Moment-Generating Algorithm for Response Time in Processor Sharing Queueing Systems

Tiberiu Chis^(✉) and Peter Harrison

Department of Computing, Imperial College London, SW7 2AZ, London, UK
{tiberiu.chis07,p.harrison}@imperial.ac.uk

Abstract. Response times are arguably the most representative and important metric for measuring the performance of modern computer systems. Further, service level agreements (SLAs), ranging from data centres to smartphone users, demand quick and, equally important, predictable response times. Hence, it is necessary to calculate moments, at least, and ideally response time distributions, which is not straightforward. A new moment-generating algorithm for calculating response times analytically is obtained, based on M/M/1 processor sharing (PS) queueing models. This algorithm is compared against existing work on response times in M/M/1-PS queues and extended to M/M/1 discriminatory PS queues. Two real-world case studies are evaluated.

1 Introduction

One could argue that performance is driving mobile [12,39,41] and cloud [1,2] technologies. For example, users wait, on average, just over nine seconds for a web page to load [27] before opting for more reliable performance from competitors. The same argument applies to delays in data centers [14,16] as part of quality of service (QoS) standards, which is incorporated, along with operational and energy costs [15,18], into service level agreements (SLAs). Whether it's using smartphones to download files using WiFi or streaming web content on the cloud, the delay principle still applies. With emerging technology companies selling increasingly more smartphones in 2015 – Xiaomi and Huawei are each aiming to sell 100 million handsets this year [47,48] – wireless communication via mobile devices will only intensify. Therefore, it is important to understand the effect of delay on asynchronous data transmission and how this impacts performance of millions of devices. From a queueing perspective, delay and response time (or latency, i.e. the time between a job arriving and leaving the system) are closely related. To meet QoS demands, application developers and content providers aim for quick response times to minimize performance bottlenecks. Modelling response times analytically requires a *fair* scheduling policy, such as processor-sharing (PS), which gives n incoming tasks an equal share of the processor (i.e. $1/n$ if service rate is 1). PS scheduling has relevant applications in web server designs and for bandwidth-sharing protocols in packet-switched networks [17,26]. PS queueing models provide an abstraction for such systems and allow analytical response time metrics to represent system delay. Minimising mean

response time alone is usually not acceptable nowadays because users tend to be equally frustrated with a highly variable service. They demand response time that is *predictable* [20, 23], which makes it important to calculate moments at least, and ideally response time distributions, which is not straightforward. In the past three decades, work has addressed response time in various ways using PS queues [19, 21, 22, 24, 36, 46]. In the present work, we introduce a novel moment-generating algorithm to calculate response times analytically. The algorithm is based on M/M/1-PS queues and offers the following contributions:

1. Iterative computation of moments, in terms of mean service rate (μ) and utilisation (ρ) of the system, using a partial differential equation for the Laplace transform of response time density.
2. Extension of the moment-generating algorithm to calculate response times for multiple job classes, which is automated for different job weights under discriminatory PS.
3. Applications include performance models dealing with smartphone data transfers, switching states for cloud servers given user demand, resource allocation for data centres, etc.

The rest of the paper is organised as follows: section 2 provides some background on queueing theory, PS scheduling and its applications and defines response times for different scheduling algorithms; in section 3, we describe related work on obtaining higher moments of response time in PS queueing systems; section 4 presents the moment-generating algorithm, which calculates response time in PS queueing systems analytically, with corresponding results under different scenarios; in section 5, we extend the moment-generating algorithm to support multiple job classes and analyse two real-world case studies in section 6; we conclude and offer extensions in section 7.

2 Background

In this section, we introduce key queueing concepts and justify the importance of queueing models with respect to diverse applications such as servers in smartphones, data centres and networks. Queueing models allow us to abstract the dynamic processes governing modern, complex computer systems and obtain representative performance measures (i.e. response times) with minimal computational cost. Fundamentally, scheduling is an integral part of queueing models for obtaining such measures.

2.1 Scheduling

There exist many scheduling disciplines for servers in queueing models. The most well-known is the first-come first-served (FCFS) discipline, which serves jobs in order of arrival and the job that waits the longest is served first. The best example of the FCFS discipline is in the first-in first-out (FIFO) queue when organising a data buffer. Other scheduling disciplines include last-come

first-served (LCFS), which selects the most recent job and serves it first. The most fundamental example of a data structure which implements LCFS is a stack. In terms of system utilisation (ρ), LCFS suffers from greater variability than FCFS as $\rho \rightarrow 1$ [7].

Organising servers under processor-sharing (PS) disciplines, such as egalitarian PS (EPS), allows for current jobs to be served at equal rates. Under EPS, if there are n jobs in a system with service rate 1, each job will be served at $1/n$ times the speed of the processor, which means there is no queueing and all jobs start immediately. One useful property of EPS is its *fairness*, where the expected response time of a job is directly proportional to its size. There are variants of PS such as discriminatory PS (DPS), where each job j in the system receives its own percentage of the server, therefore catering for multiple job classes. In DPS, a single processing system serving K job types is controlled by a vector of weights ($\alpha_j > 0, j = 1, \dots, K$). Further, assuming there are n_i class i jobs ($i = 1, \dots, K$) in the system, each class j job is served at rate:

$$r_j(n_1, \dots, n_K) = \frac{\alpha_j}{\sum_{i=1}^K \alpha_i n_i}, \quad j = 1, \dots, K \quad (1)$$

Note that when $\alpha_i = \alpha_j, i, j = 1, \dots, K$, DPS scheduling becomes EPS as each job request has equal weight. Round robin (RR) scheduling offers equal time slices for each job, assigned in circular order and without priorities. The EPS algorithm is seen as an idealisation of RR scheduling in time-shared computer systems [43]. Hereinafter, we use the terms EPS and PS interchangeably. The following section summarises PS applications for a range of computer systems.

