RC 20485 (06/12/96)
Computer Science

IBM Research Report

Rules of Thumb for Selecting Metrics for Detecting Performance Problems

Joseph L. Hellerstein

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

Research Division
Almaden · T.J. Watson · Tokyo · Zurich

ABSTRACT: This paper addresses the selection of metrics so as to facilitate the early detection of performance problems. Our approach combines results from queueing theory and statistical hypothesis testing to develop rules-of-thumb for when one metric is preferred to another. Examples of these rules include:

- Measures of queue length are more sensitive to performance problems than are measures of utilization.
- Queue length measures provide more sensitive detection than measures of response times if the performance problem is dominated by an increase in expected arrival rates.
- Response times are preferred to queue lengths if the performance problem is dominated by an increase in expected service times.

These rules are assessed for performance problems in the CPU and paging sub-systems of a production computer system. In all cases, the data are consistent with the rules.

1. Introduction

Key to managing an information system is providing acceptable performance. Indeed, one of an administrator's greatest fears is the sudden occurrence of a performance problem that causes severe service degradations. Most commonly, performance problems are detected by monitoring measurement variables (or metrics) that are compared to threshold values. An alarm is raised if a threshold is violated. For example, suppose that disk utilization is a detection metric, and its threshold value is 80%. Then, an alarm would be raised if disk utilization exceeds 80%. This paper addresses the selection of detection metrics so as to facilitate the early detection of performance problems.

To illustrate the problem we address, consider the client/server system in Fig. 1. The clients are workstations running components of business applications. These applications periodically invoke services that access data residing on a remote database server. Such request proceed along the following path:

1. The request is queued at the client's network interface to wait for access to LAN 1.
2. Transmission occurs on LAN 1.
3. Router 1 receives the message, queues it for internal processing, and then transmits it on the backbone network (which may involve additional delays).
4. Router 2 performs the same function for the interconnection between the backbone network and LAN 2.
5. The database server queues the message until the receiving application is dispatched. Then, the message is processed by the receiving application.

Replies from the server to the client follow the reverse path.

A variety of metrics can be extracted along the path of the request and its reply. For example, end-to-end response times can be obtained by using wall clock timers and inserting the appropriate application hooks ([4], [8], and [12]). Metrics that reflect queueing for CPU, memory, network access, and router processing can also be collected (although associating resource measurements with specific end-to-end response times is still an emerging area).

Which measurement variables should be used as detection metrics? An overly spartan approach can result in **false negatives**. These are situations in which performance problems are present but they are not detected. On the other hand, employing too many detection metrics can impose significant overheads, such as increased network traffic and higher disk utilizations. Further, as the number of metrics grow, there tend to be more **false alarms**. These are situations in which a threshold is violated but no performance problem is present. False alarms increase the cost of management since administrators

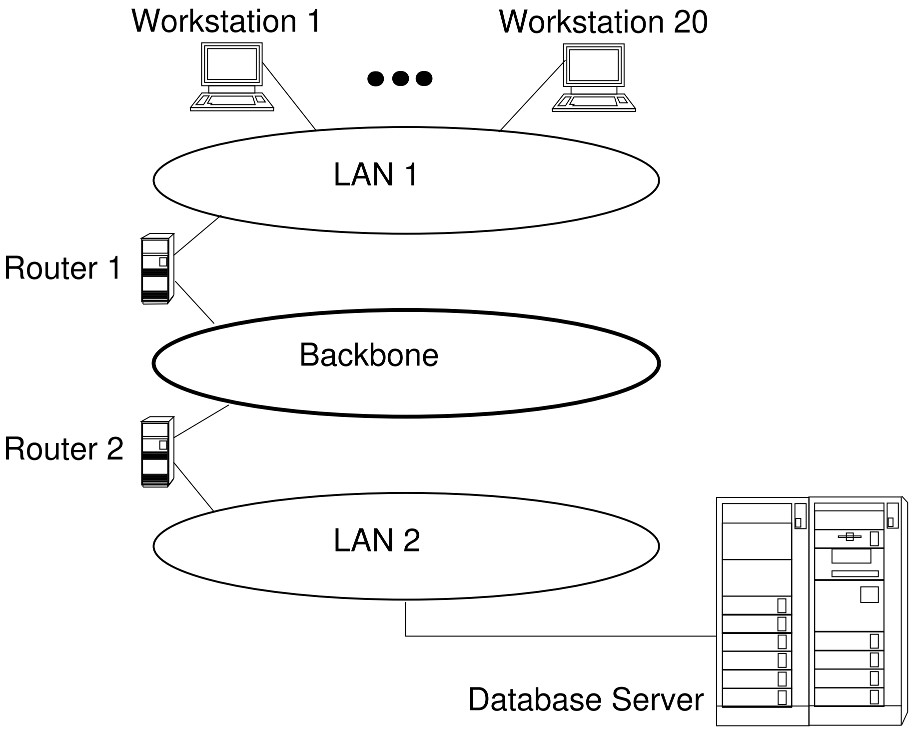


Figure 1: Client Server System

must respond to the alarms. Even worse, if false alarms are frequent, alarms may be ignored altogether.

