Capacity planning of IT infrastructure using Pareto self-similar stochastic processes

Raul V. Ramirez-Velarde [1] and Lorena Martinez-Elizalde [2]

Insituto Tecnológico y de Estudios Superiores de Monterrey- Campus Monterrey, Eugenio Garza Sada, Col. Tecnologico, Monterrey, N. L. Mexico
[1] rramirez@itesm.mx
[2] lorenamtze@gmail.com

Abstract. In this paper we introduce a simplified architecture for IT client-server environments and a model that, by capturing the burstiness and self-similarity of server-side execution times, allows for capacity planning of large scale Internet services. The data of an execution-time trace was analyzed. Hurst parameter and marginal distribution parameters were computed by the maximum likelihood technique and applied to a Pareto probability distribution based model. There seems to be a good agreement between our model, which we call the *Pareto Fractal Flow* model, the experimental execution traces, and large demand scaled simulations. The Pareto Fractal Flow model is also compared to other models based on the Gamma and Gaussian distributions and it seems to make better predictions. We believe that this model can be applied in such a manner that the information technology infrastructure can now be planned with more accurate quality of service levels for required resources.

Keywords: Pareto, capacity planning, data servers, self-similarity.

1 Introduction

As it has widely been observed, early Information Technology (IT) systems such as telegraph and telephone possessed capacity planning models. For many years, the most used model for planning the capacity in the basic telephone network has been the Erlang model. This model accurately establishes the necessary infrastructure for a certain quality of service, and it is based on Poisson arrivals and exponentially distributed service times. The Poisson assumption for the predicted traffic demand (or user behavior) and resource utilization statistics fit quite well to this model.

But modern IT infrastructure and services have very different behavior. For example, data traffic exhibit difficulty to model some characteristics such as long-range dependence, self-similarity, burstiness and non-exponential decreasing autocorrelations. These characteristics make the determination of required infrastructure for certain levels of service difficult to determine [1][2].

This difficulty in accurate determination of resource requirements gives place to two important complications for the advancement of the information society. First, it makes economic analysis difficult and less reliable, which means that often, IT infrastructure will be either over-sized or under-sized. Whenever IT infrastructures are over dimensioned, network managers face the danger of being accused of a poor management, but also it takes longer to achieve return of investment (ROI). If, on the other hand, IT infrastructures are under dimensioned, adequate quality of service levels may never be achieved or deployment times will be exceeded or budgets will be severally exceeded. Both situations favor one of the greatest risk factors for the information society: planning is inadequate, thus investment does not correspond to expectations [3].

Nevertheless, it is well known that almost every activity of our society has felt the transformation power of Information Technology: government, education, health and economy all present challenges for IT adoption and in consequence are evolving with different methodologies, goals and infrastructures. Unfortunately, there is a high level of attrition since, as we mentioned before, many projects have failed, either by not attaining stated objectives at all, by providing a poor quality of service, or simply by not offering enough services to justify all the expenses.

E-learning projects are especially vulnerable since it is common that they stretch already hard pressed budgets. In this paper we discuss a new model for capacity planning that, although it was developed for e-learning projects, will readily be apparent the same principles apply to many e-society IT projects.

E-LANE is a consortium that aims to reduce the digital divide in Latin American countries by developing an education program that uses advanced teaching methodologies and paradigms, as well as open source telecommunications and information technologies (E-LANE means Europe Latin-American New Education). E-LANE is constituted by educational institutions from both Europe and Latin America, which together have the following goals:

- To develop pedagogical models that will allow the creation of educational programs that can adapt to different needs, environments and audiences
- To develop high-quality, low-cost distance-learning technologies such as learning management systems, courseware integration tools and learning evaluation platforms
- To develop effective course, activity and evaluation design methodologies that will allow efficient learning as well as long-term retention of knowledge, abilities, competencies and skills
- To develop innovative courseware design guides and establish readily-applicable criteria which will allow the integration of technology, courseware and activities in different ways for different audiences and environments in order to enhance learning
- To establish sustainability principles, such as cost analysis tools, capacity planning models, training programs, etc., to achieve all potential impact to society and ensure continuity after the project ends.

In order to achieve those objectives, E-LANE consortium has chosen Information Technology mediated distance learning. This technology combines a large audience and large geographical coverage of traditional tele-education with the advantages of computer based learning, such as multimedia resources and improved navigation, visualization and interaction.

In order to showcase E-LANE's methodologies, technologies and practices several demonstration projects were established in different countries in Latin America in which different targets were aimed, from rural indigenous peoples to urban-university educated populations. A few of these projects are:

- Education of commercial and health practices for ethnic Guambian peoples in Cauca, Colombia
- Master degree courses for law professionals in Guatemala, Guatemala
- Digital literacy training for government employees in Monterrey, Mexico
- General public digital literacy in Santiago, Chile.

The performance models we discuss in this paper were applied in the design, proposal and development of the Nuevo Leon province's Institute for Public Administration Training in Mexico (CECAP).

