

Optimizing System Configurations Quickly by Guessing at the Performance

Takayuki Osogami
IBM Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi
Kanagawa 242-8501, Japan
osogami@jp.ibm.com

Sei Kato
IBM Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi
Kanagawa 242-8501, Japan
seikato@jp.ibm.com

ABSTRACT

The performance of a Web system can be greatly improved by tuning its configuration parameters. However, finding the optimal configuration has been a time-consuming task due to the long measurement time needed to evaluate the performance of a given configuration. We propose an algorithm, which we refer to as Quick Optimization via Guessing (QOG), that quickly selects one of nearly best configurations with high probability. The key ideas in QOG are (i) the measurement of a configuration is terminated as soon as the configuration is found to be suboptimal, and (ii) the performance of a configuration is guessed at based on the measured similar configurations, so that the better configurations are more likely to be measured before the others. If the performance of a good configuration has been measured, a poor configuration will be quickly found to be suboptimal with short measurement time. We apply QOG to optimizing the configuration of a real Web system, and find that QOG can drastically reduce the total measurement time needed to select the best configuration. Our experiments also illuminate several interesting properties of QOG specifically when it is applied to optimizing Web systems.

Categories and Subject Descriptors

C.4 [Performance of systems]: Measurement techniques

General Terms

Performance, Theory, Measurement, Experimentation.

Keywords

Performance optimization, Web system, Configuration parameters, Regression, Ranking and Selection.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMETRICS'07, June 12–16, 2007, San Diego, California, USA.
Copyright 2007 ACM 978-1-59593-639-4/07/0006 ...\$5.00.

1. INTRODUCTION

Motivation

Maximizing performance with limited resources is a major goal in designing commercial Web systems. Adjusting configuration parameters such as `MaxClients`¹ can greatly improve the performance of a Web system, but the parameter tuning has been a time-consuming trial-and-error task for an expert. There is also a significant amount of research towards automating and speeding up parameter tuning of a Web system (see [3, 7, 12, 14, 15] and references therein), but a common issue in the prior work is the long measurement times required to find the best configuration, where the configuration parameters are set optimally to achieve the maximal performance. Standard benchmarks such as TPC and SPEC require to measure the performance of a configuration for tens of minutes. Measuring the performance of only a small subset of all of the possible configurations thus can take on the order of days or weeks.

Recently, Osogami and Itoko [11] propose an algorithm that can reduce the total measurement time needed to select the best configuration of a Web system by adjusting the measurement time for each configuration. This is in contrast to all of the other prior work, where the performance is measured for a fixed duration for all of the configurations considered. Unfortunately, the algorithm in [11] requires checkpoints, which save the system state when the measurement of a configuration is stopped, so that the measurement of the configuration can be resumed with the saved state. The measurement time is reduced at the expense of increasing complexity for the checkpoints. Partly because of the complexity of the checkpoints, the algorithm in [11] has not been applied to real Web systems.

We propose an algorithm, which we refer to as Quick Optimization via Guessing (QOG), that quickly selects one of nearly best configurations with high probability. QOG does not require checkpoints and yet reduces the measurement time for each configuration. We find that QOG can drastically reduce the total measurement time needed to select the best configuration of a real Web system. A key idea in QOG is that the performance of a configuration is guessed at based on the measured similar configurations, so that better configurations are more likely to be measured before the others. If a good configuration has been measured, the measurements of poor configurations can be quickly terminated

¹`MaxClients` specifies the maximum number of users that can be connected to the Web system at the same time.

when they are found to be suboptimal. Although the ideas of guessing at the performance and terminating measurement as soon as possible may be frequently used by experts in parameter tuning, these ideas need to be formalized to automate parameter tuning. We formally specify how to guess at the performance and when to terminate measurement, and prove that QOG finds a nearly best configuration with high probability under the conditions frequently assumed in the literature.

Related work

QOG is complementary to the prior work on parameter tuning of Web systems. The focus of most of the prior work is on which configurations should be measured but not on how long to measure. For example, [12] selects, at random, the configurations and measures each selected configuration for a fixed duration. The best configuration is then estimated by regression based on the performances of measured configurations. In [14], the best configuration is searched by a hill-climbing algorithm. Hill-climbing algorithms are also used in [3, 7, 15] for optimizing the configuration of a Web system at run time under changing workload. In [3, 7, 14, 15], the performance of a configuration is measured for a fixed duration at each iteration of the hill-climbing algorithms. The ideas in QOG can be used to eliminate unnecessary measurement time in the existing hill-climbing algorithms. The only work that considers reducing the measurement time of each configuration is [11], which unfortunately requires checkpoints and has not been applied to real Web systems.

There is also a significant amount of work, known as ranking and selection algorithms, for reducing the total *simulation* time for finding the best simulated system [2, 13]. All of the existing ranking and selection algorithms exploit checkpoints to reduce the total simulation time, since checkpoints are much easier in a computer *simulation*. As we show in [10], QOG also improves upon existing ranking and selection algorithms, which we briefly review below.

A ranking and selection algorithm finds, with high probability ($\geq 1 - \alpha$ for a specified α), the configuration π_b having the best mean performance μ_b , where the goal is to minimize the total simulation time. Unless otherwise stated, we assume that a larger mean performance is better. It is common to accept an error within an *indifference zone* δ , so that any configuration whose mean performance is $> \mu_b - \delta$ is considered to be one of the “best” configurations.

Standard ranking and selection algorithms assume that simulating a system yields samples from a normal distribution, since the simulated performances can often be made arbitrarily close to independent normal random variables by appropriate batching [6]. For example, the throughput of a single server queue in a closed system can be evaluated by simulation, where the j -th sample, Y_j , is the throughput in the j -th period. Although a sample, Y_j , may not have a normal distribution, the average of a batch of B samples, $X_\ell = \frac{1}{B} \sum_{j=1}^B Y_{\ell B+j}$, can often be made arbitrarily close to a normal random variable by letting B become large. In addition, X_ℓ and $X_{\ell'}$ become independent as $B \rightarrow \infty$ for $\ell \neq \ell'$. Thus, the throughput can be estimated by the average of the sample batches, $\{X_\ell\}$, which can be made arbitrarily close to i.i.d. normal random variables.

Simpler ranking and selection algorithms consist of two stages and require a checkpoint once for each configuration. For example, in [9], a fixed number of samples are collected

from each configuration in Stage 1. Based on the samples collected in Stage 1, some of poor configurations are eliminated before Stage 2 without being simulated with high accuracy. The number of samples to be collected in Stage 2 is determined for each configuration based on the sample variances of the samples collected in Stage 1. At the end of Stage 2, the configuration having the maximum simulated performance is selected as the best configuration.

The two-stage algorithm can be extended to more than two stages or to a “sequential-stage” algorithm, where some of poor configurations are eliminated after each stage until only the best configuration is left [5]. Although a sequential-stage algorithm can reduce the total number of samples, it requires a checkpoint after *every* sample. Too often checkpoints can be expensive even in computer simulations.

To balance the number of samples and the number of checkpoints, Hong and Nelson propose a two-stage algorithm that makes decisions sequentially [4], which we refer to as the HN algorithm. HN requires checkpoints at most once for each configuration (*i.e.* two-stage). Also, HN stops simulating a configuration at any time during Stage 2 when sufficient samples have been collected to make correct decisions (*i.e.* sequential-decision). Due to this sequential decision making, HN uses far fewer samples than classical two-stage algorithms. However, HN is a rather straightforward transformation of the sequential-stage algorithm [5] into two stages, and the samples collected in Stage 2 are not fully utilized to reduce the total number of samples.

Summary of contributions

This paper contributes to the literature both from the theory side and the systems side.

Our contribution on the theory side is QOG, the first algorithm for reducing, without checkpoints, the measurement time needed to select the best configuration. Unlike the existing ranking and selection algorithms, QOG reduces the total measurement time by guessing at the performances of configurations so that better configurations are likely to be measured before the others. As we show in [10], QOG also improves upon existing ranking and selection algorithms in the sense that, when QOG is used with checkpoints and without guessing at the performance, QOG results in shorter measurement times and the same number of checkpoints as the HN algorithm.

Our contribution on the systems side is the optimization of a real Web system by QOG. We find that QOG can drastically reduce the total measurement time, while selecting one of nearly best configurations. Our experiments also illuminate several interesting properties of QOG. For example, even if QOG is terminated before all of the configurations are measured, a nearly best configuration is likely to be selected. Also, QOG can reduce the total measurement time significantly even if the performance guessed at is always wrong (the measurement time is reduced further with correct guesses). The reduction of the total measurement time is partly due to *unimportant* mistakes that QOG makes in intermediate steps. Despite these unimportant mistakes, QOG can select one of nearly best configurations after all of the configurations are measured.

