論文
2-12-8

An Analysis of the UNIX Echo Response Time
(유닉스 에코응답시간 분석)

임 종 설*
(Jong-Seul Lim)

ABSTRACT

The echo response time has been a concern in the performance of the UNIX systems, a significant tail always appears in the distribution of echo response time, though the average echo response time is less serious. This paper addresses the issue of echo response times in the UNIX systems. We explain how the Fair Share Scheduler (FSS) works and explain why the FSS might cause excessive echo response times and show by analysis how echo response time reacts to key parameters under FSS. Finally, we present a recommended solution that should improve the echo response time drastically. This solution is a refined FSS which will overcome the echo response time problem while retaining the essence of the FSG. This will enhance the UNIX performance and productivity.

요 약

에코응답시간은 유닉스 시스템 성능을 위한 관심사이며, 평균값으로 보면 중요하지 않지만 때에 따라 길어지는 등 평균에 대한 편차가 일정치 않은 문제점이 있다. 본 논문은 유닉스 시스템의 에코응답시간을 분석한다. 유닉스 커널의 Fair Share Scheduler (FSS)를 설명하며, FSS의 영향을 받아 에코응답시간이 길어지는 이유를 분석한다. 이러한 분석을 통하여 에코응답시간을 향상시키는 방법을 제시한다. 이 해결방법은 유닉스의 장점인 FSS의 본질을 잃지 않으면서 에코응답시간에 대한 문제점을 극복할 수 있는 개선된 FSS이다. 이러한 결과 도출을 위하여 수학적인 분석을 사용하였으며, 제시된 방법은 유닉스시스템의 성능 및 생산성을 높일 수 있을 것이다

1. FOREWORD

The echo response time (explained in Section 3 & [7]) has been a concern in the performance of the UNIX machines, i.e., a significant tail always appears in the distribution of echo response time, though the average echo response time is less serious. This study is motivated by the excessive echo response time is primarily caused by the Fair Share Scheduler (FSS) implemented with the UNIX systems. In this paper, we first illustrate how FSS works and explain why it may extend echo response time and then provide an analytical proof. Finally, we propose a refined FSS which will overcome the echo response time problem while retaining the essence of FSS.

* 정회원 : 선문대학교 전자정보통신공학부 교수

논문접수 : 2001. 12. 11.
심사완료 : 2001. 12. 17.

2. UNIX AND FSS

Standard UNIX employs a priority-driven scheduling discipline which assigns process priorities dynamically. There are 128 priority levels used among which the top 40 are assigned to processes which roadblock themselves during system calls and the remaining are used for processes running or waiting in the user mode. A process running in the system mode is not preemptive while one running in the user mode is. The scheduler manages a time quantum of 1 second and uses a time slice of x^* ms. The scheduler also records the CPU usage rate of each process. A process'priority is recalculated every second or when it returns from a system call based on its recent CPU usage. The FSS divides user processes into groups each of which is allocated a fixed CPU usage rate (or share). In general, the rate is proportional to the group's CPU consumption rate projected[3]. Each group is associated with a variable group priority which is determined by its recent CPU usage rate, as compared to other groups. Specifically, the ratio of the group's actual CPU usage rate and its contracted rate is the index of its relative priority. Within each Fair Share Group (FSG), UNIX scheduling discipline is employed. In other words, within a group, a process is associated with a priority number ranging from 0 to 127. Group priorities have no effect on UNIX priorities 0 through 39. In other words, processes waiting in the system mode are free of group priorities. For processes waiting or running in the user mode however, its associated group's priority completely overrides its UNIX priority. Therefore, a process with a UNIX priority of 127 will attain the CPU earlier than one with a UNIX priority of 40 if the former's group priority is higher than the latter's.

3. CASE STUDY THE ECHO RESPONSE TIME

3.1 Motivation

A keystroke generates a character interrupt and is initially scheduled in the system mode which is free of FSG priorities. After the needed system work is completed, the process with which the keystroke is associated returns to the user mode, uses the CPU to continue the processing of the character, and finally echoes the character. Note that it is quite likely that the process is preempted by system calls after it leaves the system mode. If a preemption occurs, the process is rescheduled at a UNIX user priority which is accompanied by an FSG priority; otherwise, the FSS would have no impact on the echo response time of the keystroke. Since we are primarily concerned with excessive echo response time caused by the FSS, the occurrence of the preemption event will be assumed throughout this paper.

