
World Wide Web: Internet and Web Information Systems, 7, 241–258, 2004

On the Performance Regularity of Web Servers

YIBEI LING lingy@research.telcordia.com
Applied Research Laboratories, Telcordia Technologies, One Telcordia Drive, RRC 1A216, Piscataway, NJ 08854-4157, USA

SHIGANG CHEN sgchen@cise.ufl.edu
Department of Computer Science & Information Science & Engineering, University of Florida, Gainesville, FL 32611, USA

XIAOLA LIN csxlin@cityu.edu.hk
Department of Electrical Engineering, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong

Abstract

The performance regularity is concerned with the overall performance behavior of a system in the full spectrum of working area. Such a performance characteristic is generally overlooked and does not receive proper attention. The aim of this paper is twofold. First, it raises awareness of the importance of the performance regularity of a Web server. Secondly, it introduces the Gini performance coefficient (GPC) as a scale-invariant metric for measuring the performance regularity. In this paper, 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. To illustrate the use of the proposed approach, we calculate the values of GPC for several representative systems that were used in the public SPECweb96 benchmark study. The results are completely in line with our theoretical analysis.

Keywords: performance regularity, Gini performance coefficient, performance capacity, Lorenz performance curve

1. Introduction

The World Wide Web (WWW) has exhibited exponential growth over the past several years and has become the dominant application in both public Internet and corporate intranet environment. A high-performance Web server is the key to the success of the Web-based applications. The emerging Web-based services and applications have created unique performance characterization and workload patterns, leading to constant change in system requirement. Therefore, system capacity planning needs to be frequently reexamined, and the impact of workload characterized by new services and applications needs to be carefully studied, in order to prevent consequences such as performance degradation or even crashes from happening [3].

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 [4]. For example,

SPECweb96 suite is a widely-recognized industrial benchmark for evaluating the static performance capabilities of a Web server [11], measuring the maximum system throughput in terms of HTTP GET operations/second, while SPECweb99 suite [12] 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 [13] 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 [10] and parallel processors [5]. 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 [2], 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 want to point out that the GPC and capacity metric are two complementary performance metrics, measuring system performance from different facets: how well a system generally performs and how much load a system can handle. 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.

It is known that SPECweb96 suite has been discontinued in favor of the more representative SPECweb99 suite. The reason that we use experimental results from SPECweb96 is the availability of intermediate data which are essential in evaluating the performance regularity, as well as the determination of the GPC. The SPECweb96 provides us with a rather complete picture of how system performs under different work loads, giving us a unique opportunity to evaluate system performance from a completely different angle.

This work is motivated in part by the observation in SPECweb96 benchmark results that system performance capacity does not necessarily correlate with its performance regularity, and in part by the need to have a measure to quantify the performance regularity of a system. We emphasize that the objective of this paper is to characterize the system performance regularity, rather than to provide means for accurate benchmarking or identifying the root cause of system problems.

The remainder of the paper is organized as follows. The concept of system performance regularity is described in Section 2. Section 3 introduces the Lorenz performance curve and GPC, and discusses the application of GPC in describing the performance regularity

of a server, as well as algorithms for comparing the performance regularity of systems with different system capacities. Section 4 offers an explaination of the practical meaning of GPC. In Section 5, we present our calculated results based on the performance curves from various systems reported in [11], and compare the performance regularity of different platforms in the context of GPC. Section 6 concludes the paper and highlights our contribution.

2. Performance regularity vs. system capacity

It is important to make a clear distinction between the system capacity and the performance regularity of a server: the two closely related but drastically different notions. 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 problem-specific domain. For instance, the capacity metric used by SPECweb96 suite [11] is the number of HTTP GET operations/per second, whereas the one selected by SPECweb99 suite [12] is the number of simultaneous connections. System performance regularity refers to 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 [8,9,14], i.e., the response time increases as the inverse of unutilized capacity [6,8]. 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.

Obtaining system capacity under workload has defied rigorous analysis, and the best solution known so far comes from performance benchmarking. For instance, the maximum system throughput in SPECweb96 suite [11] is obtained via a benchmarking experiment. For each load level of requests, the load generators installed on client machines send requests with randomly selected file sizes to the server according to a predetermined pattern, the average response time of each load level is measured. The workload level is incrementally increased until the server is saturated with the requests and the response time arises significantly to an unacceptable level.

Given a size distribution of request/response, the maximum throughput of a server is defined as the highest rate at which the server can process the requests while still meeting the minimum performance requirements. In other words, the system saturation point is the value at which the system is out of capacity. There is no formal definition for the system saturation point. Informally, it can be defined as a performance point at which a small change in workload results in a relatively large change in response time. The benchmark specification proposed by Doculabs [9] defines the maximum throughput of a server based on the end-to-end response time limit of three seconds. Also, in the SPECweb99 specifi-

cation [12], the maximum number of simultaneous connections of a system is defined as the number of connections that can be made and sustained at a specified maximum bit rate with a maximum segment size.