The rest of the paper is organized as follows. We propose QOG in Section 2, and apply QOG to parameter tuning of a real Web system in Section 3. Section 4 proves some of the properties of QOG.

2. QUICK OPTIMIZATION VIA GUESSING

In this section, we propose the Quick Optimization via Guessing (QOG) algorithm. We start by formally describing the properties of QOG in Section 2.1. QOG is introduced in Section 2.2, and some of the components of QOG are detailed in Sections 2.3 and 2.4.

2.1 Properties of QOG

QOG selects, with high probability, one of nearly best configurations under the i.i.d. normal assumption, which is frequently applied in the literature. Specifically, the i.i.d. normal assumption requires that (i) measuring the performance of a configuration yields a sequence of samples from a normal distribution and (ii) the samples are independent of each other. See [6] for sufficient conditions so that the i.i.d. normal assumption holds (approximately). For example, when the throughput of a Web system is measured, a sample may be collected every period of a fixed length, so that a sample represents the number of requests completed per second during a period. The sequence of samples can be made closer to i.i.d. normal random variables by choosing the period longer. Formally, let $\mathcal{C} = \{\pi_1, \dots, \pi_k\}$ be the set of candidate configurations, where $k < \infty$ is the number of candidate configurations. A sample of the simulated performance of configuration π_i is assumed to have a normal distribution with mean μ_i and standard deviation σ_i for each $\pi_i \in \mathcal{C}$. Let π_b be any one of the best configurations such that $\mu_b \geq \mu_i, \forall \pi_i \in \mathcal{C}$. Let $\mathcal{B}_\delta = \{\pi_i \in \mathcal{C} \mid \mu_i > \mu_b - \delta\}$ be the set of nearly best configurations, where $\delta > 0$ is a specifiable parameter of QOG. Let $\mathcal{W}_\delta = \mathcal{C} \setminus \mathcal{B}_\delta$.

Under the i.i.d. normal assumption, QOG has the following properties:

THEOREM 1. *Let $\pi \in \mathcal{C}$ be the configuration selected by QOG. Then $\Pr(\pi \in \mathcal{B}_\delta) \geq 1 - \alpha$.*

A proof of the theorem is sketched in Section 4. In Theorem 1, $\alpha > 0$ is a specifiable parameter that determines an upper bound on the probability that the selected configuration is not nearly best. In practice, QOG selects a nearly best configuration with higher probability than that guaranteed by Theorem 1 particularly when $|\mathcal{C}|$ is large. We find that, when $|\mathcal{C}|$ is large, α can be specified more intuitively by considering $\beta = \alpha / (|\mathcal{C}| - 1)$, an upper bound on the probability that QOG makes a mistake in selecting the better of two configurations having performances separated by δ . As we will see in Section 2.2, QOG directly compares $(|\mathcal{C}| - 1)$ pairs of configurations, and β specifies an upper bound on the probability that QOG makes an error in each comparison (see also Inequality (4) in the proof of Theorem 1).

QOG can be used even when the i.i.d. normal assumption does not hold, although Theorem 1 would not be guaranteed without the i.i.d. normal assumption. In our experiments with a real Web system in Section 3, we will see that measured performances can be made *close* to i.i.d. normal random variables by batching, and that QOG can select one of nearly best configurations even if the i.i.d. normal assumption is only approximately satisfied.

2.2 The QOG algorithm

In QOG, the measurement of a configuration is terminated as soon as it is found to be suboptimal. To reduce the measurement time, the performances of configurations are guessed at, so that better configurations are more likely to

```

01. Let  $\mathcal{D} = \mathcal{C}$  and  $\overline{X}^* = -\infty$ .
02. Let  $h$  be the smallest  $h$  satisfying (2).
03. while  $\mathcal{D} \neq \emptyset$ 
04.   Guess best  $\pi_i \in \mathcal{D}$ , and let  $\mathcal{D} = \mathcal{D} \setminus \{\pi_i\}$ .
05.   Collect  $n_0$  samples from  $\pi_i$ .
06.   Calculate the sample variance,  $S_i^2$ .
07.   Let  $N_i = \max\{n_0, \lceil h^2 S_i^2 / \delta^2 \rceil\}$ .
08.   for  $r = n_0, \dots, N_i$ 
09.     Let  $W_i(r) = \max\{0, \frac{\delta}{2r} (h^2 S_i^2 / \delta^2 - r)\}$ .
10.     if  $\overline{X}_i^{(r)} < \overline{X}^* - W_i(r)$ 
11.       break
12.     elseif  $\overline{X}_i^{(r)} > \overline{X}^* + W_i(r)$ 
13.       collect  $N_i - r$  samples from  $\pi_i$ ;
14.       if  $\overline{X}_i^{(N_i)} > \overline{X}^*$ , set  $\pi^* = \pi_i$  and  $\overline{X}^* = \overline{X}_i^{(N_i)}$ ;
15.       break
16.     endif
17.     if  $r < N_i$ , collect  $(r + 1)$ -th sample from  $\pi_i$ .
18.   endfor
19. endwhile
20. Select  $\pi = \pi^*$  as the best.

```

Figure 1: The QOG algorithm, where $\overline{X}_i^{(r)}$ is the sample mean of the first r samples from π_i .

be measured before the others. In this section, we describe QOG without specifying how to guess at the performance. A particular algorithm for guessing at the performance is described in Section 2.4, but QOG can use any algorithm for guessing at the performance. Figure 1 summarizes QOG, which we elaborate upon below.

QOG first selects a configuration $\pi_i \in \mathcal{C}$ that is most likely to be the best by *guessing at* the performances. Without loss of generality, let $\pi_i = \pi_1$ be the selected configuration. The performance of π_1 is measured for at least n_0 samples (*e.g.* the throughputs of the first n_0 periods are measured)². When n_0 samples have been collected, the sample variance, S_1^2 , of the n_0 samples is calculated as

$$S_1^2 = \frac{1}{n_0 - 1} \sum_{\ell=1}^{n_0} \left(X_{1,\ell} - \overline{X}_1^{(n_0)} \right)^2,$$

where $X_{1,\ell}$ is the ℓ -th sample from π_1 , and $\overline{X}_1^{(n_0)}$ is the sample mean of the n_0 samples. The sample variance is then used to determine the number, N_1 , of samples needed for measurement with the full accuracy as

$$N_i = \max\{n_0, \lceil h^2 S_i^2 / \delta^2 \rceil\}, \quad (1)$$

where $i = 1$. The measurement of π_1 is terminated after N_1 samples are collected, and the sample mean of the N_1 samples is calculated as

$$\overline{X}_1^{(N_1)} = \frac{1}{N_1} \sum_{\ell=1}^{N_1} X_{1,\ell}.$$

Here, h is a constant which we refer to as the confidence parameter. Roughly speaking, h is chosen so that the better of

²In theory, n_0 can be any integer ≥ 2 , but too small an n_0 can result in a greater number of samples. We set $n_0 = 20$ for our experiments in Section 3. In practice, we find that a larger n_0 only increases the total measurement time.

two configurations having performances separated by δ can be correctly identified with probability $\geq \alpha/(|\mathcal{C}| - 1)$, when the performances are measured with the full accuracy specified using h . Formally, the confidence parameter is defined as the smallest h satisfying

$$\begin{aligned} & \mathbb{E} \left[\left(1 + \exp \left(\sqrt{\frac{h^2 Q_1}{n_0 - 1}} \left(1 + \frac{Q_1}{Q_2} \right) Z + \frac{h^2 Q_1}{n_0 - 1} \right) \right)^{-1} \right] \\ & \leq \frac{\alpha}{|\mathcal{C}| - 1}, \end{aligned} \quad (2)$$

where Z is a standard normal random variable, Q_1 and Q_2 are χ^2 random variables with degree of freedom $n_0 - 1$, and the three random variables are independent. Further explanation of h and an algorithm for calculating h are postponed to Section 2.3.

