Predictability of Web-Server Traffic Congestion*

Yuliy Baryshnikov[1] Ed Coffman[2] Guillaume Pierre[3]
Dan Rubenstein[2] Mark Squillante[4] Teddy Yimwadsana[2]

1. Bell Labs, Lucent Technologies, Murray Hill, NJ 07974
2. Department of Electrical Engineering, Columbia University, New York, NY 10027
3. Department of Computer Science, Vrije Universiteit, Amsterdam
4. IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

Large swings in the demand for content are commonplace within the Internet. When a traffic hotspot happens, however, there is a delay before measures such as heavy replication of content can be applied. This paper investigates the potential for predicting hotspots sufficiently far, albeit shortly, in advance, so that preventive action can be taken before the hotpot takes place.

Performing accurate load predictions appears to be a daunting challenge at first glance, but this paper shows that, when applied to web-server page-request traffic, even elementary prediction techniques can have a surprising forecasting power. We first argue this predictability from principles, and then confirm it by the analysis of empirical data, which reveals that large server overloads can often be seen well in advance. This allows steps to be taken to reduce substantially the degradation of service quality.

1. Introduction

Traffic congestion in many settings, such as the automobile and airline transportation networks, is often predictable far enough in advance so that something can be done to mitigate its effects. The thesis of this paper is that this observation also applies to much of the page-request traffic on the Web, and that bursts of traffic can often be predicted shortly, yet adequately, before they reach worrisome dimensions. *Hotspots*, *flash crowds*, the *Slashdot effect*, and *storms* are among the colorful terms referring to these events. We shall use the term hotspot, but hasten to add that, for our purposes, the term need not connote "catastrophic" traffic; it need not mean anything more than traffic significantly higher than the norm.

Predicting hotspots, even with moderate accuracy, can be used to good effect in driving preventive measures such as replicating files/objects or migrating them to an overlay network of peer-to-peer server/cache cooperatives [18]. The results presented here demonstrate that simple, yet very effective algorithms can be developed for predicting bursts of traffic on the Internet, especially those occasional surges that bring web servers and routers to a virtual standstill.

Papers on hotspot properties generally refer to their spontaneity and their sudden, or nearly discontinuous ramp-up of traffic[1, 17]. But as noted in [9], this last property is only valid on large time scales, much larger than the time scale needed to react to an approaching hotspot. That a hotspot is visible in the distance can be explained in a number of ways. For example, there is the simple phenomenon of passing the word on a "hot" document. This process grows exponentially fast, but this in itself does not rule out adequate advance warning, as the initial growth is approximately linear. As another example, the time to gain access to the Internet varies from one person to another.

The goal of this paper is not so much the delivery of a full off-the-shelf hotspot predictor; rather, we argue the feasibility of such a predictor, and exhibit actual algorithms that demonstrate this feasibility. It is worth noting that, while the performance of our predictors is very good, even a moderate to poor predictor can offer substantial cost savings. For, in analogy with hurricane forecasting and recent IT disasters [11], the over-all cost of hotspots arriving without warning exceeds the aggregate cost of false alarms, i.e., the cost of preparing for hotspots that do not materialize.

This paper is structured as follows: Section 2 sets the context of the paper by briefly reviewing background literature. Section 3 examines the properties of the page-request traces studied in the paper and pays particular attention to autocorrelation functions, which have an intimate connection to predictability. Section 4 introduces linear least-squares extrapolation as the basis for the design of prediction algorithms. Scatterplots are presented which testify to the excellent performance of this technique.

At the heart of the paper, Section 5 then describes the structure of a parameterized prediction algorithm, the goal being a simplest algorithm with excellent prediction performance. Section 6 then carries out a case study in which the values of the parameters are estimated and the quality of resulting predictions analyzed. Section 7 complements

*This research was supported in part by NSF grant ANI-0117738.

this study with a sensitivity analysis focusing on window size and the granularity of data collection. There are a number of basic issues that must be dealt with satisfactorily if prediction is to be a viable process and the recommendations of this paper are to be followed. Among them are the quality (e.g., the age, representativeness, etc.) of the trace data, the generality (e.g., the scope of the applications) of any specific algorithm, and the complexity of the prediction process. These are covered by our conclusions in Section 8, the final and perhaps most important section of the paper.

