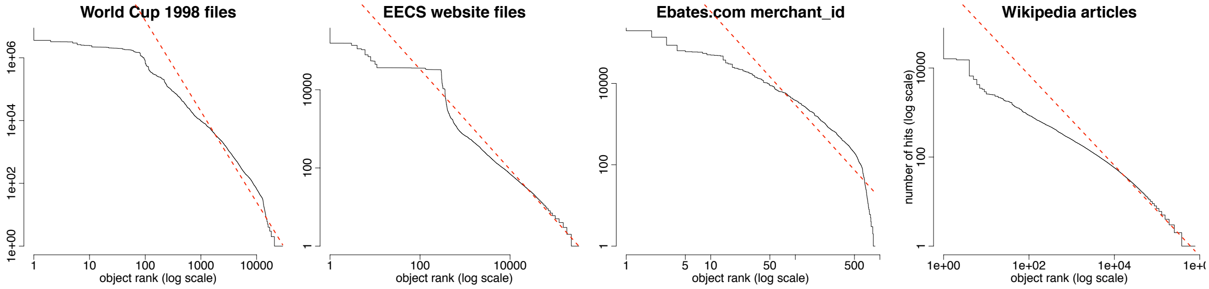

Figure 2: Object popularity distribution plotted on log-log scale along with the best-fit Zipf curve (red, dashed line). x-axis represents the rank of an object, y-axis the number of hits to that object. The most popular objects receive significantly fewer hits than the Zipf curve would predict.

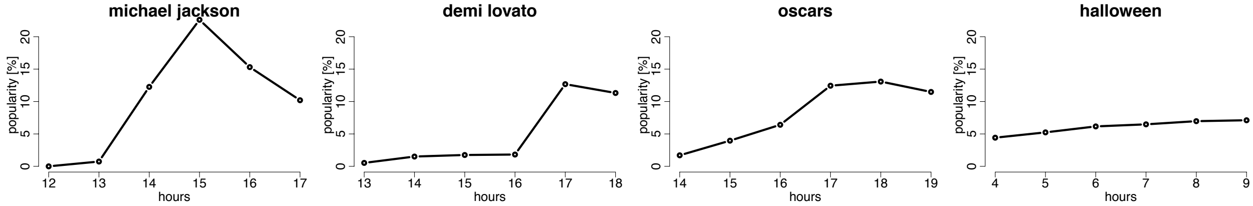

Figure 3: Topic popularity (percent) during 4 of the top 20 Twitter spikes. Spike steepness varies significantly.

4.3 Detecting data spikes

To analyze the change in object popularity during a spike, we first define the objects that are construed as *hotspots*. We consider an object to be a hotspot if the change of workload to this object during the first hour of the spike is significant compared to a *normal period* before the spike. We define the normal period to be one day before the spike, as it represents a typical interval during which various workload patterns repeat. We analyze the workload increase during the first hour of the spike since all of our spikes are longer than an hour and it is a long enough period to observe data hotspots.

More formally, let $\Delta_{i,t}$ be the change in workload to object i between time t and $t+1$ hour, and $\Delta_{max,t} = \max_i \Delta_{i,t}$ be the maximum change in workload to any object at time t.

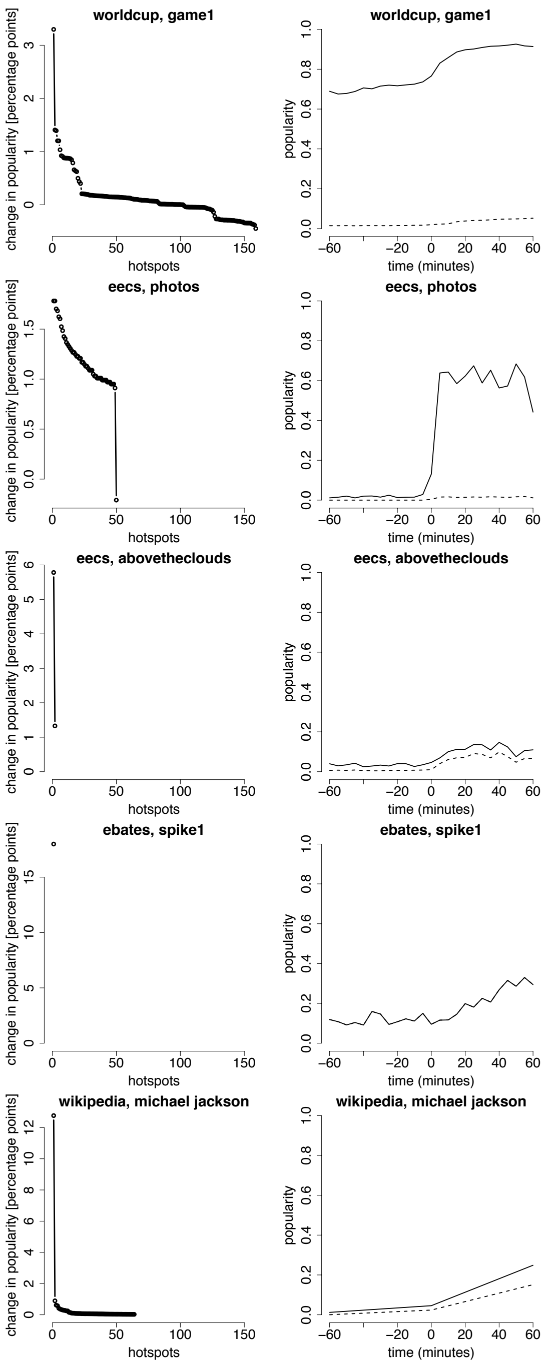

Figure 4: Left: difference in popularity of all hot objects. Right: popularity of the *single hottest* object (dashed) and *all hot* objects (solid) during spike.