After π_1 is measured, let $\pi^* = \pi_1$ be the provisional best configuration (*i.e.* $\pi^* = \pi_1$ is the best found so far), let $\bar{X}^* = \bar{X}_1^{(N_1)}$ be the sample mean of π^* , and let $\mathcal{D} = \mathcal{C} \setminus \{\pi_1\}$ be the set of configurations to be measured. Before a configuration in \mathcal{D} is measured, the performances of configurations in \mathcal{D} are guessed at. The configuration $\pi_i \in \mathcal{D}$ having the best estimated, by guessing, performance is measured next. After the performance of π_i is measured, \mathcal{D} is updated as $\mathcal{D} = \mathcal{D} \setminus \{\pi_i\}$, and π^* and \bar{X}^* are updated as needed.

Unlike the first configuration π_1 , however, QOG terminates the measurement of π_i as soon as π_i is found to be worse than π^* . The performance of π_i is measured for at least n_0 samples. When n_0 samples have been collected, QOG determines an *upper bound*, N_i , on the number of samples to be collected from π_i using Equation (1), where S_i^2 is the sample variance of the n_0 samples from π_i . The performance of π_i would be measured with the full accuracy, if N_i samples were collected. However, QOG often terminates the measurement of π_i before N_i samples are collected. More specifically, after we collect the r -th sample for $r \geq n_0$, a criterion $W_i(r)$ is calculated as

$$W_i(r) = \max \left\{ 0, \frac{\delta}{2r} (h^2 S_i^2 / \delta^2 - r) \right\}.$$

Observe that $W_i(r)$ is decreasing with r for $r < N_i$ and that $W_i(r) = 0$ for $r \geq N_i$. Roughly speaking, QOG determines that π_i is better or worse than π^* when the difference between \bar{X}^* and $\bar{X}_i^{(r)}$ becomes larger than the criterion, *i.e.* when $|\bar{X}^* - \bar{X}_i^{(r)}| > W_i(r)$, where $\bar{X}_i^{(r)}$ is the sample mean of the r samples from π_i . When π_i is determined to be worse than π^* , the measurement is terminated after the r -th sample. When π_i is determined to be better than π^* , the performance of π_i is measured with the full accuracy by collecting N_i samples from π_i . Note that a provisional best should be measured with the full accuracy since it will be compared against other configurations. A detail is that, if the performance of π_i measured with the full accuracy, $\bar{X}_i^{(N_i)}$, is smaller than \bar{X}^* , π_i will be determined to be worse than π^* contrary to our previous determination. As a result, QOG is less likely to determine that a worse configuration is better than to determine that a better configuration is worse, which turns out to be an important property of QOG as we will see in Section 3.2.

Formally, QOG takes one of the following three actions based on how $\bar{X}_i^{(r)}$ compares against \bar{X}^* . Case (i): If $\bar{X}_i^{(r)} < \bar{X}^* - W_i(r)$, we determine that π_i is suboptimal and stop

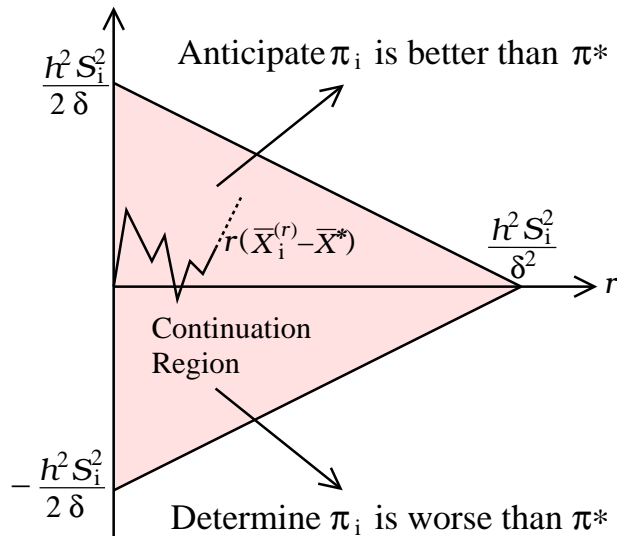


Figure 2: QOG determines whether π_i is better or worse than the provisional best π^* when $r(\bar{X}_i^{(r)} - \bar{X}^*)$ exits the continuation region.

measuring π_i (*i.e.* π^* eliminates π_i). Case (ii): If $\bar{X}_i^{(r)} > \bar{X}^* + W_i(r)$, we *anticipate* that π_i is better than π^* , and measure the performance with the full accuracy, collecting samples from π_i to the upper bound, N_i . Let $\bar{X}_i^{(N_i)}$ be the sample mean of the N_i samples. If $\bar{X}_i^{(N_i)} > \bar{X}^*$, we update $\pi^* = \pi_i$ and $\bar{X}^* = \bar{X}_i^{(N_i)}$ (*i.e.* π_i eliminates π^* as we anticipated). If $\bar{X}_i^{(N_i)} \leq \bar{X}^*$, we update neither π^* nor \bar{X}^* (*i.e.* π^* eliminates π_i , contrary to our anticipation). Case (iii): If $\bar{X}^* - W_i(r) \leq \bar{X}_i^{(r)} \leq \bar{X}^* + W_i(r)$, r samples do not suffice to differentiate between π_i and π^* , and we continue measuring the performance of π_i .

QOG continues measuring configurations in \mathcal{D} as described above until all of the configurations have been measured. When \mathcal{D} becomes empty, QOG selects $\pi = \pi^*$ as the best configuration. Finally, note that QOG can also be used to find a configuration having the *minimum* mean performance, by simply replacing each $X_{i,\ell}$ by $-X_{i,\ell}$.

2.3 Confidence parameter

2.3.1 Intuitions

Figure 2 provides a further intuition behind the choice of the criterion, $W_i(r)$, and the confidence parameter, h . Observe that $[-r W_i(r), r W_i(r)]$ for $0 \leq r \leq h^2 S_i^2 / \delta^2$ defines a triangular area, which we refer to as the continuation region. QOG determines (or anticipates) that π_i is either better or worse than π^* when $r(\bar{X}_i^{(r)} - \bar{X}^*)$ exits the continuation region. Specifically, if $r(\bar{X}_i^{(r)} - \bar{X}^*)$ exits the continuation region through the lower (or respectively, upper) edge, then QOG determines (or respectively, anticipates) that π_i is worse (or respectively, better) than π^* .

We want to choose the continuation region so that QOG makes correct decisions with high probability, while the number of samples needed to exit the continuation region is minimized. Observe that the size of the continuation region de-

depends on three parameters, δ , S_i^2 , and h . Since δ is fixed so that it reflects our indifference and the sample variance S_i^2 cannot be controlled, only the confidence parameter h can be used to determine the size of the continuation region. A larger h implies a larger continuation region, which in turn makes QOG more conservative in the sense that it collects stochastically more samples before determining whether π_i is better or worse than π^* . Hence, QOG makes correct decisions with higher probability with a larger h , but QOG requires stochastically more samples with a larger h . We choose the confidence parameter as the smallest h with which we can prove Theorem 1 (see Section 4).

2.3.2 Calculating the confidence parameter

We find the smallest h satisfying (2) by using a binary search, where the left hand side, $f_{n_0}(h)$, of Equation (2) is evaluated by Monte Carlo integration. Alternatively, one can evaluate $f_{n_0}(h)$ by numerical (Gaussian) integration, which is in general more efficient than Monte Carlo integration. We use Monte Carlo integration due to its simplicity, since the confidence parameter needs to be calculated only once in QOG and it can be calculated in a few seconds using Monte Carlo integration.

If $f_{n_0}(h)$ is a monotonic function of h , a binary search is guaranteed to find the smallest h satisfying (2). Unfortunately, a proof of the monotonicity of $f_{n_0}(h)$ does not appear to be straightforward, even though our numerical experiments suggest $f_{n_0}(h)$ is a decreasing function of h . It is however easy to show that upper and lower bounds of $f_{n_0}(h)$ are decreasing with h :

PROPOSITION 1. *Let Z be a standard normal random variable, and let Q_1 and Q_2 be χ^2 random variables with degree of freedom $n_0 - 1$, where the three random variables are independent. Let*

$$g_{n_0}(h) = \Pr \left(Z \leq -\frac{Q_1}{n_0 - 1} \left(\frac{\kappa^2}{4h^4} + \frac{Q_1}{(n_0 - 1)h^2} \left(1 + \frac{Q_1}{Q_2} \right) \right)^{-\frac{1}{2}} \right),$$

where $\kappa = 1.702$. Then,

$$g_{n_0}(h) - 0.01 < f_{n_0}(h) < g_{n_0}(h) + 0.01$$

for $h > 0$, and $g_{n_0}(h)$ is a decreasing function of h .