2. Related Work

It is well known that building a load-based taxonomy of web server traffic is both an extremely important and an extremely challenging problem [9, 8]. The behavior of the traffic is shaped by a combination of so many technological, sociological, psychological (to name a few) factors that a quantification of even basic patterns reflecting this behavior would be a major breakthrough. Web traffic forecasting on a macroscopic level has been addressed previously by [3, 2]. These time scales show discontinuities and have been adopted for presentation of results that dramatize the traffic surges of hotspot-like conditions. The discontinuities disappear on the finer time scales of interest here.

Methods for alleviating the effects of hotspots are discussed in [5], while [9] studies hotspot properties determined by traffic patterns, and by client and file-referencing characteristics. They also introduce techniques that content-delivery networks can employ to adapt to large increases in load. While this solves a vital piece of the problem, the approach is reactive rather than predictive, in that adapting to hotspots is done at their arrival rather than in advance.

Several other schemes that aim at mitigating the effects of traffic hotspots include: the Oceano project at IBM [16], which provides a farm of additional server resources that can be used to service customer demand during periods of overload; systems that can dynamically distribute requests across multiple cooperating web servers [10, 14, 15, 20]; and those proposing to route Web requests through a peer-to-peer overlay, which allows for caching and load balancing in heavy load conditions [6, 12, 13, 19]. Again, these approaches are essentially reactive rather than proactive. They can make highly-desired content more easily accessible, but do not include any mechanism to identify the content for which interest is rapidly building. We believe that the hotspot predictor we present in this article would greatly complement these approaches.

Finally, we mention the traffic bursts described by Schwartz [17] for a web site tracking earthquake events, which are something of an acid test for hotspot prediction, as passing-the-word delays do not apply to nearly the same extent. The data are rather limited but they do support the conclusions of research reported here. The web server of the earthquake site experienced somewhat faster growth, as one might expect, but hotspots are also more easily distinguished from normal traffic levels. The data were for the $10/16/99$ Hector Mine earthquake and showed a web-server hotspot growing to its peak within about 15 minutes. The paper focused on methods of coping with these surges, such as increasing link capacity and using a reverse proxy server to help distribute traffic to replica servers. Our predictability results complement this work by indicating ways to respond earlier and more gracefully to earthquake events.

3. The request flow

3.1. Definitions

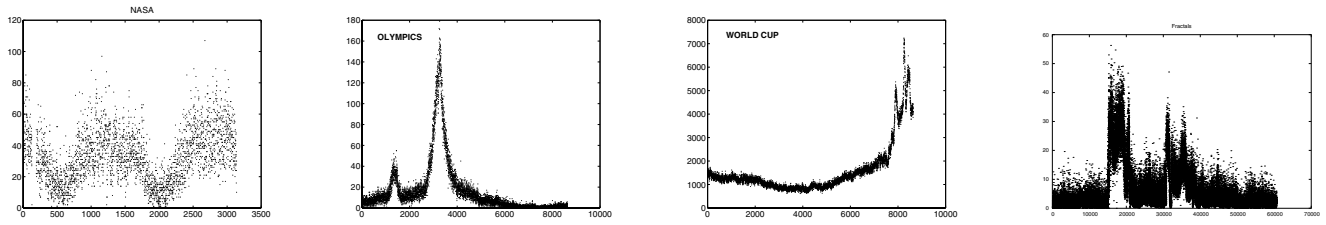
In the traces studied in this paper, a traffic observation, r_t, gives the number of page requests in a time slot, which we take as the unit of time; thus, $(t-1, t]$ is the t-th time slot. A typical time slot describing the granularity of data collection is ten seconds, but depending on the structure of the data, the efficiency required of the prediction algorithm, and the desired reaction time, the data may be accumulated into time slots of different length.