Detection may be done either off-line or on-line. In off-line detection, data are presented en mass. In on-line detection, measurement variables are monitored continuously. Our focus is off-line detection. However, it turns out that metrics that provide powerful off-line detection can also be effective at on-line detection [1].

The topic of selecting threshold values for detection metrics is of great concern in operational settings (e.g., [11] and [7]), especially for large, distributed information systems. However, little attention has been paid to comparing detection metrics. Indeed, we are aware of only one study that addresses such comparisons [3]. Unfortunately, this study is limited in that: (a) it only considers departure rates, queue lengths, and response times; and (b) it does not consider false alarms or statistical power. There is a well developed theory of statistical hypothesis testing that provides a formal approach to analyzing both off-line detection (e.g., [10]) and on-line detection (e.g., [1]). Indeed, we use the theory of off-line detection in our analysis. However, this literature focuses on the statistics employed (e.g., the use of the sample mean versus its median), not on the choice of detection metric. Last, there is a general theory for both off-line and on-line detection of changes in process control systems [1]. While this theory addresses non-linear systems in general, we are unaware of the theory being applied to queueing systems in particular.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction to statistical hypothesis testing. Section 3 employs this theory to develop rules of thumb for determining when one detection metric is preferred to another. Section 4 extends these rules to situations in which the resources consumed by a transaction are known. Our conclusions are contained in Section 5.

2. Statistical Hypothesis Testing

Our analysis of detection metrics is based on the theory of statistical hypothesis testing (e.g., [10]). Statistical tests provide an approach to selecting between two alternatives when data are available. In our case, the alternatives are:

1. A performance problem is not present.

2. A performance problem is present.

For any situation considered, one of these alternatives holds, and the other does not. The correct alternative is called the **state-of-nature**. We use the symbol θ to denote the state-of-nature when no problem is present. θ' denotes the state-of-nature when there is a performance problem.

In practice, we do not know the true state-of-nature. Rather, based on measurement data, we hypothesize that one of the two alternatives is correct. For example, if CPU utilizations exceed 80%, we may select the performance-problem alternative; otherwise, we select the no-performance-problem alternative. Let H denote the choice of the no-performance-problem alternative and H' be the choice of the problem-present alternative.

An ideal test chooses H when θ holds and H' when θ' holds. A false alarm occurs if H' is chosen when θ holds. A false negative occurs if H is chosen when θ' holds. This is summarized in Fig. 2.

We want to select detection metrics that minimize the probability of *both* false alarms and false negatives. Not surprisingly, there is a trade-off between these objectives. Changing threshold values so as to lower false alarms typically increases false negatives, and

	θ	θ'
H	OK	False Negative
H'	False Alarm	OK

- States-of-Nature (true situation)
  - θ - no performance problem
  - θ' - performance problem is present
- Hypotheses (alternative selected)
  - H - no performance problem
  - H' - performance problem is present

Figure 2: Types of Errors in Statistical Test

changing threshold values so as to decrease false negatives usually increases false alarms. In statistics, this quandary is resolved by fixing the probability of a false alarm, say at 5%. Thus, we focus on minimizing false negatives. To put it more positively, we want to maximize **statistical power**, the probability that a problem is detected when one is present.

Comparing detection tests requires comparing statements about conditional probabilities. For example, the probability of a false alarm is the probability that H' is chosen when θ holds. This is expressed as $P(H' \mid \theta)$. Similarly, a false negative is the probability that H is chose when θ' holds, or $P(H \mid \theta')$. Power is the probability that H' is chosen when θ' holds: $P(H' \mid \theta')$. Note that $P(H' \mid \theta') = 1 - P(H \mid \theta')$ since there are only two alternatives.[1] The probability of a false alarm is denoted by α, and power is denoted by π.

Choosing between H and H' is done by the outcome of a statistical test, such as *CPU utilization > 80%.* There are three components to this test: the measurement variable (CPU utilization), a relational operator (greater than), and a threshold value (80). Existing statistical theory addresses the relational operator and the threshold value. Also, this

[1] To simplify the concepts and notation employed, we have deviated somewhat from standard terminology for statistical hypothesis testing. Technically, power is the probability of rejecting H, regardless of whether θ or θ' holds. The definition of power used in this paper is actually $1 - \beta$, where β is the probability of a false negative or Type II error.

theory considers the statistic used, such as the mean CPU utilization versus the media. However, it does not consider the choice of detection metric.

3. Basic Rules

This section uses results from queueing theory and statistics to construct rules of thumb for selecting detection metrics. These rules are evaluated for several performance problems in a production computer system.