A proof of the proposition is provided in [10]. Notice, however, that the smallest h is merely preferable to a larger h , since the smallest h minimizes the number of samples needed in QOG, but a larger h also guarantees Theorem 1. Our binary search will always find an h satisfying (2), even if $f_{n_0}(\cdot)$ was not monotonic.

2.4 Guessing at the performance

The efficiency of QOG partly relies on guessing at the performance of configurations that have not been measured. Recall that different configurations of a Web system have different values of configuration parameters. The performance of a configuration thus can be guessed at based on the performances of measured similar configurations, which have similar values of configuration parameters, for example by regression or by other machine learning techniques [8]. However, unlike standard settings where machine learning techniques apply, the measured performances in QOG

have varying accuracies, since the measurement of a configuration is terminated as soon as the configuration is found to be suboptimal. Also, at an early stage of QOG, the performance of a configuration needs to be guessed at based on few measured configurations, while more measured configurations can be used to guess at the performance at later stages. In this section, we introduce a particular regression technique which we can use in QOG.

2.4.1 Regression function

In regression techniques, the performance of a system is modeled as a regression function, which is a function of configuration parameters. The coefficients of the regression function are estimated based on the performances of already measured configurations. The regression function with the estimated coefficients can then be used to guess at the performance of a configuration that has not been measured. The suitable form of the regression function depends on the particular system under consideration.

Our preliminary experiments suggest that a quadratic function of configuration parameters can provide a reasonable guess of the performance of a Web system, since the performance of a Web system often improves with the value of a configuration parameter up to a point and degrades after that point, although this does not always hold. More formally, let $\mathbf{v} = (v_1, \dots, v_n)$ be the configuration parameters. Our regression function can be written as

$$f(\mathbf{v}) = \sum_{r \in \mathcal{R}} c_r g_r(\mathbf{v}),$$

where, for each $r \in \mathcal{R}$, c_r is a coefficient to be determined and $g_r(\mathbf{v})$ is a monomial such as a quadratic term $v_i v_j$, a linear term v_i , and a constant. Without loss of generality, we assume that $g_r(\mathbf{v}) \neq g_{r'}(\mathbf{v})$ for $r \neq r'$. Note that there are $(n+1)(n+2)/2$ distinct monomials of degree ≤ 2 . When $|\mathcal{R}| = (n+1)(n+2)/2$, the quadratic regression function is called a full regression function.

2.4.2 Minimizing the weighted quadratic error

The coefficients of the regression function are determined so that the weighted quadratic error (WQE) is minimized. Formally, let \mathcal{M} be the set of already measured configurations, let $\mathbf{v}^{(m)} = (v_1^{(m)}, \dots, v_n^{(m)})$ be the values of the configuration parameters for $\pi_m \in \mathcal{M}$, let L_m be the number of samples collected from π_m , and let \bar{X}_m and S_m^2 be the sample mean and sample variance, respectively, of the L_m samples from π_m . Note that $L_m \leq N_m, \forall \pi_m \in \mathcal{M}$. Then, WQE is defined as

$$\text{WQE} = \sum_{\pi_m \in \mathcal{M}} \frac{L_m}{S_m^2} \left(\bar{X}_m - f(\mathbf{v}^{(m)}) \right)^2.$$

Observe that the quadratic error for configuration π_m is weighted by L_m/S_m^2 . It makes intuitive sense that the weight is proportional to the number of samples, L_m , since each sample should have an equal contribution. The weight is divided by the sample variance, S_m^2 , since the measured performance has less reliability when the sample variance is larger. Our intuition behind the choice of dividing by S_m^2 is as follows. The sample variance is an unbiased estimator of the true variance, and if the true variance is x times larger, we would need x times more samples so that the sample mean has the same variance (*i.e.* so that the measured performance has the same reliability).

To minimize WQE, the coefficients are determined so that $\frac{\partial}{\partial c_u} \text{WQE} = 0$ for all $u \in \mathcal{R}$. This is equivalent to solving the following system of linear equations:

$$\sum_{\pi_m \in \mathcal{M}} \frac{L_m}{S_m^2} g_u(\mathbf{v}^{(m)}) \left(\bar{X}_m - \sum_{r \in \mathcal{R}} c_r g_r(\mathbf{v}^{(m)}) \right) = 0 \quad (3)$$

for all $u \in \mathcal{R}$. A unique solution of this system of equations exists if and only if $|\mathcal{R}|$ independent and consistent configurations have been measured, which can be easily verified by examining whether the corresponding coefficient matrix is non-singular.

2.4.3 Regression with increasing data

When a regression function has R coefficients, at least R independent and consistent configurations need to be measured before the coefficients can be determined so that WQE is minimized. If n configuration parameters are under consideration, the full regression function has $(n+1)(n+2)/2$ coefficients. Thus, at least $(n+1)(n+2)/2$ configurations need to be measured before the performances of configurations can be guessed at using the full regression function. We find that the total measurement time of QOG can often be reduced by guessing at the performances of configurations using a simpler regression function having fewer coefficients until the full regression function can be used.

In our experiments in Section 3, we use a quadratic function having only independent terms as our simpler regression function. The simpler regression function $f(\mathbf{v})$ does not have terms $v_j v_{j'}$ for $j \neq j'$ and has only terms such as a quadratic term v_j^2 , a linear term v_j , and a constant. Thus, the simpler regression function has only $2n+1$ coefficients.

Before the simpler regression function can be used, we measure a particular set of $2n+1$ configurations in the random order. The $2n+1$ configurations are selected so that they are independent and consistent, as follow. Let l_j and u_j , respectively, be the minimum and maximum values of v_j for $1 \leq j \leq n$. In one configuration, the values of the configuration parameters are chosen such that $v_j = z_j$ for $1 \leq j \leq n$, where $l_j < z_j < u_j$ for $1 \leq j \leq n$. In the other $2n$ configurations, exactly one of the n configuration parameter is set at either $v_j = l_j$ or $v_j = u_j$ and the remaining $n-1$ configuration parameters are set at $v_{j'} = z_{j'}$, where $j' \neq j$.

After the $2n+1$ configurations have been measured, the simpler regression function is used until $(n+1)(n+2)/2$ independent and consistent configurations are measured. Note that the first $(n+1)(n+2)/2$ configurations measured are not usually independent and consistent, and more configurations often need to be measured before the full regression function is used. We start using the full regression function when the coefficient matrix of the system of linear equation (3) becomes non-singular.

3. OPTIMIZING A REAL WEB SYSTEM

In this section, we apply QOG to optimizing the configuration of a real Web system, and study the effectiveness of QOG specifically when it is applied to the Web system. We consider 225 candidate configurations of the Web system running a stock brokerage application, and select, as best, the configuration having the maximum throughput (or the minimum mean response time). Before describing the details of our experiments, we start by summarizing the highlights of the results of our experiments:

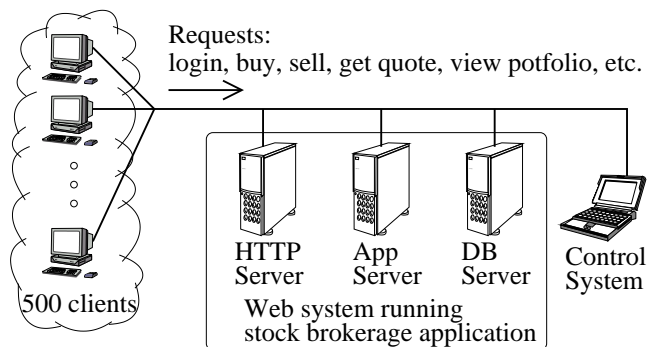


Figure 3: Experimental settings: A Web system runs a stock blockage application and receives requests from 500 clients. The Web system consists of an HTTP server, an application (App) server, and a database (DB) server.

Summary of results

- QOG can reduce the total measurement time by a factor of 22, while selecting one of the nearly best configurations.
- Even if QOG is terminated before all of the configurations are measured, a nearly best configuration is likely to be selected, since the better configurations are more likely to be measured before the others. Hence, the total measurement time of QOG can be reduced further.
- QOG can reduce the total measurement time significantly even if the throughput guessed at is always incorrect. Correct guesses can further reduce the total measurement time.
- QOG can select a nearly best configuration, although it frequently makes *unimportant* mistakes in intermediate comparisons. It turns out that these unimportant mistakes greatly contribute to reducing the measurement times.
- When there are several nearly best configurations, QOG with random guesses can also reduce the *mean* total measurement time drastically. However, the variability of the total measurement time with random guesses can be much larger than that with guesses based on regression.