To be able to detect and predict hotspots, the notion of hotspot must be formally defined. For this, we define a *hotspot level* H, whose value depends on the application. H typically corresponds to the maximum capacity of the server handling the studied load. Ideally, we would define a hotspot as any period during which $r_t \geq H$. In general, however, request rate samples taken at small granularity have wide variations from sample to sample. Therefore, a workable hotspot definition will require a smoothing of the data over a sufficiently long interval. We define a parameter W_d as the interval over which load should be computed to determine whether it is experiencing a hotspot.

> *We say that traffic is experiencing a hotspot at time t if the volume of traffic over the last W_d time slots satisfies*
>
> $$\sum_{i \in [t-W_d, t]} r_i \geq H \times W_d$$
>
> *or equivalently, the average request rate over $[t - W_d, t]$ is at least H.*

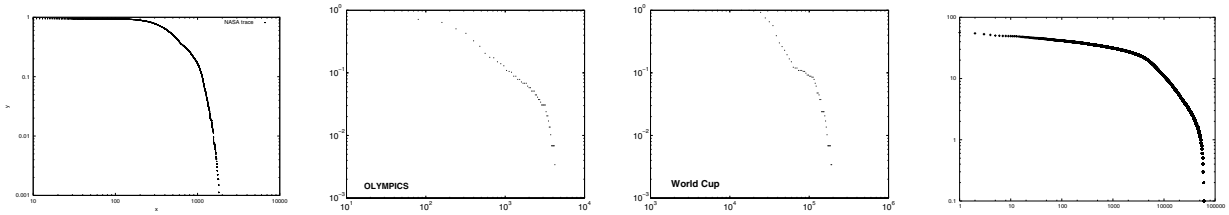
3.2. Request-flow analysis

In Figure 1, we show fragments of graphs of web server loads in four cases: A NASA server (August 1995); a Nagano Winter Olympics server(1998); a Soccer World Cup server (1998) and a slashdotted site (2004), referred to in this paper as 'Fractals' [4]. Time increases along the x-axis, and the y-axis plots the number of requests in successive 10-second slots. One sees immediately that these graphs are strikingly different; as we will see, the predictive powers of our algorithms also vary dramatically when



(a) NASA (b) Olympics (c) World Cup (d) Fractals

Figure 1. Fragments of four sample traces



(a) NASA (b) Olympics (c) World Cup (d) Fractals

Figure 2. Tail of the distribution for the number of page requests per time slot

applied to these traces. Figure 2 shows the tails of distributions of requests per time slot. Here, the number of requests that can occur within 10-second time slots is varied along the x-axis and the y-axis plots the probability that the number of requests in a slot exceeds this value. Note that the heavy-tail properties of these distributions, which are given in log-log plots, confirm the log-normal estimate of [8].

As is common in prediction problems, the performance of prediction algorithms depends strongly on the power spectrum (see [7] for example), that is, the absolute value of the Fourier transform of the time series. Figure 3 plots the autocorrelation functions (inverse Fourier transforms) of the power spectra for a sample of the four data sets we have studied. A point (x, y) in this graph means that the correlation between load values taken at time t and $t + x$ is equal to y. A high value of the autocorrelation function demonstrates that load values taken at time t and $t + x$ are highly correlated, which suggests that the load value at time t may be used to predict the load at time $t + x$. Typically, more rapidly falling autocorrelation functions are more "jittery" and more difficult from the predictive point of view.

As one can see, these four autocorrelation functions are quite different. One would clearly expect that, on the time scale of $5 - 30$ minutes, the performance of any reasonable prediction algorithm would be near optimal for the World Cup data, somewhat worse for the Olympic and Fractals data and much worse for the NASA traces. As we will see, this intuition is exactly right.

