Modeling and Evaluation of SWAP Scheduling Policy Under Varying Job Size Distributions

Idris A. Rai
Makerere University
Faculty of Computing and Informatics Technology
Kampala, Uganda
rai@cit.mak.ac.ug

Michael Okopa
Makerere University,
Faculty of Computing and Informatics Technology
Kampala, Uganda
michaelokopa@yahoo.co.uk

***Abstract*—Size-based scheduling policies have been shown to be effective resource allocation policies in computing and networked environments. One of the recently proposed size-based scheduling policy is called SWAP. It is a non-preemptive, threshold based policy that was proposed to approximate the Shortest Job First (SJF) policy by introducing service differentiation between short and large jobs such that short jobs are given service priority over the large jobs. Original study of the SWAP scheduling policy was based on only simulations, which are known to have a number of restrictions. In this paper, we derive SWAP models and evaluate the scheduling policy using workloads that have varying distributions. In contrast to simulations, the models enable fast analysis of the scheduling policy under a wide range of input parameters. Numerical results obtained from the derived models show that SWAP approximates SJF better for heavy-tailed workloads than for exponentially distributed workloads. We also show that SWAP performs significantly better than First Come First Serve (FCFS) and Processor Sharing (PS) policies regardless of the distribution of the workload.**

***Keywords*-Size-based scheduling; conditional mean response time; temporal dependence.**

I. INTRODUCTION

Motivated by the persistent evidence of heavy-tail distribution of stored and transferred file sizes, size-based scheduling policies have been widely studied for efficient resource allocation in time-sharing computing environments. Most size-based scheduling policies however require the knowledge of job sizes, which present a major limitation to their practical implementations. Examples of common size-based scheduling policies include Shortest Job First (SJF), which is a non-preemptive scheduling policy that gives service to the shortest job immediately after the job in service completes. Shortest Remaining Processing Time (SRPT), on the other hand, is a preemptive variant of SJF. It favors short jobs by giving service to the job in the queue that has the shortest remaining processing time. In order to know the remaining processing time however, one needs to know the total service required by the job (i.e., the size of the job). SRPT is known to be the optimal policy in terms of providing the minimum mean response time [3], [1]. Despite its optimum performance, SRPT scheduling is not widely used in practice particularly in network environments due to lack of information on flow sizes [2]. Some authors have as a result investigated the use of SRPT without accurate knowledge of job sizes [4], [5].

Blind scheduling policies, which are policies that don't require job sizes, are therefore often preferred in practice due to their implementation simplicity. Some popular examples of blind scheduling policies include Processor Sharing (PS), First Come First Served (FCFS), and Round Robin [3]. There also exist blind size-based scheduling policies; which are blind scheduling policies that take into account a notion of size. The most popular example is Least Attained Service (LAS) first, which is a preemptive scheduling policy that favors short jobs by giving service to the job in the system that has received the least service [3], [7], [8]. LAS doesnt require the knowledge of job sizes, and therefore can be used in network routers. However, its implementation requires a router to keep track of states of each flow that traverses the router, which may be a very daunting task in high speed networks.

In this paper, we study another recently proposed size-based scheduling policy called SWAP [6]. SWAP is size-based policy that was proposed to improve on the response time of short jobs while reducing the overhead required in identifying job sizes. The basic idea behind SWAP is to use the measured serial correlation of the service times to estimate the missing information of job sizes. Once these reliable estimates of the job service times are available, large jobs are delayed by putting them at the tail of the queue whereas short jobs are kept at the head of the queue for immediate service. SWAP uses a threshold value to decide which jobs are to be delayed. In such a way, delayed long jobs are served after most short jobs in the queue have completed their service. We review SWAP policy in detail in Section III-A.