Below, the settings of our experiments are described in Section 3.1, and the results are discussed in Section 3.2.

3.1 Experimental settings

3.1.1 Settings of our Web system

In our experiments, we consider a Web system running an online stock brokerage application as is illustrated in Figure 3. The Web system receives standard trading operations such as *buy*, *sell*, and *get quote* from 500 clients, so that the Web system is operated at a high load. We will select an optimal set of the values of the four configuration parameters shown in Table 1 so that the throughput of the Web system is maximized (or the mean response time is minimized) at the high load. Below, we describe details of our experimental settings.

We use five machines (IBM eServer xSeries 330), each of which has 1 GB main memory and 1.4 GHz dual CPUs (Intel Pentium III) and runs either Linux (RedHat Linux V8.0) or Windows (Microsoft Windows 2000 Professional). Three Linux machines run an HTTP server (IBM HTTPServer

Configuration parameter	Candidate values
MaxClients	100, 200, 300, 400, 500
ThreadPoolMax	10, 20, 30, 40, 50
HeapMax	128, 192, 256 (MB)
PSCacheSize	10, 20, 30

Table 1: Configuration parameters to be tuned.

V1.3), an application server (IBM WebSphere Application Server V5.0), and a database server (IBM DB2 UDB V8.1), respectively, and these form the Web system. Another Linux machine generates the workload, emulating 500 clients issuing requests to the Web system. The Windows machine manages and monitors the experiments.

The four configuration parameters in Table 1 are chosen out of many configuration parameters, based on our preliminary experiments, so that the performance of the HTTP server, the application server, and the database is each affected by at least one of the four configuration parameters. `MaxClients` specifies the maximum number of sessions (or clients) that can be connected to the HTTP server at the same time. If `MaxClients` is too small, resources are under-utilized. However, too large a `MaxClients` can increase the time needed for context switches, which can degrade the performance. `ThreadPoolMax` and `HeapMax` are configuration parameters of the application server. `ThreadPoolMax` specifies the maximum number of threads in the thread pool that is used by the sessions. Every active session requires a thread. If no threads are available for a new session, the session will have to wait until a thread becomes available. However, too many threads can increase the time needed for context switches. `HeapMax` specifies the maximum size of the Java Virtual Machine (JVM) on which the application server runs. Note that *initial* heap size is fixed at 128 MB. If `HeapMax` is set too small, the garbage collector runs frequently, which can degrade the performance. However, setting `HeapMax` larger than necessary results in allocating extra memory for JVM. `PSCacheSize` (prepared statement cache size) specifies the number of prepared statements of database queries that can be stored in the Statement Cache. When a database query is issued, a statement must be prepared before it can be executed. The prepared statement in the Statement Cache can be reused when the database query is issued multiple times, but setting `PSCacheSize` larger than necessary results in allocating extra memory for the Statement Cache.

For our benchmark application, we use Trade3³, which models an online stock brokerage application. Trade3 is a standard benchmark for Web systems with WebSphere Application Server, and Trade3 has also been used in the prior work on optimizing Web systems [12, 14].

The workloads are generated by Web Performance Tools V1.9 (WPT)⁴. Specifically, WPT emulates 500 clients, where each client issues a random request such as *buy*, *sell*, or other standard trading operations. After issuing a request, the client waits for the results of the request returned by the Web system. After receiving the results, the client pauses (thinks) for a random period, uniformly distributed between a second and eight seconds, before issuing a new request.

³<http://www.ibm.com/software/webservers/appserv/benchmark3.html>

⁴<http://www.alphaworks.ibm.com/tech/wptools>

All the experiments are managed and monitored by the Integrated Performance Analysis Tool (IPAT), a tool developed in our laboratory, running on a Windows machine. IPAT is used to control WPT, to change the values of the configuration parameters, and to measure and record the times when requests are issued and completed.

3.1.2 Details of measurement

For the purpose of our experiments, we first run the workload for 20 minutes on every configuration, and store the time when each request is completed and the response time of the request⁵. The stored data is then used to evaluate the total measurement time that would be needed to select the best configuration if QOG or another approach is used under the conditions that the requests are issued and processed as in the stored data. The use of the stored data allows fair comparison of different approaches. A Web system is so complex that it hardly ever reproduces the same response even if exactly the same workload is run on the same Web system with the same settings of the configuration parameters. Since the total measurement time of QOG and its variants depends on the particular responses of a Web system, different approaches should be compared to each other using the same responses of the Web system. The stored data also allows us to compare QOG against an idealized QOG where the guesses are always correct. The comparison of QOG against the idealized QOG will illuminate the effectiveness of our guesses based on regression. The use of the stored data also allows us to evaluate QOG and its variants thousands of times in varying ways, for example with different batch sizes and different regression functions, which would be infeasible without the stored data.

Since the behavior of a Web system is quite sensitive to minute conditions of the measurement, the performance of the 225 configurations are measured systematically. The 225 configurations are divided into nine groups of 25 configurations. The 25 configurations in each group have the same values of `HeapMax` and `PSCacheSize`. The configurations in each group are measured in sequence. Before each configuration is measured, the Web system is cleaned up by the following three steps: (i) the values of the configuration parameters are changed, (ii) the HTTP server, the application server, and the database server are restarted to reflect the changes in the configuration parameters and to initialize their caches and logs, and (iii) the Trade3 application is reset to initialize the Trade3 database and to log off all of the users. After the measurement of each group, the Trade3 database is re-populated with a new set of Trade3 users and stocks. We find that the performance of a Web system would degrade slowly without re-populating the Trade3 database.

When we start measuring the performance of a configuration, 500 users log in at random times so that all of the users are expected to log in within two minutes. The first three minutes is then considered to be the warmup period and ignored. Therefore, for each configuration, the completion time and the response time of each request are stored for 17 minutes after the warmup period. The stored data is used to form sample batches in QOG. When the throughputs are measured in QOG, the ℓ -th sample batch, X_ℓ , is the throughput during the ℓ -th period of B seconds, where B is the batch size. When the response times are measured

⁵The response time of a request is the time between the request is issued and completed.

in QOG, a sample batch is the average of B consecutive response times.

Although Section 3.2 only shows results when the throughputs are measured, we run the same experiments with measuring the response times. We find that the results when response times are measured are similar to those when the throughputs are measured, but they are not identical. Note that a configuration having the maximum throughput also minimizes the mean response time in closed systems such as ours. Further discussion on their similarities and differences is left as future work.

Below, the throughput of a configuration measured for 17 minutes after the warmup period is referred to as the true throughput. Also, the configuration having the largest true throughput is referred to as the best configuration, and the configuration having the i -th largest true throughput is referred to as the i -th best configuration. Table 2 lists the values of the configuration parameters (MaxClients, ThreadPoolMax, HeapMax, PSCacheSize) and the true throughput (requests per second) of the best three configurations, the median configuration, and the worst configuration of the 225 configurations under consideration.

Rank	Configuration	Throughput
1	(400, 40, 192, 20)	93.8
2	(400, 40, 128, 10)	93.7
3	(300, 40, 192, 20)	92.9
⋮	⋮	⋮
113	(400, 20, 256, 10)	80.4
⋮	⋮	⋮
225	(100, 50, 192, 10)	57.6

Table 2: The values of the configuration parameters (MaxClients, ThreadPoolMax, HeapMax, PSCacheSize) and the true throughput (requests per second). Only the best three configurations, the median configuration, and the worst configuration are shown.

3.2 Results

3.2.1 Reduction of measurement time by QOG

We start by evaluating the effectiveness of QOG in reducing the total measurement time needed to select the best configuration. Throughout Section 3.2, the measurement time of a configuration refers to the duration of the measurement of the configuration after the warmup period, and the total measurement time refers to the sum of the measurement times for all of the 225 configurations. Note that the total measurement time excludes the times needed for warmups and configuration changes, which can vary depending on the particular Web system under consideration.

In the prior work on optimizing Web system configurations, the measurement time is fixed for all of the configurations considered [3, 7, 12, 14, 15]. If the measurement time is 17 minutes for each configuration, the total measurement time of 225 configurations is 63 hours and 45 minutes, which is shown as the solid line labeled with “no QOG” in Figure 4. The total measurement time is shown as a function of the batch size, which has little effect when QOG is not used.

