UNIX 시스템의 최적 이용율 결정

임 종 설†

요 약

본 논문은 UNIX 시스템의 최적 이용율(Optimal Utilization)을 결정하는 방법을 제시한다. 제시된 방법은 사용자가 인내할 수 있는 응답시간(Tolerable Response Time)을 제공하는 범위에서 사용자에게 허용될 수 있는 최대 이용율이 최적 이용율이 된다는 정의(Definition)를 근거로하여 개발되었다. 인내할 수 있는 응답시간이 길어지면 최적 이용율도 높아지므로 최적 이용율은 인내할 수 있는 응답시간의 최대값에서 구해진다. 인내할 수 있는 응답시간은 서비스 목표(Service Objective)가 주어지면 도출해낼 수 있다. 본 논문에서는 인내할 수 있는 응답시간이 0.24초라는 것과 최적 이용율의 세가지 구성요소는 %wio, %sys, %usr 이라는 것을 보여준다. 또한 최적 이용율을 결정하는 실제의 과정을 예시하기 위하여 UNIX 운영체제를 사용하는 IBM 3081 컴퓨터의 최적 이용율을 구하는 과정을 보여준다.

Determination of The Optimal Utilization of The UNIX Systems

Jong Seul Lim†

ABSTRACT

This paper proposes a method to determine the optimal utilizations of the UNIX systems. This method is developed using the definition-the optimal utilization is the maximum allowable utilization. In other words, the optimal utilization is the maximum utilization that can be allowed by users while providing tolerable response time. As the tolerable response time increases, the optimal utilization increases. Therefore, the optimal utilization is obtained at the maximal value of tolerable response time. Our analysis shows tolerable response time is achieved when the average of the trivial response time is less than 0.24 seconds for a given service objective. It also shows the optimal utilization consists of three components-%wio, %sys, and %usr. By way of example, the optimal utilizations of a machine (IBM 3081) running under the UNIX operating system are computed using the proposed method.

1. Introduction

Of utmost importance in computer planning is to develop a method for determining the appropriate level of computing capacity to satisfy future workload requirements. Traditionally, optimal computing capacity has been determined based upon experience. In this paper, we propose a method to determine the optimal utilization (i.e., the appropriate level of computing capacity) of UNIX systems.

This method is developed using the definition - the optimal utilization is the *maximum allowable utiliazation*. In other words, the optimal utilization is the maximum utilization that can be allowed by users while providing the *tolerable response time*.[1] *Tolerable response time* can be achieved when the service objective for the trivial response time is met. As long as a system provides *tolerable response time*, the utilization of that system can be allowed by users. Because

† 정 회 원 : 선문대학교 정보통신공학과 교수
논문접수 : 1995년 4월 26일, 심사완료 : 1995년 9월 22일

1. Response time is the time between the submittal of a process and the completion of that process.

the utilization of a system gets bigger as *tolerable response time* gets longer, the utilization of the system must have a certain upper bound. This upper bound is the optimal utilization.

Section 2 discusses the distribution of the trivial response times and the relationship between the average of the trivial response times and its service objective. Section 3 derives a formula for the optimal utilization. Based on the formula, a method is proposed to determine the optimal utilizations of UNIX systems. Also, Section 3 presents an illustrative example using the proposed method. Section 4 discusses and analyzes the results. Summary and suggestions are given in Section 5.

2. Analysis Of Trivial Response Time

This section defines the distribution of the trivial response times collected during the extended prime time shift.[2] We then explain what is *tolerable response time*, which varies according to the distribution of the trivial response times and the service objective for the trivial response time.

2.1 Distribution Of Trivial Response Time

A command *acctcom(1)* computes the response times for the trivial commands. For simplicity, we arbitrarily pick the five trivial commands, i.e., *mkdir(1)*, *date(1)*, *echo(1)*, *uname(1)*, and *rmdir(1)* of all the UNIX trivial commands (see [1] for the UNIX trivial commands). Using *acctcom(1)*, we then run the above five trivial commands every 10 minutes and measures their response times. The response time for any of the above trivial commands is referred to as the trivial response time. Because the trivial response time varies randomly, we can express the trivial response time as a random variable R. Thus, the cumulative distribution function for a random variable R gives the value of $P(R \leq r)$ for any positive real r; that is, $F_R(r) = P(R < r)$ for $0 < r < \infty$. Therefore, the (density) distribution is $f(r) = \frac{d}{dr} F_R(r)$.