3.1. Rule Construction

Our goal is to make general statements about the choice of detection metrics. To do so, some abstraction is required. Our approach is to classify measurement variables as kinds of queueing parameters. Fig. 3 depicts the metrics we consider. Below are detailed descriptions.

- Departure counts (**D**). The number of departures over a fixed time interval (herein denoted by T). Examples include: the number of output octets transmitted on an interface as defined in the Simple Network Management Protocol (SNMP) MIB-II; the number of TCP output bytes reported by the Unix^{TM} *netstat* command; and drops from the CPU queue in IBM's VM operating system.

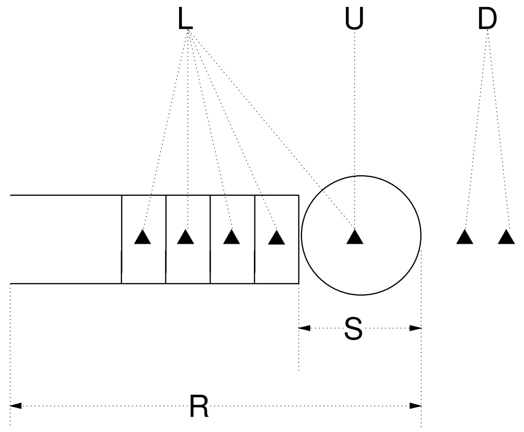
- Number of customers in the queue (**L**). Examples include: interface output queue length as defined in the SNMP MIB-II; run queue length reported by the Unix *system activity reporter*; and number of in-queue users in IBM's VM operating system.

- Response time (**R**). Examples include: response times for network file system calls as reported in the SunOS version of *nfsstat*; application end-to-end response times (e.g., [4], [8], [12]); and internal response times in IBM's VM operating system [2].

- Service time (**S**). Examples include: Unix per-process CPU consumption (from accounting data); application service times for DCE (Distributed Computing Environment) middleware services [13]; and per-user CPU consumption in IBM's VM operating system.

- Utilizations (**U**). Examples include: LAN utilizations reported by Network General's Sniffer; disk utilizations obtained by the Unix *vmstat* command; and per-user CPU utilizations in IBM's VM operating system.

While the foregoing classification is extensive, we do not claim that all measurement variables fit into our classification scheme.



(Triangles represent arrivals at the queue.)

D - Measured number of departures during an interval of length T

L - Measured number of customers in the system

R - Measured response time

S - Measured service time

U - Measured utilization

Figure 3: Metrics Studied

Herein, our focus is interactive performance because of its relationship to end-user productivity as well as to other quality of service considerations. As such, *a performance problem is an adverse impact on interactive performance that results from changes in queueing parameters.* We consider two fundamental types of performance problems:

- **Type A Performance Problem**: larger expected arrival rates (e.g., increased network traffic due to adding more users to a LAN)
- **Type B Performance Problem**: larger expected service times (e.g., due to intermittent disk failures that force retries)

The state-of-nature that we seek to detect may be a combination of Type A and Type B performance problem. Note that the qualifier *expected* appears above since we are referring to unknown parameters of the queueing model rather than measured values.

In way of notation, let λ and s denote (respectively) the expected arrival rate and expected service time when no problem is present. We use λ' and s' to denote these parameters when there is a performance problem. For a Type A performance problem, $\lambda' > \lambda$ and $s' \leq s$; for a Type B performance problem, $\lambda' \leq \lambda$ and $s' > s$. For a performance problem that is a mixture of Type A and Type B, $\lambda' > \lambda$ and $s' > s$.

We assume that data have been collected over the problem period. Further, we assume that these data are realizations of independent and identically distributed random variables from a normal distribution (e.g., by taking measurements at the same time of the day on similar days). Let $\bar{X}$ be the sample mean for a metric, where X is one of D, L, R, S, and U. For all of the five metrics, detection tests have the form $\bar{X} > C_X$, where C_X is the threshold (or **critical value** in the terminology of hypothesis testing). C_X is chosen to achieve a specified level of false alarms, say 5%.

We now show how to compute critical values (C_X) and power (π_X) under the assumption that $\bar{X}$ is normally distributed with a known mean and variance under θ and θ'. From [10],

$$C_X = \mu_X + (\sigma_X A_X)\left(\Phi^{-1}[1-\alpha)]\right), \tag{3.1}$$

where $\Phi[z]$ is the cumulative distribution function for the standard normal (mean 0 and variance 1), $\Phi^{-1}[1-\alpha]$ is the $1-\alpha$ quantile of the standard normal, and $\sigma_X A_X$ is the standard deviation of $\bar{X}$. A_X is a **standard deviation adjustment factor** for the metric X. This term accounts for batching and autocorrelations when no performance problem is present. It is fairly straightforward to show that

