Scheduling Multi-Server Systems Using Foreground-Background Processing

Rong Wu

Department of Computing and Software
McMaster University
1280 Main Street West, Hamilton, ON L8S 4L7, Canada
wur2@mcmaster.ca

Douglas G. Down

Department of Computing and Software
McMaster University
1280 Main Street West, Hamilton, ON L8S 4L7, Canada
downd@mcmaster.ca

Abstract

It is known that foreground-background processor sharing (FBPS) stochastically minimizes the number in the system with a single server when task processing times follow a Decreasing Failure Rate(DFR) distribution. Based on this, we derive an optimal policy for a system with a common queue for several identical servers. The performance of such a system will provide a lower bound for the loosely coupled servers case — a system with several identical servers, where a routing decision must be made immediately on a task's arrival. When system load and task variance are high, we propose maximum-least-elapsed-time (MLET) or join-the-shortest-queue (JSQ) for routing followed by foreground-background processor sharing scheduling at each server. Simulation results show good performance of the proposed policies.

1 Introduction

Consider scheduling tasks for a system in which the processing times of incoming tasks are unknown, such as dynamic requests for a web server system or static requests for files with fixed sizes, but where the processing times are unknown (Lu, Sheng, and Dinda [4] have a discussion of how file size and processing time may exhibit very little correlation). Righter and Shanthikumar [6] show that for a single server system, the foreground-background processor sharing (FBPS) discipline (give priority to tasks with the least attained processing time) stochastically minimizes the number of tasks in the system for decreasing failure rate (DFR) processing time distributions, while the first-come-first-served (FCFS) discipline stochastically minimizes the number of tasks in the system for increasing failure rate (IFR) processing time distributions. As a reminder, DFR (IFR) is defined as follows: Let the random variable X have distribution function F and density function f. A distribution is said to be DFR(IFR) if $f(x)/(1 - F(x))$ is decreasing (increasing).

More optimality properties of the FBPS discipline are presented in Feng and Misra [1], Nuijens [5], and Righter with her co-authors [7]. Recent research suggests that heavy-tailed distributions (most are DFR) appear to be a feature in many aspects of computing systems (see

Harchol-Balter with her co-authors [2]). This arises from a mixture of many small tasks and a small number of large tasks, but with the total processing times for large tasks being a significant proportion of the overall load. The focus of this paper is on choosing scheduling policies which perform well for multi-server systems with DFR processing time distributions. As a starting point, we use the fact that FBPS is optimal for a single server system under the assumption that the processing time distribution is DFR.

In Section 2, we look at the tightly coupled server system, i.e. one in which there is a single queue for all of the (identical) servers. We describe a policy which stochastically maximizes the number of completed tasks for a discrete time queue length process. From this, we present the policy c-FBPS for the continuous time process. The performance of such a tightly coupled server system will be a lower bound on a system with several identical servers in parallel, where a routing decision must be made immediately upon a task's arrival. In Section 3, we present a discussion on what is precisely required for the DFR assumption. In particular, the proof that FBPS is optimal requires the processing time distribution to be DFR on $[0, \infty)$. In Section 4, we propose several scheduling policies for the loosely coupled server system, i.e. one in which arriving tasks are immediately assigned to one of several identical servers in parallel. In particular, we assume that the processing times follow a DFR distribution on $[0, \infty)$. Section 5 gives some simulation results for the proposed policies where processing times follow a Shifted Pareto distribution (DFR on $[0, \infty)$). Section 6 provides some concluding remarks.