The workload of a server is mainly determined by access frequency and request/response size. Both the request and response size can be static such as HTML documents or dynamically generated by CGI. When the size distribution of request and response is statistically stationary (independent of access frequency), the workload of a server is statistically proportional to the access frequency, so does the response time of the server. It means that although statistical fluctuation in the request/response size might make system abnormal in a short run, the fluctuation would be smoothed out if long-run measurement is taken. The warmup period and the length of the experimental period in the industrial benchmark specifications [6,11,12] and the documentation of Microsoft Web Capacity Analysis Tool [8,9] are designed to eliminate the impact of short-run statistical fluctuations on the performance results, ensuring unbiased experimental results. For instance, in SPECweb96 specification [11], the default warmup time is set to be as 300 seconds and the default time duration of measurement is set to be as 600 seconds.

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.

The performance curve of a system in the SPECweb96 benchmark suite is represented by a sequence of data pairs $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathcal{R}$ is the ith 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 Figures 1 and 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 Figures 1 and 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 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 Figure 1 is irregular and abnormal, representing a sharp departure from our common sense and any theoretical projection.

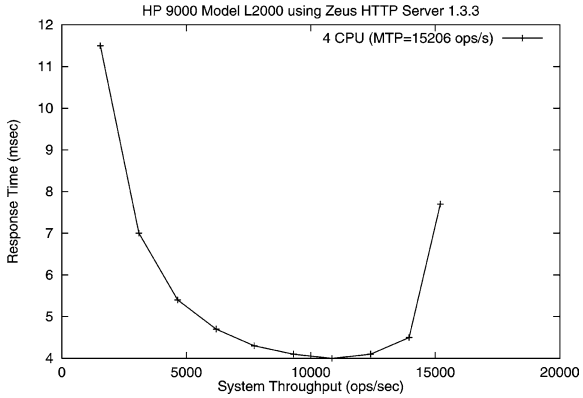

Figure 1. Performance curve of HP 9000/L2000 (MTP: maximum throughput).

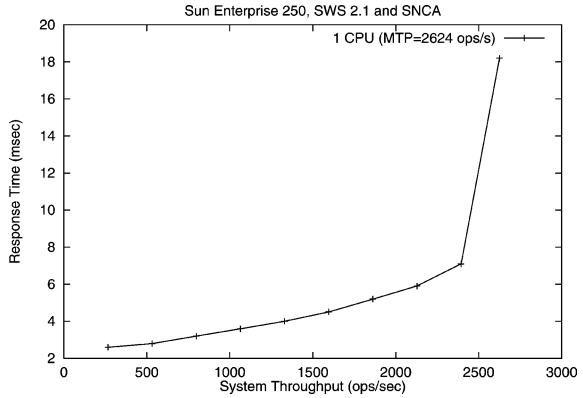

Figure 2. Performance curve of Sun Enterprise 250 (MTP: maximum throughput).

Although the apparent difference in the performance regularities of the two systems can be visually perceived by examining the respective performance curves, a quantitative measure is desirable because of the objectiveness and accuracy, allowing us to critically evaluate the performance regularity of a server. It needs to point out that the performance regularity does not receive proper attention and is generally overlooked. Its negligence is reflected in industrial benchmark reports in which intermediate results are generally omitted, with one exceptional case in SPECweb96.

The observation on the SPECweb96 benchmark studies suggests that the performance regularity of a server is independent of its system capacity, representing a distinct facet of the system performance. A new metric is needed for measuring the performance regularity of systems because the system capacity metric itself is unable to characterize the system performance regularity.

It is worth mentioning that the SPECweb96 specification [11] demands that the performance curve must consist of a minimum of 10 data points of different load levels, uni-

formly distributed across the range from zero to the maximum load. The average response time over repeated tests is used to represent the system performance at each selected system load. The detailed information refers to SPECweb96 benchmark specification [11].

We emphasize that the evaluation of the performance regularity of a server relies on the availability of performance curve. In the next section, we will present a metric called GPC for measuring the performance regularity of a server.

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 [7], 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. The basic idea can be illustrated in the following example.

Example 1. Consider a population of n individuals, with positive income denoted by a vector $z = (z_1, z_2, \ldots, z_n)$ in ascending order, i.e., $z_1 \leqslant z_2 \leqslant \cdots \leqslant z_n$, we can plot points $(k/n, S_k/S_n)$, where $S_0 = 0$, $S_k = \sum_{i=1}^{k} z_i$, $0 \leqslant k \leqslant n$. S_k represents the sum of the k lowest incomes in the population. *Lorenz curve* is then obtained by making piecewise connection between every pair of adjacent points, including the origin $(0, 0)$ and the end point $(1, 1)$. *Gini coefficient* is defined as twice the area between the line of 45 degree and the Lorenz curve.

