Performance analysis of work stealing strategies in large scale multi-threaded computing

GRZEGORZ KIELANSKI and BENNY VAN HOUDT, University of Antwerp, Belgium

Distributed systems use randomized work stealing to improve performance and resource utilization. In most prior analytical studies of randomized work stealing, jobs are considered to be sequential and are executed as a whole on a single server. In this paper we consider a homogeneous system of servers where parent jobs spawn child jobs that can feasibly be executed in parallel. When an idle server probes a busy server in an attempt to steal work, it may either steal a parent job or multiple child jobs.

To approximate the performance of this system we introduce a Quasi-Birth-Death Markov chain and express the performance measures of interest via its unique steady state. We perform simulation experiments that suggest that the approximation error tends to zero as the number of servers in the system becomes large. To further support this observation we introduce a mean field model and show that its unique fixed point corresponds to the steady state of the QBD. Using numerical experiments we compare the performance of various simple stealing strategies as well as optimized strategies.

CCS Concepts: • **Mathematics of computing** → **Queueing theory**; **Markov processes**; • **Networks** → **Network performance analysis**.

Additional Key Words and Phrases: matrix analytic methods, distributed computing

ACM Reference Format:

Grzegorz Kielanski and Benny Van Houdt. 2023. Performance analysis of work stealing strategies in large scale multi-threaded computing. 1, 1 (February 2023), 23 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Jobs in multithreaded computing systems consist of several threads [4, 26]. Upon starting the execution a main thread (which we call a parent job) several other threads are spawned (which we call child jobs). These spawned child jobs are initially stored locally, but can be redistributed at a later stage. One way of redistributing jobs is called "randomized work stealing": processors that become empty start probing other processors at random (uniformly) and if the probed processor has pending jobs, some of its jobs are transferred to the probing processor [4, 6]. Another option is to make use of "randomized work sharing", where servers that have pending jobs probe others to offload some of their work to other servers.

Work stealing solutions have been studied by various authors and are often used in practice. They have been implemented for example in Cilk programming language [3, 7], Intel TBB [21], Java fork/join framework [14], KAAPI [10] and .NET Task Parallel Library [15]. Some early studies on work sharing and stealing include [6, 18, 24]. In [6] the performance of work stealing and sharing is compared for homogenous systems with exponential job sizes. Using similar techniques the work in [6] was generalized to heterogeneous systems in [18]. The key takeaway from these papers is that work

Authors' address: Grzegorz Kielanski, Grzegorz.Kielanski@uantwerpen.be; Benny Van Houdt, benny.vanhoudt@uantwerpen.be, University of Antwerp, Middelheimlaan 1, Antwerp, Belgium, B-2020.

Manuscript submitted to ACM

stealing clearly outperforms work sharing in systems with a high load.

More recent work includes [9, 16, 17, 23, 25]. In [9] the authors analyse the system consisting of several homogeneous clusters with exponential job sizes and where half of the jobs are transferred when a probe is successful. A fair comparison between stealing and sharing strategies is given for homogeneous networks and exponential job sizes in [16, 17] and for non-exponential job sizes in [25]. Further, the comparison in [17] is extended to heterogeneous networks in [23].

The current paper is closely related to [11, 22]. Here and in [11, 22] we consider a system of homogeneous servers that uses a randomized work stealing policy. Firstly, in [22] we compared two systems: one system where parent jobs can be stolen and the other system where child jobs can be stolen one at a time. The latter of the two studied systems was novel in the sense that all previous research about work stealing and sharing focused on systems where jobs are considered to be sequential and are always executed as a whole on a single server. The key takeaway from [22] is that if probe rate r is large enough, then the second system outperforms the first. This is to be expected: for large probe rates a job gets redistributed quickly and more queues can work on it, thus lowering the mean response time. On the other hand, for small probe rates, it is better to transfer parents to empty queues as a larger amount of work is transferred per steal.

Next, in [11], presented at QEST 2021, we considered a set of policies where if a server with pending child jobs is probed by an idle server, some of its child jobs are transferred. When a server is probed that does not have pending child jobs, a pending parent job is transferred instead (if available). The major complication in the analysis of these policies, compared to [22], is that when several child jobs get stolen at once, child jobs may be transferred several times before being executed. The objective of [11] was to gain insights on how to determine the number of child jobs that should be stolen at once. We concluded that the stealing policy where half of the child jobs gets stolen every time is in general a good stealing policy for large probe rates, while stealing all children performs best when the probe rate is low. We also noted that stealing a single child usually performs the worst.