2.2 PS Applications

Within queueing systems, the PS server discipline has been of considerable interest for several decades. PS is applied to modelling performance of bandwidth-sharing protocols in packet-switched networks [17, 26] and approximating the fair-queueing server disciplines used in communication network routers [45], where delays and congestion control are key measures. Further, PS has proved useful for modelling heavy-tailed service time distributions [31] and bulk arrivals [33]. Stochastic analysis of PS systems dealing with power management and energy consumption have also been of interest. More specifically, a queueing model with PS scheduling was employed when setting bounds on performance of dynamic speed scaling [29].

When predicting queueing delays, for example, the PS discipline is more complex to model than FCFS because the remaining response times in PS systems depend on future (i.e. uncertain) arrivals and dependent service requirements. Nonetheless, the simplicity of PS, coupled with fairness properties, has made it easily applicable to a variety of high-speed, computer systems that are abstracted by queueing systems.

Typically, modern servers are often difficult to replicate precisely in a numerically tractable way; to model such servers, PS scheduling is assumed for a number of reasons:

1. PS is popular for web server design [9] and evaluating flow-level performance of end-to-end flow control mechanisms like TCP [44].
2. Under PS, there is no queueing *per se* and arriving jobs start immediately to access server resources.
3. The implicit *fairness* means expected response time of a job is directly proportional to its size.
4. PS is effective for heavy-tailed service times, which may arise, for example, as short jobs are allowed to overtake long jobs. It also facilitates tractable asymptotic analysis of heavy-tailed service time distributions [31].

2.3 Queueing Models

The most fundamental queueing model is the M/M/1-FCFS queue, with Poisson mean arrival rate λ and exponential mean service time μ for one server with FCFS scheduling. Similarly, the M/M/1 queue under PS scheduling is written as M/M/1-PS using Kendall notation. Generalising such queues, the G/G/m queue offers generally distributed arrivals and service times for m parallel servers. Note that arrivals and service times may have specific distributions such as hyperexponential, phase-type, MAP-induced, etc.

One utilises underlying continuous time Markov chain (CTMC) properties of queueing models. Additionally, classes of product-form models exist, where state equilibrium probability is a scaled product of the marginal state probabilities of Markov processes that represent individual system components [30]. Therefore, queueing models approximate modern communication systems and their long-term behaviour, without the state explosion problem limiting modelling possibilities. Often, response time is a key measurement, which we define in subsequent sections, because it is useful for approximating performance and thus provides resource allocation on large-scale storage systems, mobile technology, wireless sensor networks (WSNs), etc. Typically, response times are obtained using aforementioned queues, given queueing theoretic assumptions.

2.4 Response Times

We refer to response time (or, sojourn time) as the time a customer spends in the system before completely departing from it. In queueing terms, response time T is the sum of the queueing time and the service time (i.e. duration of customer service). Let λ be the arrival rate, μ be the service rate, and $\rho = \lambda/\mu < 1$ be the equilibrium system utilisation. Of course, under FCFS queueing discipline, the response time probability density function is well known to be $f(t) = (\mu - \lambda)e^{-(\mu - \lambda)t}$, [8]. Under PS, the mean unconditional response time may be computed using Little's law. Let $L = \rho/(1 - \rho)$ be the mean number of jobs in the system and $\mathbb{E}[T]$ be the mean unconditional response time at equilibrium. Then, it follows from Little's law that $L = \lambda\mathbb{E}[T]$ and re-arranging for $\mathbb{E}[T]$ gives us:

$$\mathbb{E}[T] = \frac{L}{\lambda} = \frac{\rho}{\lambda(1 - \rho)} = \frac{\lambda/\mu}{\lambda(1 - \rho)} = \frac{1}{\mu(1 - \rho)} \quad (2)$$

When jobs require x units of service time, the mean conditional response time is given by $\mathbb{E}[T(x)] = x/(1-\rho)$. Therefore, $\mathbb{E}[T(x)]$ is linear in x , meaning that jobs with twice the size have double the response time, on average. Note that this fairness property only applies to means. As ρ approaches 1, the unconditional mean response time $\mathbb{E}[T]$ grows as $1/(1-\rho)$ and is independent of the variability of the service time distribution. Terms only affected by the mean of the service time distribution exhibit the *insensitivity* property [38].

Calculating higher moments of response time under PS scheduling requires, in general, an advanced understanding of layered branching of incoming jobs into the system [22]. Additionally, higher moments identify variability and skewness in time-series and approximate distributions, which may help to flatten heavy-tails, for instance. The next section describes existing methods in the literature for obtaining response time in PS queues.

3 Related Work on Response Times

There are a number of works on approximating response time under PS scheduling, but few which adopt analytical queueing theory, even for the Markovian M/M/1-PS queue. Some of the earliest significant work on PS queues is by Coffman *et al* in 1970 [4], which analysed waiting time means and variance of PS systems compared to FCFS. In 1980, Fayolle *et al* [37] summarised results of Kleinrock and Mitrani for DPS and also obtained average response time (both conditionally and unconditionally on job request sizes) in M/M/1-DPS queueing systems. Further, Laplace transforms provided average waiting time for multiple class types and asymptotic behaviour of service demand was also obtained, but no results on higher response time moments were given.

The abstraction of PS scheduling as a layered branching of incoming jobs into the system was first used by Yashkov in 1987 and led to a derivation of conditional response time moments a decade later [22, 24]. The k^{th} moment of response time of a job with service requirement x , $\mathbb{E}[T(x)^k]$, is given by:

$$\begin{aligned}\mathbb{E}[T(x)^k] &= - \sum_{i=1}^k \binom{k}{i} (-1)^i \mathbb{E}[T(x)^{k-i}] \alpha_i(x) \\ \alpha_k(x) &= \frac{k}{(1-\rho)^k} \int_{t=0}^x (x-t)^{k-1} F^{(k-1)*}(t) dt \\ F^{0*}(x) &= 1 \\ F^{n*}(x) &= \int_0^x F^{(n-1)*}(x-u) dF(u) \quad \text{for } n \geq 2 \\ F(x) &= \frac{1}{\beta_1} \int_0^x (1-B(u)) du\end{aligned}$$

where $B(\cdot)$ is a general service time distribution with finite mean $\beta_1 < \infty$.