$$\pi_X = \Phi[K_X],$$

Metric	C	K
D	$\lambda T + \sqrt{\lambda T}(\Phi^{-1}[1-\alpha])A_D$	$\frac{(\lambda' T - \lambda T) - \sqrt{\lambda T}(\Phi^{-1}[1-\alpha])A_D}{\sqrt{\lambda' T}A'_D}$
L	$\frac{\rho}{(1-\rho)} + \frac{\sqrt{\rho}}{(1-\rho)}(\Phi^{-1}[1-\alpha])A_L$	$\frac{(\rho'-\rho) - \sqrt{\rho}(1-\rho')(\Phi^{-1}[1-\alpha])A_L}{\sqrt{\rho'}(1-\rho)A'_L}$
R	$\frac{s}{(1-\rho)} + \frac{s}{(1-\rho)}(\Phi^{-1}[1-\alpha])A_R$	$\frac{(s'(1-\rho) - s(1-\rho')) - s(1-\rho')(\Phi^{-1}[1-\alpha])A_R}{s'(1-\rho)A'_R}$
S	$s + (s)(\Phi^{-1}[1-\alpha])A_S$	$\frac{(s'-s)-(s)(\Phi^{-1}[1-\alpha])A_S}{s'A'_S}$
U	$\rho + \sqrt{\rho(1-\rho)}(\Phi^{-1}[1-\alpha])A_U$	$\frac{(\rho'-\rho) - \sqrt{\rho(1-\rho)}(\Phi^{-1}[1-\alpha])A_U}{\sqrt{\rho'(1-\rho')}A'_U}$

Figure 4: Critical Values (C) and Power Terms (K) for M/M/1, FCFS

where K_X is the **power term** for X and

$$K_X = \frac{\mu'_X - C_X}{\sigma'_X A'_X} = \frac{(\mu'_X - \mu_X) - (\sigma_X A_X)\left(\Phi^{-1}[1-\alpha]\right)}{\sigma'_X A'_X}. \tag{3.2}$$

Observe that multiplying a metric by a constant has no effect on K_X since the mean and standard deviation of $\bar{X}$ are multiplied by the same constant. Thus, departure counts (D) and departure rates (D/T) have the same power if T is constant.

Computing C_X and K_X for the five metrics only requires substituting the means and variances of these metrics into Eq. (3.1) and Eq. (3.2). For simplicity, we consider M/M/1 first-come-first-served (FCFS) queueing systems (e.g., [9]). Full details (including the calculation of the standard deviation adjustment factors) are reported in [5]. Fig. 4 contains the results of making the necessary substitutions.

We now compare the power of the metrics. Note that it suffices to compare the power terms of each metric since given the metrics X and Y, $K_X < K_Y$ is equivalent to $\pi_X < \pi_Y$. To proceed, we must consider the standard deviation adjustment factors (A_X and $A_{X'}$). For the case studies reported in Section 3.2, it is reasonable to assume that $A_L = A_R = A_S = A_U$ and $A_{L'} = A_{R'} = A_{S'} = A_{U'}$ (based on a detailed analysis of the measurement data [5]). Details of the pairwise comparisons of the metrics are reported in [5]. Below, we summarized the results.

- Departure counts work poorly for performance problems dominated by increases in expected service times, and service time metrics are a poor choice for performance problems dominated by larger expected arrival rates.

- Detection using queue lengths is more powerful than employing utilizations. This holds for both Type A and Type B performance problems.

- Using response times as a detection metric provides more powerful detection than employing service times.

- The choice between using response time and queue lengths as detection metrics depends on the nature of the performance problem and the utilization of the resource.

  - Queue lengths are preferred when performance problems are dominated by increases in expected arrival rates.
  - Response times are preferred for performance problems that are dominated by increases in expected service times.
  - At high utilizations, queue lengths and response times provide the same statistical power.

Fig. 5 summarizes the metric selection. How robust are these rules to deviations from the M/M/1 assumptions? Rule 1 and Rule 2 seem fairly robust, as long as service times and interarrival times are mutually independent and do not depend on the scheduling policy. In contrast, Rule 6 is not particularly robust. This rule favors R over S for performance problems dominated by an increase in expected service times. However, Rule 6 depends strongly on the variance of S. For example, suppose that service times are constant (e.g., transmission times for fixed length packets over a single network segment) and consider a performance problem dominated by changes in expected service times (e.g., an increase in packet size). If there is almost no variability in service times, then the power of a test based on S would be higher than a test based on R (since the latter would still have variability due to changes in inter-arrival times). Thus, Rule 6 is qualified by a statement about the variance of S.