We define a threshold D as the median of $\Delta_{max,t}$ for times t during the normal period before the spike. Intuitively, D represents the typical value of maximum workload increase during the normal period. The median is a robust statistic that is not affected by outliers; using the median thus accounts for normal variability of the $\Delta_{max,t}$ statistic. We consider object i to be a hotspot if (and only if) $\Delta_{i,T_s} > D$, where T_s is the start of the spike. For each hotspot i, we define the change in popularity as the difference between the popularity of i one hour after the start of the spike and the popularity of i at the beginning of the spike.

4.4 Change in object popularity during spikes

In this section we analyze the magnitude of changes in popularity of the hot objects during the spike and the temporal profile of these changes. The first column in Figure 4 shows the change in popularity of all the hot objects between the normal period before the spike started and during the spike. We observe a significant shift in popularity during the spikes. For example, the popularity of the Michael Jackson article increased by 13 percentage points during that spike. During the EECS-photo spike, the popularity of the 50 student photos increased by at least 1 point.

We notice two important differences between the spikes. In the Ebates, Above-the-Clouds, and Michael Jackson spikes, the number of hot objects is very small, while the remaining two spikes have more hot objects. However, the number of hot objects is very small compared to the total number of objects in each dataset. Also, there are significant differences in the magnitude of change among the hot objects. In the Ebates and Above-the-Clouds spikes, a single hot object is responsible for most of the shift in popularity. However, in the EECS-photo and WorldCup spikes, the change in popularity is spread out more uniformly among the hot objects. Finally, the popularity of some of the hot objects actually decreased during a spike (WorldCup and EECS-photos spikes).

The second column in Figure 4 shows the popularity of the hottest object and the total popularity of all the hot objects. Figure 3 shows the popularity of the hottest topic in four of Twitter spikes.[3] Notice that the steepness of the popularity varies significantly.

4.5 Spatial locality of data spikes

The response to data spikes is affected by the spatial locality of the hotspots. Some storage systems such as Amazon Dynamo [17] handle data in blocks and can only replicate whole blocks. Other systems such as SCADS [12] support range queries and thus store the objects in logical order. In such systems, if all the hot objects live on a single server or are logically close to each other, one could react to a spike by replicating a very small fraction of the data. However, if the hot objects are spread over many servers, one would have to copy data from multiple servers.

While we do not know how the individual objects are stored, to evaluate the spatial locality of data hotspots, we order them lexicographically by their name. This ordering preserves the logical structure of the objects; for example, files in the same directory will stay close to each other. Figure 5 shows the spatial locality of the hotspots. The x-axis represents the individual ordered hotspots and the y-axis represents the relative location of a hotspot in the range of all objects. Flat regions in these graphs thus represent hot objects located very close to each other.

We observe that the WorldCup and EECS-photos spikes have significant spatial locality. In the WorldCup spike, we notice three large clusters of hotspots. In the EECS-photos spike, all the objects except one form a contiguous range.

4.6 Summary

Before presenting our proposed quantitative characteriza-

[3] We do not have popularity data for the inauguration spike.

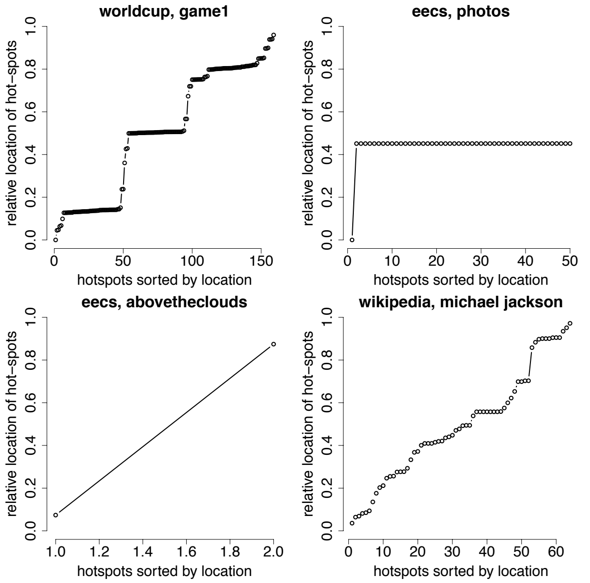

Figure 5: Spatial locality of hotspots. X-axis represents hotspots sorted lexicographically, y-axis shows their relative location. We do not show the Ebates spike that has only a single hotspot.

tion of spikes, we summarize the key observations emerging from our analysis:

In all spikes, the number of hot objects is a small fraction of the total number of objects in the system. We believe this is because the workload and spikes are generated by human users whose actions are independent of the total size of the dataset. In other words, even if Wikipedia had ten times more articles, Michael Jackson's death would likely still result in Wikipedia users visiting just the single article, not ten times as many articles. *Therefore, we characterize the number of hot objects as an absolute number instead as a fraction of all objects.*

Aside from this commonality, there is tremendous variation in volume and data spikes. The duration of the spikes varies from hours to days, the increase in workload volume ranges from insignificant increases to factors for five, and the slope of workload volume and popularity of hottest objects also varies drastically. This supports the claim that methods for automatic scaling of web applications cannot be tested on a single workload, but rather must be evaluated across a range of different and realistic spikes.

Object popularity is not necessarily Zipfian. The popularity distributions tend to be heavy-tailed, but there are significant differences from power-law distributions.

Anticipated spikes are both gentler and smaller than unanticipated spikes. In the top 20 Twitter trends, *surprising events* shoot up quickly from zero to peak, exhibiting 1-hour changes in popularity close to the maximum popularity. Examples include celebrity deaths (Michael Jackson, Patrick Swayze, Brittany Murphy, and Billy Mays) or other unexpected sensations (Nobel Prize for President Obama or the Balloon Boy). On the other hand, *anticipated events* such as holidays or other well known events (the Oscars, Eurovision) slowly build up their popularity.

5. QUANTITATIVE CHARACTERIZATION

In this section we define a set of quantitative metrics to describe the important aspects of a workload and data spike based on observations in Section 4. We present the results in Tables 3 and 4.