In 2003, Masuyama *et al* obtained a complementary response time distribution [32]. Specifically, for an M/M/1-PS queue with arrival rate λ , service rate μ and job size x , the complement of response time distribution $\bar{T}(x) = 1 - T(x)$ is defined recursively as:

$$\bar{T}(x) = \sum_{n=0}^{\infty} (1 - \rho) \rho^n \sum_{k=0}^{\infty} \frac{(\lambda + \mu)^k x^k}{k!} e^{-(\lambda + \mu)x} h_{n,k} \quad (3)$$

where $h_{n,k+1} = \frac{n}{n+1} \frac{\mu}{\lambda + \mu} h_{n-1,k} + \frac{\lambda}{\lambda + \mu} h_{n+1,k}$, $h_{n,0} = 1$ and $h_{-1,k} = 0$.

This computationally intensive recursion is more costly than Yashkov's iterative solution, although storing previous terms in a buffer would speed up calculations. However, truncating multiple infinite sums is a serious disadvantage of Masuyama's method.

In 2004, Kim *et al* offered a joint transform to obtain response time moments for K job classes with different shares of service. An M/M/1-DPS queueing system is considered, where $\rho_i = \lambda_i / \mu_i$, for all jobs $i = 1, \dots, K$, subject to $\rho = \sum_{i=1}^K \rho_i < 1$. Let N_i be the number of jobs in the system at steady state and $Q(z_1, \dots, z_K) = \mathbb{E}[z_1^{N_1} \dots z_K^{N_K}]$ be the joint probability generating function for the numbers of jobs of each class in the queue at steady state. A job i with required service time greater than x is tagged such that when it attains service x , $S_i(x)$ and $N_{ij}(x)$ denote the elapsed response time and the number of class j jobs in the system, respectively ($j = 1, \dots, K$). The joint distribution of $S_i(x)$ and $N_{ij}(x)$ is then given by the transform $T_{ix}(s; z_1, \dots, z_K) = \mathbb{E}[e^{-sS_i(x)} z_1^{N_{i1}(x)} \dots z_K^{N_{iK}(x)}]$ for $|z_i| \leq 1$, $i = 1, \dots, K$, and $s \geq 0$.

The joint transform $T_{ix}(s; z_1, \dots, z_K)$ is governed by the following partial differential equation (PDE):

$$\begin{aligned} & \frac{\partial}{\partial x} T_{ix}(s; z_1, \dots, z_K) \\ &= - \sum_{j=1}^K \frac{\alpha_j}{\alpha_i} \left\{ \left(s + \sum_{k=1}^K \lambda_k (1 - z_k) \right) z_j - \mu_j (1 - z_j) \right\} \frac{\partial}{\partial z_j} T_{ix}(s; z_1, \dots, z_K) \\ & \quad - \left(s + \sum_{j=1}^K \lambda_j (1 - z_j) \right) T_{ix}(s; z_1, \dots, z_K) \end{aligned} \quad (4)$$

Deconditioning on x , which has exponential distribution with parameter μ_i requires only taking a Laplace transform.

We define $T_i(s; z_1, \dots, z_K) = \int_0^\infty \mu_i e^{-\mu_i x} T_{ix}(s; z_1, \dots, z_K) dx$ as the joint Laplace transform of the unconditional joint density of response time and probability generating function of class populations. It is easy to see this is given by the PDE, for $i = 1, \dots, K$,

$$\begin{aligned}
& -\mu_i Q(z_1, \dots, z_K) + \mu_i T_i(s; z_1, \dots, z_K) \\
& = -\sum_{j=1}^K \frac{\alpha_j}{\alpha_i} \left\{ \left(s + \sum_{k=1}^K \lambda_k (1 - z_k) \right) z_j - \mu_j (1 - z_j) \right\} \frac{\partial}{\partial z_j} T_i(s; z_1, \dots, z_K) \\
& \quad - \left(s + \sum_{j=1}^K \lambda_j (1 - z_j) \right) T_i(s; z_1, \dots, z_K)
\end{aligned} \tag{5}$$

Unconditional moments of response time are derived by differentiating equation (5). Kim *et al* solve $(K+1)(K+2)/2$ linearly independent equations to obtain unknown moments M_i^{jk} , $0 \leq j \leq k \leq K$, for each i , $i = 1, \dots, K$, which are defined as:

$$M_i^{jk} = \frac{\partial}{\partial z_j \partial z_k} T_i(s; z_1, \dots, z_K) \Big|_{s=0, z_1=\dots=z_K=1} \tag{6}$$

where z_0 is taken to be the Laplace-parameter s . We illustrate the calculation of such moments in the case of one class ($K = 1$) in Appendix 7. The next section introduces a novel moment-generating algorithm that can *iteratively* calculate arbitrary moments of response time, thus improving an aspect of the Kims' method in this respect.

4 Moment-Generating Algorithm

In a PS queue with utilisation ρ , the response time T of an arriving customer that requires x units of service time is known to have a probability density function that has Laplace transform:

$$W^*(s | x) = \frac{(1 - \rho)(1 - \rho r^2)e^{-[\rho\mu(1-r)+s]x}}{(1 - \rho r)^2 - \rho(1 - r)^2 e^{-[1/r - \rho r]\mu x}} \tag{7}$$

where r is the smaller root of the equation $\rho r^2 - (\rho + 1 + s/\mu)r + 1 = 0$. The result is long known, see for example [4, 7], and is derived by solving a partial differential equation for a certain generating function $G(z, s, x)$, viz.

$$(\mu z^2 - (\rho\mu + \mu + s)z + \rho\mu) \frac{\partial G}{\partial z} - \frac{\partial G}{\partial x} = (\rho\mu + s - \mu z)G \tag{8}$$

which yields $W^*(s | x) = (1 - \rho)G(\rho, s, x)$. We make the following observations:

1. The unconditional response time density for an arriving customer that has exponentially distributed service time requirement with mean $1/u$ is the product of u and the Laplace transform of $W^*(s|x)$ with respect to x , evaluated at Laplace-parameter u .
2. To calculate moments, the generating function's derivatives need only be computed at $s = 0$.

