

Towards Better Performance Measurement of Web Servers

Yibei Ling⁺, Shigang Chen[‡], Xiaola Lin[#]

⁺Applied Research Laboratories, Telcordia Technologies, USA

[‡]Department of Computer Science, University of Florida, USA

[#]Department of Computer Science, City University of Hong Kong, Hong Kong

Abstract

Performance evaluation is an important issue in building efficient Web servers. It is desirable to measure the performance regularity of Web servers for overall performance behavior of the system in the full spectrum of working area, which is generally overlooked and has not received proper attention. In this paper we aim to raising the awareness of the importance of the performance regularity of a Web server by introducing Gini performance coefficient (GPC) as a scale-invariant metric for measuring the performance regularity. We present the theorems that relate the performance regularity of a Web server to the GPC, thereby providing a quantitative yardstick that complements the system capacity metric such as maximum throughput for measuring the system performance. Several representative systems that were used in the public benchmark study are examined under the proposed metric. The results are completely in line with our theoretical analysis.

1 Introduction

A high-performance Web server is the key to the success of the Web-based applications. The emerging Web-based services and applications have demanded unique performance characterization and workload patterns, leading to constant change in system requirement. In the past, various performance benchmarks have been developed to characterize the various performance problems stemming from the ever-changing computing environment. The benchmarks should be defined to reflect the problem-specific domain [6]. For example, SPECweb96 suite is a widely-recognized industrial benchmark for evaluating the static performance capabilities of a Web server [1], measuring the maximum system throughput in terms of HTTP GET operations/second, while SPECweb99 suite [2] is a more recent Web server benchmark with a focus on adding dynamic content into the traffic mix, measuring Web server performance in terms of the maximum number of simultaneous connections. The recent TCP-W benchmark [3] places emphasis on the activities of a business oriented transactional web server, and

the performance metric used by TCP-W is the number of web interactions processed per second.

In general, performance metrics defined in standard benchmarks are associated with system capacity such as maximum system throughput and maximum number of simultaneous connections. In addition, there exist aggregate performance metrics such as the harmonic mean which is used to compute the average performance of computer systems [12] and parallel processors [8]. As systems are expected to work under normal loads for most of time, it is very important to understand how well system performs in the full spectrum of system loads[4], rather than the system performance as measured by system capacity such as maximum system throughput. The performance regularity of a system, as complementary to system capacity metric, can be used to describe the overall performance behavior of a system under normal conditions, depending on the problem-specific domain.

The main interest of this paper is to bring awareness of the importance of the performance regularity, and to propose a new performance metric called Gini performance coefficient (GPC hereafter) to quantify the performance regularity. As a measure of the performance regularity, GPC is derived from the system performance curve with respect to the choice of the capacity metric being used. We formally establish a connection that links the performance regularity of a system with the corresponding GPC. Using the proposed approach, we measured and reassessed various representative systems based on the SPECweb96 benchmark suite. The obtained results are completely in line with our theoretical analysis.

2 Performance Regularity vs. System Capacity

System capacity is an outermost limit of system performance with respect to a given performance metric being used, serving a landmark dividing working area and non-working area. In the non-working area, a system is unable to provide sustainable throughput and satisfactory interactive behavior. The performance capacity metric could be selected differently, depending on the choice of the problem-specific domain. System performance regularity refers to

¹Partial funding was provided by Hong Kong CERG grant No. 9040695

the overall system behavior of a system in its working area, with its domain being determined by the corresponding system capacity.

It stands to reason that the response time of a system is proportional to system workload in general [5, 11], *i.e.*, the response time increases as the inverse of unutilized capacity [9, 5]. A lightly loaded system is very likely to generate faster response time than a heavily loaded one because a high frequency of requests from clients generates a considerable amount of simultaneous processes/threads in the server, incurring an expensive run-time overhead in context-switching, process/thread synchronization and resource contention, which in turn causes a slowdown in processing each individual request as a result. The definition below is given to classify system in terms of the overall performance behavior.