In an effort to improve the performance of systems, studies have shown that the recent past is indicative of the near future. It is a generalization of the idea of locality which can be exploited to leverage the need for knowing sizes of all jobs. Furthermore, it can be observed that the real workload data is far from independently and identically distributed. Instead, similar jobs tend to arrive within bursty periods. This observation is vividly true when the workload exhibits heavy tailed distribution. SWAP was originally proposed to take advantage of this observation in accomplishing its goals. Authors in [6] proposed SWAP to approximate the behavior of the optimal SJF scheduling policy by using workload temporal dependence to forecast job service times without any a priori

 ISBN:978-1-61208-113-7

knowledge of upcoming job demands.

Original study of SWAP was conducted through *only* simulations of the policy. Simulations techniques however have some limitations; they are restrictive in the sense that often only programmers can comfortably adopt them, and it often takes long to obtain results for a wide range of input parameters. To complement the original work on SWAP, in this paper, we derive analytical models of SWAP in terms of the conditional mean response time of jobs. The models can be used for quick performance evaluation of the policies. We investigate the performance of SWAP in terms of approximating SJF under varying job size distributions, and numerically compare the performance of SWAP to FCFS and Processor sharing (PS) policies.

The rest of the paper is organized as follows: in the next section, we present mathematical background that guides models derivation of SWAP policy. In Section III, we review SWAP scheduling and derive its models. We evaluate SWAP in Section IV, and finally conclude the paper in Section V.

II. Mathematical Background

Let's denote the probability density function (pdf) of a job size distribution as $f(x)$. The cumulative distribution function ($F(x)$) is obtained as $F(x) = \int_0^x f(t)dt$, and the survival function (or reliability function) is given as $F^c(x) = 1-F(x)$. We define $\overline{x_x^n} = \int_0^x t^n f(t)dt$ to be the n^{th} moment for jobs that are less than or equal to x. Therefore, $\overline{x_x}$ is the mean and $\overline{x_x^2}$ is the second moment of the job size distribution due to job sizes less than or equal to x. The mean and second moments are obtained when the value of subscript x is infinity.

Let x_l be a large job under SWAP scheduling, i.e., any job that is greater than a specified threshold (x_t). It follows that $\overline{x_{x_l}^n} = \int_{x_l}^\infty t^n f(t)dt$ is the n^{th} moment for job sizes greater than x_l.

The load due to jobs with sizes less than or equal to x is given as $\rho_x = \lambda \int_0^x tf(t)dt$ while the load due to jobs with sizes greater than x_l is given as $\rho_{x_l} = \lambda \int_{x_l}^\infty tf(t)dt$. Also $\rho_{x_l} = \rho - \rho_{x_l}$ where ρ is the total load in the system. We next define the expressions for the conditional mean response time under FCFS and SJF, which we shall use when deriving the models for SWAP policy.

An arriving job to a FCFS queue has to wait for all jobs it finds in the queue upon its arrival. Therefore, the conditional average response time of a job of size x in an M/G/1/FCFS system is given as

$$T(x) = x + W(x), \tag{1}$$

where $W(x) = \frac{\lambda \overline{x^2}}{2(1-\rho)}$ is the mean waiting time due to jobs in the system. Assume that an arriving job x finds only the jobs that are less than or equal to a job size x_t in the M/G/1/FCFS queue. Its conditional average response time $T(x)$ is given as

$$T(x_t) = x + W(x_t), \tag{2}$$

where $W(x_t) = \frac{\lambda \overline{x_{x_t}^2}}{2(1-\rho_{x_t})}$.

Under SJF, the shortest job in the queue is given non-preemptive priority. Thus, at every completion instant of a job in the server, the next job to receive service is the smallest job in the queue. A job of size x is therefore delayed by only jobs in the system that are less than or equal to its size. The conditional average response time of the job size of x under SJF is given as

$$T(x_x) = x + W(x_x), \forall x > 0, \tag{3}$$

where $W(x_x) = \frac{\lambda \overline{x_x^2}}{2(1-\rho_x)^2}$.