Thus capacity planning models that can predict IT requirements with precision are a must for today's information society. We hope that the information we present in this paper contributes to reaching this goal. The rest of the paper is organized as follows. In section II we describe the IT components deployed in our e-learning environment. In section III we characterize the capacity planning problem for modeling purposes and determine the important factors to consider. In section IV we present the main results of our research which include the capacity planning self-similar stochastic model and a comparison with other well-known models. Finally, we present our conclusions and acknowledgments.

2 Architecture

The CECAP was developed with on-line education from the beginning (although it is not an on-line only institution) since Internet technology is the only way CECAP could reach the 4,000 government employees that constitute the provincial government. CECAP does not reach police force employees or teachers since they have independent education facilities.

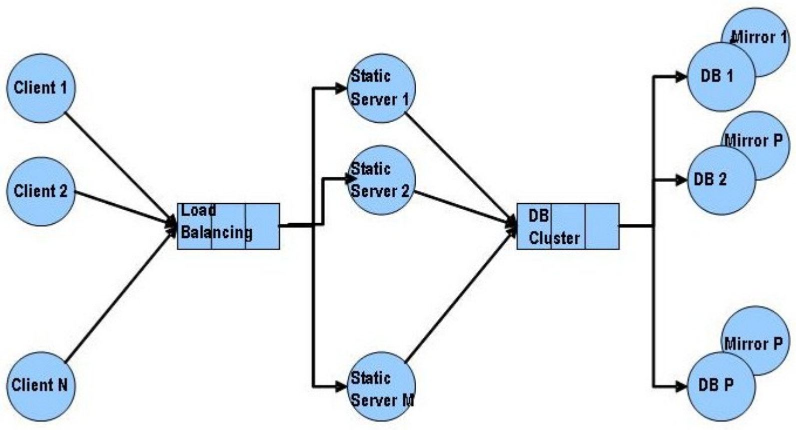


Fig. 1. IT Architecture for e-learning environments

Therefore a high load on data servers is expected and in order to ensure high availability, high reliability, high scalability, high performance and high security with adequate quality of service (QoS) a server-side scripting architecture, shown in Fig. 1, has been proposed.

The IT architecture in Fig. 1, has been implemented using open source components (with its corresponding low cost). For example, the clients can be MS Explorer, Mozilla or Opera based browsers. Load balancing can be done with Linux Virtual Server [4]. Substitution of broken static content servers and on-line replication for high availability can be done with DRDB and Heartbeat [5][6]. Database cluster (DB cluster) can be achieved with MySQL's NDB engine [7]. The e-learning environment, called the Learning Management System (LMS), is dotLRN, which is a very popular open-source LMS widely used by a large group of institutions, actively developed by its users community. It is an application that uses the TCL scripting language and PostgreSQL database.

This architecture establishes certain challenges in order to design a capacity planning model. A decomposition of all the interactions involved is shown in the delay diagram of Fig. 2.

In Fig. 2, D^jCCPU, D^jLBCPU, D^jSSCPU and D^jDBCPU mean CPU times or execution times in transaction *j* for client, load balancer, static server and database respectively. W^jLBLAN 1, W^jSSLAN 1 and W^jDBLAN 1 are LAN related delays, including transmission and contention, whereas W^jCINT means Internet delay. Q^jCCPU, Q^jLBCPU, Q^jSSCPU and Q^jDBCPU mean queuing times for client, load balancer, static server and database. All these interactions are generated by the HTTP request *i* that carries m_1 as data or message 1. Message m_1 is first received by the load balancer that forwards it to a static server. This server determines the server-side script to execute, reads it from disk and executes the program. The program may include one or several database requests, which are forwarded to the database management system (DBMS). The script execution and database requests generate an HTML file that is returned to the client, called m_2, whereas R_i is the total service time.