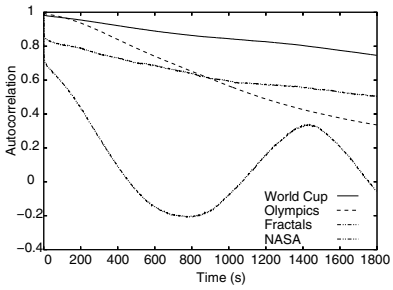


Figure 3. Autocorrelation functions

4. Linear-fit extrapolation

The prediction of Web server request flows should ideally be considered within the general framework of time series analysis, filtering, and prediction. While there exists an extended and rich theory servicing applications in such diverse areas as signal processing and econometrics, one would in fact be quite surprised to see that "standard" methods would suffice in the context of Internet traffic.

Even so, we started our investigation by testing what is arguably the simplest reasonable traffic prediction algorithm: linear extrapolation. As it turns out, it performs remarkably well. Indeed, as shown in the remainder of the paper, *our experience indicates that linear extrapolation offers an excellent balance between accuracy and simplicity/speed*. This section describes how linear-fit extrapolation is used to predict future loads, whether or not these loads constitute hotspots. The idea of linear-fit ex-

trapolation is to apply linear regression, at any time t, over a *prediction window* $W_p \geq 0$ containing the samples collected between $t - W_p$ and t. The regression allows to predict the load that should occur at time $t + \tau$, where $\tau \geq 0$ is called the *advance notice*. A prediction at time t is a mapping from the observations in the prediction window $[t - W_p, t]$ to a number $p_t \equiv p_t(\tau) \geq 0$ of requests predicted to appear in the interval $[t+\tau, t+\tau+1]$, τ time units in the future.

The *Linear Fit* (LF) mapping defines p_t and τ by the extrapolation of a least-squares linear fit, i.e., $p_t = f_t(t + \tau)$, where the coefficients of $f_t(s) = a_t s + b_t$ are chosen to minimize $\sum_{i=t-W_p}^{t} [f_t(i) - r_i]^2$, the mean quadratic deviation over the window $[t-W_p, t]$. The simplicity of LF leads to a very straightforward, linear-time implementation.

It is clear that, while linear predictors are extremely simple, they will have poor accuracy if the prediction window size is poorly adjusted to the parameters of burstiness governing the request flow. Thus, if the flow is prone to step-like bursts of activity, the linear predictor will tend to overestimate the flow at the beginning of the step; similarly, the linear predictor will underestimate the flow at the early stages of a sudden linear increase of activity. More generally, the structure of the power spectrum of the time series describing the request flow strongly influences the quality of a linear predictor. We return to this issue in Section 7.

Figure 4 gives a visual estimate of forecasting power. It plots $(r_{t+\tau}, p_t),\ t \geq 1$, with $\tau = 5$ min. Each point plotted at (x, y) therefore represents a slot containing x requests for which the predicted value issued 5 minutes in advance was y requests. The quality of the prediction is high when the points tend to be close to the diagonal. We see a striking correlation of the performance of the linear fit algorithm with the properties of the autocorrelation function: the World Cup and (to a lesser extent) the Olympics loads are well predicted, while the NASA load is very hard to predict with our method.

5. Hotspot prediction

The predictability shown by the scatterplots of the last section will now be exploited in the design of an effective hotspot prediction algorithm using LF extrapolation. The data used by the algorithm consists of a sequence of advance notices $\tau_t(H) \equiv \tau_t$ defined as the time remaining until an LF extrapolation of the traffic beyond the current time t hits the hotspot level H. These numbers lead to predictions only when positive and finite, i.e., when the linear fit of traffic in the window $[t - W_p, t]$ has a positive slope and intersects the hotspot level H at a time beyond t (See Figure 5).

