Self-similarity and Multidimensionality: Tools for Performance Modelling of Distributed Infrastructure

Raul Ramirez-Velarde[1], Cesar Vargas[2], Gerardo Castañon[3], and Lorena Martinez-Elizalde[4]

Tecnologico de Monterrey, Campus Monterrey, Sucursal de correos "J", C. P. 64849, Monterrey, N. L., Mexico
{rramirez,cvargas,gerardo.castanon}@itesm.mx, lorenamtze@gmail.com

Abstract. In this article, we present a model that uses large-deviations and the Pareto probability distribution to model self-similar data in high-performance infrastructure, such as the one found on computational and data grids, transactional and computational clusters, and multimedia streaming. We also show how Principal Component Analysis can reduce dimensionality of data, such as the one produced by different types of problems, user preferences and behaviour, and resource popularity.

Keywords: Pareto processes, digital video, performance modelling, grids, principal component analysis, self-similarity, high-performance clusters.

1 Introduction

As we begin to adopt evermore Information Technology (IT) in our lives and begin to enjoy the increased productivity and learn to depend on it, more demands are placed on infrastructure. From corporate intranets of large companies that support high-demand applications to on-line public companies that support millions of subscribers in services such as parallel computing, distributed transactions and high-definition streaming multimedia, quality of service (QoS) translates to revenue in vast amounts.

Now, large corporations such as Google, eBay, MySpace and Amazon have implemented high-performance distributed computing and data infrastructure that although may not deserve yet the name "Grid", address very much the same issues: transparent and dynamic commoditization of data and computing power based on heterogeneous systems.

The main concern here is that there needs to be techniques that will accurately calculate required resources based on QoS specifications of availability, reliability, security, consistency, transparency and performance.

In this article we address such problem. We describe statistical techniques and mathematical models that will address the main characteristics for high performance computing, non-ergodic variability and multidimensionality in behaviour of classes and cohorts. These two are important problems. Self-similarity implies that additional complexities must be taken into account to estimate waiting times and storage spaces (memory and disk), thus QoS guarantees and the cost infrastructure need to be estimated with more complicated models. Multidimensionality means that variation in types of services and resources, some times ranging in the thousands of millions complicates the determination of operational parameters.

The rest of the article is organized as follows. In section 2, we describe the assumed architecture of the computing infrastructure and show how to determine the critical variables. In section 3, we study the data collected and determine that the data is heavy-tailed. In section 4, we study self-similarity. We show how to model self-similar heavy-tailed processes using large deviations and show how we use the Pareto probability distribution to approximate the tail of the original probability distribution. In this section we also show two models used in different scenarios. In section 5, we study multidimensionality and Principal Component Analysis. We also show the results rendered by applying such technique to our data. In section 6, we present out conclusions.

2 Computation Infrastructure Characterization

In order to develop these models we have implemented a High-Performance Test Bed (HPTB). This test bed has two conceptual configurations.

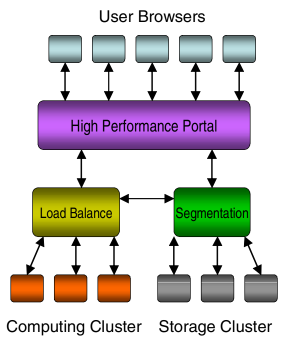

Fig. 1. High-performance transactional cluster

In Figure 1 we show the first type of configuration which corresponds more to a high-performance, high-availability enterprise transactional cluster, such as the one we would expect for e-commerce, e-government, e-learning, e-health, e-science and in general collaborative environments.

In [21], it is shown that in these types of environments the critical performance variables are computation and storage times, the latter in the form of database transactions.

In Figure 1 conforming environments, what seems to be of interest from the QoS point of view, is to know with certain probability how long computation time will take in order to determine how many user-transactions we can have per computing node and how much time will database response take. Also of interest is the ratio of computing against database nodes, which can go from 3 to 6 database nodes per computing server.

The architecture of high-performance streaming environments would usually be similar to Figure 1, with memory storage and network communications being the critical variables.