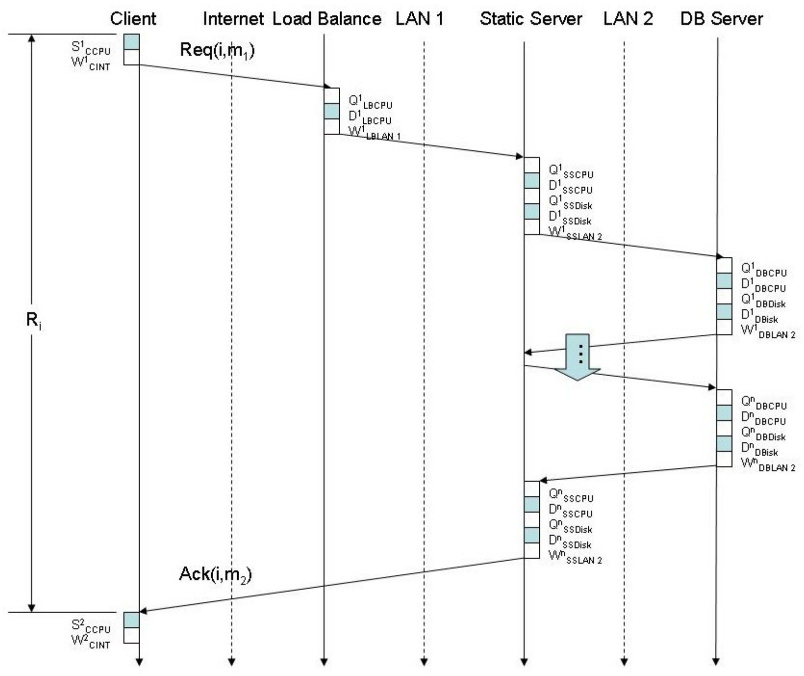


Fig. 2. Delay diagram of e-learning transactions

A comparison of the delays involved is shown in Table 1. These data were obtained by taking measurements upon a prototype server with 512 Kbytes of RAM, 3 GHz CPU, 80 Gbytes hard disk with data-transfer rate of 150 Mbps and a 1,000 Mbps switched Ethernet card. The load balance figure was taken from [4], and the LAN transmission assumes the Gigabit Ethernet network which is used to return objects with mean size of 30 Kbytes, as determined by the Web server log, to the client. WAN delay from and to client will not be taken into consideration since it depends on client Internet access technology, which cannot be controlled by our model. Static-server disk delay was computed by using disk bandwidth of 150 MBps, assuming a mean size of 30 Kbytes of executed server-side scripts and static objects stored in disk (also taken from web server logs).

Transaction	Delay (μ sec)
Load balance	60
LAN transmission	240
Static-server execution	72,233
Static-server disk	200
DB-server execution and Disk	361,566

Table 1. Comparison of transaction delays

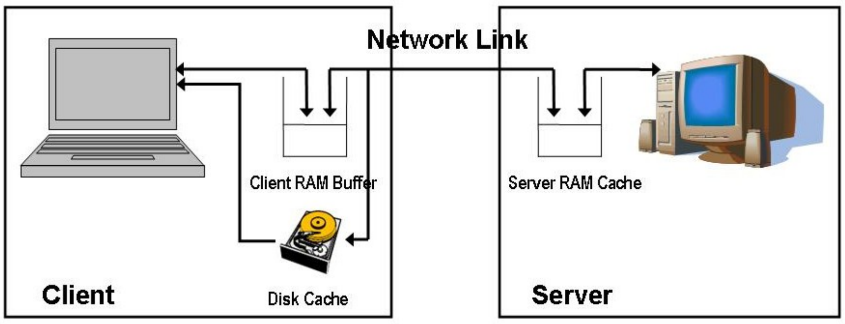


Fig. 3. Client-server model

Needless to say, as Table 1 shows, the dominant factor is the data base. This is why it is not unusual to have a 5 to 1 ratio of static content servers compared to database servers. In order to analyze the problem, it was simplified as shown in Fig. 3.

It is a well known fact that two common ways to improve performance in client-server environments is to use a client disk cache and a server RAM cache. The client disk cache improves performance by locally keeping a copy of static content. Thus, only scripts are interpreted and results are sent as long as parameters change. Meaning that, assuming a steady state operation of the environment, all static content can be ignored. Server RAM cache improves performance by keeping in fast memory rather than in slow disk as many files of the file system as possible. In e-learning environments the scripting part that make up the LMS applications are not disk space demanding, then we can assume that any server is able to keep in RAM all the necessary information and it will almost never fetch information from disk. For instance, our prototype server had only 512 Mbytes of RAM, but the entire dotLRN subtree is about 562,096 Kbytes and it includes all courses with all student files (which tend to be rather large), all administration scripts including installation, and all languages, theme and other optional files. It is important to note that the file system does not include multimedia digital materials that have been developed by the E-LANE project, and they are almost always kept in a copy near or at the client computer in order to reduce bandwidth and disk space demands. This means that, assuming a steady state operation of the environment, disk fetch delays in the static-content server can be ignored since all scripts are taken from RAM cache, which is at least two orders of magnitude faster.

Since load balance and LAN transmission times are negligible, all we have left are delays related to script execution and database transactions. For the rest of the paper, we will treat script execution and database transaction delays as an aggregated delay called X_i , i.e., the service time for script execution request *i*, whereas D_i is the total delay for script execution request *i*, including execution and queuing time.

3 System Characterization

Using the prototype server, execution times for scripts, that constitute the LMS, were recorded along with information about user ID, request time and requested script URL. Execution times for the trace are shown in Fig. 4.

We started our mathematical analysis by studying the histogram of the trace, which is shown in Fig. 5. This histogram displays a high variability in the data. The minimum execution time was 0.001 seconds, while the maximum was 120 seconds. Basic statistics for the trace are shown in Table 2.