As in the case of hotspot detection, prediction must react to *trends* over sufficiently large windows rather than a few isolated points. Also, predictions of hotspots far in the future tend to be unreliable, and there is no need to act on them in any case. Thus, for a given window size W_h and

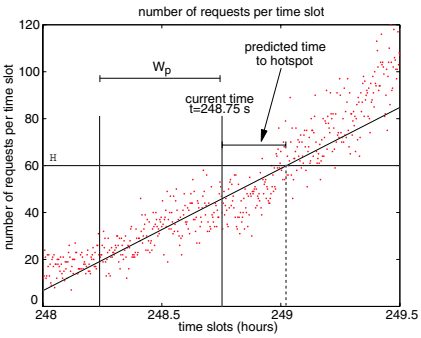


Figure 5. The definition of advance notice.

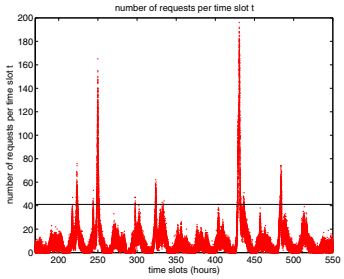


Figure 6. Full Winter Olympics trace (H=40).

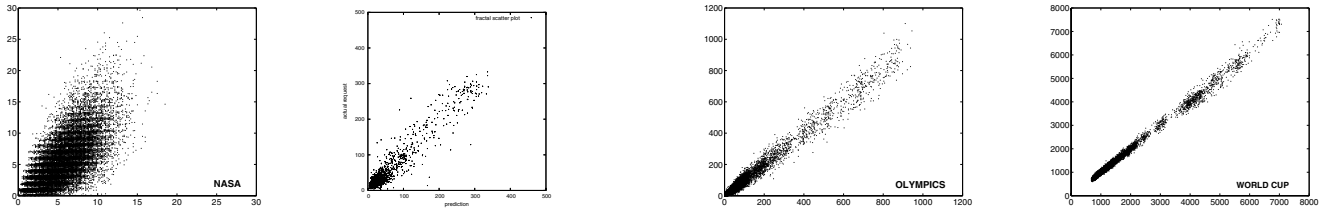
maximum advance notice τ_{max}, hotspot prediction has the following form:

> *If the traffic is currently in a hotspot (as defined at the end of the preceding section), then an alarm is set. The alarm is also set if traffic is not already in a hotspot, but an extrapolation of the trend of the advance notices in $[t-W_h, t]$ shows that a hotspot will occur some time in $[t, t+\tau_{max}]$. This is tantamount to a declaration that a hotspot will arrive within at most τ_{max} time units. If a hotspot is neither currently in progress nor predicted within τ_{max} time units, then the alarm is reset.*

Concrete versions of this algorithm depend on applications and how advance notices are extrapolated. We take one such application, study it in detail, and arrive at estimates of the parameters and a specification of the tests made by the algorithm. We have chosen the IBM Winter Olympics data for the application as it is more extensive than the Fractals data, and predictability is somewhat more of a challenge than the World Cup data. The generality of this case study will be discussed in Section 8.

6. A case study

The full trace for the IBM Winter Games data is shown in Figure 6 on a time scale where hotspots are easily identified.



(a) NASA (b) Fractals (c) Olympics (d) World Cup

Figure 4. Scatter plots contrasting real versus predicted values within the three traces.

For the sake of this case study, we set the hotspot level to $H = 40$. Throughout this and the next section, the duration of the interval over which the volume of traffic determines t_H is $W_d = 3$ minutes.

To see how performance varies with parameter values, we examine the advance-notice sequences $\ldots \tau_{t-i}, \tau_{t-i+1}, \ldots, \tau_t$ at current times t. These will be called *prediction sequences* and are superimposed on the corresponding request data, as illustrated in Figure 7, to visualize the variation of advance notices as a hotspot approaches. An approaching, well-predicted hotspot will be represented by a prediction sequence that is decreasing linearly, or nearly so, to 0 at about the hotspot arrival time t_H, i.e., when the request rate averaged over the last W_d time units increases to H.