Number of hot objects represents the number of objects identified as hotspots (see Section 4.3).

Normalized entropy of the popularity of hot objects represents the skewness of the change in popularity of hot objects. Letting $c_1, \ldots, c_n$ denote the change in popularity of n hot objects (column 1 in Figure 4), the normalized entropy H_0 is defined as follows (see Appendix):

$$H_0 = -\sum_{i=1}^{n} c_i \log_2 c_i / \log_2 n, \quad 0 \le H_0 \le 1.$$

H_0 close to 1 corresponds to distributions where all the hot objects have almost equal increase in popularity, such as in the EECS-photos spike (column 1, row 2 in Figure 4). H_0 close to 0 corresponds to distributions where a single hot object has most of the popularity, such as the Michael Jackson spike (column 1, row 5 in Figure 4).

Time to peak represents the time from the start to the peak of the spike. We first manually identify the start of the spike by visually inspecting the workload profile and changes in popularity of the hot objects. Then we find the peak time as time when the object with the most significant change in popularity (object h) reached the maximum popularity.

Duration of the spike is the time between the start of the spike and its end. We define the end of a spike to be the time when the popularity of object h returns to the level seen before the spike. We do not use the workload volume to identify the peak and end of the spike since in some cases the increase in workload volume is not large enough and using it to identify start and end of spike would be unreliable. However, the popularity profile of object h is a clear signal of the peak and end of spikes.

Relative increase in workload volume represents the workload volume increase at the peak of the spike relative to the workload at the start of the spike.

Max slope captures the increase in workload volume during the steepest part of the spike. We first normalize the workload to the start of the spike, find the steepest part, and represent its slope as *change in relative workload per minute*. Thus, a max slope of 7%/min means that during the steepest part of the spike, the workload volume was increasing by 7% every minute (relative to the workload at the start of the spike). We note that this only captures the slope of the workload volume, not the slope of workload to the individual hot objects.

Spatial locality represents the degree to which the hot objects are clustered. Let $l_1, \ldots, l_n$ represent the relative location of hot objects in the logical address space of all objects sorted lexicographically. For a given value of δ, we consider two objects i and j to be *close* if $|l_i - l_j| < \delta$. Finally, let C_δ be the minimal number of clusters of hot objects, such that each cluster contains only objects that are close. Intuitively, if we need few clusters to cover all hot objects, the hot objects exhibit high spatial locality, and vice versa. We thus define spatial locality as

$$L_\delta = 1 - \frac{C_\delta - 1}{n - 1}, \quad 0 \le L \le 1.$$

In Table 3, we report values for $\delta = 0.1\%$.

parameter	WorldCup	Above the Clouds	EECS photos	Ebates	Michael Jackson	inauguration
number of hot objects	159	2	50	1	64	n/a
0.1%-spatial locality	0.639	0.0	0.980	n/a	0.254	n/a
normalized entropy	0.765	0.695	0.990	0.0	0.478	n/a
time of peak [min]	105	40	55	130	60	n/a
duration [min]	190	800	1000	7000	>1560	n/a
relative increase in aggr.	4.97	0.97	2.43	1.13	1.05	≈ 5
max slope [%/min] of aggr.	8.7	7.0	25.4	2.9	0.09	≈ 140

Table 3: Characterization of spikes using metrics described in Section 5. The metrics for the spike *inauguration* were estimated from a post on Twitter's blog [3].

spike	change in pop. in 1 hour [pp]	maximum pop. [%]
WorldCup	1.14	1.53
EECS photos	1.47	1.48
Above the Clouds	5.77	6.42
Ebates	12.50	55.63
Michael Jackson	12.76	15.12
michael jackson (twitter)	11.52	22.61
kanye	12.64	14.39
oscars	6.03	13.08
balloon (boy)	11.76	12.89
demi lovato	10.86	12.68
thanksgiving	2.91	12.09
jonas brothers	10.42	12.06
fireworks (July 4th)	5.22	11.72
eurovision	5.36	9.93
happy easter	1.96	9.78
patrick swayze	9.67	9.67
aplusk	5.16	8.50
obama (nobel)	8.09	8.39
iphone	5.80	8.27
facebook	6.03	8.09
brittany murphy	7.60	8.01
lakers	4.47	7.56
yankees	4.62	7.14
halloween	0.92	7.11
billy mays	5.09	5.85

Table 4: Summary of change in popularity and maximum popularity of the single most popular object in each spike. The first five spikes correspond to spikes described in Table 2. The following 20 spikes are based on hourly data from the *Top 20 Trending Topics of 2009* [7].

Change in popularity in 1 hour is the biggest change in popularity of the hottest object during one hour in percentage points (pp); i.e., a change from 2% to 10% is 8pp.

Maximum popularity represents the peak popularity of the hottest object achieved during the spike.

Together, these metrics capture the most important aspects of a spike. The number of hot objects, normalized entropy, change in popularity, and maximum popularity characterize the hot objects, while the rest of the parameters characterize the changes in workload volume. We present partial results for data from [3, 7]. For the *inauguration* spike we only have approximate values of the relative increase in workload volume, while for the top 20 trending topics we only have the changes in their popularity over time and not the workload volume.

6. WORKLOAD MODEL

In this section we build on a baseline workload model for *normal* periods without any spikes or data hotspots, extending this model to support workload spikes and changes in popularity of objects. We then extend the model to support a notion of data locality. We describe the parameters of the model, explain how they control the various spike characteristics (see Table 5), and show that our model can generate workload similar to the spikes described in Section 4.

We emphasize that it is not our goal to create a complex workload model that imitates every detail of the real traces analyzed in Section 4. Instead, our goal is a model that is simple to use and captures the important aspects of volume spikes and data spikes that emerged from our analysis. To this end we use the Pitman-Yor stick-breaking process [24] to generate popularity profiles, the Dirichlet distribution to model popularities during data hotspots, and the Chinese Restaurant Process [9] to generate locations of hotspots (clusters). In the following sections we explain, justify, and validate these choices.