It can be easily checked that the Gini coefficient lies between 0 and 1. For an extremely uneven distribution with the richest possessing all, the Gini coefficient is one, while a perfectly uniform distribution with everyone having the same income results in a Gini coefficient of zero. The Gini coefficient grows in proportion with the magnitude of inequality in data distribution, and the more uneven the distribution, the larger the Gini coefficient. The Gini coefficient is *scale independent*. As a measure of inequality, it could used to measure and compare the inequality of wealth distribution of countries with distinct GNPs (Gross National Product). For instance, [1] used the Gini coefficient for comparing household per capita income for Thailand and Mexico.

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 $(x_1, y_1), \ldots, (x_n, y_n)$, where $(x_1, \ldots, x_n) \in \mathcal{R}^n$ represents the vector of system throughput in the ascending order $(x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n)$, and $(y_1, \ldots, y_n) \in \mathcal{R}^n$ 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 construct the two normalized vectors $\overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$ and $\overline{y} = (\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n)$ by the following transformation:

$$\overline{x}_i = \frac{x_i}{x_n}, \quad \overline{y}_i = \frac{y_i}{\sum_{k=1}^{n} y_k}, \qquad 1 \leqslant i \leqslant n, \tag{1}$$

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 $\overline{y}$, we construct a vector $\overline{Y} = (\overline{Y}_1, \ldots, \overline{Y}_n)$, where $\overline{Y}_i = \sum_{k=1}^{i} \overline{y}_k = \sum_{k=1}^{i} y_k / \sum_{k=1}^{n} y_k$, $1 \leqslant i \leqslant n$. It can be verified that $\overline{x}_n = \overline{Y}_n = 1$. The curve referred to as *Lorenz performance curve* (LPC for short) can be constructed by making piecewise connection between every pair of adjacent points, $(0, 0)$, $(\overline{x}_1, \overline{Y}_1), \ldots, (1, 1)$. 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, $\overline{x}$ and $\overline{y}$. There exists the componentwise correspondence between the vectors $\overline{x}$ and $\overline{y}$: the ith component $\overline{Y}_i$ represents the normalized response time measured when the system throughput is $\overline{x}_i$.
2. The components in the vector $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$ are arranged in the increasing order, but may not be uniformly distributed, i.e., $\overline{x}_i - \overline{x}_{i-1} \neq \overline{x}_j - \overline{x}_{j-1}$ for some $i \neq j$.

Note that the components in the vector $\overline{y}$ may not have an increasing order. The sequence dependence of the vector $\overline{Y}$ on the vector $\overline{x}$ demands that the order of components in the vector y be fixed when the components in the vector $\overline{x}$ are determined. As a result, the term $\overline{Y}_k = \sum_{i=1}^{k} \overline{y}_i$ might not necessarily be the sum of k smallest values in $\overline{y}$, which is fundamentally different from example 1. Therefore, traditional Lorenz curve could be viewed as a special case of the LPC in the sense that all components in the vector $\overline{y}$ are in the ascending order. We are now in a position to give the definition of GPC.

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

$$\text{GPC} = 2 \int_0^1 \big(I(\tau) - L(\tau)\big)\, \mathrm{d}\tau. \tag{2}$$

Given a system performance curve, the algorithm for calculating the GPC is given as follows.

Algorithm 1. Calculating the GPC.
Input: A normalized throughput vector $(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$, a normalized response time vector $(\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n)$;
Output: Its corresponding GPC.

(1) $\overline{Y}_i = \sum_{k=1}^{i} \overline{y}_k$, $1 \leqslant i \leqslant n$
(2) $sum = (\overline{x}_1 \cdot \overline{Y}_1)/2.0$
(3) **for** $i = 2$ **to** n
(4) $sum = sum + (\overline{x}_i - \overline{x}_{i-1}) \cdot (\overline{Y}_i + \overline{Y}_{i-1})/2.0$
(5) **endfor**
(6) $\text{GPC} = 1.0 - 2.0 \cdot sum$

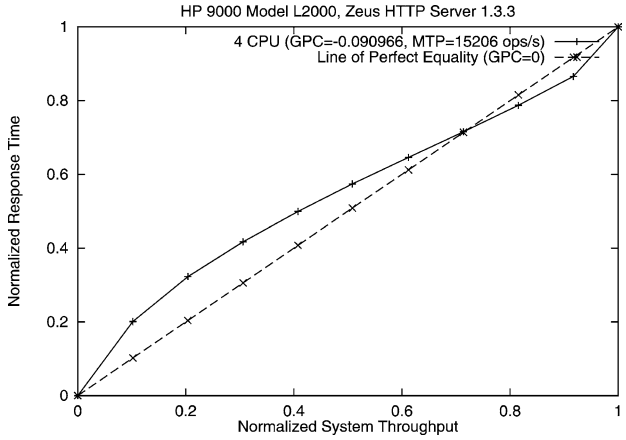

Figure 3. LPC of HP 9000/L2000.