We will numerically evaluate the SWAP models under job sizes with exponential distribution and job sizes with Bounded Pareto distribution to mitigate workloads with varying variances. The variability of a job size distribution is determined by its Coefficient of Variation (C), which is defined as the ratio of the standard deviation to the mean of a distribution. Exponential distribution has a low variability since its $C = 1$, whereas a Bounded Pareto distribution has a high variability ($C > 1$). In this paper, we specifically use exponential distribution and Bounded Pareto $BP(10, 5 * 10^5, 1.1)$ distributions with mean values of 72.7 to numerically evaluate SWAP models. Similar distributions have been used in [1], [7]. The probability density function of an exponential distribution is given as:

$$f(x) = \mu e^{-\mu x}, x \geq 0, \mu \geq 0. \tag{4}$$

Bounded Pareto distributions have commonly been used to evaluate the performance of systems under heavy tailed workloads with high variance [1], [7], [8]. In contrast to Pareto distributions which assume infinite largest job size, Bounded Pareto distributions can be used to represent realistic workload with known largest values. We denote Bounded Pareto distribution by $BP(k, P, \alpha)$ where k and P are the minimum and the maximum job sizes and α is the exponent of the power law. The pdf of the Pareto is given as:

$$f(x) = \frac{\alpha k^\alpha}{1-(k/P)^\alpha} x^{-\alpha-1}, \quad k \leq x \leq P, \quad 0 \leq \alpha \leq 2. \tag{5}$$

In the next section, we discuss SWAP scheduling and derive its models.

III. SWAP Scheduling Models

A. A review of SWAP Scheduling

SWAP is a class-based, non-preemptive, size-based scheduling policy where jobs are classified into two classes based on their sizes, namely *short* (x_s) jobs and *large* (x_l) jobs classes. SWAP uses a rather naive definition of job size based on threshold (x_t), which can be dynamic. All jobs that are less than or equal to x_t are classified as short whereas jobs that larger than x_t are classified as large jobs. The main goal of SWAP scheduling policy is to approximate the SJF scheduling policy so as to favor short jobs without apriori knowledge of job service requirements or job sizes. It reduces the mean response time for short jobs by designating higher priority to short jobs compared to large jobs.

 ISBN:978-1-61208-113-7

SWAP starts by serving all arriving jobs to a queue in FCFS manner, and compares the service given to each job to the threshold value. If a large job is served, next the entire queue is *scanned* whereby size of each job in the queue is computed and jobs in the queue are classified and marked as large or short. Once classified, short jobs are moved at the head of the queue and receive service before large jobs that are kept at the tail of the queue. We shall term these scanned large jobs as (*delayed*). Jobs within a class are serviced in their order of arrival using FCFS scheduling. Once a job has been classified under SWAP scheduling it will belong to that class for all duration of its stay in the queue.

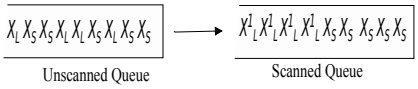

Fig. 1. Illustration of unscanned and scanned queues in SWAP(1)

It can be seen that compared to FCFS scheduling, SWAP favors short jobs to the expense of delaying large ones within the queue. Intuitively, SWAP policy should provide significant performance improvement in terms of reducing mean response time of short flows particularly for highly varying workloads where only a tiny fraction of jobs contribute to more than half of the system load.

Jobs that arrive to a scanned SWAP queue are buffered in order of their arrival and are eventually served in FCFS order once all scanned jobs in the queue have completed their service. We call the state where served jobs in the queue are not scanned an *unscanned* state of the queue. SWAP therefore alternates between scanned and unscanned states.

Large jobs under SWAP scheduling can be delayed more than once. We denote SWAP(i) as SWAP scheduling policy which delays large jobs i times. Consider SWAP(1), the service time of a large job that arrives to an unscanned queue is interrupted by the short jobs that were in the queue upon its arrival and the one large job that triggers the scanning of the queue. The scanning evet will delay the large jobs once. They will receive service immediately after the short scanned jobs have completed their service. It can be seen that under SWAP(1), the newly arriving short jobs in a scanned queue have to wait until all scanned large jobs in the queue complete their service before they can receive any service. Figure 1 illustrates SWAP(1) scheduling at scanned and unscanned states where X_S and X_L^1 denote a short job and large job that has been delayed 1 time.