3. There is no need to solve the differential equation (8) for the generating function G since the moments are given by its derivatives evaluated at $s = 0$ and $z = \rho$, corresponding to the geometric equilibrium queue length probability distribution.
4. The Laplace transform of derivative $\partial G / \partial x$ yields the term $uG^{*x}(z, s, u) - G(z, s, 0)$, where G^{*x} denotes the Laplace transform of G with respect to x and the initial value $G(z, s, 0)$ is known to be $1/(1 - z)$.
5. At $s = 0$ and $z = \rho$, the coefficient of $\partial G / \partial x$ vanishes. Thus, by successive differentiation of the Laplace-transformed equation (8), we can determine the moments recursively.

In this way, we obtain the following unconditional moments for response time:

$$\mathbb{E}[T] = \frac{1}{\mu(1 - \rho)} \quad (9)$$

$$\mathbb{E}[T^2] = \frac{4}{\mu^2(2 - \rho)(1 - \rho)^2} \quad (10)$$

$$\mathbb{E}[T^3] = \frac{12(\rho + 2)}{\mu^3(2 - \rho)^2(1 - \rho)^3} \quad (11)$$

$$\mathbb{E}[T^4] = \frac{48(48 + 52\rho - 10\rho^2 - 6\rho^3 - 24\rho^4 + 9\rho^5)}{\mu^4(2 - \rho)^3(1 - \rho)^3(3 - 2\rho)(4 - 3\rho)} \quad (12)$$

In table 1, we summarise response time moments with fixed $\mu = 1$ whilst increasing ρ and also obtain moments with fixed $\rho = 0.5$ whilst increasing μ . After calculating response time moments analytically, approximating a full response time distribution is typically straightforward [11], for example via the general lambda distribution (GLD) [25, 34]. As such approximations are not the main scope of this paper, we guide the reader to relevant material [35, 40, 42] for more information. We extend the moment-generating algorithm to multiple job types in the next section.

Table 1. Moments for varying ρ with fixed $\mu = 1$ (left) and varying μ with fixed $\rho = 0.5$ (right).

Moment	$\rho = 0.2$	$\rho = 0.5$	$\rho = 0.8$	Moment	$\mu = 0.5$	$\mu = 2.5$	$\mu = 8.5$
$\mathbb{E}[T]$	1.25	2.0	5.0	$\mathbb{E}[T]$	4.0	0.8	0.235
$\mathbb{E}[T^2]$	3.47	10.67	83.33	$\mathbb{E}[T^2]$	42.67	1.71	0.147
$\mathbb{E}[T^3]$	15.91	106.7	2.9e3	$\mathbb{E}[T^3]$	853.3	6.82	0.174
$\mathbb{E}[T^4]$	105.3	1.6e3	1.1e5	$\mathbb{E}[T^4]$	2.5e4	40.5	0.303

5 Multi-Class Algorithm

We build an automated moment-generating algorithm for multiple job classes, which supports DPS scheduling for K job classes and incorporates service weights α_i for each job class $i, i = 1, \dots, K$. For simplicity of presentation, we use equal job weights (i.e. $\alpha_i = \alpha_j, i, j = 1, \dots, K$), but this is not a requirement of our method. Adapting a multi-class version of the PDE given in equation (8), we apply similar methods of successive differentiation to determine moments recursively. Assuming two job classes (i.e. $K = 2$), with mean arrival rates λ_1 and λ_2 , mean service rates μ_1 and μ_2 , and utilisation $\rho_1 = \lambda_1/\mu_1$ and $\rho_2 = \lambda_2/\mu_2$ such that $\rho_1 + \rho_2 < 1$, we obtain respective mean response times $\mathbb{E}[T_1]$ and $\mathbb{E}[T_2]$ as:

$$\mathbb{E}[T_1] = \frac{1}{\mu_1(1 - \rho_1 - \rho_2)}; \quad \mathbb{E}[T_2] = \frac{1}{\mu_2(1 - \rho_1 - \rho_2)} \quad (13)$$

Further, we derive second moments of response time $\mathbb{E}[T_1^2]$ and $\mathbb{E}[T_2^2]$ as:

$$\mathbb{E}[T_1^2] = \frac{4(\mu_1(1 + \rho_2) + \mu_2(1 - \rho_2))}{\mu_1^2(1 - \rho_1 - \rho_2)^2(\mu_1(2 - \rho_1) + \mu_2(2 - \rho_1 - 2\rho_2))} \quad (14)$$