3 Statistical Characteristics of Critical Variables

The first characteristic that must be noted when creating performance models for these types of environments is that the observed probability distributions of storage, computing and communication times are not exponential. In fact, it has been consistently observed that in many types of human created digital data follow long or heavy tail laws. In Figure 2 we show histograms for application computation time [21] and video frame size [9],[22].

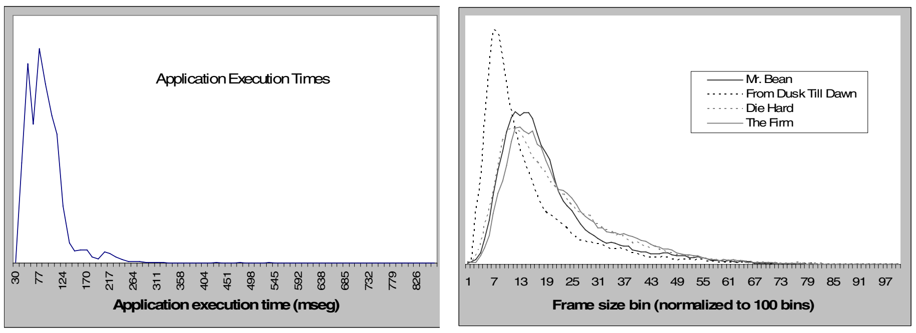

Fig. 2. Histogram for application execution and database transaction time, and video frame size

Table 1. Correlation coefficients for probability distribution fit

Trace	Gamma	Weibull	Normal	Lognormal
Script	0.97188	0.96559	0.90412	**0.97599**
Database	0.96769	0.97549	0.87440	**0.98221**
Mr. Bean	0.98584	0.97439	0.92204	**0.99239**
Die Hard	**0.99640**	0.99018	0.94122	0.98579
The Firm	0.98388	0.97764	0.89058	**0.98965**

In the next figures we show how heavy-tailed distributions are a better fit for out environments under study. Although sometimes Gamma pdf gives a better overall fit, at the tail extreme usually Weibull or Lognormal give a better fit. This is shown in Figures 3 and 4. In Table 1 we show that Lognormal will be a better overall fit.

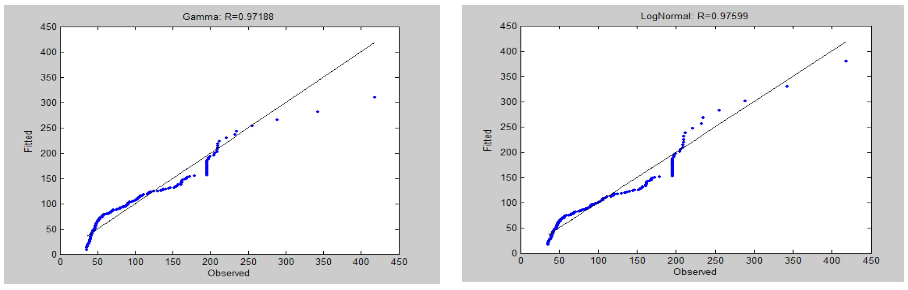

Fig. 3. Probability plots for application execution traces

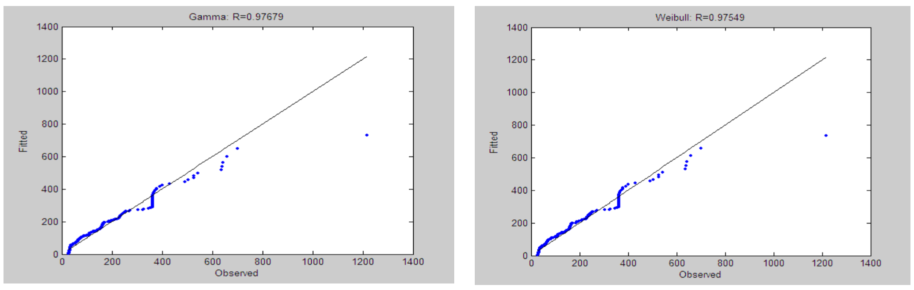

Fig. 4. Probability plots for database transaction times

4 Self-similarity

Research has shown that many types of IT data such as local and wide area network traffic (See [1], [6], [10], [14], [17], and references therein), video frame size ([2], [3], [12], [19], [20]), and others, are self-similar [7]. 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.