As it can be seen in Fig. 4 and statistically determined in Table 2, execution times are highly unsymmetrical as shown by the skewness statistic (symmetric distributions have skewness of zero) and present a heavy tail, as shown by the kurtosis statistic (a kurtosis statistic larger than 4 is considered to be heavy-tailed). This is a type of system that cannot be modeled neither by a symmetric probability distribution such as Gaussian or short-term memory processes such as Erlang.

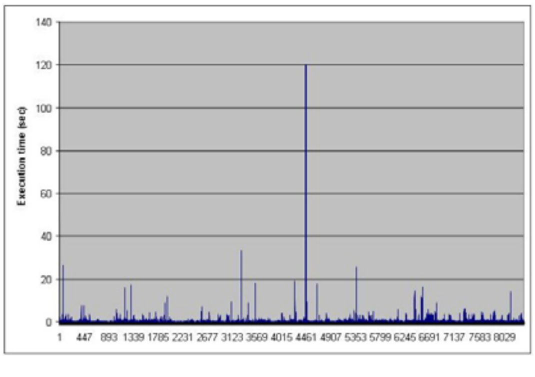


Fig. 4. Execution-times trace

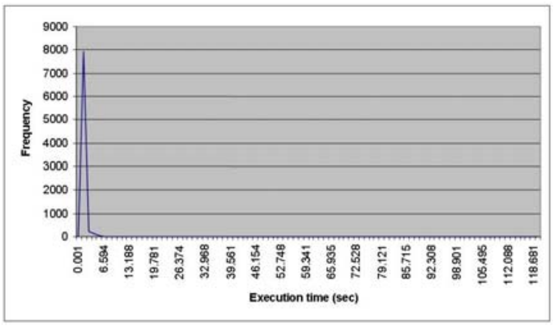


Fig. 5. Histogram of execution times

Mean (sec)	0.434
Variance	4.672
Std Deviation	2.161
Skewness	42.940
Kurtosis	2302.000

Table 2. Basic statistics of execution times

Nevertheless, Normal, Gamma and Pareto probability density functions (pdf) where tested against the data to establish the best fit. Results are shown in Figs. 6 and 7.

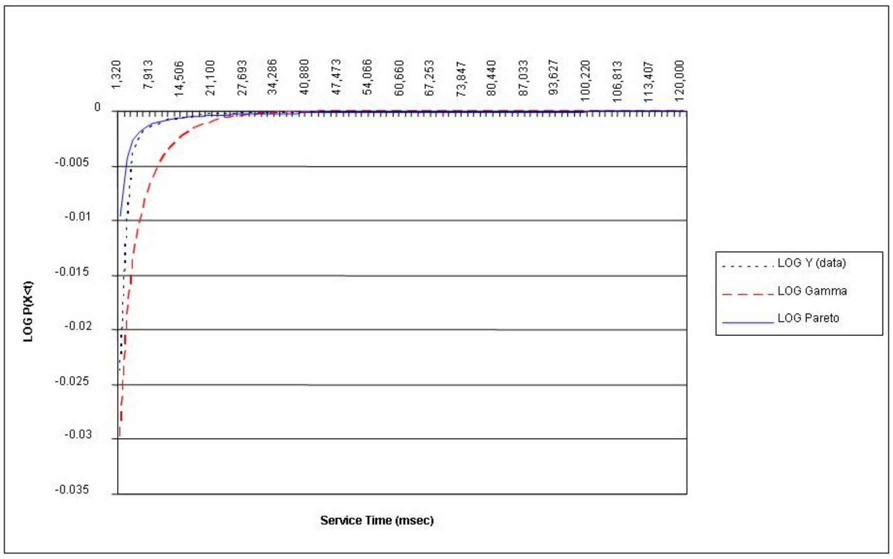


Fig. 6. Gamma and Pareto cumulative distribution fit against data trace

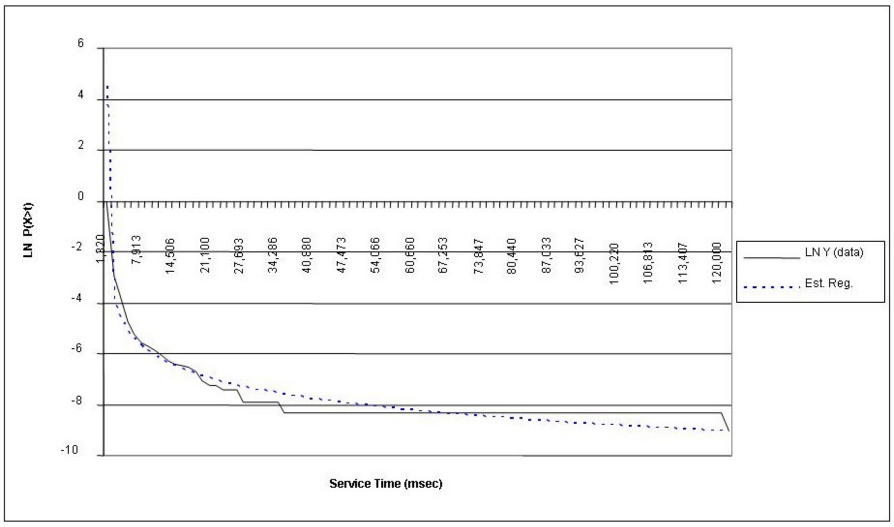


Fig. 7. Pareto fit against data trace for marginal distribution

Gamma and Gaussian distributions were fitted by matching moments (Gaussian fit is not shown on Fig. 6 since it is so far off that renders the graph unreadable), while the Pareto distribution was fitted using the least square estimator after the following linearization on the Pareto cumulative distribution function (cdf)