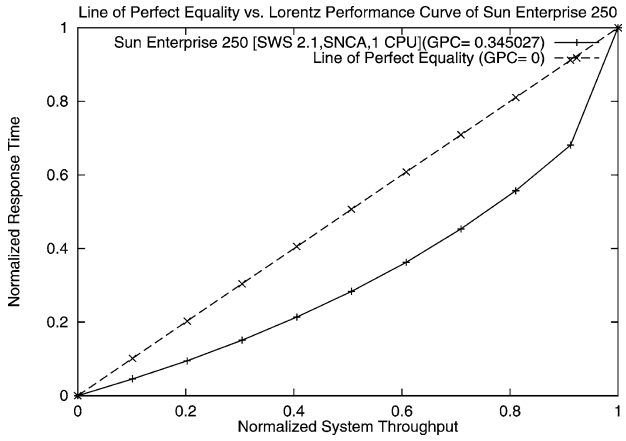

Figure 4. LPC of Sun Enterprise 250.

Throughout the paper, the LPC stands for *Lorenz performance curve*. In the following, we will present theorems illustrating a link between the GPC and the performance regularity.

As an immediate application of the algorithm, we transform the performance curves of the HP 9000/L2000 and Sun Enterprise 250 into the corresponding Lorenz performance curves and calculate the values of GPC. In Figures 3 and 4, the 45 degree line (line of perfect equality) serves a reference line. The inspection of Figure 4 indicates that Lorenz performance curve of the HP 9000/L2000 is initially concave, lying above the line of 45 degree, then becomes convex as system throughput approaches a saturation point, making its curve lying below the line of 45 degree. The presence of concavity in the Lorenz performance curve makes Gini coefficient small. Overall, the value of GPC is calculated as

-0.090966, reflecting the erratic overall performance behavior. In contrast, Figure 4 shows that Sun Enterprise 250 performs regularly in its working area, and the corresponding LPC is strictly convex, lying below the line of 45 degree. The value of Gini coefficient is calculated as 0.345027, much greater than that of the HP 9000/L2000.

The calculation results suggest a direct connection between the GPC and the performance regularity, indicating that the performance regularity can be quantitatively described by the GPC. Namely, the GPC of one corresponds to a perfect performance regularity, the GPC of zero is tantamount to a poor performance regularity, inasmuch as the negative GPC signifies the presence of erratic behaviors as illustrated in Figure 1. The following theorem gives a necessary condition that ensures the positiveness of the GPC.

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

Proof: Let $\overline{x}_0 = 0$ and $\overline{y}_0 = 0$. The Lorenz performance curve is constructed by making piecewise connection between every pair of adjacent points $(\overline{x}_{i-1}, \sum_{k=1}^{i-1} \overline{y}_k)$, and $(\overline{x}_i, \sum_{k=1}^{i} \overline{y}_k)$, $1 \leqslant i \leqslant n$. Since $\sum_{k=1}^{i} \overline{y}_k \leqslant \overline{x}_i$, for $1 \leqslant i \leqslant n$, all points are lying under the line of equality (the line of 45 degree). Hence the entire LPC is below the line of equality. By Definition 2, the 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. We start with a closely-related definition of the λ-weighted normalized response time, followed by a lemma and its proof.

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

$$\overline{y}_\lambda = \sum_{i=1}^{n} \lambda_i \overline{y}_i = \frac{\sum_{i=1}^{n} \lambda_i y_i}{\sum_{i=1}^{n} y_i}, \tag{3}$$

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

It can be verified that the weight sequence $\lambda = (\lambda_1, \ldots, \lambda_n)$, constructed from the workload vector x, is in descending order, i.e., $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$. 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 $(y_1, y_2, \ldots)$ and minimize the contributions from heavily loaded states $(y_n, y_{n-1}, \ldots)$. The reason that we use λ-weighted normalized response time instead of average response time is illustrated in the following example.

Table 1. Performance comparison.

System A								
$x^{[A]}$	10	20	30	40	50	60	70	80
$y^{[A]}$	100	100	1	1	1	1	1	200
System B								
$x^{[B]}$	10	20	30	40	50	60	70	80
$y^{[B]}$	1	1	1	1	1	100	100	200

Example 2. Consider two systems with the same response time y_n at the respective system capacity x_n, the performance data are tabulated in Table 1. Based on Definition 3, the λ-weighted normalized response time $\overline{y}_\lambda$ for the system A is calculated as 0.47, and for the system B is 0.165. The system A is apparently irregular by Definition 1, while the system B is regular. This is captured by their λ-weighted normalized response times, but not by the average response times, which are the same.