2 Tightly Coupled Server Systems

We consider a system where there is a common queue for c identical servers. We make no assumptions on the arrival process and we assume that the task processing times are unknown and follow a DFR distribution. We first give the description of a policy we call D-c-FBPS and show that it stochastically maximizes the number of completed tasks for a sampled version of the system (with the sampling rate chosen sufficiently fast). By letting the sampling rate tend to infinity, we develop c-FBPS for a continuous version of the system. Suppose we only make scheduling decisions at times $n\Delta t$, $n = 0, 1, 2, ...$, where $\Delta t > 0$ is the sampling time. Define

$$\mu_i(s_i) = \frac{f_i(s_i)\Delta t}{1 - F_i(s_i)},$$

where s_i is the elapsed processing time of task i. The value $\mu_i(s_i)$ is the probability that task i will finish in a time Δt. Note that the distribution itself may depend on the task. A direct consequence of the DFR assumption is that $\mu_i(\cdot)$ is a decreasing function. For a tightly coupled server system, the policy D-c-FBPS gives priority to tasks with the first c highest values of $\mu_i(s_i)$ at a particular scheduling epoch (ties may be broken arbitrarily).

Theorem 2.1 *Consider the problem with periodic scheduling epochs parameterized by sampling time Δt. For a tightly coupled server system with DFR processing time distribution, D-c-FBPS stochastically maximizes the number of completed tasks for suitably small Δt.*

Proof. The proof for Theorem 2.1 is similar to the proof of the $c\mu$ rule in [6] and as a result, our notation will be chosen to that in [6]. If the number of tasks in the system is less than c, all of the tasks will get service. So, without loss of generality, we consider the situation that the number of tasks to be scheduled at an epoch is greater than c. Let N be the number of tasks that have been completed, we want to show that D-c-FBPS maximizes $P\{N \geq k\}$ for all k.

The result trivially holds for $T = 0$. We proceed by induction, assuming the result holds for $T = n\Delta t$ and considering the time horizon $T' = T + \Delta t$.

Suppose that for the problem with time horizon T', π is an optimal policy that does not follow D-c-FBPS for the first Δt time units. In other words, at time 0, task j is processed under π, but there exists a task i such that $\mu_i(s_i) > \mu_j(s_j)$. Note that π and D-c-FBPS are equivalent after time Δt. If τ is the first time that π schedules task i, then as the DFR assumption yields $\mu_j(s_j + \Delta t) \leq \mu_j(s_j) < \mu_i(s_i)$, and as π and D-c-FBPS are equivalent from Δt to T', j will not be scheduled again before time τ.

Suppose π' is constructed from π by scheduling task i at time 0 and task j at time τ. Also, let N' be the number of tasks that are correctly completed during the first Δt time units and $\hat{N}$ be the number of tasks that are correctly completed after time Δt. Then

$$\begin{aligned} P_\pi\{N \geq k\} = {} & P_\pi\{N \geq k | T' \geq \tau\} \cdot P_\pi\{T' \geq \tau\} \\ & + P_\pi\{\hat{N} \geq k, T' < \tau\} \\ & + \sum_{h=1}^{c} P_\pi\{\hat{N} = k - h, T' < \tau\} \cdot P_\pi\{N' = h\}. \end{aligned}$$

The order of i and j is irrelevant if $T' \geq \tau$, so $P_\pi\{N \geq k | T' \geq \tau\}$=$P_{\pi'}\{N \geq k | T' \geq \tau\}$. If $T' < \tau$, $\hat{N}$ under π or π' is the same, so $P_\pi\{\hat{N} \geq k, T' < \tau\} = P_{\pi'}\{\hat{N} \geq k, T' < \tau\}$ and $P_\pi\{\hat{N} = k - h, T' < \tau\} = P_{\pi'}\{\hat{N} = k - h, T' < \tau\}$, where $h = 1, 2, ..., c$.