The solid line labeled with QOG in Figure 4 shows the total measurement time when QOG is used with fixed $n_0 =$

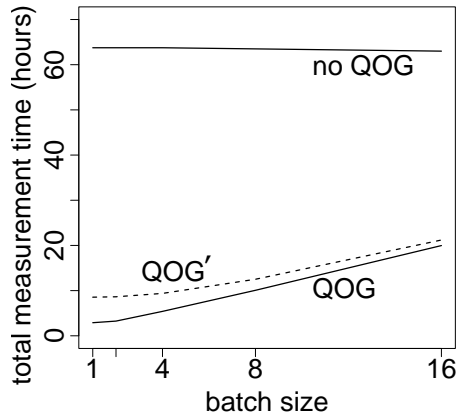


Figure 4: The total measurement time needed to determine the best configuration when QOG is used and when QOG is not used (solid lines). The dashed line shows the total measurement time of QOG', a modified QOG where the upper bound on the measurement time is fixed for all of the configurations.

20, $\delta = 3$, and $h = 1.13$. The value of h is chosen so that the largest upper bound on the measurement time calculated by Equation (1) is 17 minutes. With $h = 1.13$, the error probability for each comparison is bounded by $\beta = \alpha/(|C| - 1) = 0.3$. Note that each data point for QOG is generated as the average of 1,000 runs, since the total measurement time varies depending on the particular configurations measured before the full regression function is used (see Section 2.4.3). Observe that, when the batch size is one, the mean total measurement time is 2.9 hours, which is *only 1/22 of the time when QOG is not used*.

Since the total measurement time increases with the batch size, smaller batch sizes are preferable. Note that, in all cases studied in Figure 4, the selected configuration is one of the two best configurations. When the batch size is one, the average skewness of a batch is -0.52 and the average lag-1 autocorrelation is 0.27 under our conditions⁶. Since the i.i.d. normal random variables have zero skewness and zero lag-1 autocorrelation, the batches only approximately satisfy the i.i.d. normal assumption. This suggests that QOG is robust against mild violations of the i.i.d. assumption.

The huge reduction of the total measurement time by QOG is partly due to the fact that the upper bound on the measurement time varies between different configurations, and the largest upper bound is set at 17 minutes. Therefore, even if each configuration is measured until the upper bound is reached, the total measurement time is 2.6 to 4.2 times shorter than that when QOG is not used.

To isolate the QOG's effect of terminating the measurement before the upper bound is reached, we consider a modified QOG, which we refer to as QOG'. The procedure of QOG' is the same as QOG except that h is varied for each configuration so that the upper bound on the measurement

⁶The skewness and the lag-1 autocorrelations are calculated for each configuration, and they are averaged over all of the configurations.

time calculated by Equation (1) becomes 17 minutes *for all of the configurations*⁷. Thus, each configuration is measured for 17 minutes unless the configuration is determined to be suboptimal within 17 minutes.

The solid line labeled with QOG' in Figure 4 shows the total measurement time, averaged over 1,000 runs, of QOG'. Similarly to QOG, the total measurement time is minimized when the batch size is one, and the selected configuration is one of the two best configurations for a range of the batch sizes. When the batch size is one, the total measurement time of QOG' is 7.5 times shorter than that when neither QOG nor QOG' is used.

Below, we will study the effectiveness of QOG' in more detail. QOG' is studied instead of QOG, since the majority of the configurations are measured for only a few minutes in QOG, which often obscures essential features of QOG. However, our experiments suggest that the results that we find with QOG' also carry over to QOG.

3.2.2 Effectiveness of guesses

Figure 5 illuminates the effectiveness of regression in guessing at the throughput by showing the total measurement time of QOG' (dotted line) and that of no QOG (dotted line), where neither QOG nor QOG' is used, together with the total measurement times of two modified QOG' algorithms, DEC and INC (solid lines). The procedure of DEC (and respectively, INC) is the same as QOG' except that DEC (and respectively, INC) measures the configurations in the decreasing (and respectively, increasing) order of their true throughputs. Thus, DEC is an idealized QOG' which can always guess the best configuration without mistakes. In practice, DEC and INC are not feasible, since the true throughput is not known in advance. Our experimental procedure introduced in Section 3.1 allow us to compare QOG' to DEC and INC.

First observe that the total measurement time of QOG' is very close (2% to 8%) to that of DEC. This suggests the effectiveness of the regression used in QOG' in estimating the throughput so that better configurations are more likely to be measured before the others.

A counter-intuitive result in Figure 5 is that the total measurement time of INC is much shorter than that of no QOG. If INC made no mistake in intermediate comparisons, every configuration would be measured for 17 minutes, since the better configurations are measured after the others. It turns out that INC makes mistakes approximately one in two comparisons, so that half of the configurations are measured for less than 17 minutes. However, all of these mistakes are unimportant, since a mistake is made only when the true throughputs of the configurations compared to each other differ by at most $\delta = 3$. As a result, INC chooses one of the two best configurations after all of the configurations are measured. Thus, *QOG' is superior to not using QOG' even if the guesses are always wrong, but correct guesses can greatly shorten the total measurement time.*

Below, we will always set the batch size at one. Recall that the total measurement time of QOG' increases with the batch size, and QOG' always selects one of the two best configurations for all of the batch sizes studied. Since a batch size is defined to be the length of the period where a sample throughput is observed, one could consider a batch

⁷As a result, h varies between 1.13 and 12.0, and $\beta = \alpha/(|C| - 1)$ varies between 1.8×10^{-9} and 0.30.

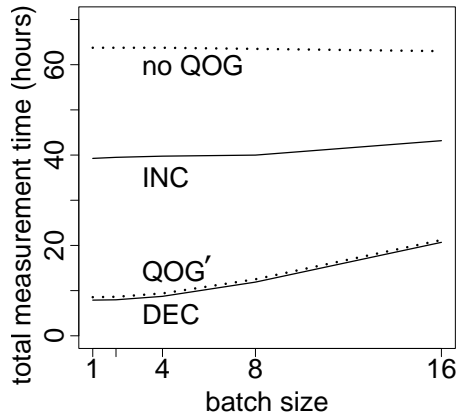


Figure 5: The total measurement time needed to determine the best configuration in two modified QOG' algorithms, DEC and INC, where configurations are measured in decreasing and increasing, respectively, orders of their true throughputs (solid lines). The dotted lines show the total measurement time when QOG' is used and when neither QOG nor QOG' is used.

size smaller than one. We do not consider a smaller batch size, since a smaller batch size does not reduce the total measurement time significantly but requires more frequent decisions to be made as to whether the configuration under the measurement is worse than the provisional best.

3.2.3 Variability of the total measurement time

The total measurement time of QOG' varies depending on the particular configurations measured before the full regression function is used. Figure 6(a) shows the distribution of the total measurement time of QOG'. Observe that the total measurement time of QOG' is consistently close to 7.9 hours, the total measurement time of DEC.

For comparison, we create RND, which is based on QOG' but measures the configurations in a random order. Figure 6(b) shows the distribution of the total measurement time of RND. Although the *mean* total measurement time of RND is only 12% (and respectively, 21%) longer than that of QOG' (and respectively, DEC), the variability of RND is much higher than that of QOG'. In the worst case, RND may measure the configurations in the increasing order of their true throughputs, and hence its total measurement time can be as long as that of INC, which is four times longer than that of DEC.

Our intuition behind the short *mean* total measurement time of RND is that there are several nearly best configurations under our conditions, and at least one of them is likely to be measured at an early stage. Specifically, 10 configurations have true throughputs within $\delta = 3$ of the best configuration. The measurements of poor configurations can be terminated quickly after one of nearly best configurations becomes the provisional best. The mean total measurement time of RND will be longer if the best configuration is much better than the other configurations.

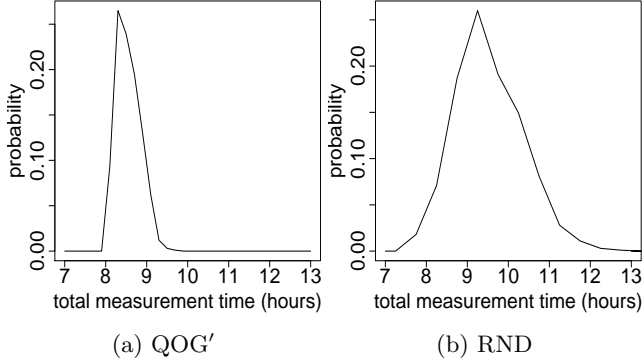


Figure 6: The distribution of the total measurement time in (a) QOG' and in (b) RND, a modified QOG' where configurations are measured in a random order. Each figure is generated based on 1,000 runs.