We notice immediately by inspection that the LF algorithm performs extremely well in forecasting the hotspot shown in Figure 7 over a period of at least 15 to 20 minutes before the onset t_H of the hotspot. For example, assuming that preparations for a hotspot take no more than a couple of minutes, a useful choice for τ_{max} could be about 5 minutes with a window of about 15 minutes for fitting the trend of the prediction sequence. The alarm would be set close to 5 minutes before the onset t_H of the hotspot. That is, a lead time of close to 5 minutes would be given, where lead time is defined as the actual advance notice, i.e., the time $t_H - t$ remaining until the hotspot arrival.

The trade-off in setting τ_{max} is clear. If it is too large, there will be many false alarms where a decreasing prediction sequence will tail off and start to increase before reaching the abscissa. This happens when a surge of traffic does not in fact reach hotspot proportions, even though early projections say it would. An example is shown in Figure 8, where if τ_{max} were chosen to be about 15 minutes or more, a hotspot would have been falsely predicted at time 244.1 hours. On the other hand, τ_{max} can not be taken too small, since that would not give enough time to prepare for the hotspot. Table 1 shows this trade-off by giving the average lead times corresponding to the advance notices computed by LF, and the number of false alarms as a function of τ_{max}.

Figure 8 also illustrates an important point about false alarms. Because τ_{max} is small in the time scale of hotspot development, the traffic levels setting off false alarms are not far from hotspot levels. Thus, the "errors" represented by false alarms tend to be quite minor.

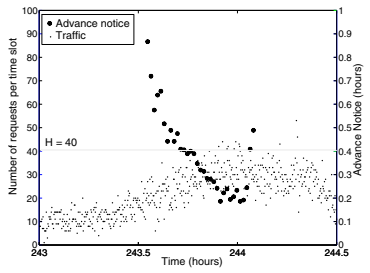


Figure 8. A false alarm when τ_{max} is too large.

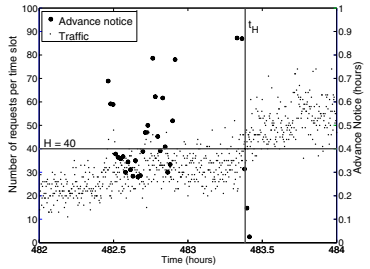
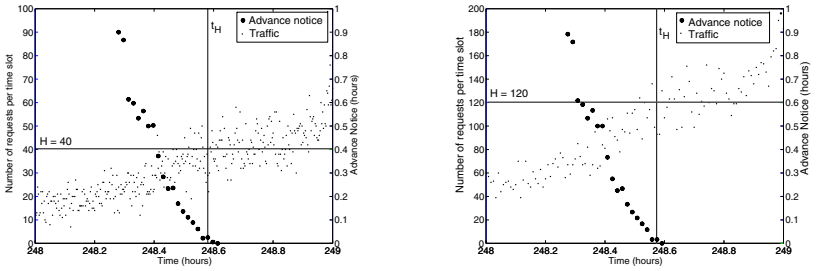


Figure 9. A hard-to-predict hotspot

Figure 9 shows an example of a hotspot that is very difficult to predict with linear extrapolation. As can be seen, the growth past the hotspot level of 40 is quite abrupt, first leading to a number of false alarms, and then failing to correctly predict the actual hotspot.

7. Robustness

The prediction algorithm relies on a number of constants whose value may influence the performance of the predictors. For example, increasing the sample granularity (i.e., the size of the time slot) also smooths the data and speeds up prediction algorithms. The effect of increasing granularity is shown in Figure 7. As suggested, predictions are relatively robust to the choice of time slot; granularity can



(a) 10 seconds per time slot

(b) 30 seconds per time slot

Figure 7. Prediction sequence plots for different time slot granularities

τ_{max} (mins)	♯ of false alarms	♯ of late alarms	average lead time (mins)
6	0	2	2
12	2	1	5
18	3	1	9
24	5	1	12

Table 1. The performance of linear prediction on the IBM Winter Olympics trace for different values of τ_{max}. A late alarm occurs when the alarm is not given until the arrival of the hotspot. The trace contains 5 hotspots.

vary over an order of magnitude without a serious degradation of the prediction process. Of course, the time slot must be kept appreciably smaller than lead times expected of the predictions.