In [11, 22] we assumed that parent and child jobs have exponentially distributed service requirements, while jobs in a real system are typically more variable in size. In this paper, which is an extended version of [11], we relax this assumption by generalizing the analysis from [11] to phase-type (PH) distributed parent and child job sizes. As any positive valued distribution can be approximated arbitrarily close by a phase-type distribution [2, Section 3.2.1], this relaxation is significant, without complicating the analysis too much. We show that the insights obtained in [11] still hold for PH child and parent jobs. Compared to [11] we also present a mean field model and prove that it has a unique fixed point that coincides with the steady state of a structured Markov chain (a similar result was presented in [22], where the proof is considerably easier due to the more restricted setting).

The rest of this paper is organized as follows. In Section 2 we describe the system of N queues. The subsequent sections contain the main contributions of the paper, namely:

- To approximate the performance of the work stealing system of N queues, we introduce in Section 3 a Quasi-Birth-Death Markov chain (QBD for short) that the describes the evolution of a single server queueing system with negative arrivals. We prove that this QBD has a unique stationary distribution, which can be quickly calculated. In Section 4, we indicate how to compute the waiting time distribution and mean service time.
- We compare the performance of several stealing strategies in Section 5. We confirm the main insights gained from [11], namely that the strategy of stealing half of the child jobs performs well for low loads and/or high probe rates and that stealing all child jobs performs best when the load is high and/or the probe rate is low. We

further conclude that stealing becomes more worthwhile as the job sizes become more variable (that is, with increased squared coefficient of variation).

- For some strategies we present simulation results in Section 6 that suggest that as the number of servers becomes large, the approximation error of the QBD model tends to zero.
- In Section 7, we introduce a mean field model and prove that it has a unique fixed point which is given by the stationary distribution of the QBD.

We finish the paper with Section 8, where we present some concluding remarks.

2 SYSTEM DESCRIPTION AND STRATEGIES

We consider a system with N homogeneous servers each with an infinite buffer to store jobs. Parent jobs arrive in each server according to a local Poisson arrival process with rate λ. Upon entering service a parent job spawns $i \in \{0, 1, \ldots, m\}, m \geq 1$, child jobs, the number of which follows a general distribution with finite support p_i (i.e., $p_i \geq 0$ for every i and $\sum_{i=0}^{m} p_i = 1$). These child jobs are stored locally and have priority over any parent jobs (either already present or yet to arrive), while the spawning parent job continues service. Thus, when a (parent or child) job completes service the server first checks to see whether it has any waiting child jobs, if so it starts service on a child job. If there are no child jobs present, service on a waiting parent job starts (if any are present). We assume that parent and child jobs have phase type (PH) distributed service requirements with representations (α^p, S^p) and (α^c, S^c) respectively, where S^p is an $n_p \times n_p$ matrix and S^c an $n_c \times n_c$ matrix. This means that the probability that the service requirement of a child job exceeds t is given by $\alpha^c e^{S^c t} 1_{n_c}$, where 1_k denotes a column vector of ones of height k, and the same holds for the parent jobs if we replace the superscript c by p. PH distributions are distributions with a modulating finite state Markov chain (see also [13]). Moreover there are various fitting tools available for PH distributions (see e.g. [12, 20]).

When a server is idle, it probes other servers at random at rate $r > 0$, where r is a system parameter. Note that r determines the amount of communication between the servers and increasing r should improve performance at the expense of a higher communication overhead. When a server is probed (by an idle server) and it has waiting (parent or child) jobs, we state that the probe is successful. When a successful probe reaches a server without waiting child jobs, a parent job is transferred to the idle server. Note that such a transferred parent job starts service and spawns its child jobs at the new server.

When a successful probe reaches a server with pending/waiting child jobs, several child jobs can be transferred at once. If the probed server is serving a parent job and there are i child jobs in the buffer of the probed server, $j \leq i$ child jobs are stolen with probability $\phi_{i,j}$ (i.e., for every i we have $\sum_{j=1}^{i} \phi_{i,j} = 1$). On the other hand if a child job is being processed by the probed server and there are i child jobs waiting in the buffer of the probed server, $j \leq i$ child jobs are stolen with probability $\psi_{i,j}$ (i.e., for every i we have $\sum_{j=1}^{i} \psi_{i,j} = 1$). For ease of notation we set $\phi_{i,j} = \psi_{i,j} = 0$ if $j > i$. Probes and job transfers are assumed to be instantaneous.