$$\begin{aligned} & P_{\pi'}\{N' = h\} - P_\pi\{N' = h\} \\ = {} & (\mu_i(s_i) - \mu_j(s_j)) \\ & \cdot \sum_{r_1,...,r_{h-1} \neq i,j} \left(\prod_{d=1}^{h-1} \mu_{r_d}(s_{r_d}) \cdot \prod_{l \neq r_1,...,r_{h-1},\, l \neq j} (1 - \mu_l(s_l)) \right) \\ & + \sum_{r_1,...,r_h \neq j} \left(\prod_{d=1}^{h} \mu_{r_d}(s_{r_d}) \cdot \prod_{l \neq r_1,...,r_h,\, l \neq i,j} (1 - \mu_l(s_l)) \right) \\ = {} & (\mu_i(s_i) - \mu_j(s_j)) \cdot \left(\sum_{r_1,...,r_{h-1} \neq i,j} \left(\prod_{d=1}^{h-1} \mu_{r_d}(s_{r_d}) - k_1 o(\Delta t^h) \right) \right), \end{aligned} \tag{1}$$

where $h = 1, 2, ..., c$, $r_1, ..., r_h, m, l$ represent tasks that are in service during the first Δt time units, and k_1 is a positive constant. As we have assumed that $\mu_i(s_i) - \mu_j(s_j) > 0$, we can always take Δt small enough such that the right hand side of equation (1) is greater than 0 for each value of h. Then

$$P_\pi\{\hat{N} = k - h, T' < \tau\} \cdot P_\pi\{N' = h\} - P_{\pi'}\{\hat{N} = k - 1, T' < \tau\} \cdot P_{\pi'}\{N' = 1\} > 0,$$

for each $h = 1, 2, ..., c$. Therefore,

$$\begin{aligned} & P_{\pi'}\{N \geq k\} - P_\pi\{N \geq k\} \\ & = \sum_{h=1}^{c} (P_{\pi'}\{N' = h\} - P_\pi\{N' = h\}) \cdot P_\pi\{\hat{N} = k - h, T' < \tau\} > 0, \end{aligned} \tag{2}$$

and (2) implies that π cannot be optimal. ∎

If we let $\Delta t \to 0$, we can create a continuous version of D-c-FBPS, which we will call c-FBPS.

The policy c-FBPS is described as follows. Let SE_i represent the set of tasks whose elapsed time equals task i's elapsed time in the system, n_i represent the number of elements in SE_i, and m_i represent the number of tasks whose elapsed time is less than task i's elapsed time in the system. For a tightly coupled system with c servers, c-FBPS gives service to task i if $m_i < c$. Further, if $n_i > c - m_i$, all the tasks in SE_i processor share amongst $c - m_i$ servers, otherwise, each task in SE_i is assigned a single server. Note that this policy may be impossible to implement due to the feature of multiple processor pooling. In fact, the discrete time version may be a more natural version of the problem in the multiple server setting. In [6], this type of argument is used to show that FBPS is optimal. They begin with a discrete time process and suggest that if one takes a continuous version of the policy, a simple limiting argument provides optimality. The work in [5] suggests that while true, the limiting argument is somewhat more subtle. It appears we can use the same approach as in [5] to get $P_{\pi'}\{N \geq k\} - P_{\pi}\{N \geq k\} > 0$ for a continuous version of the problem operating under c-FBPS.

There is another subtlety in the argument here (and in [6] on which it is based). We require $\mu_j(s_j + \Delta t) \leq \mu_j(s_j) < \mu_i(s_i)$ on $[0, \infty)$, not just over the range of the processing time distributions. We discuss this in more detail in the next section.

3 DFR Distribution Range

FBPS is optimal for a DFR distribution with range starting from 0. If the range does not start from 0, it may not be optimal. Suppose all the tasks in the system follow a DFR distribution on $[b, \infty)$, where $b > 0$. The elapsed time of each task always starts from 0, which is not in the range of the processing time distribution $[b, \infty)$. In particular, this implies that a task's failure rate is 0 if its elapsed time is in $[0, b)$. When a new task arrives, if the system has other tasks whose elapsed times are greater than b, then these tasks' failure rates are greater than the newly arrived task. FBPS gives priority to the new arriving task (least elapsed time) instead of tasks with the highest failure rates in this case. Hence the proof for optimality of FBPS cannot work. We can also illustrate this by an example. Consider a system where task processing times follow a DFR distribution on $[100, \infty)$, *e.g.* a Pareto distribution $P[b, \infty)$ with density function $f(x) = \alpha b^{\alpha}/x^{(\alpha+1)}$, where $\alpha = 1.5$ and $b = 100$. The expected processing time for a task in such a system is 300. Eighty percent of the arrivals are tasks whose processing times are less than 300. Consider a system which has only one task with elapsed time 99 when a new task arrives. FBPS gives priority to the new arrival whose expected remaining processing time is 300. But the expected remaining processing time for the older task is 201 (by $\int_0^{\infty} uf(x+u)du/(1 - \int_b^x f(v)dv)$, where $x = 99$), which is much smaller than 300. It therefore seems to be advantageous to at least provide some service to the old task. It seems in this case that the optimal policy may be very complicated. This is not to say that FBPS might not perform well in this case, just that it is not optimal.