Definition 1 *Suppose the size distribution of request and that of response are statistically stationary (independent of access frequency), the performance of a system is said to be regular if the response time of processing a request is statistically non-decreasing as the access frequency increases. Otherwise, the performance of a system is said to be irregular.*

To illustrate the importance of system performance regularity, we start with two examples reported in SPECWeb96 suite as a case study.

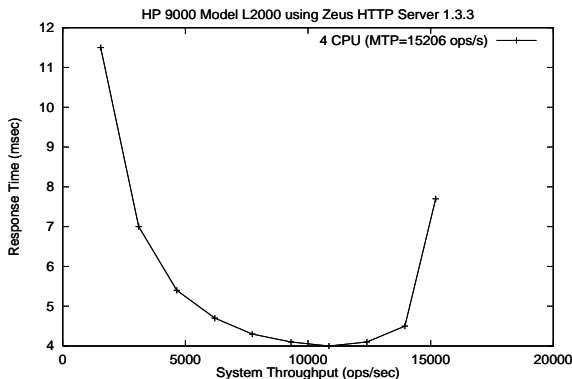


Figure 1: Performance Curve of HP 9000/L2000

The performance curve of a system in the SPECWeb96 benchmark suite is represented by a sequence of data pairs $(x_1, y_1), \dots, (x_n, y_n)$, where $x_i \in \mathcal{R}$ is the i th request load level, and $y_i \in \mathcal{R}$ is the corresponding response time under the request load level x_i . The performance curves depicted in Fig(1) and Fig(2) are the reproduction of SPECWeb96 results of HP 9000/L2000 published in the fourth quarter of 1999 and of Sun Enterprise 250 published in the second quarter of 1999, respectively. It is clear from Fig(1) and Fig(2) that HP 9000/L2000 clearly outperforms Sun Enterprise 250, scoring 15206 ops/s on the SPECweb96, as opposed to 2624 ops/s by Sun Enterprise 250. On the other

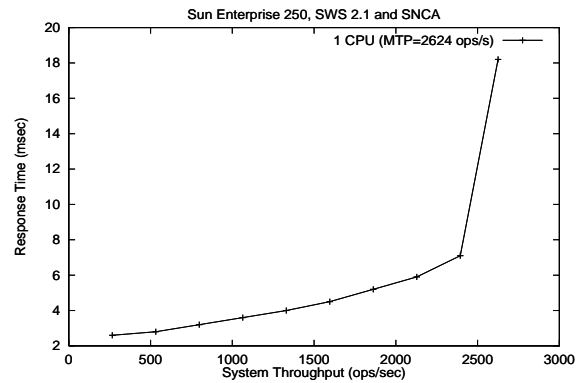


Figure 2: Performance Curve of Sun Enterprise 250

hand, Sun Enterprise 250 performs more regularly than HP 9000/L2000 counterpart: its response time grows slowly but monotonically with the increase in workload. By contrast, the HP 9000/L2000 exhibits the erratic performance behavior reflected in the apparent anomaly in its response time and workload relationship. The response time pathologically decreases with the increase in workload in a wide range: from 1550 ops/s to 10855 ops/s. It is contrary to expectations that the response time under system throughput 1550 ops/sec is about 11.5 msec, almost twice of that under the maximum system throughput (7.7 msec for 15206 ops/s), meaning that the system needs more time in processing when it is lightly loaded. Such a system behavior illustrated in Fig(1) is irregular and abnormal, representing a sharp departure from our common sense and any theoretical projection.

3 A Measure for Performance Regularity

In order to better understand the Gini performance coefficient, a good place to begin with is to review the Gini coefficient and Lorenz curve used in economics. A measure of inequality, referred to as *Gini coefficient*, was proposed by Gini in 1912 [10], and has been widely used in economics and social sciences for measuring the magnitude of inequality in data distributions such as wealth and income. The Gini coefficient is based on the Lorenz curve which is represented by a cumulative frequency curve.