Robustness considerations for the other rules are more difficult to assess. For example, the choice between L and U in Rule 3 is affected by the distribution of interarrival and service times as well as scheduling policies. Consider a queue with general service times. Clearly, the mean and variance of L increase with the variance of the service time distribution. Further, L is affected by scheduling policies. In contrast, the power of a utilization metric is insensitive to service time distributions and scheduling policies. Similar considerations are present for Rules 4, 5, 6, and 7. Nevertheless, simulation studies that we have done for constant and hyperexponential service times have produced results consistent with Rule 3 through Rule 7 (with an appropriate caveat about the variance of S in Rule 6).

For equal standard deviation adjustment factors:

- Rule 1: D is a poor choice for performance problems dominated by an increase in expected service times.
- Rule 2: S is a poor choice for performance problems dominated by an increase in expected arrival rates.
- Rule 3: L is preferred to U.
- Rule 4: L is preferred to R for performance problems dominated by an increase in expected arrival rates.
- Rule 5: R is preferred to L for performance problems dominated by increases in expected service times.
- Rule 6: R is preferred to S for performance problems dominated by increases in expected service times (if the variance of S is not too small).
- Rule 7: L and R are comparable choices at high utilizations.

Figure 5: Rules for Selecting Detection Metrics (From M/M/1, FCFS Queueing Systems)

Even so, data from production systems are needed to understand how these rules fare in practice.

3.2. Case Studies

To assess the metric selection rules, we present several case studies that use data from a large time sharing system running the IBM VM/SP HPO Operating System [6] at a major telephone company. The workload is a mixture of text processing, scientific applications, and program development. Three performance problems are considered. The first two relate to the CPU sub-system. This sub-system can be characterized as a two server queue with a complex, deadline scheduler. The third performance problem relates to the paging sub-system, a complex queueing network of input/output processes, controllers, and disk drives. The performance problems are summarized below:

- **Problem 1**: Increased CPU service times due to an inefficiently programmed application.
- **Problem 2**: Increased CPU service times due to an operating system bug.
- **Problem 3**: Increased paging rates due to an operating system bug.

From the foregoing descriptions, it seems reasonable that Problem 1 and Problem 2 are dominated by an increase in expected services times, and Problem 3 is dominated by an increase in expected arrival rates.

For the CPU problem, measurement variables were collected that correspond to each of the five detection metrics considered in this paper: **QDROPS**, departure rate from the CPU queue (D/T); **CPUQ**, number of transactions waiting for or being served by the CPU sub-system (L); **CPUR**, response time for CPU transactions (R); **CPUS**, CPU service time (S); and **SYSTCPU**, the CPU utilization in percent (U). (Since there are two CPUs, $0 \leq SYSTCPU \leq 200$.) Data for the paging sub-system is more sparse. The variables reported are: **PREAD**, departure rate from the paging sub-system (D/T); **PAGEQ**, number of transaction waiting for or receiving service from the paging sub-system (L); and **PAGER**, response time to page-in requests (R).

Five data sets are used in our study. There is one data set for each of the problems, and separate reference data sets are used to establish baseline values for measurement variables in the CPU and paging sub-systems. The sensitivity of a measurement variable to the presence of a performance problem is indicated by how much larger the variable is in the problem data compared to the reference data. The statistical significance of this difference is quantified by the t-statistic.

CPU Variable	*Type*	*Problem 1*	*Problem 2*
QDROPS	D	-7.49	-4.02
CPUQ	L	0.71	1.68
CPUR	R	1.25	2.03
CPUS	S	1.17	0.03
SYSTCPU	U	-13.65	-6.62

Page Variable	*Type*	*Problem 3*
PREAD	D	0.31
PAGEQ	L	5.10
PAGER	R	4.44

- *A more sensitive test is indicated by a larger t-statistic.*

Figure 6: t-statistics in the Case Studies

Fig. 6 reports the t-statistics of the five metrics for the three performance problems considered. These data allow us to assess the rules for selecting detection metrics, with the exception of Rule 2 (since service times are not reported for Performance Problem 3). **In all cases, our metric selection rules are consistent with the data.** Specifically,

- Rule 1: D has a very small t-statistic in Problems 1 and 2, which is consistent with D being a poor for detecting performance problems dominated by an increase in expected service times.
- Rule 3: L has a larger t-statistic than U, which is consistent with L being preferred to U.
- Rule 4: L has a larger t-statistic than R in Problem 3, which is consistent with L being preferred to R for performance problems dominated by an increase in expected arrival rate.
- Rule 5: R has a larger t-statistic than L in Problems 1 and 2, which is consistent with R being preferred to L for performance problems dominated by an increase in expected service times.
- Rule 6: R has a larger t-statistic than S in Problems 1 and 2, which is consistent with R being preferred to S.

- Rule 7: This rule states that R and L are comparable choices at high utilizations. The CPU utilization in Problem 1 is 158, and it is 184 in Problem 2. In both cases, R has a larger t-statistic than L. However, the difference between these t-statistics is greater in Problem 1 (.54) than in Problem 2 (.35), which is consistent with Rule 7.