One can create a distribution that is DFR on $[0, \infty)$ from one which is DFR on its range by performing a simple shift. For example, for a Pareto distribution $P[b, \alpha)$, we can shift the range to get a DFR distribution on $[0, \infty)$. Let X be a random variable with a Pareto distribution $P(b, \alpha)$ and f be its density function. Let Y be a random variable with $Y = X - b$, then Y has a Shifted Pareto distribution $SP[b, \alpha)$ with density $f(y) = \alpha b^{\alpha}/(y + b)^{\alpha+1}$, where $y \geq 0$. The resulting Shifted Pareto distribution with density $f(y)$ is DFR on $[0, \infty)$. In section 5, we use a Shifted Pareto distribution to perform simulations.

4 Loosely Coupled Server Systems

We have characterized the optimal policy for tightly coupled servers. The case of loosely coupled server systems is also of interest. In such a system, each server has its own queue and a routing decision is made immediately at the time of a task's arrival. We assume the routing decision takes zero time.

The scheduling policy for a loosely coupled server system has two components: the *dispatch* policy, which assigns tasks to servers, and the particular *service* policy employed at each of the servers to which the tasks are assigned. Examples of *dispatch* policies are Random (a task is assigned to a particular server with probability $1/c$), Round-Robin (RR, the ith arriving task is assigned to the $(i \mod c)$th server), and Join the Shortest Queue (JSQ, each task is assigned to the server with the least number of waiting tasks). Note that Random and Round-Robin do not need any information for routing. JSQ requires the queue length information at each server. Examples of service policies are First Come First Served (FCFS), in which tasks are served in the order that they arrive, Shortest Remaining Processing Time (SRPT), where priority is given to the task with the least processing time remaining, Processor Sharing (PS), where all tasks get an equal portion of CPU time, and FBPS.

In [5], it is shown that FCFS performs poorly for heavy-tailed processing time distributions since the number of tasks in queue is likely to grow very large once a large task is in service. The mean queue length of FCFS is known to be directly proportional to the task size variance (*e.g.* Pollaczek-Khinchin formula). In fact, the mean queue length may not exist if the second moment of the processing time does not exist. On the other hand, FBPS and PS (Processor Sharing) perform well since large tasks cannot monopolize the server if there are short tasks waiting. The proof of optimality for FBPS in [6] does not have any assumption on the arrival process. Therefore, for any fixed routing policy, choosing FBPS as the service policy at each server achieves the best performance. The choice of dispatching policy depends on many factors. If communication overhead is a big issue, we suggest Round-Robin since Round-Robin does not require any information to make routing decisions. If higher quality of performance is needed, we propose to use Maximum Least Elapsed Time (MLET) or JSQ as the dispatching policy. Let ls_i represent the least task elapsed time at queue i. If all the servers are busy, MLET assigns incoming tasks to server k where $k = \arg\max_{1 \le i \le c} ls_i$. If there is an idle server, MLET assigns incoming tasks to one of them. Both JSQ and MLET have communication overhead. The intuition behind MLET is to try to approximate c-FBPS as much as possible. The routing decision is based on looking at a list of the $c + 1$ tasks (the c tasks in service and the new arrival) and putting the c tasks with the least elapsed processing time into service. Note that MLET may not match c-FBPS exactly since it only considers those tasks with least elapsed time at each server. Better performance might possibly be achieved by a more complicated policy which considers all the elapsed times at each server. Our intuition to choose JSQ is that tasks in a short queue have likely received more service than tasks in a long queue. In this way, the shortest queue is more likely to have the task with the maximum least elapsed service, so intuitively JSQ should operate in a similar manner to MLET (and in turn, c-FBPS). In the next section, we give some simulation results that suggest that this intuition is indeed the case.