$$\mathbb{E}[T_2^2] = \frac{4(\mu_1(1 - \rho_1) + \mu_2(1 + \rho_1))}{\mu_2^2(1 - \rho_1 - \rho_2)^2(\mu_1(2 - 2\rho_1 - \rho_2) + \mu_2(2 - \rho_2))} \quad (15)$$

```

K = 2;
L1sub = Solve[(L1 == λ1 / μ1 + Sum[αj (λj L1 + λj Lj) / (αj μj + α1 μ1), {j, 1, K}]) & /@
  Range[K], L1 & /@ Range[K]] // Simplify;
Do[M0,i = ((-L1 / λi) /. L1sub) // Simplify, {i, 1, K}];
e1 =
  ((μ1 + μ1 α1 / α1) M0,n,i - λ1 Sum[M0,j,i αj / αi, {j, 1, K}] + Sum[Mn,j,i αj / αi, {j, 1, K}] ==
    λ1 M0,i - (1 + α1 / α1) Mn,i) & /@ Range[K];
e2 = ((μ1 + 2 μ1 α1 / α1) Mn,i - 2 λ1 Sum[Mn,j,i αj / αi, {j, 1, K}] ==
  μ1 L1,n + 2 (1 + α1 / α1) λ1 Mn,i) & /@ Range[K];
e3 = Table[(μ1 + μk αk / α1 + μm αm / α1) Mn,k,i - λk Sum[Mn,j,i αj / αi, {j, 1, K}] -
  λm Sum[Mk,j,i αj / αi, {j, 1, K}] == μi Lm,k + λk (1 + αm / α1) Mn,i + λm (1 + αk / α1) Mk,i,
  {m, 1, K}, {k, 1, m - 1}] // Flatten;
eM3 = Join[e1, e2, e3] /. Mx_,y_,i /; x > y -> My,x,i /. Lx_,y_ /; x > y -> Ly,x;
eM = ((μ1 + μ1 α1 / α1) Mn,i == λ1 Sum[Mj,i αj / αi, {j, 1, K}] + μ1 L1 + λ1) & /@ Range[K];
M2sub = Solve[eM /. L1sub] // Flatten, Mn,i & /@ Range[K]] // First;
eL2 = Table[λj - Sum[αi (λj Lk,i + λk L1,j + λi Lj,k) / (αj μj + αk μk + αi μi), {i, 1, K}] ==
  (αj + αk) (λj Lk + λk Lj) / (αj μj + αk μk), {k, 1, K}, {j, 1, k}];
L2sub = Solve[eL2 /. L1sub /. Lx_,y_ /; x > y -> Ly,x] // Flatten,
  Table[L1sub, {b, 1, K}, {a, 1, b}] // Flatten] // First // Simplify;
M0,0,i_ := (-2 M0,i - 2 Sum[M0,j,i αj / αi, {j, 1, K}]) / μi;
M3sub = Solve[eM3 /. L2sub /. M2sub, Join[Table[M0,b,i, {b, 1, K}],
  (Table[M1,b,i, {b, 1, K}, {a, 1, b}] // Flatten)]] // First;
M0,1

```

Fig. 1. Mathematica code for K -class moments up to 2

These expressions were obtained by solving the moment equations obtained by repeatedly differentiating equation (5) up to two times. The algorithm to do this, written in Wolfram's Mathematica, is shown in figure 1. Obtaining the variance (i.e. $\sigma_i^2 = \mathbb{E}[T_i^2] - \mathbb{E}[T_i]^2$) of a class i job reveals the spread of the response time distribution from the mean. Further, calculating higher moments of response time is useful for predicting performance in a variety of multi-class applications where jobs have different priorities – or shares of a PS server. As with the second moment, higher moments are derived by differentiating equation (5) and defining the steady state generating function $Q(\cdot)$, which is straightforward to derive. This is the approach used in figure 1 for just two moments, but which is easy to extend to any higher moments. The difficulty that arises is the number of calculations needed, since every partial derivative up to p is required to calculate moment p – a rapidly increasing number, especially if there are many classes. A symbolic solution is surely intractable, but mathematical software could easily cope with a numerical solution when values are pre-set for the parameters of the model. Using such an automated multi-class algorithm, it is straightforward to estimate the probability distribution of response time for K job classes; good approximations can usually be found from the first four moments or so.

6 Case Studies

We obtained workload traces from two applications, which we abstract using M/M/1-PS queueing models, each with two job classes (i.e. $K = 2$). The first application is an HTC One (M7) smartphone transmitting data via 4G cell radio, where a time-stamped trace was recorded from a transmission period of 30 minutes. We summarise this HTC trace with the following mean service rates for each job class: $\mu_1 = 0.6$ and $\mu_2 = 2.4$. The second application is an Apache CloudStack VM executing programs on an Intel Core i7-2600 CPU @ 3.40GHz host machine. The CloudStack trace was recorded with mean service rates $\mu_1 = 1.4$ and $\mu_2 = 6.1$. Using equation (13), we plot mean response times (i.e. $\mathbb{E}[T_i]$, for $i = 1, 2$) in figure 2 with increasing system load (i.e. $\rho_1 + \rho_2$) for the HTC and CloudStack traces. Further, using equations (14) and (15), we plot variance (i.e.

Fig. 2. $\mathbb{E}[T_1]$ and $\mathbb{E}[T_2]$ for HTC (left) and CloudStack (right) traces under increasing load.

Fig. 3. σ_1^2 (left) and σ_2^2 (right) for the HTC trace under increasing load.

Fig. 4. σ_1^2 (left) and σ_2^2 (right) for the CloudStack trace under increasing load.

σ_i^2 , for $i = 1, 2$) of response time for increasing values of ρ_1 and ρ_2 in figures 3 and 4. Note that for the variance, the total system load (i.e. $\rho_1 + \rho_2$) does not exceed 1. For systems with more than two job classes, it is important to measure performance via response time moments for resource provisioning whilst considering different system load. Indeed, the moment-generating algorithm allows such measurements for any K job classes.

7 Conclusion and Future Work

We proposed an automated moment-generating algorithm for calculating response times analytically in M/M/1-PS queues in terms of mean service rate (μ) and utilisation (ρ) of the system. This incremental algorithm uses a partial differential equation for recursively evaluating terms in a Laplace transform and is extended for multiple job classes. Further, we examined two case studies, specifically workloads from a smartphone and a VM exhibiting two job classes each, and obtained response time means and variance for both workloads. Other possible applications include resource allocation in data centres, run-time analysis of multi-class workload in storage systems, and online server provisioning with switching states. Indeed, response times have become essential components of SLAs and thus support the long-term performance goals of many systems.