To define the distribution of the trivial response times of the trivial commands, f(r), we use one of *probability plots*, which is a Quantile-Quantile (Q-Q) plot [2]&[3]. In a Q-Q plot, distributions having the same shape (but which may differ in scale and location) have quantities that lie linearly along the straight line (y = x). In other words, we can judge the distribution of the data by recognizing whether or not a set of plotted data appears to lie more or less along the straight line. To find out the best distribution for a given set of the trivial response times, we use trial and error. That is, we keep trying out the various theoretical distributions in Q-Q plots until we find out the best fit for the distribution of the trivial response times.

Fig. 1 shows a Q-Q plot of the data from IBM 3081 for five business days. In addition to a Q-Q plot for IBM 3081, we tried out Q-Q plots for several other Large Main Frame UNIX systems[3] (the results are not shown in this paper because of the similarity). Also, note that the average of r of *gamma* $(0.4, \beta)$ distribution is 0.4β. In Fig. 1, x-axis represents the ordered quantiles[4] of gamma 0.4 distribution and y-axis represents the ordered quantiles of response times for the trivial commands gathered from IBM 3081. The points on the plot more or less falls near the straight line (y = x). See [2]&[3] for more details on the Q-Q plots.

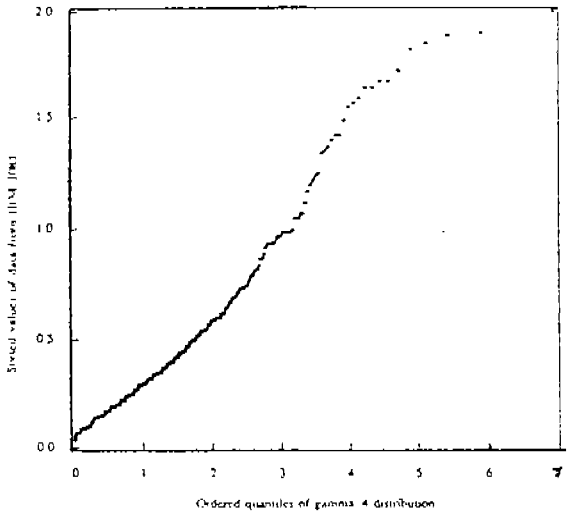
We also conducted Chi-square goodness-of-fit hypothesis tests. The results confirmed that the distribution of the trivial response times for the Large Main Frame UNIX systems is as follows:

The distribution for the Large Main Frame UNIX systems can be estimated with *gamma*

2. The extended prime time shift is a contiguous block of 10 hours that encompasses local prime shift. It is usually 8:00 A.M. - 6:00 P.M. with a few exceptions of 7:30 A.M. - 5:30 P.M.

3. The Large Main Frame UNIX systems represent machines running under S370 - e.g., IBM 3081, IBM 3090, Amdahl 5970, Amdahl 5890, etc.

4. The quantile refers to a fraction of the set of data.



(Fig. 1) A Q-Q plot of the trivial response times for IBM 3081 running under UNIX OS.

$(0.4, \beta)$ distribution [4], i.e.,

$$f(r) = \begin{cases} \dfrac{\beta^{-0.4} r^{-0.6} e^{-r/\beta}}{\Gamma(0.4)} & \text{if } r > 0 \\ 0 & \text{otherwise,} \end{cases} \quad (2.1)$$

where β is a parameter and r is a trivial response time.

2.2 Average Of Trivial Response Time And Service Objective : Relationship

In the previous section, we defined the distribution of the trivial response times. Using that distribution, we will discuss what the average of the trivial response times, which satisfies the service objective, should be. Suppose one wants to use the service objective 95% for the Large Main Frame UNIX systems. The service objective 95% refers to the probability that the response times of the trivial commands on the Large Main Frame UNIX systems take less than one second should be greater than or equal to 95%. Therefore, equation (2.1) must satisfy

$$\int_0^1 f(r)dr \geq 0.95 . \quad (2.2)$$

From (2.1) and (2.2), we obtain

$$\beta \leq 0.60 . \quad (2.3)$$

Note that (2.3) was computed using a short simulation program because there is no way of evaluating (2.2) exactly.