Consider the following situation: a keystroke with a low FSG priority is preempted while groups with higher FSG priorities have CPU-bound jobs to run. Although this keystroke should receive the top user priority within its own FSG, it will not be chosen to run until its FSG priority turns around. This situation results in an excessive echo response time and will take place constantly because current scheduling discipline places FSG priorities on top of all UNIX user priorities. In general, a batch job has a lower UNIX priority than an interactive job. Thus, it may not be trivial that running batch jobs in an FSG may prolong the echo response times of interactive users within that FSG. As a matter of fact, however, because an FSG running batch jobs a result, a keystroke entering the system right after a substantial service was given to its FSG is expected to wait longer. Typically, a keystroke needs a minimal amount of CPU time to complete.

Therefore, after being preempted, it is expected to stay in the highest user-priority queue within its FSG. This implies that the echo response time is heavily dominated by the group wait time the amount of time from a random arrival instant until the associated FSG earns its turn of using the CPU.

3.2 Analysis

Let x_k denote the group service time per CPU visit of FSG k. Certainly x_k is related to the workload of FSG k and fluctuates with time. For simplicity. We assume that x_k is quasi-static, i.e., x_k changes very slowly with time. We will treat x_k as a constant over several services. Also, we assume that FSG k steadily receives its share of CPU usage rate, say S_k, $0 < S_k < 1$. Then, FSG k obtains one service (of length x_k) per $\frac{x_k}{S_k}$ seconds and therefore, the interservice time is given by $\frac{x_k}{S_k} - x_k$ or $\frac{x_k[1-S_k]}{S_k}$. A preempted keystroke may arrive any time during each period of $\frac{x_k}{S_k}$ seconds with a uniform probability density $\frac{S_k}{x_k}$. If it arrivers during the service interval of FSG k, the group wait time w_k would be zero. Otherwise, w_k can range from zero to $\frac{x_k}{S_k}[1-S_k]$ with an equal density $\frac{S_k}{x_k}$. Therefore, the density of w_k can be written as

$$f[w_k | x_k] = S_k \delta[w_k] + \frac{S_k}{x_k} \quad 0 \le w_k \le t_{\max}[x_k] \tag{1}$$

where $\delta(\cdot)$ is the Dirac delta function and where

$$t_{\max}[x_k] = \frac{x_k}{S_k}[1-S_k] \tag{2}$$

is the worst-case wait time given x_k. The group expansion factor $\frac{1-S_k}{S_k}$ plays a major role in $t_{\max}[x_k]$. For $S_k = 20\%$, this factor is four and for $S_k = 10\%$, this factor is nine. Also, note that x_k may exceed 1 second even though a 1-second service quantum is maintained by FSS. The average group wait time can be computed as follows:

$$E[w_k | x_k] = \int_0^{tmax[x_k]} w_k f[w_k | x_k] dw_k$$
$$= \frac{S_k}{2x_k} t^2_{\max}[x_k] = \frac{x_k}{2S_k}[1-S_k]^2 \tag{3}$$

Also, we define a 95-percintile group wait time, $t_{95}[x_k]$, to be the number such that

$$\Pr ob\{w_k \ge t_{95}[x_k]\} = 5\%$$

It is readily shown that

$$t_{95}[x_k] = \frac{0.95 - S_k}{1-S_k} t_{\max}[x_k] = \frac{x_k}{S_k}[0.95 - S_k] \tag{4}$$

Apparently, given S_k, echo response time (the average, the worst, or the 95-percentile) is linearly increasing with x_k.

Tables 1 and 2 show $t_{\max}[x_k]$, $E[w_k | x_k]$ and $t_{95}[x_k]$ for various values of x_k with $S_k = 40\%$ and $S_k = 10\%$ respectively. A comparison of Table 1 and 2 exhibits a substantial impact of S_k on echo response time. In reality, system traffic fluctuates and thus, the FSS cannot consistently provide every FSG with its contracted share. As a result, the instantaneous CPU share allocated to FSG k (say, S'_k) fluctuates, even though

$E[S_k]=S_k$ may be maintained by the FSS. Naturally it would be more accurate to use S'_k instead of S_k in determining echo response time. It is essential to point out that S'_k is related to system load. If FSG k has a considerable amount of traffic, then S'_k would tend to be larger when system load is light and smaller when system load is heavy. This implies that system load has a substantial impact on echo response time. In summary, excessive echo response time is caused by FSS through the following two factors:

(i) An FSG with a smaller CPU share has a larger group expansion factor.
(ii) If an FSG consumes a larger CPU time per service, it is associated with a longer interservice time.