Window size (W_p) is a more important parameter as it must be adjusted to the burstiness (and spectral density) of the data. Robustness is still present to a fair degree, but there are clear limits. If W_p is taken too small, then the LF predictions over-react to local variations, but if it is taken too large, then the predictions become sluggish as they use data that is too old and hence irrelevant to current traffic behavior. Figure 10 shows performance for 3 choices of window size. As compared to the choice that we have been using (20 minutes), the choice of $W_p = 10$ minutes gives more erratic prediction sequences; and while those for the case $W_p = 45$ minutes are quite smooth, they are late in their predictions. The autocorrelation function can serve as an initial guide to selecting window sizes.

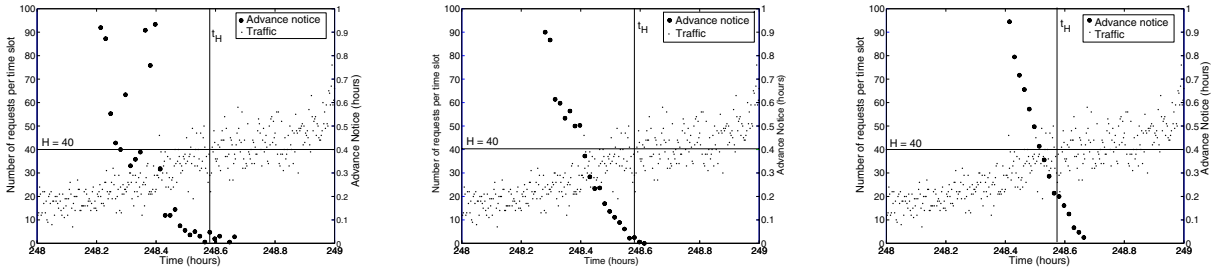
Finally, we tested possible improvements in linear extrapolation. In particular, we examined the performance of quadratic least squares extrapolation. Interestingly, the potential improvements were found to be minor, at most 10%, and not worth the added complexity. More generally, we took a closer look at the leading edges of traffic surges to see if there was a useful relation between the growth-rate derivative and the ultimate peak height. However, our experiments revealed no significant dependence between these traffic measures. It is worth emphasizing that this may not hold in other applications, where high traffic peaks are signaled early on by accelerating increases in traffic.

8. Conclusions

We have dealt with essentially an empirical problem and with algorithms containing parameters that are estimates drawn from traffic traces rather than the solutions to well-defined mathematical problems; this scenario is of course inherent to the concept of prediction. Bearing this in mind, we have presented convincing evidence that even very simple prediction algorithms have a significant predictive power. One meta-conclusion is that they perform especially well in situations where they are needed, that is, when sustained traffic surges are likely to occur.

The generality of our problem, and hence our contribution, lies in its structure, and not so much in the parameter values supplied for specific applications. This structure is defined by a resource (web servers here) with a demand that fluctuates, on occasion increasing to levels too high to support. *This paper has shown that there is a useful predictability in Internet traffic that can be exploited in the use of resources that experience strong surges in traffic.* Our proof of concept has included the design of an effective prediction algorithm for web-server traffic, together with the parameter values of a case study, for applications resembling sports tournaments in terms of the demand placed on the servers. Our specific algorithm is especially useful for situations in which simplicity and speed are required with little sacrifice in accuracy. Our parameter values apply to empirical data spanning several years, and they can easily be rescaled to account for technological advances in the future.