Let $\overline{R}$ denote the average of the trivial response times. Since $\overline{R}$ in (2.1) is 0.4β, (2.3) gives

$$\overline{R} = 0.4\beta \leq 0.4^{*}0.60 = 0.24 .$$

The above equation implies that, to meet the service objective 95%, the average of the trivial response times cannot exceed 0.24 seconds. That is, the maximum allowable value of $\overline{R}$ is 0.24 seconds. Letting $\overline{R}^{*}$ denote the maximum allowable value of $\overline{R}$, we have

$$\overline{R}^{*} = 0.24 \quad (2.4)$$

3. Derivation Of Optimal Utilization And An Illustrative Example

3.1 Derivation Of Optimal Utilization

From (2.4), we deduce that *tolerable response time* is achieved when the average of the trivial response times is less than 0.24 seconds for the Large Main Frame UNIX systems (note this is based on a service objective 95% and *gamma* $(0.4, \beta)$ distribution).

Lim[5] has derived T(x) and is more simplified as (3.1). However, one may develop his own T(x) based on his model. T(x) is the average time between the submittal of a process requiring x seconds of the central processing unit (CPU) time[5] and the completion of the process in the UNIX systems.

5. CPU time is the required processing time of a process. CPU time consists of system time and user time. System time is the time a process spends in system mode. User time is the time a process spends in user mode. System mode is a UNIX mode where a process executes the UNIX kernel codes and accesses the system data segment. User mode is a UNIX mode where a process executes user programs and accesses the user data segment.

$$T(x) \approx \frac{x}{1 - (\%sys + \%usr + \gamma)}, \qquad (3.1)$$

where x is the CPU time of a process; %sys and %usr represent the portion of the time that the CPU runs in user mode and runs in system mode, respectively; γ is shown in (3.3).

The average of the response time is obtained by

$$E[T(x)] = \overline{R} \approx \int_0^\infty T(x)f(x)dx = \frac{\overline{x}}{1 - (\%sys + \%usr + \gamma)}, \qquad (3.2)$$

where $\overline{x}$ is the average of the CPU times of processes.

We derived γ (the derivation of γ is omitted in this paper) as

$$\gamma = \frac{\%wio}{\%sys + \%usr + \%wio}, \qquad (3.3)$$

where %wio represents the portion of the time that the CPU is idle while some process is waiting for block I/O.

Substituting (3.3) for γ in (3.2) and letting ρ = %sys + %usr + %wio, we obtain

$$\overline{R} \approx \frac{\overline{x}\rho}{-\rho^2 + \rho + (\%wio)\rho - \%wio}. \qquad (3.4)$$

Rewriting (3.4),

$$\rho = \frac{\overline{R} + (\%wio)\overline{R}\overline{x} + \sqrt{[\overline{R} + (\%wio)\overline{R}\overline{x}]^2 - 4(\%wio)\overline{R}^2}}{2\overline{R}}. \qquad (3.5)$$

Since ρ in (3.5) increases with increasing $\overline{R}$ when %sys + %usr ≥ 0 (see Appendix) and since $\overline{x}$ is independent of ρ, the *maximum allowable utilization* ρ^* (i.e., optimal utilization) is given by

$$\rho^* = \frac{\overline{R}^* + (\%wio)\overline{R}^*\overline{x} + \sqrt{[\overline{R}^* + (\%wio)\overline{R}^*\overline{x}]^2 - 4(\%wio)(\overline{R}^*)^2}}{2\overline{R}^*},$$

where $\overline{R}^*$ is 0.24 by (2.4) for the Large Main Frame UNIX systems.

Since (3.6) has the unknown parameters $\overline{x}$, %wio, and $\overline{R}^*$, we need the method comprising the following four steps to determine the optimal utilization for a given system.

STEP 1 : Compute $\overline{x}$ (i.e., the average of the CPU times for the trivial commands[6]) by using the data generated from *acctcom(1)*.

STEP 2 : Compute the average of %wio by using the data generated from *sar(1)* during the extended prime time shift week.[7]

STEP 3 : Estimate the distribution of the trivial response times and then compute $\overline{R}^*$ (i.e., the maximum of the average of that distribution) according to your own service objective for trivial response times.[8]

STEP 4 : Compute the optimal utilization ρ^* by substituting $\overline{x}$, %wio, and $\overline{R}^*$ into (3.6).

3.2 An Illustrative Example

To demonstrate the above method, we compute the optimal utilization of IBM 3081 running under the Large Main Frame UNIX. We drew the data generated from acctcom(1) in IBM 3081 during 30 minutes. This sampled data consisted of 12,290 data points for the various commands. We extracted the data for the five trivial commands: *mkdir(1)*, *date(1)*, *echo(1)*, *uname(1)*, *and rmdir(1)*. We then computed the

6. The trivial commands that we picked are the five commands: *mkdir(1)*, *date(1)*, *echo(1)*, *uname(1)*, *and rmdir(1)*. However, one may pick a set of the trivial commands by his choice.