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

Lemma 1. $0 \leqslant \overline{y}_\lambda \leqslant 1$.

Proof: By Equation (3), the λ-weighted normalized response time can be expressed as follows:

$$\overline{y}_\lambda = \frac{\sum_{i=1}^{n} \lambda_i y_i}{\sum_{i=1}^{n} y_i} \leqslant \lambda_1 \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} y_i} = \lambda_1 \tag{4}$$

and

$$\overline{y}_\lambda \geqslant \lambda_n \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} y_i} = \lambda_n, \tag{5}$$

because the weight sequence λ is in the descending order, i.e.,

$$0 \leqslant \lambda_n \leqslant \overline{y}_\lambda \leqslant \lambda_1 \leqslant 1. \tag{6}$$

Then the lemma is proved. □

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.

Proof: Let $\overline{x}_0 = 0$ and $\overline{y}_0 = 0$. It follows by Algorithm 1 that the GPC is expressed as

$$\text{GPC} = 1.0 - 2.0 \cdot \sum_{i=1}^{n} \left[(\overline{x}_i - \overline{x}_{i-1}) \frac{\overline{Y}_i + \overline{Y}_{i-1}}{2} \right]. \tag{7}$$

Because $\lambda_i = 1 - (\overline{x}_i + \overline{x}_{i-1})/2$, $1 \leqslant i \leqslant n$, we have

$$\begin{aligned}
&\sum_{i=1}^{n} \left(\overline{x}_i - \overline{x}_{i-1}\right)\left(\overline{Y}_i + \overline{Y}_{i-1}\right) \\
&\quad = \frac{\sum_{i=1}^{n-1}[(\overline{x}_{i+1} - \overline{x}_{i-1}) \sum_{k=1}^{i} y_k] + (\overline{x}_n - \overline{x}_{n-1}) \sum_{k=1}^{n} y_k}{\sum_{k=1}^{n} y_k} \\
&\quad = \frac{\sum_{i=1}^{n}(2 - \overline{x}_i - \overline{x}_{i-1}) y_i}{\sum_{k=1}^{n} y_k} \\
&\quad = 2\frac{\sum_{i=1}^{n} \lambda_i y_i}{\sum_{i=1}^{n} y_i} = 2\overline{y}_\lambda. \qquad (8)
\end{aligned}$$

Therefore, we have

$$\mathrm{GPC} = 1.0 - 2\overline{y}_\lambda.$$

The proof is completed. □

Hence, the GPC is linearly related to the λ-weighted normalized response time. It directly follows Lemma 1 that the value of GPC lies between -1 and 1. Its range is different from traditional Gini coefficient. Notice that the traditional Lorenz curve is always below the line of equality. With the respect to the GPC, the sequence dependence of the vector $\overline{y}$ upon the vector $\overline{x}$, plus the relaxation on the order-preserving property among the components in the vector $\overline{y}$ could move the Lorenz performance curve above the line of 45 degree, thus the range of GPC is extended from [0, 1] to $[-1, 1]$. Theorem 2 has established the intimate relationship that relates the performance behavior (performance regularity) to the GPC. Such a connection allows us to quantitatively evaluate the performance regularity of a server.

4. Interpretation of GPC

To address the question of how to interpret the GPC value of a system, we first discuss a few reference cases.

- For a *perfect* system whose response time remains near zero and shoots up only when the workload reaches the system capacity, its GPC will be close to one.
- For a system whose response time remains high for light and heavy workloads, its GPC will be close to zero.
- For a system whose response time is very high when the workload is near zero, decreases sharply when the load increases, and rises back up when the load reaches the system capacity, its GPC will be negative.

Generally speaking, a negative GPC means an "abnormal" system; a non-negative GPC means a "regular" system. The workload range from zero to the system capacity is called the working range, in which the segment with relatively low response time is called the

desirable working range. Among the positive GPCs, a larger value means a larger desirable working range. For instance, consider two systems with the same capacity, defined as the workload under which the response time exceeds 3 seconds. Suppose the response time of the first system is near zero until the workload is 90% of the capacity, and that of the second system is above 2 seconds after 50% of the capacity. Even though the two have the same capacity, the first system is apparently better. Its desirable working area is up to 90% of the capacity, while the desirable working area of the second system is only up to 50%. Given its mathematical formulation, GPC will catch this difference.

In summary, system capacity should not be the sole metric for comparison. The performance regularity in the entire working range should also be compared. The proposed metric (GPC) quantifies this additional comparison. It is an abstraction of the response-time chart, but more than the chart because it is normalized, which removes the capacity-related factors; it is quantified, which allows an unambiguous direct comparison; and it is biased in favor of the low end of the capacity range and consequently captures the desirable working range under which the system performs best.