$$Y = P_{Pareto}(D > a) = \left(\frac{A}{a}\right)^{\alpha} \tag{1}$$

hence,

$$\ln Y = \alpha \ln(A) - \alpha \ln(a)\,. \tag{2}$$

After a linear regression that returns the intercept *b* and slope *m*, we find the Pareto parameters as follows

$$\alpha = -m$$
$$A = \exp(b/\alpha)$$

4 Model Description

In this paper we wish to present a model that, given a maximum response time *W* for script execution requests, it will determine the maximum number of users that a server can bear. Accordingly, given a known number of users for an IT application (called user-load), a correct estimate of the number of servers can be determined and the project can be ROI-analyzed and implemented within budget constrains and with correct quality of service levels.

Therefore, we characterize D_i in the following manner. Suppose that at time t, $\{t>0\}$, which is the beginning of an epoch of duration T, all jobs arrive that will be executed in such epoch. Let $Y_i=\{Y_{i,t},\ i=1,\ldots,n,\ t>0\}$ be a discrete-time random process where Y_i is the execution time of request i at epoch t. Let $X_{i,s}=Y_{i,t+s}-Y_{i,\mathrm{t}}$ be its strictly positive increment process. We assume, as a necessary condition, that Y_i has stationary increments [8]. Also $\mathrm{E}[Y_i]=\mu$ and $\mathrm{Var}[Y_i]=\sigma^2$. The waiting-time process of interest is the worst case scenario $D_{max}=\max\{D_i\}$, which conforms to the discrete accumulated execution-time series $D_{\max} = \sum_{i=1}^{n} X_i$.

4.1 Self-Similar Nature of Collected Data

To determine the maximum user load per server we must take into consideration the statistical properties of the execution times. Recent research has shown that many types of IT data such as local and wide area network traffic (See [9], [10], [11] and references therein), video frame size [12], and others, are self-similar. This means that if we plot the data averaged over several periods of time (say 100 msec to minutes and more), the plots produce profiles that are visually similar to each other over a wide range of time scales.

In our case, execution-time data was analyzed using the Variance of Residual technique and a value of 0.976 for the Hurst parameter H was determined. In this method a log-log plot of the aggregation level versus the average of the variance of the residuals of the series should be a straight line with slope of $H/2$.

There are several, not equivalent, definitions of self-similarity. The standard one states that a continuous-time process $Y=\{Y(t),\ t\geq 0\}$ is self-similar (with self-similarity or Hurst parameter H) if it satisfies the condition [13]:

$$Y(t) =_{\mathrm{d}} a^{-H}Y(at),\ \forall t\geq 0,\ \forall a>0,\ 0.5<H<1 \qquad (3)$$

where the equality means that the expressions have equivalent probability distribution. A typical self-similar process is also a Fractional Brownian Motion [14], and it has been widely used to model different types of systems, including ATM traffic, Ethernet traffic, Internet WWW traffic, etc.

A process satisfying Eq. (3) can never really be stationary as the condition for stationarity requires that $Y(t) =_{\mathrm{d}} Y(at)$, (or rather the distribution of $\{Y(t+s)-Y(t)\}$ does not depend on t), so $Y(t)$ is assumed, as we have mentioned, to have stationary increments.

Let us have $t=1$ and $a=t$ in Eq. (3), thus

$$Y_t =_{\mathrm{d}} t^H Y_1 \tag{4}$$

Also, supported by Eq. (8) and [12],

$$f_{Y_1}(x) = \left|t^H\right| f_{Y_t}(t^H x)\ , \tag{5}$$

where $f_{Y_t}(x)$ is the pdf of Y_t. It is assumed that

$$\mathrm{E}[Y_t^2] = \sigma^2 t^{2H} \tag{6}$$

where $\sigma^2 = \mathrm{E}[Y_1^2]$. We should note that Eq. (6) states, for self-similar variable aggregation, that the resulting variance does not scale linearly, leading to interesting behaviors, such as lack of smoothness under aggregation, which complicates user load planning.