Table 1 $S_k=40\%$

x_k	$E[w_k \mid x_k]$	$t_{\max}[x_k]$	$t_{95}[x_k]$
0.1 sec	0.045 sec	0.15 sec	0.1375 sec
0.5 sec	0.225 sec	0.75 sec	0.6875 sec
1 sec	0.45 sec	1.5 sec	1.375 sec
2 sec	0.9 sec	3 sec	2.75 sec

Table 2 $S_k=10\%$

x_k	$E[w_k \mid x_k]$	$t_{\max}[x_k]$	$t_{95}[x_k]$
0.1 sec	0.405 sec	0.9 sec	0.85 sec
0.3 sec	1.215 sec	2.7 sec	2.55 sec
0.5 sec	2.025 sec	4.5 sec	4.15 sec
1 sec	4.05 sec	9 sec	8.5 sec

3.3 Echo Response Time Versus System Load

In this section, we study the relationship between group wait time w (which dominates echo response time) and CPU utilization rate ρ (i.e., system load) based on a simple FSS model. This model has the essenceof FSS and is intended for analytical tractability. The result is thus, model-dependent and should not represent the functional relationship between w and ρ in general. Nevertheless, it does provide a fundamental guideline as to how echo response time reacts to system load in FSS. We assume that there are M fair share groups, FSG 1, FSG2, , FSG M , in the system and that each FSG has an equal contracted share of CPU usage: i.e., $S_i=\frac{1}{M}$, $1\le i\le M$. Furthermore, we assume that each FSG has an equal CPU utilization rate, i.e.,

$$\rho_1=\rho_2=A=\rho_M=\frac{\rho}{M}$$

where ρ is the overall CPU utilization rate. Then we have:

ρ= Prob {an FSG has a runnable process when the CPU is available to it}

Now consider a test keystroke arriving at FSG 1 whose instantaneous group priority is n, Note that, by symmetry, the random integer n is equally likely to assume any integer from 1 to M ; that is

$$p(n)=\frac{1}{M},\quad 1\le n\le M \tag{5}$$

There are n 1 groups ahead of FSG 1. Each group has a probability ρ to offer a process when CPU calls it. Let Jbe the number of groups that the test job of FSG 1 has to wait, then,

$$p(J=j \mid n)=\binom{n-1}{j}=p^j(1-p)^{n-1-j}\quad 0\le j\le n-1 \tag{6}$$

Unconditioning on n, we obtain:

$$p(J=j)=\sum_{n=j+1}^{M} p(J=j \mid n)p(n)$$
$$=\frac{1}{M}\left[\frac{p}{1-p}\right]^j \sum_{n=j+1}^{M}\binom{n-1}{j}(1-p)^{n-1}\quad 0\le j\le n-1 \tag{7}$$

Let $\bar{x} = \bar{x}(p)$ be average service time per group per CPU visit. Then we have the average group wait time given J = j :

$$E[w \mid J=j] = 0 \text{ for } j=0 \tag{8a}$$

$$E[w \mid J=j] \geq \left[j - \frac{1}{2}\right]\bar{x} \text{ for } 1 \leq j \leq M-1 \tag{8b}$$

Note that $\bar{x}(\rho)$ increases with group utilization rate $\frac{\rho}{M}$, or with ρ.

The deduction of $\frac{1}{2}$ from j accounts for the fact that the FSG currently being serviced has only an average residual time $\frac{\bar{x}}{2}$ remaining. The inequality is necessary because an FSG may receive more than one service before its priority falls below FSG 1's. Using (7) and (8), we may compute a lower for echo response time:

$$\begin{aligned} E[w] &= E[E(w \mid J=j] = \sum_{j=0}^{M-1} E[w \mid J=j]p(J=j) \\ &\geq \left[\sum_{j=1}^{M-1} jp(J=j) - \frac{1}{2}\sum_{j=1}^{M-1} p(J=j)\right]\bar{x} \\ &\geq \left[\frac{1}{2}\sum_{j=0}^{M-1} jp(J=j)\right]\bar{x} = \frac{1}{2}E[J]\bar{x} \\ &= \frac{1}{2}E[E(J \mid n)]\bar{x} = \frac{1}{2}E[(n-1)\rho]\bar{x} \text{ by (6)} \\ &= \frac{M-1}{4}p\bar{x} \text{ by (5)} \end{aligned} \tag{9}$$

