Understanding the Slowdown of Large Jobs in an M/GI/1 System [*]

Mor Harchol-Balter[†] Karl Sigman[‡] Adam Wierman[§]

ABSTRACT

We explore the performance of an M/GI/1 queue under various scheduling policies from the perspective of a new metric: the it slowdown experienced by largest jobs. We consider scheduling policies that bias against large jobs, towards large jobs, and those that are fair, e.g., Processor-Sharing. We prove that as job size increases to infinity, all work conserving policies converge almost surely with respect to this metric to no more than $1/(1-\rho)$, where ρ denotes load. We also find that the expected slowdown under any work conserving policy can be made arbitrarily close to that under Processor-Sharing, for all job sizes that are sufficiently large.

1. INTRODUCTION

It is well-known that choosing the right scheduling algorithm can have a big impact on performance, both in theory and in practice. For example, changing the scheduling algorithm in a CPU from Processor-Sharing (PS) to a scheduling policy that biases towards small jobs, such as Shortest-Remaining-Processing-Time-First (SRPT), or a scheduling policy that biases towards young jobs, such as Least-Attained-Service (LAS), can improve mean response time (a.k.a. sojourn time) dramatically.

However, less well understood is the performance impact of different scheduling policies on large jobs. For example, how does a policy that biases towards small jobs, such as SRPT, compare against a policy that biases towards large jobs, such as Longest-Remaining-Processing-Time-First (LRPT), when the performance metric is the response time of the large jobs?

Throughout we limit our discussion to a stable M/GI/1 queue with load $\rho < 1$ and arrival rate λ. We show that all *work conserving* scheduling policies have the same performance as PS with respect to large jobs. In particular, we show that the slowdown as job size tends to infinity under any work conserving policy is at most $\frac{1}{1-\rho}$ almost surely; even for policies that clearly bias against large jobs. We also consider the expected slowdown for jobs that are not the very largest. We show that all "sufficiently-large" jobs have slowdown arbitrarily close to that of PS, where the definition of "sufficiently-large" depends on ρ and includes most jobs provided ρ is not too high.

A job's *size* (service requirement) will be denoted by the random variable X and will be chosen i.i.d. from a continuous distribution with *finite mean* and *finite variance*. We will use T to denote the steady-state response time (a.k.a. sojourn time) and $T(x)$ to denote the steady-state response time for a job of size x; a customer arriving in steady-state bringing a service time of length x has a response time $T(x)$.

DEFINITION 1.1. *For any given policy, the slowdown, S, is defined as response time divided by job size, namely, $S = \frac{T(X)}{X}$. The slowdown for a job of size x, $S(x)$, is thus given by*

$$S(x) = \frac{T(x)}{x}.$$

The expected slowdown for a job of size x, $E[S(x)]$, is given by

$$E[S(x)] = \frac{E[T(x)]}{x}.$$

Mean slowdown is often used as a measure of system performance as opposed to the more traditional mean response time for two reasons [1, 2, 3]. First, it is desirable that a job's response time be correlated with its size (processing requirement). We'd like small jobs to have small response times and big jobs to have big response times. A second reason why we care about mean slowdown is that it is more representative of the performance of a large fraction of jobs.

[*]This is a brief introduction to our recent Technical Report CMU-CS-02-118.

[†]Carnegie Mellon University, Computer Science Department. Email: harchol@cs.cmu.edu. This work was supported by NSF Career Grant CCR-0133077 and by Pittsburgh Digital Greenhouse Grant 01-1.

[‡]Columbia University, Department of Industrial Engineering and Operations Research. Email: sigman@ieor.columbia.edu

[§]Carnegie Mellon University, Computer Science Department. Email: acw@cs.cmu.edu.

Let us now introduce the scheduling policies investigated in this article:

Processor-Sharing (PS): Under PS the processor is shared fairly among all jobs currently in the system [5]. It is well known that for an M/GI/1/PS queue, $E[S(x)]^{PS} = \frac{1}{1-\rho}$. This says that for any given load $\rho < 1$, under PS scheduling, all jobs have the same expected slowdown; hence PS is *fair*.

Shortest-Remaining-Processing-Time-First (SRPT): Under SRPT, at every moment of time, the server is processing that job with the shortest remaining processing time. The SRPT policy is well-known to be optimal for minimizing mean response time [4].

Preemptive-Last-Come-First-Served (P-LCFS): Under P-LCFS, whenever a new arrival enters the system, it immediately preempts the job in service. Only when that arrival completes does the preempted job get to resume service.