A second definition of self-similarity, more appropriate in the context of standard time series theory, involves a stationary sequence $X=\{X_i,\ i\geq 1\}$. Let

$$X^{m,k} = \frac{1}{m}\sum_{i=(k-1)m+1}^{km} X_i \tag{7}$$

be the corresponding aggregated sequence with level of aggregation m, obtained by dividing the original series X_i into non-overlapping blocks of size m and averaging over each block. The index k, label the blocks. If X_i is the increment process of a self-similar process Y_i as defined in Eqs. (3) and (6), that is $X_i=Y_{i+1}-Y_i$. Then for all integers m,

$$X =_{\mathrm{d}} m^{1-H} X^m\ . \tag{8}$$

A stationary sequence $X=\{X_i,\ i\geq 1\}$ is called *exactly self-similar* if it satisfies Eq. (8) for all aggregation levels m. A stationary process $X=\{X_i,\ i\geq 1\}$ is said to be *asymptotically self-similar* if Eq. (8) holds as $m\rightarrow\infty$.

4.2 Stochastic Framework

Our stochastic model tries to find a user-load such as

$$\phi = \mathrm{P}[D_i > W] = \mathrm{P}[X_1 + ... + X_n > \varphi], \tag{9}$$

where D_i is script execution-time for request i and W is the maximum response time permitted by the service level agreement (SLA). That is to say, that script execution delay is an aggregation of queueing and execution delays. We propose the *Pareto Fractal Flow* model (PFF) model to fit the data trace, which is closely related to the one found in [15]. We compute φ using a continuous-time process $Y(t)$ just as was done in [12], and later tie it to the discrete-time increment process X_n. Let $Y(t)=\mu t+Z(t)$, with $\mathrm{E}[Y(t)]=\mu t$, be the accumulated execution time up to time t. The stochastic process is self-similar with $\mathrm{E}[Z^2(t)]=\sigma^2 t^{2H}$. It is assumed that $Z(0)=0$.

Let $D(t,s)$ be the accumulated execution delay or model state, which is given by

$$D(t,s)=R(t-s)-Y(t)+Y(s), \tag{10}$$

where R is the time between job arrivals (in our case, for user-load equals 1 user). We wish to minimize

$$\mathrm{P}[D_{\max} = \sup_{0<n<t} \{R(t-s) - Y(t) + Y(s)\} > W]. \tag{11}$$

We have, in accordance to [16], that

$$\mathrm{P}[\sup_{0<n<t} \{R(t-s) - Y(t) + Y(s)\} > W] \geq$$
$$\sup_{0<n<t} \mathrm{P}[R(t-s) - Y(t) + Y(s) > W]. \tag{12}$$

Let $n=t-s$ and $A(t-s)=Z(t)-Z(s)$. By stationary increments,

$$Z(t)-Z(s)= Z(t-s)-Z(0)=Z_n \tag{13}$$

and by self-similarity we obtain from (3), (4) and (8),

$$A(t-s) =Z_n=n^H Z_1 \tag{14}$$

Therefore

$$\mathrm{P}[D_{\max} > W] \geq \mathrm{P}[n^H Z_1 - (R-\mu)n > W]. \tag{15}$$

Furthermore, if we assume that the tail probability follows a hyperbolically decreasing law, such as Pareto pdf, and following up on the fact that

$$\mathrm{P}[D_{\max} > W] \geq \sup_n \mathrm{P}\left[Z_1 > \frac{W + (R-\mu)n}{n^H} = a \right], \tag{16}$$

we find that

$$\mathrm{P}[D_{\max} > W] \geq \mathrm{P}\left[Z_1 > \inf_n \left\{ \frac{W + (R-\mu)n}{n^H} \right\} \right]. \tag{17}$$

In order to establish a scaling law for user-load, we establish that if a server is going to execute scripts for *N* users, then we must assume that the inter-arrival time between job requests will scale by the law *R/N*.

It can be shown by minimizing Eq. (17) that [12]

$$\tau_0 \equiv \arg\inf_n \left\{ \frac{W + (R-\mu)n}{n^H} \right\} = \frac{HW}{\left(\frac{R}{N} - \mu \right)(1-H)}, \tag{18}$$

t₀ is called the critical-time scale. Accordingly,

$$\mathrm{P}[D_{\max} > W] \geq \mathrm{P}\left[Z_1 > \left(\frac{R}{N} - \mu \right)^H \frac{H^H}{(1-H)^{1-H}} W^{1-H} \right] \tag{19}$$

Pareto Fractal Flow

We now define $Z(t)$ as a Pareto process with marginal pdf

$$\mathrm{P}[Z(t) = a] = \frac{\alpha (At^H)^\alpha}{a^{\alpha+1}}, \tag{20}$$

and cdf

$$\mathrm{P}[Z(t) < a] = 1 - \left(\frac{At^H}{a} \right)^\alpha . \tag{22}$$

That is to say that $Z(t)$ has a marginal time-scaled Pareto pdf. Also, it has the following characteristics:

$$\mathrm{P}[Z(0)=0]=1 \tag{23}$$

and

$$\mathrm{Var}[Z(t)]=\frac{(At^{H})^{2}\alpha}{(\alpha-2)(\alpha-1)}=\sigma^{2}t^{2H}, \tag{24}$$