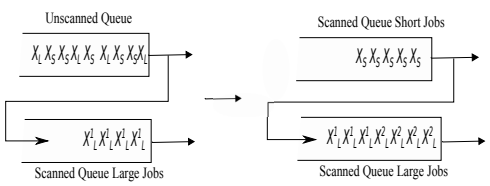

Fig. 2. Illustration of SWAP(2)

Increasing delays of large jobs under SWAP improves the service of short jobs further by avoiding disruptions of their service due to some large jobs that arrived in the queue before them. Consider SWAP(2), for instance, once the first short scanned jobs are served, the large scanned jobs wont receive service immediately. Instead, the server will serve the arriving unscanned jobs in the queue until the scanning is triggered again by an unscanned large job. At that point, the entire queue will be scanned to classify the jobs. The server will then give priority to the scanned short jobs in the queue before it can serve the large jobs that have been delayed twice.

Once large jobs have been delayed 2 times under SWAP(2), they receive service immediately after all classified short jobs in the queue have completed their services. Figure 2 demonstrates SWAP(2), where again X_L and X_S denote large and small jobs, and X_L^i are large jobs that have been delayed i times. Note that we represent SWAP with two queues where the second queue hosts delayed large jobs after they are scanned. Delayed scanned jobs wait in the second queue until they are delayed for i times equivalent to $i-1$ new scanning events since the time they were scanned.

B. Modeling SWAP Scheduling

In this section, we derive models of SWAP scheduling in terms of conditional mean response time of short jobs and large jobs. We consider a tagged job that arrives to an M/G/1/SWAP(i) system at scanned and unscanned states separately.

1) Models for arrivals at unscanned state: Assume a tagged job arriving to a SWAP queue in an unscanned state, but *just* before a large job receives service. Recall that such a large job will trigger the scanning of the queue. If the tagged job is short it will be placed at the tail of the short jobs class, otherwise it will be placed at the tail of the queue.

Let's consider SWAP(1), the tagged short job will be delayed by the mean residual life of the large job that triggered the scanning and will then wait for all scanned short jobs it finds in the queue before it receives service. The waiting time of the tagged job due to these jobs is given by $W(x_t)$ defined in Section II. On the other hand, the tagged large job will be delayed by all jobs it finds in the system upon its arrival by the mean waiting time denoted by $W(x)$. Note that $W(x)$ represents the delay of short and large scanned jobs. The corresponding delays due to separate jobs classes are denoted as $(W(x_l))$ for large jobs and $W(x_t)$ for short jobs. The resulting conditional mean response times under SWAP(1) at unscanned state are given as:

$$T(x) = \begin{cases} W(x_t) + x & x \leq x_t \\ W(x_l) + W(x_t) + x & x > x_t \end{cases}$$

Note that the conditional mean response time for a large job here is the same as the conditional mean response time of the job under the FCFS queue (Equation 1).

We now derive the models for SWAP(2), under which large jobs are delayed twice before they receive service. We assume the steady state case of the queueing system at which arriving jobs will always find scanned large jobs that have been delayed

once waiting in the queue. These delayed jobs will receive service immediately after the newly scanned short jobs have completed their service.

Again, assume a tagged job as the last job that arrives to the queue before the queue turns from unscanned state to scanned state. Let the tagged job be a short job; it will be delayed by all short jobs it finds in the queue plus the remaining service of the large job it finds in the server when it arrived. The expression of its conditional mean response time is the same as the conditional mean response time of short job arriving at SWAP(1) at unscanned state shown in Equation (III-B1). If the tagged job is a large job, it will be delayed by all jobs it finds in the queue, which include all the short and large jobs that have just been scanned, and the large jobs that were scanned in the previous scanning event and the large jobs that were delayed in the queue upon its arrival. Since itself has to be delayed twice, the job will also be delayed by arriving new short jobs, both unscanned and scanned. The conditional mean response time of the tagged job arriving during unscanned state of SWAP (2) is therefore given as follows:

$$T(x) = \begin{cases} W(x_t) + x, & x \le x_t \\ 2W(x_l) + 2W(x_t) + \overline{x}F^c(x_l) + x, & x > x_t \end{cases} \quad (6)$$