3.2.4 Another benefit of QOG

Figure 7 plots the true throughputs of the configurations in the order where they are measured by QOG' (solid line in Column (a)) and by RND (solid line in Column (b)). Dashed lines show the true throughputs in their decreasing order and thus correspond to DEC. Observe that, although QOG' does not always measure better configurations before the others, it is certainly biased towards better configurations. This suggests that *a nearly best configuration is likely to be selected even if QOG' is terminated before all of the configurations are measured*. Terminating before measuring all of the configurations can drastically reduce the *total time* by eliminating much of the time wasted for warmups and configuration changes. Although the measurement times of poor configurations measured at later stages are quite short, the savings of the times needed for their warmups and configuration changes can be significant.

The two vertical lines in Figure 7(a) show when QOG' changes how to guess at the throughput. QOG' does not use regression for the first nine configurations before the first vertical line. After the first vertical line, QOG' guesses at the throughput using the simpler regression function. Figure 7(a) shows that the best configuration is measured soon after the simpler regression function is used. QOG' starts using the full regression function when sufficient configurations have been measured (second vertical line). In the figure, 21 configurations are measured between the two vertical lines. Soon after the full regression function starts being used, the second best configuration is measured by QOG'.

Figure 7(a) also shows that quite good configurations are measured after 183 configurations have been measured. We find that this is due to the fact that our solution space is multi-modal. When the solution space is multi-modal, our quadratic regression function cannot guess at the throughput of all of the configurations very well. The multi-modal solution space also suggests that simple hill-climbing methods can fail in finding the best configuration. Under our conditions, the values of the configuration parameters are similar for the best twelve configurations. However, the values of the configuration parameters of the 13-th best configuration are (100,10,192,30), very different from those of the best configuration (see Table 1). The 13-th best config-

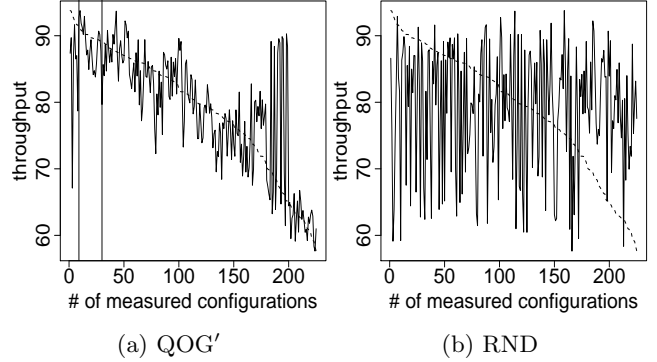


Figure 7: The throughputs of the configurations in the order where they are measured by (a) QOG' and by (b) RND (solid lines). Dashed lines shows the throughputs in their decreasing order. The two vertical lines in Column (a) show when QOG' changes how to estimate the throughput: Regression without dependent terms is used for the next 21 configurations, regression with dependent terms is used for the rest of the configurations.

uration forms the second peak in our solution space. Our experiment (not shown) suggests that this second peak includes configurations whose throughputs drop significantly when the load becomes higher. The configurations near the second peak are not robust against changes in the workload.

3.2.5 Error probability

In the above experiments, QOG' and its variants such as DEC, INC, and RND always select one of the two best configurations after all of the configurations have been measured. To evaluate the probability that QOG' makes incorrect decisions *at intermediate comparisons*, we run the experiments of selecting the better of two configurations by QOG'. Specifically, for all ordered pairs of the 225 configurations, the first configuration is measured for 17 minutes and the second configuration is measured until it is determined to be better or worse by QOG'. The error probability is defined to be the probability that the configuration having the smaller true throughput is selected by QOG'. Recall that the true throughput is defined to be the throughput measured for 17 minutes. Hence, the error probability is zero if the first configuration has larger true throughput, since the second configuration needs to be measured for 17 minutes to be selected as the better.

The solid line in Figure 8 shows the error probability as a function of the difference between the true throughputs (the true throughput of the first configuration minus that of the second one). The error probability is surprisingly high when the second configuration is only slightly better. As the difference between the true throughputs becomes larger, the error probability becomes smaller. The figure shows that QOG' can make a mistake even when the difference is close to 20, but this can occur only when the variability of the batches is very high.

To isolate the effect of variability on the error probability, we classify the 225 configurations into two classes, high C^2

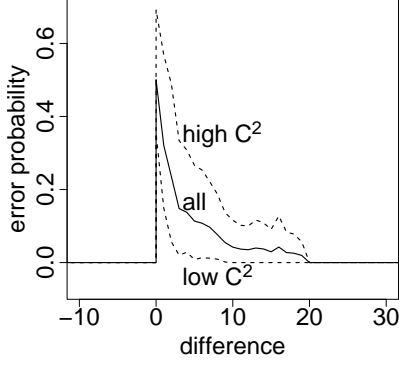


Figure 8: Error probability of QOG' in selecting the better of two configurations as a function of the difference between their true throughputs.

and low C^2 . The class of high C^2 consists of half of the 225 configurations having higher variability⁸. The dashed line in Figure 8 shows the error probability for each class of the second configuration as labeled. When the difference between the true throughputs is greater than $\delta = 3$, the error probability is less than 0.03 for the class of low C^2 but can be as high as 0.3 for the class of high C^2 . Note that (standard) QOG will adjust the duration of the measurement time such that the error probability becomes independent of the variance of the sample batches.

Finally, note that QOG' successfully selects a nearly best configuration after all of the configurations are measured, even though the error probability in intermediate comparisons is rather high. There are a few intuitions behind this result. First, most of the mistakes in intermediate comparisons are unimportant in the sense that the mistakes are made only when the throughputs of the two configurations are close. In fact, by making these unimportant mistakes, QOG' can reduce the total measurement time (recall Figure 5). Second, QOG' never determines that a worse configuration is better due to our definition of the true throughput. Thus, once a nearly best configuration becomes the provisional best, a nearly best configuration will be selected after all of the configurations are measured. Also, under our conditions, nearly best configurations have relatively low variability, and hence QOG' is less likely to make mistakes for nearly best configurations. Finally, note that the *exact* error probability of QOG' cannot be studied, since the *exact* throughput of each configuration is not known.

4. PROOF

In this section, we provide an outline of a proof of Theorem 1. In [10], we provide a complete proof for a two-stage variant of QOG, which can be easily translated into a complete proof of Theorem 1.

First consider the case where $\mathcal{B}_\delta = \{\pi_b\}$. Let \mathcal{C}_{bef} and \mathcal{C}_{aft} , respectively, be the set of configurations that are measured before and after π_b in QOG. Let E_i be the event that QOG makes an incorrect decision that $\pi_i \in \mathcal{W}_\delta$ is better than

⁸The sample squared coefficient of variation of a batch is at least 0.055 for a configuration in the class of high C^2 .

π_b (i.e. $\pi_i \in \mathcal{W}_\delta$ eliminates π_b). Since such an incorrect decision is made at most once, it suffices to prove

$$\Pr(E_i) \leq \frac{\alpha}{|\mathcal{C}| - 1} \quad (4)$$

for each of $\pi_i \in \mathcal{C}_{\text{bef}}$ and $\pi_i \in \mathcal{C}_{\text{aft}}$.

To prove (4) for $\pi_i \in \mathcal{C}_{\text{bef}}$, let π' be the provisional best immediately before π_b is measured. When $\pi_i \in \mathcal{C}_{\text{bef}}$, E_i occurs only when (i) $\pi' = \pi_i$, and (ii) there exists an r such that $\bar{X}_b^{(r)} < \bar{X}' - W_b(r)$ and $r < s$ for any s such that $\bar{X}_b^{(s)} > \bar{X}' + W_b(s)$, where $\bar{X}' = \bar{X}_i^{(N_i)}$. Thus, $\Pr(E_i)$ is no greater than the probability that Event (ii) occurs. Formally,

$$\Pr(E_i) \leq \Pr\left(\exists r \leq N_b \text{ s.t. } \bar{X}_b^{(r)} < \bar{X}' - W_b(r) \text{ and } r < s \forall s \text{ s.t. } \bar{X}_b^{(s)} > \bar{X}' + W_b(s)\right). \quad (5)$$

Observe that the right hand side of (5) is the probability that $r(\bar{X}_b^{(r)} - \bar{X}')$ exits the continuation region through the lower edge (see Figure 2).