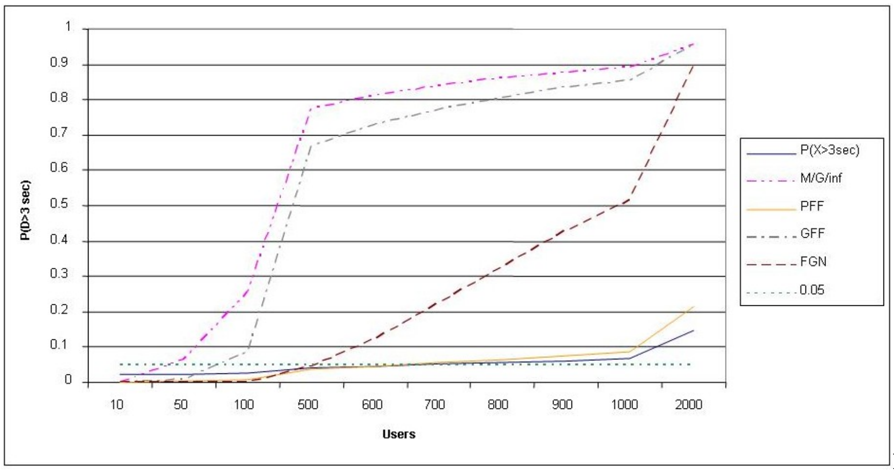
for a>2. Also, Var[Z_1] =σ^2 and E[Z_1]=μ.

Following on Eqs. (1) and (19) we have

$$\phi=\mathrm{P}[D_{\max}>W]\geq\left(\left(\frac{R}{N}-\mu\right)^{H}\frac{H^{H}}{A(1-H)^{1-H}}W^{1-H}\right)^{-\alpha}. \tag{25}$$

From Eq. (25) we derive the following formula which gives the maximum number of users for a server given a SLA

$$N\leq\frac{R}{\mu+\left(\frac{AH^{H}(1-H)^{1-H}\phi^{-1/\alpha}}{W^{1-H}}\right)^{1/H}}. \tag{22}$$



F

Fig. 7. Comparing predicted probability that request delay will exceed 3 seconds of different models versus simulation results

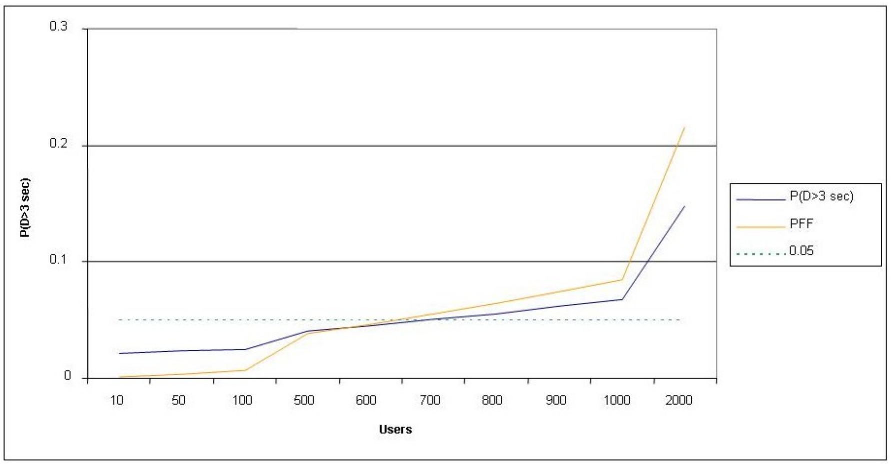


Fig. 8. A closer look at the results of the PFF model against simulation

The performance of the PFF model was compared against the performance of other known models, for example Fractional Gaussian Noise of [14], Gamma Fractal Flow of [12], and the M/G/∞ of [17]. It can be seen in Figs. 7 and 8 that the PFF seems to make better predictions about self-similar user-load scalability than the other models. All results are compared against a simulation that uses captured real-life execution times from the prototype server. The purpose of the simulation was to determine the probability of having delay times above 3 seconds given a user load. For example, in Fig. 7, in we see that the M/G/∞ and the GFF models give very pessimistic predictions, between 50 and 100 users and between 100 and 500 respectively. Simulation results show that a server can give adequate quality of service (that is, probability of D_{max}>3 sec is 0.05) between 600 and 700 users. As it can be seen in Fig. 7, the FGN model at first gives too optimistic predictions and then turns to be too pessimistic, although not as far from simulation results as the M/G/∞ and the GFF models. The PFF model gives predictions very close to simulation results. In Fig. 8 we can take a closer look at how close those predictions are in relation to simulation results.

5 Conclusions

As we have shown, the Pareto probability distribution is a good descriptor of the behavior and performance of advanced IT services, such as self-similarity and extreme variability. It enables the formulation of simple models that take into consideration the extreme variability of such data. It also has the advantage that there exist strong techniques for estimating shape and location parameters, not to mention the Hurst parameter.