For $M=10$ and $\bar{x} = a\rho$ second, we have $E(w) \geq 2.25a\rho^2$. Let us now consider the worst case in which $J = M-1$ and

$$E[w \mid J=M-1] \geq \left[M - \frac{3}{2}\right]\bar{x} \tag{10}$$

This worst-case group wait time would increase with ρ since $\bar{x}$ does. Above all, the probability that this worst case would occur is, by (7):

$$p[J=M-1]\frac{1}{M}\rho^{M-1} \tag{11}$$

which increases rapidly with ρ, especially when M is large. Alternatively, we may write:

$$p\left\{\text{average group wait time exceeds}\left[M - \frac{3}{2}\right]\bar{x}\right\} \geq \frac{1}{M}\rho^{M-1} \tag{12}$$

Therefore, not only the worst-case group wait time become unacceptable, but does the probability of its occurrence when ρ approaches 1. Although these results were obtained based on the assumption that all M FSGs have an identical CPU share $\frac{1}{M}$, they can be generalized to encompass more practical situations. For example, suppose FSG 1 has a CPU share of $\frac{1}{k}$ and that the CPU shares of the remaining $(M-1)$ FSGs are arbitrary. These $(M-1)$ FSGs have a combined CPU share of $\frac{k-1}{k}$. Then, as far as the group wait time for FSG 1 is concerned, the remaining $(M-1)$ FSGs are equivalent to $(k-1)$ FSGs each of which has a CPU share of $\frac{1}{k}$. Then, the results derived above apply (to FSG 1), provided that M is replaced by k. Essentially, it is the CPU share of an FSG rather than the total number of FSGs that determines its group wait time. Therefore, in general, we may apply the above results to an FSG whose CPU share is S by substituting $\frac{1}{M}$ by S and M by $\frac{1}{S}$.

4. CONCLUSIONS

One crucial feature of UNIX is its capability of offering interactive jobs excellent response time. The FSS degrades this capability by allowing FSG priorities to override all UNIX user priorities. To bring the UNIX advantage for interactive users, we need to release those non-CPU-bound interactive processes from FSS. Specifically, if these processes

are either made nonpreemptive or chosen to run regardless of their FSG associations, the group wait time, will no longer be part of echo response time. For the FSS, basically, each FSG has to compete against the entire system traffic for use of CPU. In Section 3, we derived a low bound for average group wait time, which increases quadratically with CPU utilization rate ρ. It was also indicated that the group wait time dominates echo response time. Now for the refined FSS, a non-CPU-bound interactive process competes only against other non-CPU-bound interative processes in the system in the system. These Processes altogether represents simply a small fraction, say C, of CPU utilization rate, i.e., c << 1. Let $\bar{y}$ be the average service time a keystroke needs; we have $\bar{y} << \bar{x}$, of course, where $\bar{x}$ has been used to denote the average service time per group per CPU visit. Then, using an $M/M/1$ queue model for keystrokes leads to an average echo response time[4]:

$$T_{RFSS} = \frac{\bar{y}}{1 - c\rho} \tag{13}$$

The equation (13) implies that T_{RFSS} increases nearly linearly with ρ. The improvement is very visible. Basically, the proposed idea entails a characterization of non-CPU-bound interactive processes that would enable the operating system to distinguish them other processes. The goal, of course, is to accommodate all qualified interactive processes while keeping others under the control of FSS. Achieving this goal entails novel classification schemes of non-CPU-bound interactive processes which are currently being explored.

※ References

[1] UNIX System V Release 2.0 User Reference Manual, AT&T Bell Laboratories, 1995.

[2] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey, Graphical Methods For Data Analysis, Wadsworth Statistics/Probability Series, 1993.

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

[4] L. Kleinrock, Queueing System Volume II: Computer Applications, Wiley, 1976.

[5] H. W. Lee, Queueing Theory, Sigma Press, 1998.

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

[7] J. S. Lim, S. H. Kim, "The Optimal Utilization Detremined By The Response Time," Vol. 2, No. 9, 2001.

[8] S. S. Lavenberg, Computer Performance Modeling Handbook, Academic Press, Inc, 1988.

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

[10] M. Sakata, S. Noguchi, and J. Oizumi, "An Analysis of M/G/1 Queue under Round-Robin Scheduling," Operations Research, Vol. 19, 371-385, 1971.

[11] C. Y. Lo, "Performance Analysis and Application of a Two-priority Packet Queue," AT&T Tech. J. Vol. 66, No. 3, 82-99 May-June 1987.

임 종 설

1979년 서울대학교 공과대학 공학사
1986년 Polytechnic University, New York 통신공학 공학박사
1986년-91년 AT&T 벨연구소 이동통신시스템개발 책임연구원
1991년-93년 한국이동통신 연구소 책임연구원
1993년-현재 선문대학교 전자정보통신공학부 교수
관심분야 : 이동통신, 데이타통신, 컴퓨터 네트워크