6.1 Model of normal workload

A model of normal workload consists of a workload volume time series U and a popularity profile for individual objects. We do not focus on the workload volume time series U, as this can be based on other studies of Web workloads, such as [10]. Instead we concentrate on the novel challenge that arises in data spike modeling: creating a realistic characterization of object popularity. We note at the outset that the popularities, while drawn from a distribution, are held constant during workload generation. The transformation from popularity profile to workload, however, is also stochastic, and this stochastic process is responsible for capturing the variance of the workloads of individual objects.

We showed in Section 4.2 that the popularity profiles in our datasets do not tend to follow the Zipf law often assumed for Web data. Therefore we instead use *stick-breaking*, a probabilistic modeling method used to generate heavy-tailed distributions. The idea is straightforward—one starts with stick of unit length and breaks off a piece according to a draw from a beta distribution. This procedure is applied recursively to the remaining segment of the stick. The result is a set of increasingly small pieces of the stick whose lengths are taken to represent the popularity profile of our objects.

The particular form of stick-breaking we use is the *Pitman-Yor process* [24]. This process, denoted $\mathcal{PY}(d, \alpha)$, is parameterized by a *discount parameter* $d, 0 \leq d < 1$, and a *concentration parameter* $\alpha > -d$. To generate the popularity profile of n objects, we first generate n beta random variables as follows:

$$\beta_i \sim \text{Beta}(1-d, \alpha + id), \quad i = 1, \ldots, n, \quad 0 \le \beta_i \le 1.$$

Next, we set the length of the first piece of stick as $\pi_1 = \beta_1$, and set the lengths of the remaining pieces as follows:

$$\pi_i = \beta_i \prod_{l=1}^{i-1} (1 - \beta_l), \quad i = 2, \ldots, n$$

Intuitively, the length π_i of the i'th piece is the length of the stick after breaking off the first $i-1$ pieces multiplied by β_i. It turns out that the lengths of the pieces generated by this procedure follow a power-law distribution with parameter $1/d$ in expectation. However, an individual stick-breaking realization generates π_i's whose lengths drop off faster than a power law. This behavior more closely matches the behavior we observed in real workload traces in Figure 2.

To generate a popularity profile for n objects with parameter a, we first generate a vector $(\pi_1, \ldots, \pi_n)$ using the $\mathcal{PY}(1/a, 0.5)$ process, and we then set the object popularities, $(b_1, \ldots, b_n)$, as a random permutation of the π_i's.

6.2 Modeling volume spikes and data spikes

To add a spike to a normal workload, we need to increase the workload volume during the spike and create a new object popularity distribution with a higher popularity for the hotspots. This process is parameterized by the following parameters: t_0, t_1, t_2, and t_3 represent the times when the spike starts, when it reaches its peak, the end of peak period with flat workload, and the end of the spike, respectively. M represents the magnitude of the spike, or relative increase in workload volume. N represents the number of hotspots and V the variance of hotspot popularity (which determines the normalized entropy described in Section 5). We first describe how we adjust the workload profile and then the object popularity during a spike.

The four t_i parameters and the magnitude parameter define the change in the workload profile. The workload profile is multiplied by a factor c_t to obtain the new workload profile during a spike. c_t is 1.0 for times t before t_0, increases linearly between t_0 and t_1 up to M, stays flat between t_1 and t_2, decreases linearly to 1.0 between t_2 and t_3, and is 1.0 after t_3 (see Figure 6). We model the change in workload volume using a piece-wise linear factor to keep the model simple and minimize the number of parameters. Figure 1 also justifies this decision as most of the workload profiles can be approximated well using a piece-wise linear function.

To generate object popularity during a spike, we take the baseline popularity $B = (b_1, \ldots, b_n)$ and adjust it by putting more weight on the hotspots. In particular, we construct a vector $H = (h_1, \ldots, h_n)$ that represents the popularity of only the hotspots during a spike. The final object popularity profile at time t is $P_t = (1 - c_t^*)B + c_t^* H$—a weighted average of the baseline popularity and the increased popularity of the hotspots. For times t before and after the spike, $c_t^* = 0$, so that the object popularity will be the baseline B. During the spike, c_t^* is adjusted in a piece-wise linear fashion (similar to c_t) such that at the peak of the spike $c_t^* = (M-1)/M$. This guarantees that as the workload volume increases by a factor of M, all the additional traffic will be directed towards the hotspots using distribution H. For example, for $M = 3$, 2/3 of the workload at the peak will be directed to hotspots.

We generate the popularity of hotspots H as follows. We first pick locations $l_1, \ldots, l_N$ of N hotspots uniformly at random; H has non-zero popularity only at these locations. The values of popularity of the hotspots, $h_{l_1}, \ldots, h_{l_N}$, are obtained as a sample from a Dirichlet distribution (see Appendix): $(h_{l_1}, \ldots, h_{l_N}) \sim \text{Dir}(\alpha_1, \ldots, \alpha_N)$. A sample from a Dirichlet distribution results in N non-negative numbers that sum to 1 and thus represent valid object popularities.

As described in Section 4, different spikes have different values of normalized entropy of the hot object popularity distribution. In the workload model, we control the entropy of the hot object popularity distribution by adjusting the variance of the Dirichlet distribution. We set the parameters of the Dirichlet distribution as follows:

$$\alpha_i = \frac{N - 1 - VN^2}{VN^3}.$$

With this setting of the α_i parameters, the expected value of all h_{l_i} is equal to $1/N$ and variance is equal to V; $\text{E}[h_{l_i}] = 1/N, \text{Var}[h_{l_i}] = V$. High variance V results in a distribution with high entropy and vice versa, which allows us to control the normalized entropy of the hot objects. This process is illustrated in Figure 7.