Although two systems with the same GPC can behave differently, the general trends of their performances are likely to be very similar after the capacity factors are removed by normalization. This is confirmed during our evaluation of real-system data, which is presented in the next section. If two systems behave vastly different over their working ranges, their GPC values will also differ greatly. Using Example 2, the GPC for system A is calculated as 0.06, reflecting the strong irregularity in its response time distribution, and that for system B is 0.67. The calculated results provide a quantitative description of their performance regularity differential. The comparison of two systems in terms of the performance regularity is thus reduced to comparing the values of their GPC.

It should be emphasized that GPC is measured based on the average response times over a sufficiently long period of time, during which the temporary system irregularity (due to other concurrent processes, scheduled tasks, temporary surge of one activity type) is smoothed out. One particular request under heavy load may have longer response time than another particular request under light load even for a regular system, but under normal conditions this should not happen consistently in an average sense. The purpose of GPC is to quantatively measure and compare the system regularity. While it serves as an indication of the overall performance behavior of systems in the working area, it does not point out the root causes of the problem by itself.

5. Assessment of system performance regularity

In this section, we will assess the performance regularity of systems reported in SPECweb96 [11] 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. To simplify terminology, we use MTP hereafter to represent maximum system throughput. We have chosen to focus on investigating:

1. The impact of number of CPUs on the performance regularity of a system.
2. The impact of different versions of operating system on performance regularity.

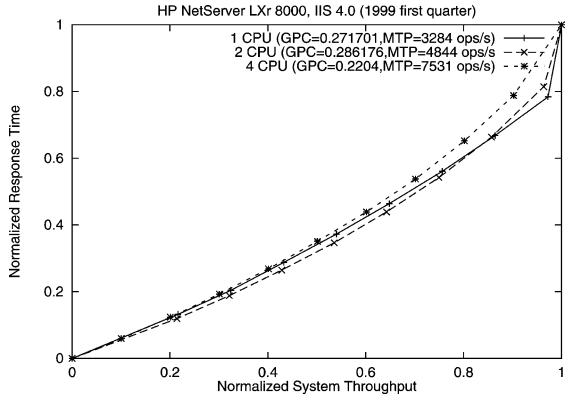

Figure 5. GPC vs. no. of CPUs.

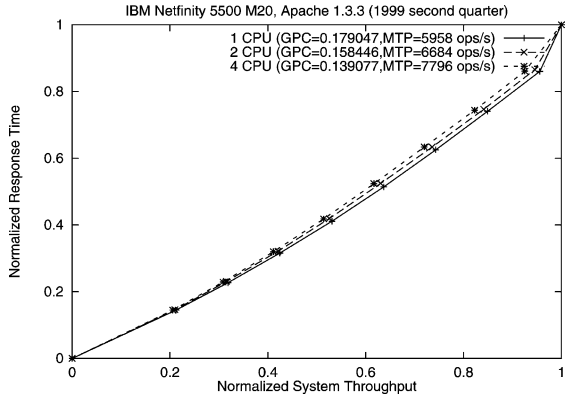

Figure 6. GPC vs. no. of CPUs.

It is well known that an increase in the number of CPUs in SMP system architecture is generally accompanied with an increase in system capacity such as maximum system throughput. It would be of interest to see its impact on the performance regularity.

In Figures 5–14, 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 (see `http://www.spec.org` for detail).

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 Figures 5, 7, 10, 13, 14 suggests that Microsoft and Sun Microsystem have 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.

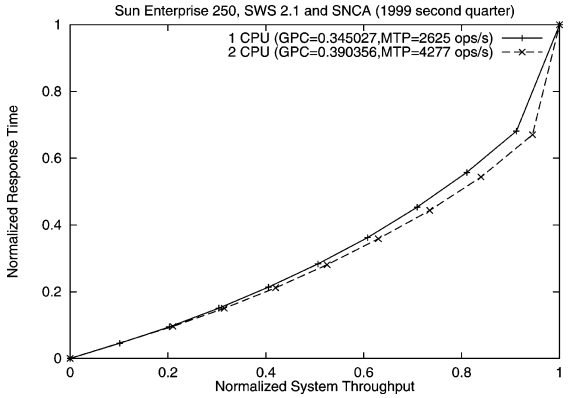

Figure 7. GPC vs. no. of CPUs.