This characterization of IT-services performance has important implications for the design, deployment, cost estimation and capacity planning of today's information infrastructure. It implies for example, that certain suppositions long held as true cannot be used any more, such as the Markov memoryless property and smooth behavior under time averaging. Known laws such as Little's, Mean Usage and other known queueing computations are no longer useful.

The model we have developed enables the simple computation of important parameters such as user load, delay time and quality of service. This model can be easily extended and applied to other areas of technology related to the information society such as link capacity estimation, buffer dimensioning and delay computation, multimedia engineering and others.

Appendix. Self-similarity of the time-scaled pareto pdf

As was previously done in [12], we show that the time-scaled Pareto pdf is self-similar by showing that if $Y =_{\mathrm{d}} aX$ [18], then

$$f_Y(z) = \frac{1}{|a|} f_X(\frac{z}{a}) \ . \tag{A1}$$

A fractal random variable scales in time following the law

$$Y_{\mathrm{t}} =_{\mathrm{d}} t^H Y_1. \tag{A2}$$

Then, we obtain from Eqs. (5) and (A1)

$$f_{Y_1}(x) = t^H f_{Y_t}(t^H x) \ , \tag{A3}$$

since t is always positive.

Substituting in Eq. (20)

$$t^H f_{Y_t}(t^H x) = t^H \frac{\alpha (A t^H)^\alpha}{(x t^H)^{\alpha+1}} = f_{Y_1}(x) \tag{A4}$$

which proves the fractal behavior of the time-scaled Pareto random variable.

Acknowledgment

We thank Tecnologico de Monterrey, Campus Monterrey, through the Research Chair in Telecommunications, and the European Union through the @LIS program (E-LANE project) for the support provided in the development of this work.

References

[1] W. E. Leland, M. S. Taqqu, W. Willinger and D.V. Wilson. “On the self-similar nature of Ethernet traffic”. In ACM SIGCOMM 93, 1993.

[2] B. A. Cipra. “Oh, What a Tangled Web We’ve Woven”. SIAM News, Volume 33, Number 2, 1994.

[3] Gupta, M. P. and Jana, D. E-government evaluation: a framework and case study. Government Information Quarterly, 20 (2003), 365-387.

[4] Zhang, W. “Linux Virtual Server for Scalable Network Services”. Proceedings of the Ottawa Linux Symposium 2000. Ottawa, Canada.

[5] Robertson, A., 2004. Highly-affordable High Availability. *Linux Magazine*, November 2004.

[6] Ellenberg, L., 2004. Data Redundancy by DRBD. *Linux Magazine*, November 2004.

[7] Zawodny, J., 2004. MySQL Cluster Configuration. *Linux Magazine*, November 2004.

[8] Raul V. Ramırez-Velarde and Ramon M. Rodrıguez-Dagnino. “Performance Analysis of a VBR video server with Gamma distributed MPEG data”. SPIE, Performance and Control of Next Generation Communication Networks, vol.5 244-16, Orlando, Florida, U.S.A., pp. 131-142, September 2003.

[9] M. Garret and W. Willinger. “Analysis, Modeling and Generation of Self-Similar VBR Video Traffic”. ACM SigComm, London, pp. 269-280, September 1994.

[10] B. N. Bashford and C. L. Williamson. “Statistical Multiplexing of Self-Similar Video Streams: Simulation Study and Performance Results”. Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1998.

[11] J. Beran, R. Sherman, M. S. Taqqu and W. Willinger. “Long-range Dependence in Variable-Bit-Rate Video Traffic”. IEEE Transactions on Communications 43 (2-4) 1566-1579, April 1995.

[12] Raul V. Ramırez-Velarde and Ramon M. Rodrıguez-Dagnino. “A gamma fractal noise source model for variable bit rate video servers”. Computer Communications, Volume 27, Issue 18, 1 December 2004, Pages 1786-1798.

[13] W. Willinger, V. Paxton. “Where Mathematics Meets the Internet”, Notices of the American Mathematical Society, Vol. 45, 1998.

[14] I. Norros. “A Storage Model with Self-Similar Input”. Queuing Systems-Theory and Applications 16 (1994) 387-396.

[15] J. R Gallardo, D. Makrakis and L. Orozco-Barbosa. “Use of a-stable self-similar stochastic processes for modeling traffic in broadband networks”. Performance Evaluation, Elsevier (2000) 71-98.

[16] N. G. Duffield and N. O’Connell. “Large Deviations and Overflow Probabilities for general Single-Server Queue, with Applications”. Mathematical Proceedings of the Cambridge Philosophical Society, 1995.

[17] M. Parulekar and A. Makowski. “Tail probabilities for a multiplexer with self-similar traffic”. Fifteenth Annual Joint Conference of the IEEE Computer Societies, Networking Next Generation, Preceedings IEEE (1996) pp. 1452-1469.

[18] A. Leon-Garcia. “Probability and Random Processes for Electrical Engineering”. Addison-Wesley, London, 1994.