The general expression for SWAP(i) can be obtained using iterative method, and is given in Equation (7).

$$T(x) = \begin{cases} W(x_t) + x, x \le x_t \\ iW(x_l) + iW(x_t) + (i-1)\overline{x}F^c(x_l) + x, \\ \qquad\qquad x > x_t \end{cases} \quad (7)$$

Observe that the derived expressions for conditional mean response times dont show performance gain acquired by short jobs from delaying large jobs more times. To intuitively see the reduction on short jobs response times, note that if large jobs arent delayed, the short jobs would be served after the scanned large jobs which would in turn increase their mean response time by $W(x_l)$.

2) Models for arrivals at scanned state: We now consider a tagged job arriving to a scanned queue. We derive models for the worst case scenario where the tagged job finds in the queue scanned short jobs being serviced, all scanned large jobs and other unscanned jobs including at least one large unscanned job waiting in the queue. This tagged job will experience the longest mean response time under SWAP. Other scenarios that we shall skip due to space limitation include the tagged job arriving just after a scanning event and a tagged job arriving to a scanned queue with only short unscanned jobs. The analyses of these skipped scenarios however are straight forward.

Let's assume the tagged short job is arriving to a scanned queue of SWAP(1) policy under the worst case scenario presented above, its service will be delayed by all scanned jobs it finds in the queue, all unscanned short jobs that it finds in the queue, and finally one large unscanned job that will trigger the next scan. The conditional mean response time of the tagged short job will include mean waiting time due to the service of the remaining scanned short jobs ($W_r(x,t)$), mean service time of scanned large jobs ($W(x_l)$), the service of the single large job that triggers the scanning ($\overline{x}F^c(x_l)$), and finally the mean service time of unscanned short jobs it finds in the queue ($W(x_t)$). If we assume that the tagged job arrives at the queue at a random point after the queue is scanned, we can further approximate $W_r(x_t)$ as $W(x_t)/2$.

The tagged large job, on the other hand, will additionally be delayed by large jobs that were newly scanned along with itself by a mean waiting time of $W(x_l)$. The conditional mean response time for the tagged job is obtained as follows:

$$T(x) = \begin{cases} W(x_l) + 3W(x_t)/2 + x + \overline{x}F^c(x_l), & x \le x_t \\ 2W(x_l) + 3W(x_t)/2 + x + \overline{x}F^c(x_l), & x > x_t \end{cases}$$

For SWAP(2), the tagged short job arriving after scanning will see in the queue scanned short jobs, scanned large jobs that have been delayed once, and unscanned short jobs and large jobs. In contrast to a short job under SWAP(1), the service of the tagged short job here will not be interrupted by the delayed large jobs since they have to be delayed once more. Therefore, the job's response time will be due to the remaining scanned short jobs by approximate of $W(x_t)/2$, unscanned short jobs it finds in the queue by mean waiting time of $W(x_t)$, and the one unscanned large jobs that will trigger the next scanning event.

For a tagged large job, it will be additionally delayed by the scanned large jobs it finds in the queue by mean waiting time of $W(x_l)$, all short jobs that will arrive until the next scanning event by mean waiting time of $W(x_t)$, the large job that will trigger the next scanning event, and any large jobs that it finds in the queue by their mean waiting delay of $W(x_l)$. Using similar arguments as before, we obtain the expression for conditional response time for the job as follows:

$$T(x) = \begin{cases} 3W(x_t)/2 + x + \overline{x}F^c(x_l), & x \le x_t \\ 2W(x_l) + 5W(x_t)/2 + x + 2\overline{x}F^c(x_l), & x > x_t \end{cases}$$

The general expression for SWAP(i) model for short jobs isnt very obvious to derive. We present instead the general model large jobs arriving at scanned states as follows:

$$T(x) = iW(x_l) + (i+1/2)W(x_t) + x + i\overline{x}F^c(x_l), i \ge 1,\ x > x_t$$