Important phenomena that increase applicability have



(a) W_p = 10 minutes (b) W_p = 20 minutes (c) W_p = 45 minutes

Figure 10. Prediction-sequence plots for W_p of 10 minutes, 20 minutes, and 45 minutes. We used H = 40 requests per slot as the hotspot level for the IBM Winter Olympic trace.

been observed, e.g., robustness with respect to granularity of data collection. Further, it is important to recognize that the mechanism we are proposing is quite efficient and economical in use of resources, so it can only improve system performance; even with ill-adapted parameter values or "ill-tempered" hotspots, a "late-alarm failure" of the algorithm simply converts hotspot prediction into hotspot detection, which is no less (and is in some cases more) than the status quo. Further, while false alarms are created by traffic at less than hotspot level, the traffic is close to hotspot level, and so hotspot responses to such traffic are not wholly inappropriate.

Finally, we note that the scenario we have studied is conservative in the following sense. Prediction algorithms in practice may be supplemented with partial prediction data such as the general timing of particular events. For example, it may be known that an announcement will appear within a couple of hours on a given day, but the timing is otherwise unknown. Consistent information of this sort can in general be exploited in the parameter settings of a prediction algorithm.

References

[1] S. Adler. The slashdot effect: An analysis of three internet publications. `http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html`.

[2] A. Aussem and F. Murtagh. A neuro-wavelet strategy for web traffic forecasting. *Research in Official Statistics*, 1(1):65–87, 1998.

[3] J. Bolot and P. Hoschka. Performance engineering of the world wide web: Application to dimensioning and cache design. In *Proc. Intl. World-Wide Web Conference*, Paris, France, May 1996.

[4] P. Bourke. Googleblatted and SlashDotted, Feb. 2004. `http://astronomy.swin.edu.au/~pbourke/fractals/quatjulia/google.html`.

[5] E. Coffman, P. Jelenkovic, J. Nieh, and D. Rubenstein. The Columbia hotspot rescue service. Technical Report CU/EE/TR 2002-05-131, Columbia University, May 2002.

[6] M. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, San Fransisco, CA, USA, Mar. 2004.

[7] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, New York, NY, 1992.

[8] A. Iyengar, M. Squillante, and L. Zhang. Analysis and characterization of large-scale web server access patterns and performance. *World Wide Web*, 2(1-2):85–100, 1998.

[9] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *Proc. Intl. World-Wide Web Conference*, Honolulu, Hawaii, May 2002.

[10] Q. Li and B. Moon. Distributed cooperative Apache web server. In *Proc. Intl. World Wide Web Conference*, Hong Kong, May 2001.

[11] P. Neumann. Anticipating disasters. *Communications of the ACM*, 48(3):128, Mar. 2005.

[12] V. Padmanabhan and K. Sripanidkulchai. The case for cooperative networking. In *Proc. Intl. Workshop on Peer-to-Peer Systems*, Cambridge, MA, Mar. 2002.

[13] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proc. NOSSDAV Conference*, Miami Beach, FL, May 2002.

[14] G. Pierre and M. van Steen. Design and implementation of a user-centered content delivery network. In *Proc. IEEE Workshop on Internet Applications*, June 2003.

[15] M. Rabinovich and A. Aggarwal. RaDaR: a scalable architecture for a global web hosting service. In *Proc. Intl. World-Wide Web Conference*, May 1999.

[16] I. Research. The Oceano project. `http://www.research.ibm.com/oceanoproject`.

[17] S. Schwartz. Web servers, earthquakes, and the Slashdot effect, Feb. 2000. `http://www-socal.wr.usgs.gov/stans/slashdot.html`.

[18] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen. Replication for web hosting systems. *ACM Computing Surveys*, 36(3):291–334, 2004.

[19] T. Stading, P. Maniatis, and M. Baker. Peer-to-peer caching schemes to address flash crowds. In *Proc. Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, Mar. 2002.

[20] W. Zhao and H. Schulzrinne. DotSlash: A self-configuring and scalable rescue system for handling web hotspots effectively. In *Proc. Intl. Workshop on Web Caching and Content Distribution*, Beijing, China, Oct. 2004.