There are several none 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 [21]:

$$Y(t) \equiv a^{-H}Y(at),\ \forall t\geq 0,\ \forall a>0,\ 0.5<H<1 \tag{1}$$

where the equality means that the expressions have equivalent probability distribution.

The accumulation process of self-similar input can be modelled as follows:

Let $Y(t)=\mu t + Z(t)$ be an increment process for some time unit t_u, with $E[Y(t)]=\mu t$. Let $Z(t)$ be a self-similar process with $E[Z(0)]=0$ and $E[Z^2(t)]=\sigma^2 t^{2H}$.

The accumulation process is then

$$\begin{aligned}\sum_{i=1}^{n} X(i) &= Y(n)-Y(n-1)+Y(n-1)-Y(n-2)+\ldots+Y(1)-Y(0)\\ &= n\mu + Z(n) \equiv n\mu + n^{H}Z(1),\end{aligned}$$

where the equality indicates equality in distribution.

The accumulation probability (or saturation probability), that is the probability that an input process will eventually overflow a buffer or, as in a case of withdrawal process, that it will eventually empty a repository is

$$\begin{aligned} \mathrm{P}(X_1+X_2+\ldots+X_n>\varphi) &\equiv \mathrm{P}(n\mu+n^H Z(1)>\varphi) \\ &\equiv \mathrm{P}\left(Z(1)>\frac{\varphi-n\mu}{n^H}=a\right)=\phi, \end{aligned} \tag{2}$$

When modelling concurrently a birth-death process we complete the model as follows:

Let Q be the steady-state of the birth-death, input process. Then

$$\mathrm{P}[Q>W]\ge \mathrm{P}[Y(n)-R\cdot n>W]=\mathrm{P}[n^H Z(1)-(R-\mu)n>W]. \tag{3}$$

where R is a fixed withdrawal process. Furthermore,

$$\mathrm{P}[Q>W]\ge \sup_n \mathrm{P}\left[Z(1)>\frac{W+(R-\mu)n}{n^H}=a\right], \tag{4}$$

Thus we find through large deviations theory that [8]

$$\mathrm{P}[Q>W]\ge \mathrm{P}\left[Z(1)>\inf_n\left\{\frac{W+(R-\mu)n}{n^H}\right\}\right]. \tag{5}$$

It can be shown by minimizing Eq. (5) that [17], [20]

$$\tau_0\equiv \arg\inf_n\left\{\frac{W+(R-\mu)n}{n^H}\right\}=\frac{HW}{(R-\mu)(1-H)}, \tag{6}$$

τ_0 is called the critical-time scale. Accordingly,

$$\mathrm{P}[Q>W]\ge \mathrm{P}\left[Z(1)>(R-\mu)^H\frac{H^H}{(1-H)^{1-H}}W^{1-H}\right]=\phi. \tag{7}$$

Then, one must only determine an appropriate pdf for $Z(1)$ and the model is solved.

Although this is an upper bound, it is usually assumed that the bound is tight. Liu in [16] proves that indeed the bound is tight, making the inequality and equality for Pareto and Lognormal pdfs.

But we are interested in being accurate in the extreme tail of the probability distribution. That is to say, that since quality of service probabilities (such as loss, saturation and blocking probabilities) are usually specified in the 10^{-3} to 10^{-6} range, a pdf that is accurate in the 95% of the body of the data and inaccurate on the tail will not be useful.

Given its analytic simplicity, we have chosen the Pareto pdf to model the tail of our self-similar data. In Figure 5 we show how this pdf fits aggregated execution and database transaction times in the tail interval, obtained from the HPTB and available upon request [21], and compare against the fit provided by the Gamma and Gaussian pdf. It seems that Pareto provides a better fit.