Extensions include generalising response time analysis for G/G/1-DPS queues and catering for bursty arrivals through an online MMPP or HMM used

for possible workload prediction. Further, incorporating energy cost into our performance models would match the SLA requirements more realistically. Indeed, battery models are popular in the literature [3, 5, 6, 10, 13, 28], but there is scarce analysis on power consumption related to performance via higher response time moments for multiple job classes.

References

1. Curtis, J.: 10 top cloud computing providers for 2014. <http://tinyurl.com/top-10-cloud-providers-2014>
2. Velazco, C.: Google gives students unlimited cloud storage. <http://www.engadget.com/2014/09/30/google-drive-for-education/>
3. Huria, T., Ceraolo, M., Gazzarri, J., Jackey, R.: High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells. In: Proc. IEEE IEVC, Greenville, pp. 1–8 (2012)
4. Coffman Jr., E.G., Muntz, R.R., Trotter, H.: Waiting Time Distributions for Processor-Sharing Systems. *Journal ACM* **17**, 123–130 (1970)
5. Prabhu, B.J., Chockalingam, A., Sharma, V.: Performance analysis of battery power management schemes in wireless mobile devices. In: Proc. IEEE WCNC, Orlando, vol. 2, pp. 825–831 (2002)
6. Open Battery. <http://www.doc.ic.ac.uk/~gljones/openbattery/index.php>
7. Harrison, P.G., Patel, N.M.: Performance Modelling of Communication Networks and Computer Architectures. Addison-Wesley (1993)
8. Stewart, W.J.: Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling, p. 409. Princeton University Press (2009)
9. Kjaer, M.A., Kihl, M., Robertsson, A.: Response-time control of a processor-sharing system using virtualised server environments. In: Proc. IFAC, Korea, vol. 17, p. 3612–3618 (2008)
10. Rohner, C., Feeney, L.M., Gunningberg, P.: Evaluating battery models in wireless sensor networks. In: Tsoussidis, V., Kassler, A.J., Koucheryavy, Y., Mellouk, A. (eds.) WWIC 2013. LNCS, vol. 7889, pp. 29–42. Springer, Heidelberg (2013)
11. Au-Yeung, S.W.M., Dingle, N.J., Knottenbelt, W.J.: Efficient approximation of response time densities and quantiles in stochastic models. In: Proc. ACM WOSP, Redwood Shores, vol. 4, pp. 151–155 (2004)
12. Shye, A., Scholbrock, B., Memik, G., Dinda, P.A.: Characterizing and Modeling User Activity on Smartphones, Technical Report, Northwest University (2010)
13. Rao, V., Singhal, G., Kumar, A., Navet, N.: Battery model for embedded systems. In: Proc. IEEE VLSID, Washington, DC, vol. 18, pp. 105–110 (2005)
14. Gao, P.X., Curtis, A.R., Wong, B., Keshav, S.: It's not easy being green. In: Proc. ACM SIGCOMM, Helsinki, vol. 44, pp. 211–222 (2012)
15. Wray, J.: Where's The Rub: Cloud Computing's Hidden Costs. <http://tinyurl.com/cloud-computing-hidden-costs>
16. Alawnah, R.Y., Ahmad, I., Alrashed, E.A.: Green and Fair Workload Distribution in Geographically Distributed Data. *Journal Green Eng.* **4**, 69–98 (2014)
17. Massoulié, L., Roberts, J.W.: Bandwidth sharing and admission control for elastic traffic. *Telecomm. Systems* **15**, 185–201 (2000)
18. AISO.net. <http://www.aiso.net/index.html>
19. Ott, T.J.: The Sojourn-Time Distribution in the M/G/1 Queue with Processor Sharing. *Journal of Applied Probability* **21**, 360–378 (1984)

20. Wierman, A.: Scheduling for Today's Computer Systems: Bridging Theory and Practice, PhD Thesis, School of Computer Science, Carnegie Mellon University (2007)
21. Kim, J., Kim, B.: Sojourn time distribution in the M/M/1 queue with discriminatory processor-sharing. *Performance Evaluation* **58**, 341–365 (2004)
22. Yashkov, S.F.: Processor-Sharing Queues: Some Progress In Analysis. *Queueing Systems* **2**, 1–17 (1987)
23. Wierman, A., Harchol-Balter, M.: Classifying scheduling policies with respect to higher moments of conditional response time. In: *Proc. ACM SIGMETRICS* (2005)
24. Zwart, A.P., Boxma, O.J.: Sojourn time asymptotics in the M/G/1 processor sharing queue. *Queueing Systems* **35**, 141–166 (2000)
25. Lakhany, A., Mausser, H.: Estimating the parameters of the General Lambda Distribution. *Algo. Research Quarterly* **3**, 47–58 (2000)
26. Roberts, J.W.: A survey on statistical bandwidth sharing. *Computer Networks* **45**, 319–332 (2004)
27. Lohr, S.: For Impatient Web Users, an Eye Blink Is Just Too Long to Wait. *New York Times*. <http://tinyurl.com/eye-blink-too-long-to-wait>
28. Jones, G.L., Harrison, P.G., Harder, U., Field, T.: Fluid queue models of battery life. In: *Proc. IEEE MASCOTS*, vol. 19, pp. 278–285 (2011)
29. Wierman, A., Andrew, L.L.H., Tang, A.: Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Performance Evaluation* **69**(12), 601–622 (2012)
30. Casale, G., Harrison, P.G.: AutoCAT: automated product-form solution of stochastic models. In: *Matrix-Analytic Methods in Stochastic Models*, vol. 27, pp. 57–85 (2013)
31. Queija, R.N.: Sojourn times in non-homogeneous QBD processes with processor sharing. *Stochastic Models* **17**, 61–92 (2001)
32. Masuyama, H., Takine, T.: Sojourn time distribution in a MAP/M/1 processor-sharing queue. *Op. Res. Letters* **31**, 406–412 (2003)
33. Bansal, N.: Analysis of the M/G/1 processor-sharing queue with bulk arrivals. *Op. Res. Letters* **31**, 401–405 (2003)
34. Lebrecht, A.: Queueing network models of Zoned RAID system performance, PhD Thesis, Department of Computing, Imperial College London (2009)
35. Ramberg, J., Schmeiser, B.: An approximate method for generating asymmetrics random variables. *Comm. ACM* **17**, 78–82 (1974)
36. Ward, A.R., Whitt, W.: Predicting reponse times in processor-sharing queues. In: *Proc. of Fields Institute Conference on Communication Networks* (2000)
37. Fayolle, G., Iasnogorodski, R., Mittrani, I.: Sharing a Processor Among Many Job Classes. *Journal ACM* **27**(3), 519–532 (1980)
38. Kelly, F.: *Stochastic Networks and Reversibility*, vol. 1. Wiley (1979)
39. Embedded Microprocessor Benchmark Consortium (EEMBC). <http://eembc.org/>
40. Ramberg, J., Dudewicz, E., Tadikamalla, P., Mykytka, E.: A probability distribution and its uses in fitting data. *Technometrics* **21**, 201–214 (1979)
41. AndEBench-Pro. http://eembc.org/andebench/index_pro.php
42. Freimer, M., Mudholkar, G., Kollia, G., Lin, C.: A study of the generalized Tukey Lambda family. *Comm. in Statistics* **17**, 3547–3567 (1988)
43. Aalto, S., Ayesta, U., Borst, S., Misra, V., Nunez-Queija, R.: Beyond Processor Sharing. *ACM SIGMETRICS Perform. Eval. Rev.* **34**, 36–43 (2007)
44. Kherani, A.A., Kumar, A.: On processor sharing as a model for TCP controlled HTTP-like transfers. In: *Proc. IEEE ICC, Paris*, vol. 4, pp. 2256–2260 (2004)