In the next section, we present numerical results showing the performance of SWAP and its comparison with SJF and PS scheduling policies

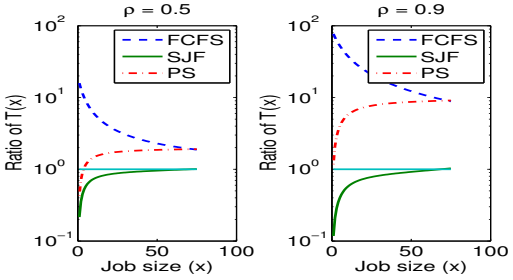

Fig. 3. Ratios of $T(x)$ vs x exponential workloads, $x_t = 75$

IV. Performance Evaluation

In this section, we use the derived SWAP models to evaluate its performance. We look at how SWAP approximates the SJF for short and large jobs, and we also compare its performance with that of FCFS and PS policies. We also investigate the impact of threshold values (x_t) to the performance of SWAP. Processor sharing (PS) is one of the mostly studied policy in time-sharing operating systems. It is also known as a fair scheduling policy providing conditional mean response time of $x/(1-\rho)$ for a job with size x.

We use exponential and Bounded Pareto distributions presented in Section II at low and high system load values of $\rho = 0.5$ and $\rho = 0.9$ respectively. For each set of result, the threshold values were chosen such that ρ_{x_t} under both considered distributions are close to each other. Due to space limitations, we numerically evaluate SWAP models for jobs that arrive to unscanned state only which we derived in Section III-B1.

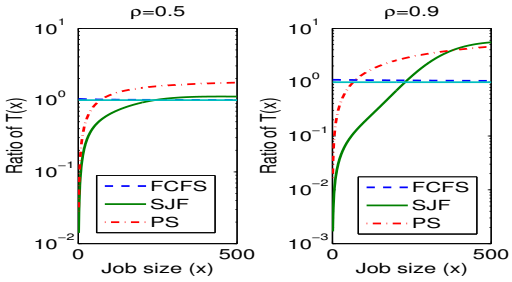

Fig. 4. Ratios of $T(x)$ vs x for exponential workloads, $x_t = 500$

Figures 3 and 4 show the ratios of conditional mean response time of short jobs under FCFS, PS, and SJF policies to that of SWAP for threshold values of $x_t = 75$ and $x_t = 500$ respectively for the case of exponentially distributed workload. It can be observed from the figures that SWAP indeed approximates the SJF for low load and short threshold values where the response time ratio of SWAP to SJF is always close to one. The approximation is more accurate for larger jobs compared to short jobs. The estimates under SWAP are less accurate for larger thresholds and higher loads as seen in Figure 4 at $\rho = 0.9$.

We can also see from the figures that SWAP performs much better than FCFS and PS for small threshold values regardless of the load. At high threshold values, SWAP offers similar mean response time as FCFS since the scanning event is triggered by very large jobs which makes SWAP scheduling for even short jobs the same as SJF. This can be seen in Figure 4 where $x_t = 500$ and $\rho_{500} = 0.89$.

Similarly, Figures 5 and 6 show results of SWAP in comparison with SJF, FCFS, and PS under Bounded Pareto distributed workloads presented in Section II. Compared to the results for exponential workloads, we can quickly see that SWAP approximates SJF better under Bounded Pareto distribution for all load values and varying threshold values. Bounded Pareto workloads are heavy tailed meaning more than 99% of their