As conclusion we show two models we have derived, one for high-performance cluster, the other for multimedia streaming. Eqn. (8) shows the maximum user load for an application server according to a maximum execution delay *W*. The model is called Pareto Fractal Flow Performance for this model is shown in [21].

$$N \le \frac{R}{\mu + \left(\frac{AH^{H}(1-H)^{1-H}\phi^{-1/\alpha}}{W^{1-H}} \right)^{1/H}}, \tag{8}$$

where *N* is maximum user load per server, *A* is the Pareto location parameter, α is the Pareto shape, μ is the execution time mean, ϕ is the probability of exceeding maximum allowed time and *H* is the Hurst parameter.

Eqn. (9) shows video server maximum user-load whereas Eqn. (10) shows user video buffer size. This model is called Pareto Fractal Noise. Model derivation and performance for this model is shown in [22].

$$N \le \frac{n_v^{\min}}{R_{pl} I_v + \frac{1}{R_{dr}} \left(R_{pl} n_v S_{vf} + \frac{A(R_{pl} n_v)^{H} \kappa}{\phi^{1/\alpha}} \right)}, \tag{9}$$

where $\kappa = k_{\max}^{H-1}$, $k_{\max}$ being the practical limit to the a constant called the proportionality constant *k* that the video server designer is willing to accept. Variable *k* is proportional to the amount of video frames that are needed in user buffers in order to ensure continuous video reproduction. R_{pl} is video playback-rate, S_{vf} is the mean video frame size, n_v is block video grouping in disk (number of video blocks grouped), I_v is block separation is seconds and R_{dr} is hard disk throughput. The proportionality constant *k* indicates how many blocks containing n_v video frames need to be read and stored in user video buffer in each service cycle.

User buffer *b* is:

$$b = kR_{pl} n_v S_{vf} + \frac{A(kR_{pl} n_v)^{H}}{\phi^{1/\alpha}}. \tag{10}$$

See [20] and [22] for a formula to determine *k* and a more thorough explanation of this model's parameters.

5 Principal Component Analysis

So far we have occupied ourselves with solving the problem of modelling self-similarity and heavy-tails. But there is another characteristic that presents unique challenges in performance modelling of high-end infrastructure which is multidimensionality.

Multidimensionality is a phenomenon found in most modern information technology based services. For example, a video-on-demand server will need to store thousands of videos. Thus, out of thousands of videos, what characteristics should we assume that play-back time videos will have? Should we design for the average of all videos? Should we design for the worst case scenario? Should we try to figure out which videos will be more popular and what percentage of all play-back will recall those popular videos?

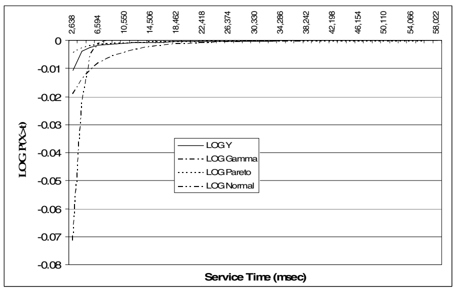

Fig. 5. Fitting the pdf tail. Comparing Pareto, Gaussian and Gamma. Y is trace data.

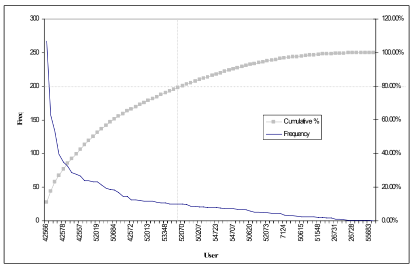

Fig. 6. Frequency of user contributions in a collaborative on-line community. There are 80 users in the community. Data taken from HPTB.

In HPC environments different kinds of applications will be executed by different kinds of processors. Moreover, the information will be stored in different types of devices and transmitted using different types of transmission media. Thus should we design this infrastructure averaging performance data for all participating processors, links and storage devices? Should we also average characteristics for all different kinds of jobs? That is find and average for chemical modelling, atmospheric sciences, materials modelling, cosmology and other types of problems?