45. Dukkupati, N., Kobayashi, M., Zhang-Shen, R., McKeown, N.: Processor sharing flows in the internet. In: de Meer, H., Bhatti, N. (eds.) IWQoS 2005. LNCS, vol. 3552, pp. 271–285. Springer, Heidelberg (2005)
46. Harrison, P.G.: Response time distributions in queueing network models. In: Donatiello, L., Nelson, R. (eds.) SIGMETRICS 1993 and Performance 1993. LNCS, vol. 729, pp. 147–164. Springer, Heidelberg (1993)
47. Huilgol, M.: Xiaomi aims to sell 100 million smartphones in 2015. <http://tinyurl.com/xiaomi-100-million-smartphones>
48. Moore, M.: Huawei Looks To Shift 100 Million Smartphones in 2015. <http://tinyurl.com/huawei-100-million-smartphones>

Appendix: The Kims' Method of Response Time Moments for $K = 1$

Conditional and unconditional joint transforms of response time are given in equations (4) and (5), respectively. This allows calculation of conditional and unconditional moments of response time [21], where there are K job classes. For the $K = 1$ case, let us assume the following conditions for the M/M/1-PS queueing model:

1. The mean arrival rate is λ and the mean service rate is μ .
2. Utilisation is $\rho = \lambda/\mu < 1$.
3. $z_1 = z/\rho$, where z is the parameter from equation (8).

Let $Q(z_1) = \mathbb{E}[z_1^{N_1}]$ be the probability generating function in the system at steady state for one job type, where N_1 is the number of jobs in the system at steady state. Note that $z_1 = z/\rho$ is the difference of deconditioning on ρ in Kim's method. Further, Kim *et al* tag a job with required service time greater than x ; when the tagged job attains service x , let $S_1(x)$ and $N_1(x)$ denote the elapsed response time and the number of jobs in the system, respectively. Then, Kim *et al* use a joint transform to derive a relation on the joint distribution of $S_1(x)$ and $N_1(x)$:

$$T_x(s; z_1) = \mathbb{E}[e^{-sS_1(x)} z_1^{N_1(x)}]$$

which is defined for $|z_1| \leq 1$ and $s \geq 0$.

The proof of this relation is given in [21] and is omitted here. For the $K = 1$ case, we evaluate the expression for the PDE given in equation (4), such that we obtain equation (8) as follows:

$$\begin{aligned} & \frac{\partial}{\partial x} T_x(s; z_1) = \\ & -\frac{\alpha_1}{\alpha_1} \left((s + \lambda(1 - z_1))z_1 - \mu(1 - z_1) \right) \frac{\partial}{\partial z_1} T_x(s; z_1) - (s + \lambda(1 - z_1))T_x(s; z_1) \end{aligned}$$

Simplifying terms gives us

$$\frac{\partial}{\partial x} T_x(s; z_1) = - \left(sz_1 + \lambda z_1 - \lambda z_1^2 - \mu + \mu z_1 \right) \frac{\partial}{\partial z_1} T_x(s; z_1) - (s + \lambda - \lambda z_1)T_x(s; z_1)$$

Substituting z/ρ for z_1 , we have

$$\frac{\partial}{\partial x} T_x(s; z_1) = -\rho \left(s \frac{z}{\rho} + \lambda \frac{z}{\rho} - \lambda \frac{z^2}{\rho^2} - \mu + \mu \frac{z}{\rho} \right) \frac{\partial}{\partial z} T_x(s; z_1) - (s + \lambda - \lambda \frac{z}{\rho}) T_x(s; z_1)$$