Least-Attained-Service (LAS): Under LAS the job with the least attained service gets the processor to itself. If several jobs all have the least attained service, they time-share the processor via PS. This is a very practical policy, since a job's *age* (attained service) is always known, although it's size may not be known.

Longest-Remaining-Processing-Time-First (LRPT): Under LRPT, at every moment of time, the server is processing the job with the longest remaining processing time. If multiple jobs in the system have the same remaining processing time, they time-share the processor via PS. Since the LRPT policy biases towards the *longest* jobs, it is of little practical value.

2. RESULTS

THEOREM 2.1. *As $x \to \infty$, expected slowdown for SRPT, P-LCFS, LAS, and LRPT is the same as for PS:*

$$\begin{aligned}\lim_{x\to\infty} E[S(x)]^{SRPT} &= \lim_{x\to\infty} E[S(x)]^{P-LCFS} = \lim_{x\to\infty} E[S(x)]^{LAS} \\ &= \lim_{x\to\infty} E[S(x)]^{LRPT} = \frac{1}{1-\rho}.\end{aligned}$$

That is, the expected slowdown for the largest job is the same under policies that bias towards short jobs, policies that bias towards long jobs, and policies that treat all jobs fairly.

THEOREM 2.2. *For any work conserving scheduling policy*

$$\lim_{x\to\infty} E[S(x)] \leq \frac{1}{1-\rho}.$$

If the policy is also non-preemptive, then $E[S(x)] \to 1$ as $x \to \infty$.

REMARK 2.1. *Theorem 2.2 does not extend to policies that are not work conserving. In fact, for every $z \in [1,\infty)$ there is a non work conserving policy such that $E[S(x)] \to z$ as $x \to \infty$.*

REMARK 2.2. *The $\frac{1}{1-\rho}$ bound in Theorem 2.2 is tight. In fact, for every $z \in [1, \frac{1}{1-\rho}]$ there is a work conserving policy such that $E[S(x)] \to z$ as $x \to \infty$.*

The above remarks show that the metric $\lim_{x\to\infty} E[S(x)]$ defines a taxonomy on all scheduling policies. Non work conserving policies have a value in $[1,\infty)$ under this metric. Preemptive work conserving policies have a value in $[1, \frac{1}{1-\rho}]$ under this metric. Non-preemptive work conserving policies all have a value of 1 under this metric. Each class is complete in that for each value in the range, there exists a policy with that value.

Until now we have concentrated on the limiting behavior as the job size $x \to \infty$. We now show that for all "sufficiently large" jobs, under any work conserving policy the performance can be made arbitrarily close to that under PS. Let V be the amount of work in the system when a job arrives, i.e. $E[V] = \frac{\lambda E[X^2]}{2(1-\rho)}$.

THEOREM 2.3. *Fix $\varepsilon > 0$. Under any work conserving scheduling policy P, if $x \geq \frac{1}{\varepsilon}E[V]$, then*

$$E[S(x)]^P \leq (1+\varepsilon)E[S(x)]^{PS} = \frac{1+\varepsilon}{1-\rho}.$$

If the policy is also non-preemptive and $x \geq \frac{1}{\varepsilon(1-\rho)}E[V]$, then

$$E[S(x)]^P \leq 1+\varepsilon$$

Finally we state stronger versions of Theorem 2.1 and Theorem 2.2:

THEOREM 2.4. *Under Processor-Sharing it holds a.s. that*

$$\lim_{x\to\infty} S(x)^{PS} = \frac{1}{1-\rho}$$

.

THEOREM 2.5. *Under work conserving scheduling policies it holds a.s. (assuming the limit exists) that*

$$\lim_{x\to\infty} S(x) \leq \frac{1}{1-\rho}.$$

If the policy is also non-preemptive, then the limit does exist and $S(x) \overset{a.s.}{\to} 1$ as $x \to \infty$.

3. REFERENCES

[1] Baily, Foster, Hoang, Jette, Klingner, Kramer, Macaluso, Messina, Nielsen, Reed, Rudolph, Smith, Tomkins, Towns, and Vildibill. Valuation of ultra-scale computing systems. White Paper, 1999.

[2] Allen B. Downey. A parallel workload model and its implications for processor allocation. In *Proceedings of High Performance Distributed Computing*, pages 112–123, August 1997.

[3] M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3), 1997.

[4] Linus E. Schrage and Louis W. Miller. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research*, 14:670–684, 1966.

[5] Ronald W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.