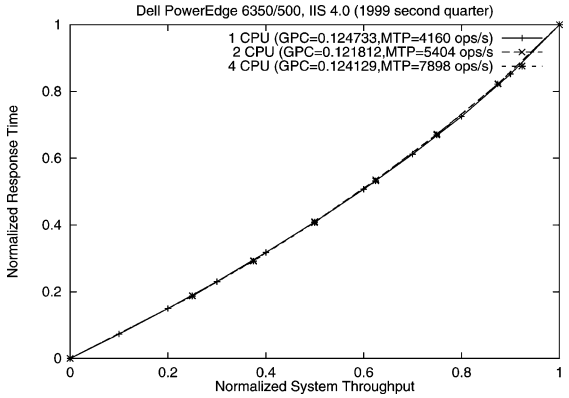

Figure 8. GPC vs. no. of CPUs.

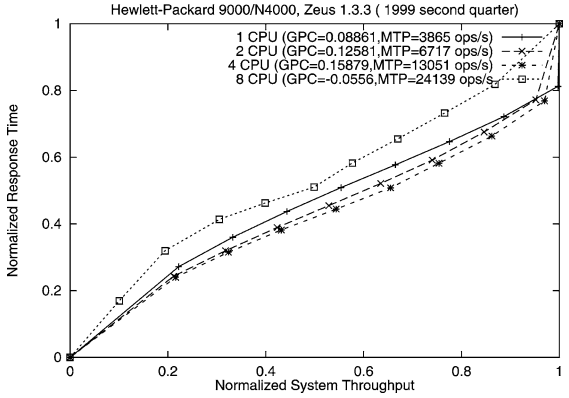

Figure 9. GPC vs. no. of CPUs.

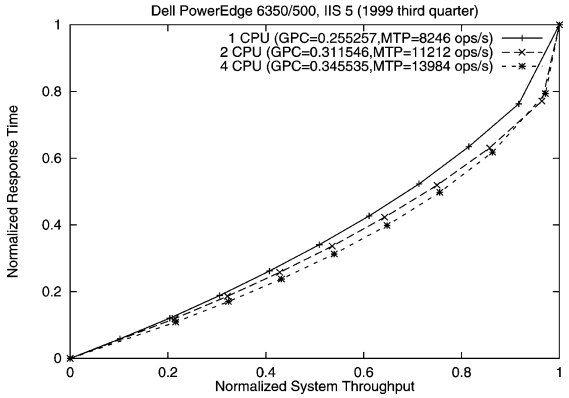

Figure 10. GPC vs. no. of CPUs.

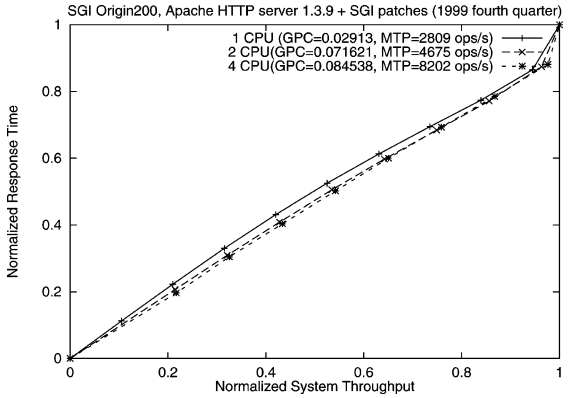

Figure 11. GPC vs. no. of CPUs.

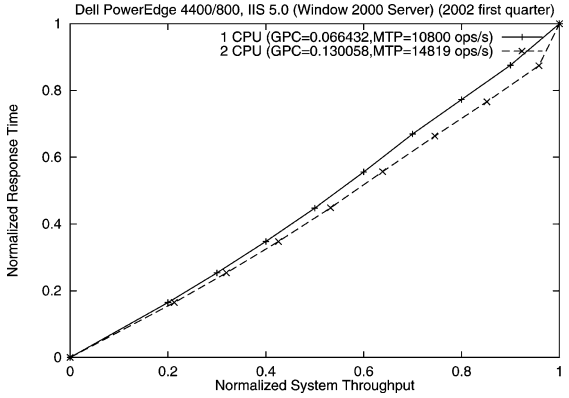

Figure 12. GPC vs no. of processors.

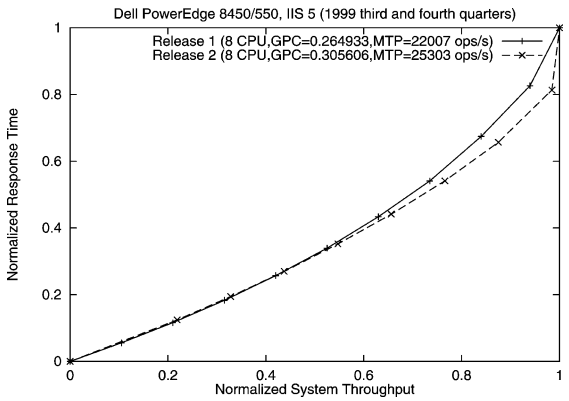

Figure 13. GPC vs. versions of Microsoft 2000.