Based on the original Gini coefficient, we introduce GPC in connection with the performance regularity of a system. Consider the system performance curve that consists of n data pairs $(x_1, y_1), \dots, (x_n, y_n)$, where where $(x_1, \dots, x_n) \in \mathcal{R}^n$ represents the vector of system throughput in the ascending order ($x_1 \leq x_2 \leq \dots \leq x_n$), and $(y_1, \dots, y_n) \in \mathcal{R}^n$ represents its corresponding of system response time vector. Notice that x_n denotes the system capacity, and y_n represents the response time at system capacity. We con-

struct the two normalized vectors $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ and $\bar{y} = (\bar{y}_1, \bar{y}_2, \dots, \bar{y}_n)$ by the following transformation:

$$\bar{x}_i = \frac{x_i}{x_n}, \quad \bar{y}_i = \frac{y_i}{\sum_{k=1}^n y_k}, \quad 1 \leq i \leq n \quad (1)$$

where x_n and $\sum_{k=1}^n y_k$ are the normalizing factors for the system throughput and the response time, respectively. The normalization transformation scales down the x -axis by a factor of x_n which is the maximum system throughput obtained, and scales down the y -axis by a factor of $\sum_{i=1}^n y_i$. From the normalized response time vector \bar{y} , we construct a vector $\bar{Y} = (\bar{Y}_1, \dots, \bar{Y}_n)$, where $\bar{Y}_i = \sum_{k=1}^i \bar{y}_k = \sum_{k=1}^i y_k / \sum_{k=1}^n y_k, 1 \leq i \leq n$. It can be verified that $\bar{x}_n = \bar{Y}_n = 1$. The curve referred to as *Lorenz performance curve* (LPC for short) can be constructed by making piecewise connection between every pair of adjacent points, $(0, 0), (\bar{x}_1, \bar{Y}_1), \dots, (1, 1)$. With the normalizing transformation, we are able to map the performance curve of a system into the corresponding LPC. The LPC has the two salient features, making it distinguished from a traditional Lorenz curve:

1. the traditional Lorenz curve is constructed from one vector, and Lorenz performance curve is constructed by the two vectors, \bar{x} and \bar{y} . There exists the componentwise correspondence between the vectors \bar{x} and \bar{y} : the i th component \bar{Y}_i represents the normalized response time measured when the system throughput is \bar{x}_i ;
2. The components in the vector $\bar{x} = (\bar{x}_1, \dots, \bar{x}_n)$ are arranged in the increasing order, but may not be uniformly distributed, i.e., $\bar{x}_i - \bar{x}_{i-1} \neq \bar{x}_j - \bar{x}_{j-1}$ for some $i \neq j$.

Definition 2 Let $L(\tau)$ be the Lorenz performance curve defined as a continuous curve over $[0, 1]$, and $I(\tau)$ be the line of equality (45 degree). The GPC is defined as

$$GPC = 2 \cdot \int_0^1 (I(\tau) - L(\tau)) d\tau \quad (2)$$

Given a system performance curve, the algorithm for calculating the GPC can be easily developed. Due to the page number limitation, the proofs of the following lemma and theorems have to be omitted.

Theorem 1 Let the LPC be constructed by using the normalized vector of system workload $\bar{x} = (\bar{x}_1, \dots, \bar{x}_n)$, and the normalized vector of response time $\bar{y} = (\bar{y}_1, \dots, \bar{y}_n)$. If $\sum_{k=1}^i \bar{y}_k \leq \bar{x}_i$, then GPC is positive.

The following theorem establishes a link between the performance regularity and the GPC with respect to the choice of performance metric, serving a pivotal theorem of this paper.

Definition 3 Given a workload vector $x = \{x_1, \dots, x_n\}$ and the response time vector $y = \{y_1, \dots, y_n\}$, and the components in the vector x are in the ascending order, i.e., $x_1 \leq \dots \leq x_n$, the λ -weighted normalized response time is defined as

$$\bar{y}_\lambda = \sum_{i=1}^n (\lambda_i \cdot \bar{y}_i) = \frac{\sum_{i=1}^n (\lambda_i \cdot y_i)}{\sum_{i=1}^n y_i} \quad (3)$$

where the assignment of weights $(\lambda_1, \dots, \lambda_n)$ is determined as $\lambda_i = 1 - \frac{x_i + x_{i-1}}{2 \cdot x_n}, 1 \leq i \leq n$, and $x_0 = y_0 = 0$.