The fact that JSQ is a reasonable routing policy is appealing. It would imply that in the case where processing times are unknown, as long as one gets the scheduling policy correct (FCFS for IFR distributions, FBPS for DFR distributions), the routing policy may be independent of the underlying assumptions. It also suggests that the insight in Whitt [8] is that JSQ is not optimal because the scheduling policy was not chosen well.

5 Simulation Results

Tables 2-4 give 90% confidence intervals for the expected queue length for several policies under varying numbers of servers and system loads. These policies are represented in a format A-B, where A denotes the dispatching policy and B denotes the service policy used at each server. The simulations use a Shifted Pareto distribution for the processing times, where $b = 512$ and $\alpha = 1.2$. The expected processing time is 2453. Table 1 gives the parameters used in the simulation.

Number of Servers (c)	ρ	$1/\lambda$
4	0.7	876.07
4	0.85	721.47
4	0.95	645.53
8	0.7	438.04
8	0.85	360.74
8	0.95	322.76
16	0.7	219.02
16	0.85	180.37
16	0.95	161.38

Table 1: Parameters Used in Simulation

Tasks arrive to the system according to a Poisson process with rate λ, where λ is varied to set the load on the system. We ran 30 replications for each policy, with each replication consisting of 1.0×10^6 arrivals.

Policy	$\rho = 0.7$	$\rho = 0.85$	$\rho = 0.95$
RR-FBPS	(1.173, 1.224)	(1.747, 1.834)	(2.408, 2.533)
JSQ-FBPS	(0.929, 0.969)	(1.202, 1.271)	(1.525, 1.617)
MLET-FBPS	(0.819, 0.934)	(1.230, 1.294)	(1.694, 1.793)
RR-PS	(1.703, 1.916)	(3.179, 3.606)	(5.285, 5.984)
JSQ-PS	(1.010, 1.112)	(1.522, 1.689)	(2.239, 2.568)
MLET-PS	(1.012, 1.123)	(1.583, 1.736)	(2.536, 2.883)
c-FBPS	(0.803, 0.854)	(1.042, 1.091)	(1.273, 1.374)

Table 2: 90% CI of Expected Queue Length with $c = 4$

The simulation results show that the greater the number of servers, the better the system performance. For any fixed service policy, comparing with other dispatching policies, Round-Robin is less sensitive to the number of servers. The performance of MLET-FBPS and JSQ-FBPS are close to each other which matches our intuition to choose either MLET or JSQ as a dispatching policy. They both outperform RR-FBPS. The difference of the expected queue length between the proposed policies (MLET-FBPS, JSQ-FBPS) and optimal policy c-FBPS increases as the system load increases. If the system load is not high, RR-FBPS is a good choice. The results also show that for any fixed dispatching policy, the performance when FBPS is the service policy is better than using PS. Of course, this holds as FBPS is optimal, but the results quantify the difference. When the system load increases, the gap between FBPS related policies and PS related policies (here the dispatching policy is fixed) increases. Note that if the distribution is

Policy	$\rho = 0.7$	$\rho = 0.85$	$\rho = 0.95$
RR-FBPS	(1.125, 1.169)	(1.676, 1.767)	(2.248, 2.362)
JSQ-FBPS	(0.792, 0.817)	(1.036, 1.084)	(1.238, 1.305)
MLET-FBPS	(0.778, 0.801)	(0.996, 1.033)	(1.279, 1.355)
RR-PS	(1.621, 1.731)	(2.966, 3.225)	(4.794, 5.240)
JSQ-PS	(0.821, 0.846)	(1.133, 1.211)	(1.561, 1.725)
MLET-PS	(0.820, 0.843)	(1.174, 1.249)	(1.738, 2.006)
c-FBPS	(0.679, 0.701)	(0.844, 0.882)	(1.043, 1.085)