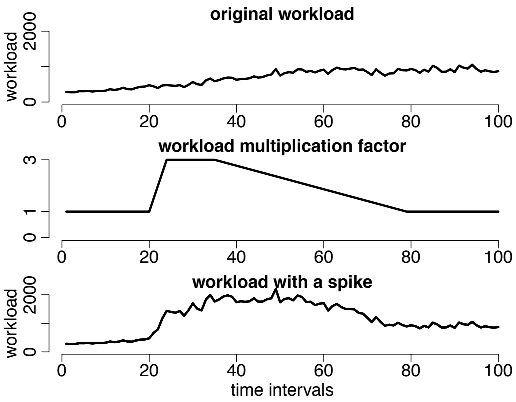

Figure 6: Top, an example of a normal workload profile without a spike. Center, workload multiplication factor c_t with the following parameters: $t_0 = 20$, $t_1 = 25$, $t_0 = 35$, $t_0 = 80$, $M = 3.0$. Bottom, the workload profile during a spike is obtained as a multiplication of the normal profile (top) and c_t (center).

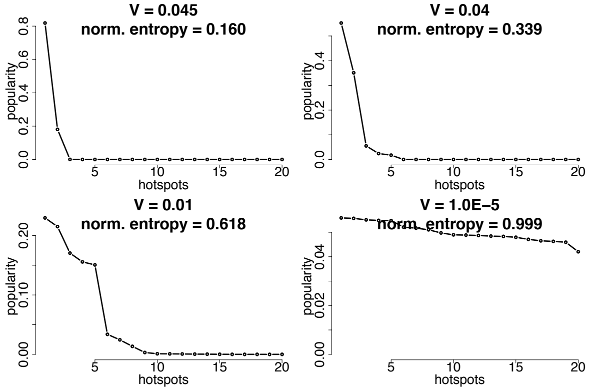

Figure 7: Popularity of $N = 20$ hot objects for different values of the variance parameter V resulting in different values of entropy.

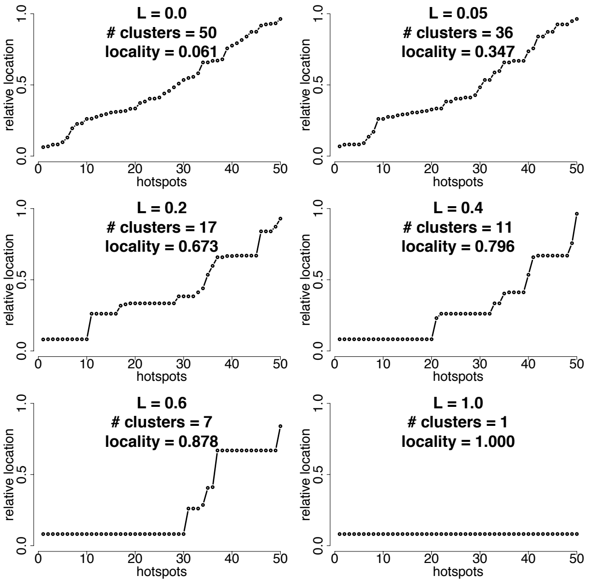

Figure 8: Locations of hot spots generated using a Chinese Restaurant Process. The title of each plot shows the L parameter used, number of clusters, and the resulting 0.1%-spatial-locality characteristic.

6.3 Adding spatial locality to data spikes

The model of a spike described in Section 6.2 picks the locations of the hot objects uniformly at random. However, as we demonstrated in Section 4.5, hot objects often exhibit significant spatial locality. Here we describe a process that clusters the locations of the hotspots in two steps.

We first create clusters of hotspots using an iterative process known as the Chinese Restaurant Process (CRP) [9], a statistical model of clustering that is parameterized by a single parameter $0 \leq L \leq 1$. We start with a single cluster that contains the first hotspot. All the subsequent hotspots are either assigned to a new cluster, with probability proportional to $1/L - 1$, or pick an existing cluster i, with probability proportional to k_i, which is the number of hotspots in cluster i. The parameter L thus determines the number of clusters; in particular, large values of L imply a low probability of starting a new cluster and thus a smaller number of clusters. Given N hotspots, the expected number of clusters turns out to grow logarithmically as $O((1/L - 1) \log N)$.

Second, we pick the location of cluster i, l_i, uniformly at random from all the available objects, and mark objects $l_i, \ldots, l_i + k_i - 1$ as hotspots. After selecting the locations of hot objects, we assign their probabilities as described in Section 6.2. Figure 8 shows sample locations of hot objects for different values of the locality parameter L. Since CRP is a stochastic process, the cluster sizes vary across multiple runs with the same L parameter. If we wish to control the variance of the cluster sizes we can run the CRP multiple times and average the sizes of clusters.

6.4 Summary of workload model

To model a workload without spikes, we select the number of active users at each point in time, $u_1, \ldots, u_T$, from a section of an existing workload trace. The popularity of individual objects, b_i, is constructed using the Pitman-Yor stick-breaking process. We add a volume spike by multiplying the number of active users by a piece-wise linear function (Figure 6). Finally, we add a data spike by selecting N hotspots with a particular spatial locality and entropy.

The model parameters are summarized in Table 5. For most of the parameters, the effect on the various spike characteristics is straightforward. In particular, the magnitude and number of hot objects are directly the spike characteristics. The spike coordinates $t_0, \ldots, t_3$ determine the maximum slope and duration. As for the normalized entropy and the spatial locality of the hotspots, these are controlled by the V and L parameters. As we show in Figures 7 and 8, there is a mapping between these quantities such that we can control the normalized entropy and spatial locality of the hotspots by the choice of V and L. In summary, for any values of the spike characteristics (see Table 3), we can find model parameters that generate such a spike.

6.5 Model validation

In this section we validate our workload model in two ways. First, we compare the variance of workload to individual objects in real traces with the variance obtained using our model. Second, we compare the workload volume and workload to the hottest object observed during the EECS-photos spike with workload generated from our model.