Simplifying terms further and using relation $\rho = \lambda/\mu$ gives us

$$\frac{\partial}{\partial x} T_x(s; z_1) = \left(-sz - \lambda z + \lambda \frac{z^2}{\rho} + \rho\mu - \mu z \right) \frac{\partial}{\partial z} T_x(s; z_1) - (s + \lambda - \lambda \frac{z}{\rho}) T_x(s; z_1)$$

$$\frac{\partial}{\partial x} T_x(s; z_1) = \left(-sz - \rho\mu z + \mu z^2 + \rho\mu - \mu z \right) \frac{\partial}{\partial z} T_x(s; z_1) - (s + \rho\mu - \mu z) T_x(s; z_1)$$

Replacing G for $T_x(s; z_1)$ and rearranging terms gives us equation (8) as follows:

$$\left(\mu z^2 - (\rho\mu + \mu + s)z + \rho\mu \right) \frac{\partial G}{\partial z} - \frac{\partial G}{\partial x} = (\rho\mu + s - \mu z)G$$

Obtaining unconditional moments of response time uses repeated differentiation of the PDE given in equation (5), where we use the joint transform $T(s; z_1)$ for the $K = 1$ case. To obtain the first moment of response time T (i.e. $\mathbb{E}[T]$), we use equation (2) and Little's law. The second moment requires derivation of $(K+1)(K+2)/2$ linearly independent equations with unknown moments $L^j, M^0, M^j, M^{00}, M^{0j}, j = 1, \dots, K$, and $L^{jk}, M^{jk}, 1 \leq j \leq k \leq K$. For $K = 1$, the moments are defined as follows:

$$\begin{aligned} L^1 &= \left. \frac{\partial}{\partial z_1} Q(z_1) \right|_{z_1=1}, \quad M^0 = \left. \frac{\partial}{\partial s} T(s; z_1) \right|_{s=0, z_1=1}, \quad M^1 = \left. \frac{\partial}{\partial z_1} T(s; z_1) \right|_{s=0, z_1=1}, \\ M^{00} &= \left. \frac{\partial^2}{\partial s^2} T(s; z_1) \right|_{s=0, z_1=1}, \quad M^{01} = \left. \frac{\partial^2}{\partial s \partial z_1} T(s; z_1) \right|_{s=0, z_1=1}, \\ L^{11} &= \left. \frac{\partial^2}{\partial z_1^2} Q(z_1) \right|_{z_1=1}, \quad M^{11} = \left. \frac{\partial^2}{\partial z_1^2} T(s; z_1) \right|_{s=0, z_1=1} \end{aligned} \tag{16}$$

Evaluating derivatives for these moments gives us

$$\begin{aligned} L^1 &= \mathbb{E}[N_1 z_1^{(N_1-1)}] \Big|_{z_1=1}, \\ M^0 &= \mathbb{E}[-S_1 e^{-sS_1} z_1^{N_1}] \Big|_{s=0, z_1=1}, \quad M^1 = \mathbb{E}[e^{-sS_1} N_1 z_1^{(N_1-1)}] \Big|_{s=0, z_1=1}, \\ M^{00} &= \mathbb{E}[S_1^2 e^{-sS_1} z_1^{N_1}] \Big|_{s=0, z_1=1}, \quad M^{01} = \mathbb{E}[-S_1 e^{-sS_1} N_1 z_1^{(N_1-1)}] \Big|_{s=0, z_1=1}, \\ L^{11} &= \mathbb{E}[N_1(N_1-1) z_1^{(N_1-2)}] \Big|_{z_1=1}, \quad M^{11} = \mathbb{E}[e^{-sS_1} N_1(N_1-1) z_1^{(N_1-2)}] \Big|_{s=0, z_1=1} \end{aligned} \tag{17}$$

Substituting values for s and z_1 , we have

$$\begin{aligned} L^1 &= \mathbb{E}[N_1], \quad M^0 = \mathbb{E}[-S_1], \quad M^1 = \mathbb{E}[N_1], \quad M^{00} = \mathbb{E}[S_1^2], \quad M^{01} = \mathbb{E}[-S_1 N_1], \\ L^{11} &= \mathbb{E}[N_1(N_1 - 1)], \quad M^{11} = \mathbb{E}[N_1(N_1 - 1)] \end{aligned} \quad (18)$$

Note that $L^1 = M^1$ and $L^{11} = M^{11}$ such that these terms are used interchangeably hereinafter. Further, it is known that $\mathbb{E}[N_1] = \rho/(1 - \rho)$ and $\mathbb{E}[-S_1] = -1/\mu(1 - \rho)$. In the $K = 1$ case, taking partial derivatives of equation (5) gives us three linearly independent equations from which we obtain the moments. The first equation is obtained by taking partial derivatives twice in equation (5) with respect to s and evaluating at $s = 0, z_1 = 1$:

$$\mu M^{00} + 2M^{01} = -2M^0 \quad (19)$$

Then, we take partial derivatives of equation (5) with respect to s and z_1 and evaluate at $s = 0, z_1 = 1$:

$$(2\mu - \lambda)M^{01} + M^{11} = \lambda M^0 - 2M^1 \quad (20)$$

Again, we take partial derivatives twice in equation (5), but this time with respect to z_1 and evaluate at $s = 0, z_1 = 1$:

$$(\mu - \lambda)M^{11} = 2\lambda M^1 \quad (21)$$

Solving equations (19), (20) and (21), we obtain the following values for the moments:

$$M^{00} = \frac{4}{\mu^2(2 - \rho)(1 - \rho)^2}; \quad M^{01} = \frac{-\lambda(3 - \rho)}{\mu^2(2 - \rho)(1 - \rho)^2}; \quad M^{11} = \frac{2\rho^2}{(1 - \rho)^2} \quad (22)$$

Therefore, we verify that values for M^{00} from equation (22) and $\mathbb{E}[T^2]$ from equation (10) are indeed the same for the $K = 1$ case. Extending analysis to the third moment is computationally more complex and Kim *et al* do not provide explicit values for M^{000} as we do for $\mathbb{E}[T^3]$ in equation 11. Hence, this is an advantage of the moment-generating algorithm proposed in our work over existing work.