It can be verified that the weight sequence $\lambda = (\lambda_1, \dots, \lambda_n)$, constructed from the workload vector x , is in descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. The λ -weighted normalized response time is the sum of weighted normalized response time, with the dual effect to amplify the contributions from lightly loaded states (y_1, y_2, \dots) and minimize the contributions from heavily loaded states (y_n, y_{n-1}, \dots) . The following example is given to illustrate the sensitivity of the λ -weighted normalized response time \bar{y}_λ to the performance irregularity.

The following lemma is very useful in simplifying the proof of the main theorem.

Lemma 1

$$0 \leq \bar{y}_\lambda \leq 1$$

We present the main theorem of this paper as follows.

Theorem 2 Gini performance coefficient (GPC) is directly proportional to the overall performance regularity of a server, with respect to a given response time at system capacity.

4 Assessment of System Performance Regularity

In this section, we will assess the performance regularity of systems reported in SPECWeb96 [1] in the context of the GPC. We emphasize that the main reason of using the experimental results from SPECWeb96 benchmark suite is the availability of performance curves.

In Figs(3)-(6), we transform the original performance curves reported in SPECWeb96 into the corresponding Lorenz performance curves and calculate the values of the GPC. We are unable to produce similar calculation on the benchmark results submitted in 2000 because of the absence of performance curves. The original performance curves available on <http://www.spec.org> are omitted due to page limit.

In an effort to investigate such effects, we intentionally group the results of system with the different number of

processors into one graph for easy comparison and presentation clarity. Our study based on Figs(3)-(6) suggests that Microsoft has done a better job in utilizing SMP architecture. There exists a positive correlation between the GPC and the number of processors. An increase in the number of processors could produce an additional gain in the GPC, signifying an improvement in the performance regularity. Note that there exists one exception case for Microsoft IIS, which was reported in SPECWeb96's first quarter 1999, the GPC for HP NetServer running on IIS 4.0 declines as the number of processor increases (Fig(3) for details). By contrast, Apache HTTP servers in In these figures do not improve system performance regularity with an increase of the number of processors. The comparison results between Microsoft IIS 5 and Apache server appeared to be supported by the conclusion [7] that Apache HTTP server has a problem in maintaining satisfactory performance on a SMP architecture.

The graphs of the GPC versus different versions of Window 2000 advanced server are showed in Fig(5). With eight Pentium III Xeon processors at 550 MHz, running Microsoft IIS 5.0, we compare two Dell PowerEdge 8450/550 systems running on the two versions of Window 2000 advanced server, with the same hardware configurations. The benchmark results were reported in the third quarter 1999, and the fourth quarter 1999, respectively. In Fig(6), with four Pentium III Xeon processors at 500 MHz, running Microsoft IIS 5.0, we compare two IBM Netfinity 7000/M10 systems running on two versions of Window 2000 advanced server, the benchmark results were reported in the first quarter 2000, and the third quarter 1999, respectively.

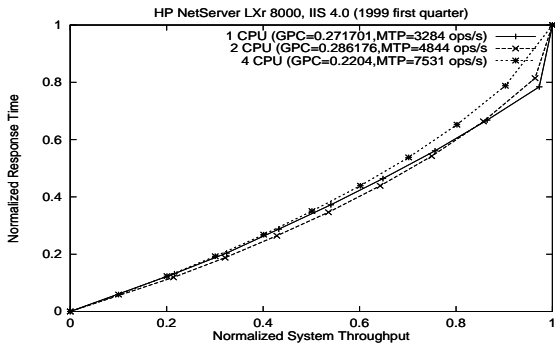


Figure 3: GPC vs. No. of CPUs

5 Conclusion

This work is motivated by the importance of the performance regularity and by the necessity of making finer distinction of the system performance. We establish theorems