That these data are consistent with our rules is encouraging since neither the CPU nor the paging sub-systems remotely resemble an M/M/1 queueing system (which formed the basis for the derivation of the metric selection rules).

3.3. Detection With Multiple Metrics

Several of the selection rules hold only for certain kinds of performance problems. For example, Rule 1 and Rule 4 apply only if the performance problem is dominated by an increase in expected arrival rates. Unfortunately, prior knowledge of performance problems is rarely available in practice. However, by selecting detection metrics that jointly cover many kinds of performance problems, we can construct a **hybrid test** that, hopefully, is reasonably powerful for a broad range of performance problems.

A hybrid test employs multiple metrics, each with its own threshold. For example, we may wish to combine the strengths of L and R metrics in the case study by employing the hybrid test $CPUQ > 3$ *or* $CPUR > .3$. An alarm is raised if either threshold is violated.

How do we obtain threshold values for detection metrics in hybrid tests? Consider the hybrid test $\bar{L} > C_L^*$ *or* $\bar{R} > C_R^*$, where C_L^* and C_R^* are the critical values in the hybrid test. Clearly, the probability of either $\bar{L} > C_L^*$ or $\bar{R} > C_R^*$ is at least as large as just one of the thresholds being violated. Thus, to maintain the same level of false alarms, we expect that $C_L^* \geq C_L$ and $C_R^* \geq C_R$. An exact computation of the thresholds for a hybrid test requires knowing the joint distribution of the variables employed. Unfortunately, joint distributions are often complex (or unknown), unless the variables are statistically independent. However, a simple bound can be used to compute critical values. Specifically, for L and R,

$$\begin{aligned} P(\bar{L} > C_L^* \text{ or } \bar{R} > C_R^* \mid \theta) &\leq P(\bar{L} > C_L^* \mid \theta) \\ &+ P(\bar{R} > C_R^* \mid \theta). \end{aligned} \tag{3.3}$$

By choosing C_L^* and C_R^* so that the right hand side sums to α, we ensure that the probability of a false alarm does not exceed α. Note that Eq. (3.3) tends to choose critical values that are larger than necessary. Hence, the achieved level of false alarms will, in general, be lower than α, which means that the power of the test will be lower as well.

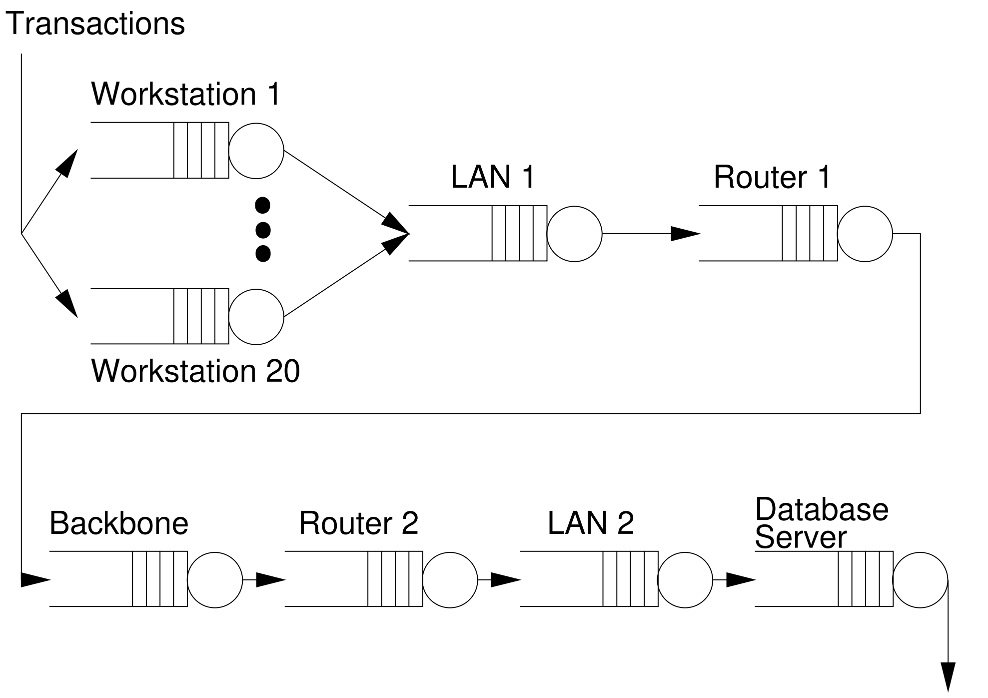


Figure 7: Queueing Model for Client Server System

4. Extended Rules

The rules described in Section 3 provide a starting point for selecting detection metrics. However, we can do better still if more information is available. This section addresses the selection of detection metrics when the resources used by a transaction are known.

To motivate our discussion, consider Fig. 7, which displays a queueing network representing the client/server system in Fig. 1 for the path taken by requests from the clients to the database server. Here, there are queues for each workstation, LAN, router, and the database server. Should we have separate threshold tests for each resource? If not, for what subset of the resources should there be threshold tests?