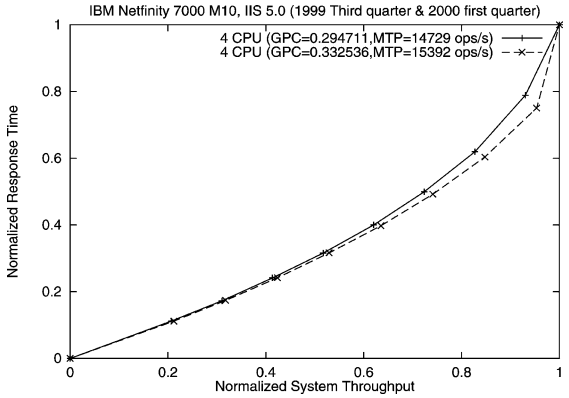

Figure 14. GPC vs. versions of Microsoft 2000.

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 (see Figure 8 for details). By contrast, Apache and Zeus HTTP servers in Figures 6, 9, 11 do not improve system performance regularity with an increase of the number of processors. The obtained results also suggest that for Apache HTTP server, a great deal of effort is needed to narrow its performance difference with other high-end HTTP servers, such as ones from Sun, Microsoft, and IBM, with a focus on improving the performance regularity by better utilizing SMP architecture.

The graphs of the GPC versus different versions of Window 2000 advanced server are showed in Figures 13 and 14. In Figure 13, 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 Windows 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 Figure 14, 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 Windows 2000 advanced server, the benchmark results were reported in the first quarter 2000, and the third quarter 1999, respectively.

The comparison results indicate that Microsoft Windows 2000 advanced server release candidate 2 consistently outperforms its predecessors in both system capacity and performance regularity, with the gain in the GPC by $(0.332536 - 0.294711 = 0.037825)$ in the case of IBM 7000/M10 model, and with the gain in the GPC by $(0.305606 - 0.264933 = 0.040673)$ in the case of Dell PowerEdge 8450/550 model. The improvement, though too small to be distinguished visually due to the striking similarity in the performance curve, is distinguishable quantitatively in the context of the GPC.

6. Conclusion

The paper has mainly focused on system performance regularity, instead of concerning with accurate benchmarking and identifying the root cause of the performance irregularity of servers as illustrated in Figure 1. We have shown that system capacity and performance regularity are two distinct aspects of system performance, and that the performance regularity of a system often does not necessarily correlate with its capacity. Measuring the performance regularity of a system is an important aspect of the system performance because a system typically operates below its maximum capacity. However, evaluating the performance regularity of a server is largely overlooked in system benchmarking study, as evidenced by the omission of intermediate results in almost all benchmark reports.

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

It is worth noting that the GPC is scale-independent and could be derived from the performance curve of the system with respect to the chosen capacity metric. For instance, it can be used to measure the performance regularity with respect to the performance metric such as the number of simultaneous connections as well. Measuring the performance regularity of Web servers with the Gini performance coefficient represents the first step towards the identification of performance problems in Web servers.

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

Acknowledgements

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

References

[1] O. P. Attanasio and M. Szekely, "Household saving in developing countries: Inequality, demographics and all that," World Bank Document, April 2000.
[2] G. Banga and P. Druschel, "Measuring the capacity of a web server," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997, pp. 244–263.
[3] S. L. Gaede, "Perspectives on the SPEC SDET Benchmark," Lone Eagle System Inc., January 1999, `http://www.loneagle.com/SDET/SDETPerspectives.html`
[4] J. Gray, *The Benchmark Handbook for Database and Transaction Processing Systems*, Morgan Kaufmann, New York, 1993.
[5] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Analysis*, McGraw-Hill, New York, 1988.
[6] K. Kant and Y. Won, "Server capacity planning for web traffic workload," *IEEE Transactions on Knowledge and Data Engineering*, September/October 1999, 731–747.
[7] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*, Academic Press, New York, 1979.
[8] Microsoft Corporation, *User Guide: Microsoft Web Capacity Analysis Tool Version 4.35 Windows 2000 Operating System*, Microsoft Corporation, 2000.
[9] M. Pendleton and G. Desai, "@Bench test report: Performance and scalability of Windows 2000," Doculabs, August 2000.
[10] J. Smith, "Charactering computer performance with a single number," *Communications of ACM* 31(10), 1988, 1202–1206.
[11] SPECweb96, 1996, `http://www.spec.org/osg/web96`
[12] SPECweb99, 1999, `http://www.spec.org/osg/web99`
[13] Transaction Processing Performance Council, "TPC Benchmark W (Web Commerce)," October 2001.
[14] B. L. Wong, *Configuration and Capacity Planning for Solaris Servers*, Prentice Hall PTR/Sun Microsystems Press, 1997.