One approach, used in industry for a long time has been to create classes and find 20% of classes used the most and concentrate the most. Nevertheless, not even knowledge of user preferences in video, availability of most used infrastructure and frequency of most worked problems will really solve the problem, because of a phenomenon called the Long-Tail, which is related to the Heavy-Tail phenomenon. The Long Tail means that in order to find most common classes of behavior, we need to cover most of the population (there is no Pareto 20/80, but rather 30/70, or 40/60). That is, all cases are almost as important.

An example from HPTB [21]; Figure 6 shows the frequency of contributions for an 80 user cooperative environment. 80% of contributions are not reached until contributions from the 34th most active user are taken into consideration, that is, 42% of all users.

Thus there is a need to find means of reducing data dimensionality. We use Principal Component Analysis for such purpose. The objective is clear, to change the data in order to reduce the classes of services to just a few ones, and if possible just one, to contain must of the original variability of data [13].

Principal component analysis (PCA) is the most common form of factor analysis. It creates new variables and dimensions which are uncorrelated linear combinations of the original ones. New dimensions are orthogonal in the original dimension space and are arranged in such a way that the first dimensions capture as much of the original variance in the data as possible allowing us to discard low variable dimensions. The first dimension or first principal component (PC) is the direction of greatest variability in the data. Second PC is the next orthogonal (uncorrelated) direction of greatest variability. It can be viewed as a rotation of the existing axes to new positions in the space defined by original variables. We analyzed 26 sequences of video. Figure 7 shows a plot of the variability captured by each principal component, called Scree plot. In this plot we see that the first two principal components capture 41.5% of variability of original data. 80% variability is not achieved until the 11th principal component is included, that is 42% of all PCs.

At first it seems disappointing to not be able to dramatically reduce the dimensions with something close to 80% of variability, as it is usually desired. But this is deceiving for PCA has achieved much. Let us first consider that in 23 videos, one individual one should have only about 1/23 variability or about 4%. In a 64,000 videos database the variability that one video can contribute would be around 0.001%. But although we are nowhere near conducting PCA on thousands of videos, it does seem that PCA is able to discover a few factors that can contribute up 40% of variability serving as a base for server design. Using those high variability PCs we can construct a characteristic video sequence that serves as a proxy for most of the videos currently available in the server, and hopefully, for videos that will available in the future.

Just as the long-tail makes video server design difficult, the long-tail makes computing infrastructure design difficult. In a grid environment, certain types of problems will be solved more often than others. The same scenario will occur in transactional environments, some users with their particular needs will make use of the environment more often than others. PCA can be used to derive characteristic behaviour independent of long-tail scenarios.

Figure 8 shows the Scree plot for application execution. The first two PCs accumulate 53% of all variability, whereas in the database case (not shown) the first two PCs accumulate 42%.

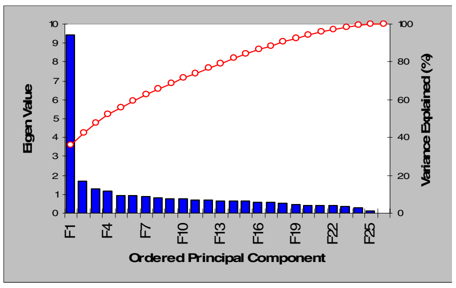

Fig. 7. Scree plot for video data PCA; first 10 PCs

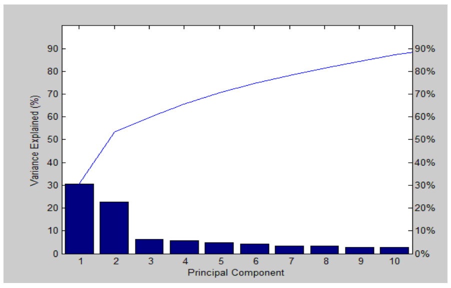

Fig. 8. Scree plot for PCA of application execution and database transactions

As is evident from simple observation of the Scree plots presented, the PCs in all cases decay with a long-tail. That is that the Scree graph presents a few large value PCs and then a long-tail of very slowly decaying PCs with a rule similar to $d^{-\beta}$, where d is the distance between PCs. This power law decay is expected and characteristic of self-similar data as reported in [11].