7. The extended prime time shift week is Monday through Friday, excluding holidays. Each day is usually 8:00 A.M. - 6:00 P.M.

8. $\overline{R}^*$ is 0.24 for the Large Main Frame UNIX systems and for a service objective given in this paper, for example (see Section 2.2).

average of the above five trivial commands: $\overline{x}$ = 0.026 seconds. Also, the average of %wio during an extended prime time week (8:00 A.M. - 6:00 P.M) was computed to be 0.1. Equation (2.4) gives $\overline{R}^*$ = 0.24 seconds. Then, from (3.6), we can obtain the value of ρ^* equal to 0.88.

4. Discussion

4.1 Three Components Of Optimal Utilization

In this paper, we have used the optimal utilization ρ^* such that

$$\rho^* = \%sys + \%usr + \%wio . \qquad (4.1)$$

Equation (4.1) shows that ρ^* consists of three components: %sys, %usr, and %wio. Typically, the optimal utilization has been the sum of two components: %sys and %usr. Generally, %wio is proportional to the delay by waiting resources (e.g., block I/O). That is, the larger the %wio a system has, the more resources a process requires. The more resources required by a process, the slower I/O speed. The slower I/O speed, the longer delay experienced by users. Therefore, the optimal utilization must include %wio. The optimal utilization then reflects the actual delay that users experience. For example, although the quantity %sys + %usr is small, users may experience a long delay if a system has a large %wio. In this case, we can pose a question: Does a small quantity %sys + %usr really represent that a system is under-utilized? The answer is NO. Suppose that %sys + %usr = 0.3 (we assume 0.3 is small) and %wio = 0.6, it would be unreasonable to say that a system is under-utilized because users still experience the long delay caused by the high %wio.

4.2 Optimal Utilization Is A Function Of The Service Objective

Equation (2.2) shows that β is a function of the service objective. Therefore, $\overline{R}^*$ (the average of the trivial response times) is also a function of the service objective. Equation (3.6) then shows that the optimal utilization ρ^* is a function of $\overline{R}^*$. Therefore, ρ^* is a function of the service objective. This implies that if we need to change the service objective for the trivial response times we have to change the optimal utilization ρ^* accordingly. The implication here seems very reasonable in such a sense that "the optimal utilization should decrease as the service objective gets tightened."

4.3 Sensitivity Of %wio

From the illustrative example in the previous section, we consider how the changes in %wio affect the optimal utilization ρ^*.

To do that, we substitute various %wio into (3.6), while fixing two parameters (i.e., $\overline{x}$ and $\overline{R}^*$) to the same values as in the illustrative example. The results are shown in the following table.

<Table 1> Sensitivity of %wio

%wio for IBM 3081	
0.05	$\rho^* = 0.89$
0.10	$\rho^* = 0.88$
0.15	$\rho^* = 0.87$
0.20	$\rho^* = 0.86$
0.30	$\rho^* = 0.83$

The data in the above table shows that ρ^* is not sensitive to the change in %wio. More precisely, if %wio ranges from 0.05 to 0.30, the change in the optimal utilization ρ^* for a given system is much smaller than the change in %wio. This implies that we can still trust the optimal utilization computed by the proposed method using four steps (STEP1 through STEP4 in Section 3.1), even if the quantity of %wio computed by the STEP 2 in Section 3.1 is somewhat inaccurate.

4.4 Where To Use The Optimal Utilization

In this section, we show where to use the optimal utilization obtained by the proposed method using four steps. Suppose that, by using

the proposed method, the optimal utilization of a certain machine running under UNIX OS is computed to be 0.8. From a UNIX command sar(1), we can gather the current ρ values (i.e., %sys+%usr+%wio) of the machine during a certain time interval. Then, we can average those values gatherd.

If the average of the current ρ values is bigger than 0.8, the machine is over-utilized. This means that the machine is running above its capacity. Therefore, response time of the machine is longer than users expect. To make this machine optimally utilized, the system administrators should reduce the number of users entered into the machine or reduce the number of processes sent by users. If the system administrators would not want to reduce the number of users, they had better increase the capacity of the machine by replacing compartments such as CPU, memory, and hard disks. Otherwise, they need to purchase a new machine which gives a more capacity.