Short-term volatility of object popularity

As described above, the popularity of an object i that is not a hotspot is b_i and is thus independent of time t. One might be concerned that this implies that the actual workload received by object i over time will either be constant or have variance much smaller than observed in real traces. Here we demonstrate that the variance of workload generated by our model is comparable to variance in real workloads.

Generating k requests to n objects with popularities of $B = (b_1, \ldots, b_n)$ is equivalent to taking k samples from a multinomial distribution $\mathrm{Mult}(B)$. The expected value and variance of the number of hits to object i, w_i can be expressed as follows: $\mathrm{E}[w_i] = kb_i$, $\mathrm{Var}[w_i] = kb_i(1 - b_i)$. In Figure 9, we compare the variance observed during a normal workload period before the WorldCup spike with variance of a multinomial distribution. We see that the variance of a multinomial distribution closely matches the variance of most of the objects.

Comparing real and generated workload

To compare real and generated workload during the EECS-photos spike, we set the model parameters such that the spike characteristics match values presented in Table 3 and generate requests to objects based on the workload volume. In Figure 10 we compare the workload volume and the workload to the hotspot with the largest increase in workload during the spike. We note that while the generated curves do not exactly match the observed workload, the overall profile is very similar. We reiterate that our goal was not to replicate every detail of the real workloads, but only to capture the most important aspects of the spikes.

7. WORKLOAD GENERATION

We built a closed-loop workload generator in which a single user is simulated by a single client thread that creates and executes requests in a loop, a common design for workload generators [1, 6]. Each thread selects a request type (such as read or write), selects request parameters using our

parameter	contraint	description	spike characteristic	section
n	$n > 0$	total number of objects	-	6.1
U	$u_i \geq 0$	workload profile	-	6.1
a	$a > 1$	power-law parameter	-	6.1
N	$0 < N \leq n$	number of hot objects	number of hot objects	6.2
V	$0 < V < \frac{N-1}{N^2}$	variance of hot object popularity	normalized entropy	6.2
$(t_0,)\ t_1, t_2, t_3$	$t_0 \leq t_1 \leq t_2 \leq t_3$	spike coordinates	time of peak, duration, max slope	6.2
M	$0 < M$	magnitude of spike	relative increase in workload volume	6.2
L	$0 \leq L \leq 1$	spatial locality	0.1%-spatial locality	6.3

Table 5: Parameters of the workload model, their description, the corresponding spike characteristics that they control, and the section in the paper that describes the parameter in more detail. The first three parameters model the *normal* workload and thus do not correspond to any spike characteristic. The spike model has seven main parameters (N, V, t_1, t_2, t_3, M, and L) and the start of the spike (t_0) that has no effect on spike characteristics.

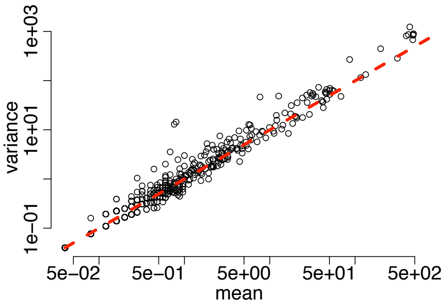

Figure 9: Mean vs. variance of workload to randomly selected 1000 objects in the WorldCup dataset during a two hour normal workload period in log-log scale. Each circle corresponds to a single object, the x-axis shows the mean workload to the object, the y-axis shows the variance of workload. The dashed (red) line represents the variance of these objects when they are sampled from a multinomial distribution.

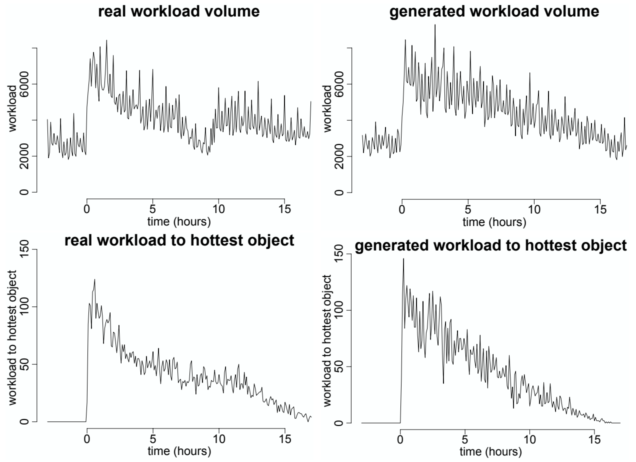

Figure 10: Comparison of real and generated workload for the EECS-photos spike. Top, actual and generated workload volume. Bottom, actual and generated workload to the hotspot with largest increase in workload.

model, sends the request to the system under test, waits for a response, and repeats.

We discretize time into T short intervals to create the *volume profile*, a time series $u_1, \ldots, u_T$ in which u_t is the number of users active during interval t. Each thread stores a copy of this profile; to simulate changes in the number of active users, a thread sleeps during interval t if its thread ID is larger than u_t. Otherwise, at time t, the workload generator selects object i with probability $p_{i,t}$.

We remark that while open-loop workload generation would allow specifying a particular workload rate (e.g., 2000 requests per second), a closed-loop workload generator only allows us to specify the number of active users; the resulting workload rate depends on the number of users, their think times, and the latency of the individual requests. For example, when 100 active users generate requests that take 100ms to execute, the workload rate would be 1000 requests per second. If the request latencies drop to 10 ms, the workload rate would increase to 10,000 requests per second.

Initializing the model for the experiment in Section 6.5 with 338276 objects and 50 hotspots took 14.5 seconds. Generating 20 hours of workload took 2.1 seconds.