6 Conclusions

The use of self-similar stochastic processes with power-law decaying tails, such as the Pareto Fractal Noise and Pareto Fractal Flow, can in fact be used to model the extreme variability present in different kinds of computing and multimedia digital phenomena, such as application execution times, database transactions and transmission and storage of data present in modern information technology infrastructure such as grids and high-performance clusters and streaming services. The approach we have used is simple, given a resource limit and a maximum probability for resource starvation; we determine maximum user loads and other required resources. It is simple but it works.

We have also shown that principal components analysis is effective in simplifying infrastructure design by using just the very few PCs and producing new data that reduce the complexity of the original one. PCA based design can overcome unknown resource popularity and extreme differences of measurable characteristics.

References

[1] Addie, R.G., Zukerman, M., Neame, T.D.: Broadband Traffic Modeling: simple solutions to Hard Problems. IEEE Communication Magazine 36(8) (August 1998)

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

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

[4] Beran, J.: Statistics for long-memory processes. Chapman & Hall, New York (1994)

[5] Bodamer, S., Charzinsky, J.: Evaluation of effective bandwidth schemes for self-similar traffic. In: ITC Specialist Seminar on IP Traffic Measurement, Modeling, and Management. Monterey, California, September 14 (2000)

[6] Cipra, B.A.: Oh, What a Tangled Web We've Woven. SIAM News 33(2) (1994)

[7] Park, K., Kim, G., Crovella, M.: On the Relationship between File Sizes, Transport Protocols and Self-Similar Network Traffic. In: Proc. IEEE International Conference on Network Protocols, pp. 171–180 (October 1996)

[8] Duffield, N.G., O'Connell, N.: Large Deviations and Overflow Probabilities for general Single-Server Queue, with Applications. In: Mathematical Proceedings of the Cambridge Philosophical Society (1995)

[9] Fitzek, H.P., Reisslein, M.: MPEG-4 and H.263 Video Traces for Network Performance Evaluation. Technical University Berlin, Technical Report TKN-00-06 (October 2000)

[10] Gallardo, J.R., Makrakis, D., Orozco-Barbosa, L.: Use of a-stable self-similar stochastic processes for modeling traffic in broadband networks. Performance Evaluation, pp. 71–98. Elsevier, Amsterdam (2000)

[11] Gao, J.B., Cao, Y., Lee, J.-M.: Principal component analysis of $1/f^{\alpha}$ noise. Physics Letters A 314, 392–400 (2003)

[12] Garret, M., Willinger, W.: Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In: ACM SigComm, London, pp. 269–280 (September 1994)

[13] Jolliffe, I.T.: Principal Component Analysis. Springer, U. S. A (2002)

[14] Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V.: On the self-similar nature of Ethernet traffic. In: ACM SIGCOMM 1993 (1993)

[15] Leon-Garcia, A.: Probability and Random Processes for Electrical Engineering. Addison-Wesley, London (1994)

[16] Liu, Z., Nain, P., Towsley, D., Zhang, Z.: Asymptotic behavior of a multiplexer fed by a long-range dependent process. J. Appl. Probab. 36(1), 105–118 (1999)

[17] Norros, I.: A Storage Model with Self-Similar Input. Queuing Systems-Theory and Applications 16, 387–396 (1994)

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

[19] Ramirez-Velarde, R.V., Rodrıguez-Dagnino, R.M.: Performance Analysis of a VBR video server with Gamma distributed MPEG data. In: SPIE, Performance and Control of Next Generation Communication Networks, Orlando, Florida, U.S.A, vol. 5244-16, pp. 131–142 (September 2003)

[20] Ramirez-Velarde, R.V., Rodrıguez-Dagnino, R.M.: A gamma fractal noise source model for variable bit rate video servers. Computer Communications 27(18), 1786–1798 (2004)

[21] Ramirez-Velarde, R., Rodrıguez-Dagnino, R.M.: A Scalability Model for High-Performance Computing Using Pareto Self-Similar Stochastic Processes. Technical report

[22] Ramirez-Velarde, R., Rodrıguez-Dagnino, R.M.: Design of a High Definition Video on Demand Server with Large Title Availability Using Principal Component Analysis of Self-Similar Flows'. Technical report