Table 3: 90% CI of Expected Queue Length with $c = 8$

Policy	$\rho = 0.7$	$\rho = 0.85$	$\rho = 0.95$
RR-FBPS	(1.125, 1.169)	(1.676, 1.767)	(2.248, 2.362)
JSQ-FBPS	(0.760, 0.770)	(0.898, 0.928)	(1.059, 1.115)
MLET-FBPS	(0.748, 0.754)	(0.865, 0.886)	(1.032, 1.064)
RR-PS	(1.561, 1.636)	(2.763, 2.915)	(4.332, 4.725)
JSQ-PS	(0.774, 0.791)	(0.924, 0.962)	(1.219, 1.301)
MLET-PS	(0.758, 0.769)	(0.944, 0.998)	(1.212, 1.311)
c-FBPS	(0.643, 0.663)	(0.799, 0.824)	(0.922, 0.952)

Table 4: 90% CI of Expected Queue Length with $c = 16$

close to but not exactly DFR on $[0, \infty)$, (***e.g.*** Pareto distribution), the FBPS related policies still work well on such a distribution. We also did simulations based on a Pareto distribution with the same α and b, the results are close to the Shifted Pareto distribution case.

6 Conclusion and Future Work

We discuss scheduling policies for distributed systems when task sizes are unknown and task processing times follow a DFR distribution. All the results for loosely coupled system are based on simulation. It would be of interest to study whether the proposed policies approach optimality in any (asymptotic) sense. Liu and Towsley in [3] show that Round-Robin minimizes response times and queue lengths among the policies without using any server and task information if processing times follow an IFR distribution (here FCFS is the service policy). Based on this, we conjecture that RR-FBPS is also the optimal policy under DFR distributions where no server information is provided for routing. We are currently examining this issue. It would also be of interest to develop limiting approximations for FBPS in the single server case, which could in turn aid in examining the multiple server setting.

References

[1] Hanhua Feng and Vishal Misra. "Mixed Scheduling Disciplines for Network Flows". In *Proceedings of The Fifth Workshop on Mathematical Performance Modeling and Analysis (MAMA 2003)*, San Diego, California, USA, 2003.

[2] Mor Harchol-Balter, Mark E. Crovella, and Cristina D. Murta. "On Choosing a Task Assignment Policy for a Distributed Server System". *Journal of Parallel and Distributed Computing*, 59(2):204–228, November 1999.

[3] Zhen Liu and Don Towsley. "Optimality of the Round Robin Routing Policy". *Journal of Applied Probability*, 31:466–478, 1994.

[4] Dong Lu, Huanyuan Sheng, and Peter A. Dinda. "Effects and Implications of File Size/Service Time Correlation on Web server scheduling policy". Technical Report NWU-CS-04-33, Northwestern University, 2004.

[5] Misja Nuijens. *The Foreground-Background Queue*. PhD thesis, University of Amsterdam, Simon Stevinstraat 46-II, 1097 CA Amsterdam, 2004.

[6] Rhonda Righter and J.George Shanthikumar. "Scheduling Multiclass Single Server Queueing Systems to Stochastically Maximize the Number of Successful Departures". *Probability in the Engineering and Informational Sciences*, 3:323–333, 1989.

[7] Rhonda Righter, J.George Shanthikumar, and Genji Yamazaki. "On Extremal Service Disciplines in Single-stage Queueing Systems". *Journal of Applied Probability*, 27:409–416, 1990.

[8] Ward Whitt. "Deciding Which Queue to Join: Some Counterexamples". *Operations Research*, 34(1):226–244, January 1986.