Today, the most widely used approach to metric selection with known transaction flows is to have one or more detection metrics for each resource consumed. This results in a hybrid test in which an alarm is raised if any of the metrics violates its threshold value. Specifically, for Fig. 7 we would have twenty-six detection metrics and hence twenty-six threshold tests (even more if we employ hybrid test tests at individual queues).

Such an approach should be pursued with caution since it can result in a high level of false alarms, lower power, or both. To see this, suppose there are I detection metrics. Let α^* be the probability of a false alarm that must not be exceeded in each simple test in order that the hybrid test has a false alarm probability of α. To simplify matters, assume that the tests are independent. Then, $\alpha = 1-(1-\alpha^*)^I$. So, $\alpha^* = 1-(1-\alpha)^{1/I}$. This is quite small. For example, when $\alpha = .05$ and $I = 26$, $\alpha^* \approx .002$. A small α^* means that critical values are large and hence that power is small.

The foregoing suggests that it is inadvisable to construct hybrid tests from a large number of simple tests. An alternative is to consider simple tests based on measurements of end-to-end performance, such as end-to-end queue lengths (e.g., the number of outstanding requests from workstations to the database server) or end-to-end response times. At first glance, this offers the benefit of being sensitive to changes in all queues traversed, while avoiding the pitfall of having a large number of tests.

Unfortunately, difficulties arise with this approach as well. These difficulties relate to the mean and variance of end-to-end measurements in systems with unbalanced resource utilizations. Specifically, from Fig. 5 it seems that the best choice for a detection metric is either queue lengths or response times. However, the mean and variance of queue length and response time measurements are dominated by the most heavily utilized resource (since L and R increase exponentially with larger service times and arrival rates at high utilizations). From Eq. (3.2), we know that the power of a simple test is determined by the mean and variance of the detection metric. Thus, tests based on L and R will be very sensitive to changes at the most highly utilized resources. Hence, these tests may fail to catch performance problems in resources with lower utilizations in systems. As a result, nascent problems may go unnoticed. This can lead to unpleasant surprises for system administrators.

We propose a third approach to selecting detection metrics. In our approach, resources are segregated by utilization levels. For each grouping of resources, we compute an aggregate metric (e.g., queue lengths, components of response times). A hybrid test is then constructed from these aggregate metrics. Specifically, we proceed in the following manner.

1. Identify the resources consumed for the transactions of interest.

2. Partition this set of resources into

    - $\mathcal{RES}_{hi}$: highly utilized resources

- $\mathcal{RES}_{lo}$: remaining resource

3. Form aggregate metrics. For example, if there are measures of time spent at each resource, then $\mathcal{R}_{hi} = \sum_{i \in \mathcal{RES}_{hi}} R_i$ and $\mathcal{R}_{lo} = \sum_{i \in \mathcal{RES}_{lo}} R_i$, where R_i is the time spent at resource i. A similar calculation can be used for queue lengths, utilizations, and other measurement variables.

4. Construct a hybrid test based on the aggregate metrics $\mathcal{R}_{hi}$ and $\mathcal{R}_{lo}$. For example, if response time data are available, we would have the hybrid test $\mathcal{R}_{hi} > C_{\mathcal{R}_{hi}}$ or $\mathcal{R}_{lo} > C_{\mathcal{R}_{lo}}$.

To illustrate this approach in the context of Fig. 7, suppose that there are response time measurements available for each queue and that only the routers are highly utilized (e.g., $> 80\%$). We consider response times for workstation 1.

1. The resources are:
   - Workstation 1: WS1
   - LAN 1: LAN1
   - Backbone: BB
   - LAN 2: LAN2
   - Router 1: RTR1
   - Router 2: RTR2
   - the database server: DB

2. Resources are partitioned as:
   - $\mathcal{RES}_{hi} = \{RTR1, RTR2\}$
   - $\mathcal{RES}_{lo} = \{WS1, LAN1, BB, LAN2, DB\}$

3. Form the aggregate metrics.
   - $\mathcal{R}_{hi} = R_{RTR1} + R_{RTR2}$
   - $\mathcal{R}_{lo} = R_{WS1} + R_{LAN1} + R_{BB} + R_{LAN2} + R_{DB}$

4. Construct the hybrid test. Suppose that measurements over the last couple of weeks have shown that 97.5% of the $\mathcal{R}_{hi}$ measurements are no larger than .9 seconds, and the 97.5-th quantile for $\mathcal{R}_{lo}$ it is .5 seconds. This suggests that there is a 2.5% probability that $\mathcal{R}_{hi}$ exceeds .9 seconds if there is no problem present, and there is a 2.5% probability that $\mathcal{R}_{lo}$ exceeds .5 seconds if there is no problem present. Setting $\mathcal{C}_{\mathcal{R}_{hi}}$ to .9 and $\mathcal{C}_{\mathcal{R}_{lo}w}$ to .5, we have following hybrid test:

   $$\mathcal{R}_{hi} > .9 \text{ or } \mathcal{R}_{lo} > .5$$

   This test has a probability of a false alarm that is no greater than 5%.