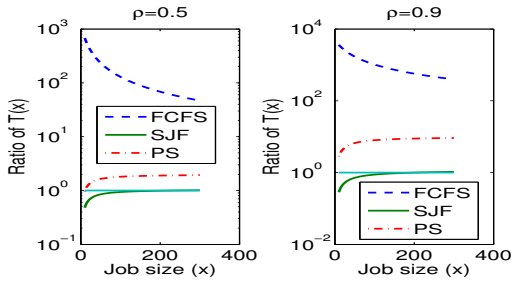

Fig. 5. Ratios of $T(x)$ vs x for Bounded Pareto workloads, $x_t = 300$

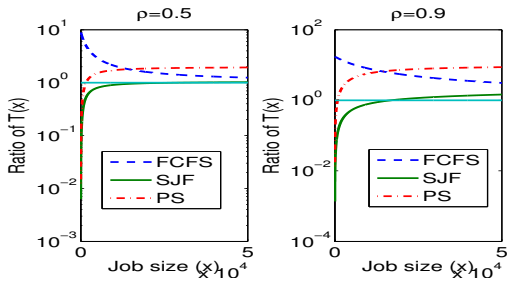

Fig. 6. Ratios of $T(x)$ vs x for Bounded Pareto workloads, $x_t = 50000$

jobs are short and constitute to only half of the total load. Consequently, most of the jobs under Bounded Pareto workloads are classified as short, and by delaying the few very large ones, the performance of short jobs is improved significantly. However, at large thresholds, regardless of system load, very short jobs under SWAP experience longer mean response times than under SJF (see Fig. 6), meaning SWAP is inaccurate in approximating SJF.

We also observe that the mean response time provided by SWAP compared to that of FCFS and PS under Bounded Pareto job size distributions follow similar trends to the performance under exponential distribution. In general, SWAP performs much better than FCFS for short jobs. SWAP also performs better than PS except for very few short jobs under SWAP with large threshold values such as in Fig. 6.

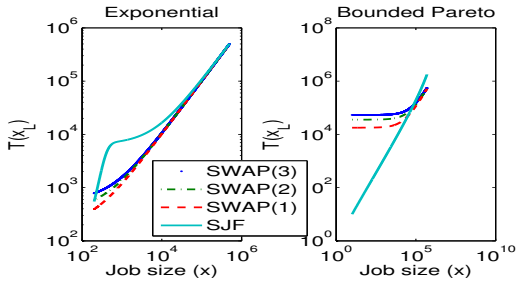

Fig. 7. $T(x)$ vs x_L for exponential with $x_t = 200$ and Bounded Pareto with $x_t = 5000$, $\rho = 0.9$

We finally investigate the conditional mean response time of large jobs under exponential and heavy tailed workloads for SWAP system with high and low loads. We specifically compare the performance of large jobs under SWAP and SJF to see how well SWAP approximates SJF for large jobs, and

 ISBN:978-1-61208-113-7

under SWAP(i) for $i = 1, 2, 3$ to see how much additional response time large jobs experience as a result of being delayed further. We exclude results that compare SWAP to PS and FCFS for large jobs due to space limitation.

We observe from Figure 7 that for the considered parameters, all SWAP(i) offers similar performance and all approximate SJF well for the very large jobs under exponential workload. We also observe that some shorter jobs experience longer mean response time under SJF than under SWAP policies. The situation is however very different for heavy tailed workloads where SWAP with larger i offering noticeably worse conditional mean response time than SWAP with smaller i. The performance of SWAP also significantly differs from SJF specifically for shorter jobs where SWAP performs worse and for the largest jobs where SWAP instead performs better that SJF. The largest jobs under SJF are interrupted by all jobs in the system which is not the case for SWAP. On the other hand, shorter jobs under SWAP are interrupted by large jobs of any size compared to only jobs that are less than their sizes under SJF.

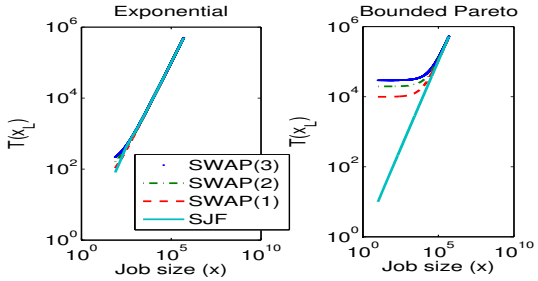

Fig. 8. $T(x)$ vs x_L for exponential with $x_t = 75$ and Bounded Pareto with $x_t = 300$, $\rho = 0.5$

Figure 8 shows the results at low load of $\rho = 0.5$, and smaller threshold values of $x_t = 75$ and $x_t = 300$ for exponential and heavy tailed workloads respectively. We can easily see that the performance of all SWAP(i) policies is similar for the case of exponential distribution. In this case, we also see that SWAP approximates SJF very well. However, for heavy-tailed workloads the performance of shorter jobs is still worse under SWAP than under SJF but is just slightly less than for the case of $\rho = 0.9$. The performance of the largest jobs however is the same under all policies showing better approximation of SJF by SWAP for the largest jobs compared to the case for high load and large threshold shown in Figure 7.