8. EXTENDING THE MODEL

Longer-term trends in object popularity. As demonstrated above, the variance in workload to individual objects on a time scale of five minutes is well modeled by a multinomial distribution. However, on longer time scales (hours), we notice slow upward and downward trends in workloads to many objects. We can simulate these trends by perturbing the object popularity using the Dirichlet distribution. Let $(b_1, \ldots, b_n)$ be the baseline popularity of all objects in the system and let $(p_{1,t}, \ldots, p_{n,t})$ be the actual popularity of objects used at time t. To simulate trends in popularity that change every T minutes, we update $(p_{1,t}, \ldots, p_{n,t})$ every T minutes by sampling from the following Dirichlet distribution: $\mathrm{Dir}(Kb_1, \ldots, Kb_n)$. For any value of K, the expected value of p_i is b_i, which means that the sampled values of $p_{i,t}$ will oscillate around the baseline popularity b_i. However, the value of K affects the variance of $p_{i,t}$; smaller values of K imply larger variance and thus more significant trends in object popularity and vice versa (see Appendix).

Other extensions. The Chinese Restaurant Process could be used to add spatial locality even to the baseline object popularity b_i by first creating some number of clusters and then assigning the objects with the largest popularity to these clusters. Second, the current model assumes

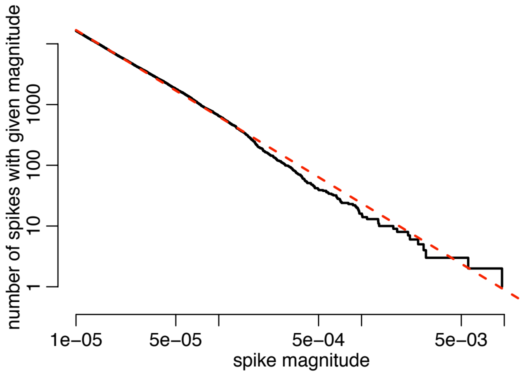

Figure 11: Spike magnitudes on log-log scale based on one month of Wikipedia data (solid line) and linear fit to the data (dashed line).

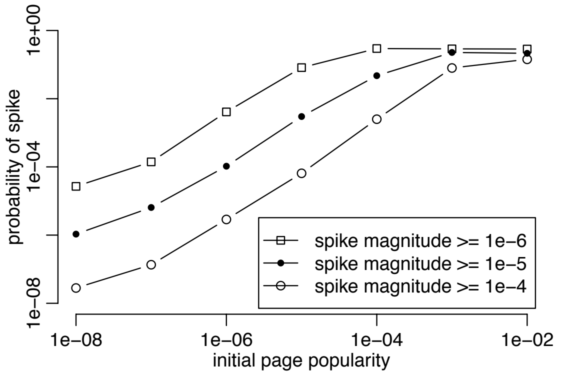

Figure 12: Probability of a spike of certain magnitude (10^{-6}, 10^{-5}, and 10^{-4}) for pages of different popularities on log-log scale.

that only the additional workload (with magnitude M) is directed at the hotspots. By increasing the values of c_t^* (see Section 6.2), we can redirect more workload at the hotspots. Finally, we can easily create a mixed spikes by creating new hotspots while other spikes are still in progress.

9. CONJECTURES

Below we present two conjectures based on initial analysis of spike magnitudes and frequencies; validating or refuting them is our future work. We studied the daily change in page popularities on Wikipedia during one month and treated spikes on individual pages during one day as separate events. The *spike magnitude* represents the increase in popularity of a particular page.

Spike magnitudes follow Zipf's law. Figure 11 shows that the spike magnitudes on Wikipedia during one month follow Zipf's law; there are many spikes with small magnitude and few spikes with large magnitude. By fitting the distribution (red line in Figure 11), we could predict the frequency of spikes of larger magnitudes.

More popular objects are more likely to become hotspots. In Figure 12, we first bin Wikipedia pages based on their popularity one day before the spike and compute the empirical probability of observing a spike of certain magnitude. We see that the more popular pages have higher likelihood of getting a larger spike.

10. CONCLUSION

As significant workload spikes are becoming the norm for popular Web sites such as Facebook or Twitter, it is important to evaluate computer systems using workloads that resemble these events. In many cases a surge in workload occurs together with the emergence of data hotspots, which has a crucial impact on stateful systems. Because it is very difficult to obtain realistic traces of such events, it is important to model and synthesize workloads that contain realistic spikes in workload volume and changes in data popularity.

In this paper we provide a methodology for characterizing the most important aspects of spikes in stateful systems. We characterize a spike in terms of changes in workload volume (maximum slope, relative increase, time to peak, and duration) and changes in data popularity (the number of hot objects, spatial locality, and normalized entropy).

We use this methodology to analyze five spikes in four datasets and observe two important facts. First, the number of hotspots in all the spikes is a very small fraction of the total number of objects in the system. Second, the spikes differ dramatically in all of the other characteristics. This suggests that there is no "typical" workload spike and computer systems should be evaluated using a wide range of spiky workloads.

Our workload model uses a small number of parameters to capture the most important aspects of spikes in stateful systems using well-known probability distributions and generative processes. We first model a typical workload with no spikes using a workload profile and a distribution describing the object popularity. Second, we add a volume spike by increasing the workload and add a data spike by increasing popularity of a small number of objects. Finally, we add a spatial locality of hotspots using a simple clustering process. The workload model serves as input to a closed-loop workload generator, where simulated clients select objects based on the generated object popularity that evolves over time.

Appendix: Terms and Models Used in Paper

Entropy. In information theory, entropy is a measure of the uncertainty associated with a random variable. Entropy H of a discrete random variable X with possible values $x_1, \ldots, x_n$ measured in *bits* is $H(X) = -\sum_i p(x_i) \log_2 p(x_i)$. Entropy achieves the minimum of 0 if one of the values x_i has probability 1 (no uncertainty). Entropy achieve the maximum of $\log_2 n$ if all the values have equal probability of $1/n$ (maximal uncertainty about the outcome of the random variable X). The normalized entropy with values between 0 and 1 is thus defined as $H_0(X) = H(X)/\log_2 n$.