5. Conclusions

Early detection of performance problems is essential to limit their scope and impact. Most commonly, performance problems are detected by applying threshold tests to one or more detection metrics. These metrics should be chosen so as to minimize false alarms and to maximize statistical power (the probability of detecting a problem when one exists). Unfortunately, the ad hoc manner in which detection metrics are currently chosen often leads to low power and/or a large number of false alarms.

This paper takes a first step towards providing a systematic approach to selecting detection metrics. We develop rules of thumb based on an analysis of statistical power equations of M/M/1 queueing systems for five widely used detection metrics: departure counts (D), number in system (L), response times (R), service times (S), and utilizations (U). Our rules include: L is preferred to U; L is preferred to R if the performance problem is dominated by an increase in expected arrival rates; and R is preferred to L if the performance problem is dominated by an increase in expected service times. The effectiveness of our rules is assessed for performance problems in the CPU and paging sub-systems of a production computer system. Even though these sub-systems bear little resemblance to M/M/1 queueing systems, the empirical results are consistent with our selection rules.

We also address detection considerations when the set of resources used by a transaction is known. One could apply our detection rules to each resource in the network (e.g., each LAN, disk, and processor). However, this results in a large number of threshold tests, which in turn can cause a high level of false alarms (since there is a high probability that at least one threshold will be violated) or low power (if thresholds are set so high that false alarms are avoided). Alternatively, one could apply the detection rules to end-to-end measurements of performance. Here, difficulties arise if resource utilizations are not balanced since the test will fail to catch nascent performance problems at resources with lower utilizations. Thus, we propose using a hybrid test that deals separately with highly utilized resources.

Much work remains. In particular, our metric selection rules are obtained from simple M/M/1 queueing models and are based on statistical models that assume a known population variance. The limitation of these assumptions needs to be explored. Also needed are more data from product computer systems and networks to assess the selection rules and to give more insight into key parameters (e.g., the standard deviation adjustment factors). These data should include performance problems, their cause, and appropriate reference data. Further, more analysis is required to construct additional selection rules, especially for queueing networks, other metrics (e.g., expansion factors), tradeoffs between metrics, and different classes of traffic.

References

[1] **M. Basseville and I. Nikiforov**: *Detection of Abrupt Changes: Theory and Applications*, Prentice Hall, 1993.

[2] **Robert Berry, Jeffrey Hedglin, Joseph Hellerstein, and Paul VanLeer** "Fine-Grain Analysis (FGA): A Methodology for Analyzing Intermittent Performance Problems," *Proceedings of the Computer Measurement Group*, 1990.

[3] **Robert F. Berry and Joseph L. Hellerstein**: "An Approach to Detecting Changes in the Factors Affecting the Performance of Computer Systems," *Proceedings of ACM Sigmetrics*, May, 1991.

[4] **Dave Glover and Doug McBride**: "Measuring and Monitoring Business Transactions in an Open Systems Environment," *Proceedings of the Computer Measurement Group*, 1995, pp. 924-1011.

[5] **Joseph L. Hellerstein**: "An Approach to Selecting Metrics for Detecting Performance Problems in Information Systems," *Proceedings of the Second IEEE International Conference on Systems Management*, Toronto, Canada, June 19-21, 1996.

[6] **IBM**: *VM/SP HPO Performance Tuning Guide*, IBM Corporation, (GG22-9392) 1985.

[7] **IBM**: *CICS MRO Tuning and Performance Guide*, IBM Corporation, (GG24-3224) 1988.

[8] **Mark W. Johnson**: "Measuring the Response Time of Distributed Applications," *Proceedings of the Computer Measurement Group*, 1995, pp. 853-864.

[9] **Leonard Kleinrock**: *Queueing Systems, Volume 1*, John Wiley and Sons, 1975.

[10] **B. W. Lindgren**: *Statistical Theory*, The Macmillian Company, 1968.

[11] **Mike Loukides**: "System Performance Tuning," O'Reilly & Associates, Inc., Sebastopol, CA, 1991.

[12] **Mark M. Maccabee, Anna Long, and Walter Schwane**: "Distributed Client/Server End-to-End Response Time Instrumentation, Methodology, and Experience with a Commercial Product," *Proceedings of the Computer Measurement Group*, 1995, pp. 150-161.

[13] **J. Rolia and B. Lin**: "Consistency Issues in Distributed Application Performance Metrics," *Proceedings of CAS Conference*, October, 1994, Tornoton, Canada, pp. 121-129.