We conclude therefore that the performance of SWAP in terms of jobs mean response time depends on the distribution of workloads. Similarly, its accuracy in approximating SJF also depends on the distribution of the workloads. In particular, SWAP performs well under both respects for short jobs that arrive to unscanned queue for heavy-tailed workloads. In contrast, SWAP performs poorly in terms of offering higher mean response time than SJF to large jobs just above the threshold value regardless the load and threshold value. SWAP scheduling is also inaccurate in approximating SJF for large jobs except for low load and shorter thresholds.

V. Conclusion

We modelled and evaluated the recently proposed SWAP scheduling policy under varying workload distributions. The numerical results that we obtained from the derived models show that SWAP can be used to approximate Shortest Job First (SJF) however it is more accurate for workloads with highly varying job sizes such heavy-tailed job size distributions. This is because heavy-tailed distributions exhibit expectation paradox that is a clear manifestation of temporal dependence, which is the basic assumption under which SWAP was proposed. The comparison of SWAP with FCFS and PS also show that SWAP is a more superior policy in terms of reducing the mean response time of short jobs. We further observed that in contrast to exponentially distributed workloads, especially at high threshold and load values, large jobs under SWAP suffer very negligible penalty.

In this paper, we presented numerical results of SWAP at unscanned state only due to space limitations. In the future, we will numerically investigate SWAP at scanned state as well. We will also explore the use of SWAP for networked environments with Internet flows as transferred entities. In contrast to jobs, flows don't arrive at a system all at once, making it very hard to immediately infer their sizes. We will check to see if per-connection buffer occupancy will approximate SWAP in such environments. Finally, we intend to validate the SWAP models derived in this paper using simulations of the policies.

Acknowledgement

This work was partially funded by CISCO University Research Fund, a corporate adviced fund of Silicon Valley Community Foundation.

References

[1] N. Bansal and M. Harchol-Balter. Analysis of SRPT Scheduling: Investigating Unfairness. In *Sigmetrics 2001 / Performance* 2001, pp. 279-290, June 2001.

[2] M. Harchol-Balter, M. Schroeder, B. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems (TOCS)* 21, 2 (May 2003).

[3] Leonard Kleinrock, *Queueing Systems*, Volume II. Computer Applications: John Wiley & Sons, 1976.

[4] D. Lu, P. Dinda, Y. Qiao, H. Sheng, F. Bustamante. Applications of SRPT Scheduling with Inaccurate Scheduling Information. In *Proceedings of IEEE MASCOTS*, October (2004).

[5] D. Lu, H. Sheng, and P. Dinda. Size-based scheduling policies with inaccurate scheduling information.In *Proceedings of IEEE MASCOTS* (2004).

[6] N. Mi, G. Casale, and E. Smirni. Scheduling for Performance and Availability in Systems with Temporal Dependent Workloads. In *The International Conference on Dependable Systems and Networks*, pp. 336-345, 2008.

[7] I. A. Rai, G. Urvoy-Keller, and E. Biersack. Analysis of LAS Scheduling for Job Size Distributions with High Variance. In *Proc. of ACM SIGMETRICS'03*, pp. 218-228, June 2003.

[8] I. A. Rai, G. Urvoy-Keller, M. K. Vernon and E. W. Biersack. Performance Analysis of LAS-based Scheduling Disciplines in a Packet Switched Network. In *Proc. ACM SIGMETRICS*, June 2004.

 ISBN:978-1-61208-113-7