On the othe hand, if the average of the current ρ values is smaller than 0.8, the machine is under-utilized. Then, the system administrators are able to do the capacity plan. That is, they can estimate how many more number of users the machine can accommodate in addition to the current users or they can estimate when the the surplus capacity of the machine will be saturated as the number of users increases. Before the machine is saturated, the system administrators should have a plan to increase the capacity of the machine.

5. Summary And Suggestions

A method is proposed to determine the optimal utilizations of UNIX systems. We showed that, for a given service objective, *tolerable response time* was achieved when the average of the trivial response times is less than 0.24 seconds for the Large Main Frame UNIX systems. The *maximum allowable utilization* (i.e., optimal utilization) was computed at the maximal value of *tolerable response time* because *allowable utilization* increases with increasing *tolerable response time*.

From the analysis in this paper, we suggest that (A) the optimal utilization include %wio (the portion of the time that the CPU is idle while some process is waiting for block I/O), in addition to including %sys (the portion of the time that the CPU runs in system mode) and %usr (the portion of the time that the CPU runs in user mode); (B) the optimal utilization be updated as the service objective for the trivial response times changes. An illustrative example using the proposed method showed that the optimal utilization is 88% for IBM 3081 (note that this percentage includes %wio in addition to including %sys and %usr). Likewise, the optimal utilizations for the other machines running under the UNIX operating system can be computed using the proposed method.

References

[1] UNIX System V Release 2.0 User Reference Manual, AT&T Bell Laboratories, October 1985.

[2] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey, Graphical Methods For Data Analysis, Wadsworth Statistics/Probability Series, 1983.

[3] R. A. Becker and J. M. Chamber, An Interactive Environment For Data Analysis And Graphics, Wadsworth Statistics/Probability Series, 1984.

[4] A. M. Law and W. D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Book Company, 1982.

[5] J. S. Lim, "Scheduling Algorithms and Queueing Response Time Analysis of the UNIX Operating System," The Transactions of The Korea Information Processing Society, Vol. 1, No. 3, pp. 367-379, Sept. 1994.

[6] G. J. Henry, "The Fair Share Scheduler," AT&T Bell Lab. Tech. J., Vol. 63, No. 8, pp. 1845-1857, 1984.

[7] L. Kleinrock, "Time-Shared Systems : A Theoretical Treatment," J. of A.C.M., Vol. 14, pp. 242-261, 1967.

[8] K. Thompson, "UNIX Time-Sharing System: UNIX Implementation," The Bell System Tech. J., Vol. 57, No. 6, pp. 1931-1946, 1978.

[9] M. J. Bach, The Design of The UNIX Operating System, Prentice-Hall, 1986.

[10] H. J. Larson, Introduction to Probability Theory and Statistical Inference, Wiley, 1982.

Appendix

< A proof that ρ increases with increasing $\overline{R}$ >

Differentiating $\overline{R}$ in (3.4) with respect to ρ, we have

$$\frac{d\overline{R}}{d\rho} = \frac{\overline{x}\rho^2 - \overline{x}(\%wio)}{[-\rho^2 + \rho + (\%wio)\rho - \%wio]^2} . \quad (A.1)$$

For (A.1), $\frac{d\overline{R}}{d\rho} \geq 0$ if

$$\rho^2 - \%wio = (\%sys + \%usr + \%wio)^2 - \%wio \geq 0 .$$

The above relationship gives

$$\%sys + \%usr \geq \sqrt{\%wio} - \%wio . \quad (A.2)$$

The right hand side of (A.2) is always smaller than or equal to 0.25 (this can be proved simply by letting $dg(\%wio)/d(\%wio) = 0$, where $g(\%wio) = \sqrt{\%wio} - \%wio$).

Therefore, for $(\%sys + \%usr) \geq 0.25$, ρ increases with increasing $\overline{R}$ because $d\overline{R}/d\rho > 0$ (note that, in practice, we do not have to consider the case for $\%sys + \%usr < 0.25$ because $\%sys + \%usr$ is typically greater than 0.25 in the real systems).

임 종 설

1979년 서울대학교 섬유공학과 (학사)
1983년 University of Cincinnati 산업공학과(공학석사)
1986년 Polytechnic University Operations Research(공학박사)
1986년~91년 AT & T 벨연구소 책임연구원
1993년~현재 선문대학교 정보통신공학과 교수