Beta distribution. The beta distribution with two parameters α and β is a continuous probability distribution defined on an interval $(0, 1)$. Examples of the beta density are shown in Figure 13. The mean and variance are

$$\mathrm{E}[X] = \frac{\alpha}{\alpha+\beta}, \quad \mathrm{Var}[X] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}.$$

Multinomial distribution. The multinomial distribution is a probability distribution of the number of occurrences of events 1 through k in n independent trials, when the events have probabilities $p_1, \ldots, p_k$, and $\sum_i p_i = 1$. The mean and variance of number of occurrences of event i, X_i, are

$$\mathrm{E}[Xi] = np_i, \quad \mathrm{Var}[X_i] = np_i(1-p_i).$$

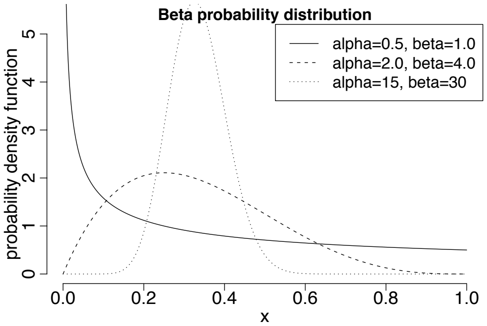

Figure 13: Probability density function of a Beta distribution for different α and β parameters. All three distributions have the same mean, $1/3$, but different variance.

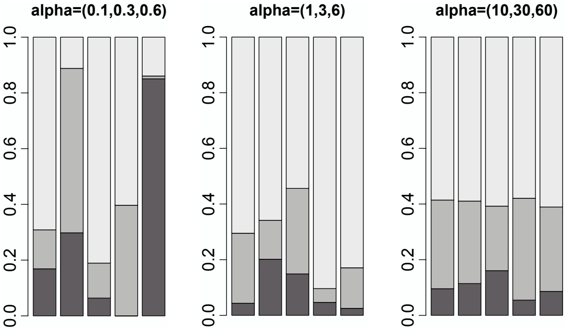

Figure 14: Five samples from three different Dirichlet distributions; $\mathbf{Dir}(0.1, 0.3, 0.6)$, $\mathbf{Dir}(1, 3, 6)$, and $\mathbf{Dir}(10, 30, 60)$. Each sample $(X_1, X_2, X_3) \sim Dir(\alpha)$ is represented by a single bar with three components with heights X_1, X_2, and X_3. Each of the distributions has the same expected values of the three components, 0.1, 0.3, and 0.6, but the variance varies significantly.

Dirichlet distribution. The Dirichlet distribution is a continuous multivariate probability distribution, denoted $\mathrm{Dir}(\alpha)$, parameterized by vector $\alpha = (\alpha_1, \ldots, \alpha_K)$, $\alpha_i > 0$. A single sample from a Dirichlet distribution is **vector** of numbers: $X = (X_1, \ldots, X_K) \sim \mathrm{Dir}(\alpha)$, where $\sum_i X_i = 1$. Samples from three different Dirichlet distributions are illustrated in Figure 14. The mean and variance of X_i are

$$\mathrm{E}[X_i] = \frac{\alpha_i}{\alpha_0}, \quad \mathrm{Var}[X_i] = \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)}.$$

11. REFERENCES

[1] Faban project web site. `http://faban.sunsource.net/`.

[2] Httperf project web site. `http://code.google.com/p/httperf/`.

[3] Inauguration Day on Twitter. `http://blog.twitter.com/2009/01/inauguration-day-on-twitter.html`.

[4] Jmeter project web site. `http://jakarta.apache.org/jmeter/`.

[5] Outpouring of searches for the late Michael Jackson. `http://googleblog.blogspot.com/2009/06/outpouring-of-searches-for-late-michael.html`.

[6] Rubis project web site. `http://rubis.ow2.org/`.

[7] Top 20 Twitter trends in 2009. `http://trendistic.com/\_top-twenty-trending-topics-2009/`.

[8] Wikipedia page counters. `http://mituzas.lt/2007/12/10/wikipedia-page-counters/`.

[9] D. Aldous. Exchangeability and related topics. In *Ecole d'Ete de Probabilities de Saint-Flour XIII 1983*, pages 1–198. Springer, 1985.

[10] M. Arlitt and T. Jin. Workload characterization of the 1998 World Cup Web site. Technical Report HPL-1999-35R1, HP Labs, 1999.

[11] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A Berkeley view of Cloud Computing. Technical Report UCB/EECS-2009-28, UC Berkeley, Feb 2009.

[12] M. Armbrust, A. Fox, D. Patterson, N. Lanham, H. Oh, B. Trushkowsky, and J. Trutna. SCADS: Scale-independent storage for social computing applications. In *CIDR*, 2009.

[13] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *SIGMETRICS*, 1998.

[14] P. Bodík, G. Friedman, L. Biewald, H. Levine, G. Candea, K. Patel, G. Tolle, J. Hui, A. Fox, M. I. Jordan, and D. Patterson. Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization. In *ICAC*, 2005.

[15] X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *IEEE Computer*, 2003.

[16] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic evidence and possible causes. *IEEE/ACM Transactions on Networking*, 1996.

[17] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *SOSP 2007*.

[18] A. Gulati, C. Kumar, and I. Ahmad. Storage workload characterization and consolidation in virtualized environments. In *VPACT*, 2009.

[19] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *WWW*, 2002.

[20] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda. Characterization of storage workload traces from production windows servers. In *ISWC*, 2008.

[21] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, 2004.

[22] N. Mi, G. Casale, L. Cherkasova, and E. Smirni. Injecting realistic burstiness to a traditional client-server benchmark. In *ICAC*, 2009.

[23] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy web server: Findings and implications. In *SIGCOMM*, 2000.

[24] J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Ann. Probab.*, 25(2):855–900, 1997.

[25] B. Schroeder and M. Harchol-Balter. Web servers under overload: How scheduling can help. *ACM Trans. Internet Technol.*, 6(1):20–52, 2006.

[26] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic provisioning of multi-tier Internet applications. In *ICAC*, 2005.

[27] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. In *SOSP*, 1999.