After a lengthy calculation, we find an upper bound on the right hand side of (5):

$$\Pr(E_i) \leq \Pr\left(\exists r \leq N_b \text{ s.t. } Y(r) < \min\left\{0, -\frac{h^2 S_b^2}{2\delta\sigma_b} + \frac{\delta}{2\sigma_b} r\right\} \text{ and } r < s \forall s \text{ s.t. } Y(s) > \max\left\{0, \frac{h^2 S_b^2}{2\delta\sigma_b} - \frac{\delta}{2\sigma_b} s\right\}\right), \quad (6)$$

where $Y(t) = \sum_{\ell=1}^t \left(Z_{b,\ell} + \frac{1}{\sigma_b} \left(\delta + \sqrt{\sigma_i^2/N_i} Z_i\right)\right)$, and $Z_i = -(\bar{X}' - \mu_i)/\sqrt{\sigma_i^2/N_i}$ and $Z_{b,\ell} = (X_{b,\ell} - \mu_b)/\sigma_b$ for $\ell = 1, 2, \dots$ can be shown to be i.i.d. standard normal random variables.

Next, we use the following lemma, which provides a bound on the probability that a “random walk” with a drift *having a normal distribution* exits a triangular area through the lower edge.

LEMMA 1. *Let $\alpha > 0$ and $\beta > 0$, and let $L(r) = -\alpha + \beta r$ and $U(r) = \alpha - \beta r$ for $0 \leq r \leq \frac{\alpha}{\beta}$ and $L(r) = U(r) = 0$ for $r > \frac{\alpha}{\beta}$. Let X_1, X_2, \dots be independent standard normal variables, Δ be a normal random variable with mean $\mu \geq 0$ and standard deviation $\sigma \geq 0$, and $S_\Delta(r) = \sum_{i=1}^r X_i + r\Delta$. Let \mathcal{E} be the event that there exists an r such that $S_\Delta(r) < L(r)$ and $r < s$ for any s such that $S_\Delta(s) > U(s)$. Then,*

$$\Pr(\mathcal{E}) \leq \mathbb{E}\left[\left(1 + \exp(2(\sqrt{\alpha\beta}Z + \alpha\Delta))\right)^{-1}\right]$$

where Z is a standard normal random variable.

The lemma can be proved by using the results in [1], where Δ is a constant.

Applying Lemma 1 to (6) and simplifying the formula, we obtain, after a lengthy calculation,

$$\Pr(E_i) \leq \mathbb{E}\left[\left(1 + \exp\left(\sqrt{\frac{h^2 Q_b}{n_0 - 1}} \left(1 + \frac{Q_b}{Q_i}\right) Z + \frac{h^2 Q_b}{n_0 - 1}\right)\right)^{-1}\right],$$

where $Q_b = (n_0 - 1)S_b^2/\sigma_b^2$ and $Q_i = (n_0 - 1)S_i^2/\sigma_i^2$ are χ^2 random variables with degree of freedom $n_0 - 1$, Z is a standard normal random variable, and Q_b , Q_i , and Z are

independent of each other. Now, (4) follows from the way we choose h (see (2)).

When $\pi_i \in \mathcal{C}_{\text{aft}}$, the event E_i occurs only when there exists an r such that $\bar{X}_i^{(r)} > \bar{X}_b + W_i(r)$ and $r < s$ for any s such that $\bar{X}_i^{(s)} < \bar{X}_b - W_i(s)$, where $\bar{X}_b = \bar{X}_b^{(N_b)}$. Thus,

$$\Pr(E_i) \leq \Pr\left(\exists r \leq N_i \text{ s.t. } \bar{X}_i^{(r)} > \bar{X}_b + W_i(r) \text{ and } r < s \forall s \text{ s.t. } \bar{X}_i^{(s)} < \bar{X}_b - W_i(s)\right)$$

The last expression can be shown to be equivalent to (5) and hence $\leq \alpha/(|\mathcal{C}| - 1)$.

The case of $|\mathcal{B}_\delta| \geq 2$ may be argued as in the proof of HN [4] that selecting the best configuration when $\mathcal{B}_\delta = \{\pi_b\}$ is more difficult than selecting one of nearly best configurations when $|\mathcal{B}_\delta| \geq 2$. In [10], we formally prove that this argument holds in QOG.

5. CONCLUSION

In this paper, we propose the Quick Optimization via Guessing (QOG) algorithm for optimizing the configuration of a Web system. We find that QOG can, by a factor of 22, reduce the total measurement time needed to select a nearly best configuration, when it is applied to a real Web system. By guessing at the performance, QOG seeks to measure better configurations before the others, so that the measurements of poor configurations are terminated as soon as possible. We provide a criterion as to when the measurement of a configuration should be terminated, so that QOG selects a nearly best configuration with high probability. Our experiments also suggest that QOG is robust in the sense that it can quickly find a nearly best configuration even when the i.i.d. normal assumption is only approximately satisfied, when the performances guessed at are always wrong, and when the solution space is multi-modal.

An interesting future direction is to combine the ideas in QOG and existing search techniques [3, 7, 14, 15], so that only a subset of configurations are measured for minimal duration. The use of ideas in QOG and [14] would be effective when a Web system has many candidate configurations. In fact, the idea of reducing both the number of configurations to be measured and the measurement time of each configuration is investigated in [11], but it has not been applied to a real Web system due to the complexity for checkpoints required in [11]. The use of ideas in QOG and [3, 7, 15] would allow us to quickly adjust the configuration of a Web system at run time under changing workload and to reduce the time when the Web system runs with poor configurations.

Although we apply QOG to optimizing the configuration of a Web system, QOG can be used for other computer and communication systems having multiple candidate configurations. Also, we define a configuration of a Web system so that different configurations have different values of configuration parameters, but a configuration can be defined flexibly depending on what changes can be made on the Web system to improve its performance. For example, different configurations may have different hardware configurations, although various issues would need to be resolved as to how we change the hardware configurations.

Acknowledgment

We thank Shannon Jacobs for his English rewiring.

6. REFERENCES

- [1] T. W. Anderson. A modification of the sequential probability ratio test to reduce the sample size. *Annals of Mathematical Statistics*, 31:165–197, 1960.
- [2] R. E. Bechhofer, T. J. Santner, and D. M. Goldsman. *Design and analysis of experiments for statistical selection, screening, and multiple comparisons*. John Wiley & Sons, 1995.
- [3] I. Chung and J. K. Hollingsworth. Automated cluster-based web service performance tuning. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pages 36–44, June 2004.
- [4] L. J. Hong and B. L. Nelson. The tradeoff between sampling and switching: New sequential procedures for indifference-zone selection. *IIE Transactions*, 37(7):623–634, 2005.
- [5] S. Kim and B. L. Nelson. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation*, 11(3):251–273, 2001.
- [6] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, third edition, 2000.
- [7] X. Liu, L. Sha, S. Froehlich, J. L. Hellerstein, and S. Parekh. Online response time optimization of Apache Web server. In *Proceedings of the 11th International Workshop on Quality of Service (IWQoS 2003)*, pages 461–478, June 2003.
- [8] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [9] B. L. Nelson, J. Swann, D. Goldsman, and W. Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, 49(6):950–963, 2001.
- [10] T. Osogami. Finding probably best systems quickly via simulations. Technical Report RT0684, IBM Tokyo Research Laboratory, 2006.
- [11] T. Osogami and T. Itoko. Finding probably better system configurations quickly. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/PERFORMANCE 2006)*, pages 264–275, June 2006.
- [12] M. Raghavachari, D. Reimer, and R. D. Johnson. The deployer’s problem: Configuring application servers for performance and reliability. In *Proceedings of the 25th International Conference on Software Engineering*, pages 484–489, 2003.
- [13] J. R. Swisher, S. H. Jacobson, and E. Yücesan. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation*, 13(2):134–154, 2003.
- [14] B. Xi, Z. Liu, M. Raghavachari, C. H. Xia, and L. Zhang. A smart hill-climbing algorithm for application server configuration. In *Proceedings of the 13th International Conference on World Wide Web*, pages 287–296, 2004.
- [15] Y. Zhang, W. Qu, and A. Liu. Automatic performance tuning for J2EE application server systems. In *Proceedings of the 6th International Conference on Web Information Systems Engineering (WISE 2005)*, pages 520–527, November 2005.