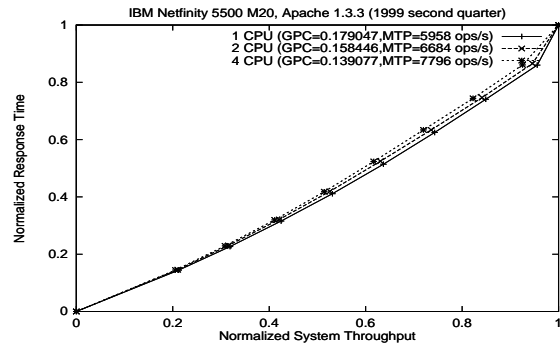


Figure 4: GPC vs. No. of CPUs

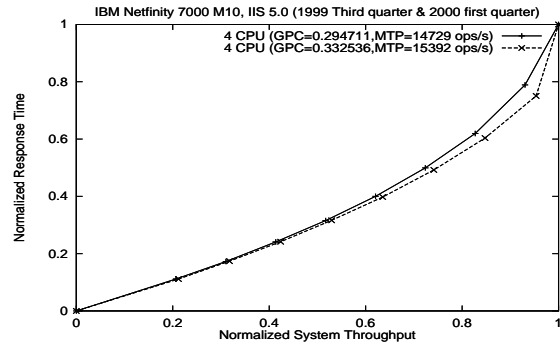


Figure 5: GPC vs. No. of CPUs

that relate the GPC to the system performance regularity, thereby providing a quantitative description of the performance regularity. We have also presented the algorithm for constructing the Lorenz performance curve based on the available performance curve and calculating the GPC.

Our study suggests that the performance curves (intermediate data points) should be considered as an integral part of a benchmarking report, because they contain valuable information about not only the system capacity but also how well a system performs in its working area. A better understanding of system performance could be enhanced by analysis of the performance regularity of a server. The use of GPC, in conjunction with any performance met-

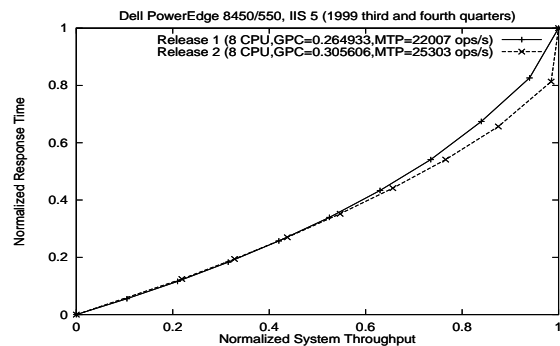


Figure 6: GPC vs. Versions of Microsoft 2000

ric (capacity), can lead a better and comprehensive assessment of system performance.

References

- [1] *SPECWeb96*. <http://www.spec.org/osg/web96>, 1996.
- [2] *SPECWeb99*. <http://www.spec.org/osg/web99>, 1999.
- [3] Tcp benchmark w (web commerce). In *Transaction Processing Performance Council*, October 2001.
- [4] Gaurav Banga and Peter Druschel. Measuring the capacity of a web server. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 244–263, December 1997.
- [5] Microsoft Corporation. *User Guide: Microsoft Web Capacity Analysis Tool Version 4.35 Windows 2000 Operating System*. Microsoft Corporation, 2000.
- [6] Jim Gray. *The benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann Publishers, New York, 1993.
- [7] James Hu. *JAWS: Understanding High Performance Web Systems*. <http://www.cs.wustl.edu/~jxh/research.research.html>.
- [8] K. Hwang and F.A. Briggs. *Computer Architecture and Parallel Analysis*. McGraw-Hills, New York, 1988.
- [9] Krishna Kant and Youjip Won. Server capacity planning for web traffic workload. In *IEEE Transactions on Knowledge and Data Engineering*, pages 731–747, September/October 1999.
- [10] Albert W. Marshall and Ingram Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, New York, 1979.
- [11] M. Pendleton and G. Desai. *@Bench Test Report: Performance and Scalability of Window 2000*. Doculabs, August 2000.
- [12] J. Smith. Charactering computer performance with a single number. *Communications of ACM*, 31(10):1202–1206, 1988.