The main objective of this paper is to study how the probabilities $\phi_{i,j}$ and $\psi_{i,j}$ influence the response time of a job, where the response time is defined as the time between the arrival of a parent job and the completion of the parent and all its spawned child jobs. Given the above description, it is clear that we get a Markov process if we keep track of the number of parent and child job and the phase of the job in service in each of the N servers. This Markov process however does not appear to have a product form, making its analysis prohibitive.

Instead we use an approximation method, the accuracy of which is investigated in Section 6. The idea of the approximation exists in focusing on a single server and assuming that the queue lengths at any other server are

independent and identically distributed as in this particular server. Within the context of load balancing, this approach is known as the cavity method [5]. In fact all the analytical models used in [6, 9, 16–18, 22–25] can be regarded as cavity method approximations. A common feature of such an approximation is that it tends to become more accurate as the number of servers tends to infinity, as we demonstrate in Section 6 for our model. The cavity method typically involves iterating the so-called cavity map [5]. However, in our case the need for such an iteration is avoided by deriving expressions for the rates at which child and parent jobs are stolen.

3 QUASI-BIRTH-DEATH MARKOV CHAIN

As the system of N queues uses work stealing, where (parts of) a job can be transferred upon a successful steal, these N queues are coupled in a non-trivial way. However, due to homogeneity of the processors we are still able to analyze the system of N queues by approximating it by a single queue with negative arrivals. This queue can described by a Quasi-Birth-Death (QBD) Markov chain, which we introduce in this section.

Let $\lambda_p(r)$ denote the rate at which parent jobs are stolen when the server is idle. Let $\lambda_{c,1}(r), \ldots, \lambda_{c,m}(r)$ denote respectively the rates at which $1, \ldots, m$ child jobs are stolen. We provide formulas for these rates further on. The evolution of a single server has the following characteristics, where the negative arrivals correspond to steal events:

(1) When the server is busy, arrivals of parent jobs occur according to a Poisson process with rate λ. When the server is idle, parent jobs arrive at the rate $\lambda + \lambda_p(r)$, while a batch of i child jobs arrives at rate $\lambda_{c,i}(r)$ for $i = 1, \ldots, m$.
(2) Upon entering service, a parent job spawns $i \in \{0, 1, \ldots, m\}$, $m \geq 1$, child jobs with probability p_i. Child jobs are stored locally.
(3) Child jobs have priority over any parent jobs *waiting* in the queue and are thus executed immediately after their parent job when executed on the same server.
(4) Parent and child jobs have PH distributed service requirements with n_p and n_c phases and with representations (α^p, S^p) and (α^c, S^c), respectively. We assume that these representations have $\alpha^c 1_{n_c} = 1$ and $\alpha^p 1_{n_p} = 1$. We denote $s^p = -S^p 1_{n_p}$ and $s^c = -S^c 1_{n_c}$.
(5) If there are parent jobs and no child jobs waiting in the buffer of the server then a negative parent arrival occurs at the rate rq, where $q = 1 - \rho$ is the probability that a queue is idle (where ρ is defined in (1)).
(6) If a parent job is in service and there are $i \in \{1, \ldots, m\}$ child jobs in the buffer of the server, a batch of j negative child job arrivals occurs at the rate $rq\phi_{i,j}$, for all $j \in \{1, \ldots, i\}$.
(7) If a child job is in service and there are $i \in \{1, \ldots, m-1\}$ child jobs pending in the buffer of the server, a batch of j negative child job arrivals occurs at the rate $rq\psi_{i,j}$, for all $j \in \{1, \ldots, i\}$.

Note that the load of the system can be expressed as

$$\rho = \lambda \left(\alpha^p (-S^p)^{-1} 1_{n_p} + \alpha^c (-S^c)^{-1} 1_{n_c} \sum_{n=1}^{m} n p_n \right), \tag{1}$$

where $\alpha^p (-S^p)^{-1} 1_{n_p}$ and $\alpha^c (-S^c)^{-1} 1_{n_c}$ is the mean parent and child job size, respectively. Denote by $X \geq 0$ the number of parent jobs waiting, by $Y \in \{0, 1, \ldots, m\}$ the number of child jobs in the server (either in service or waiting), by $Z \in \{0, 1\}$ whether a parent job is currently in service ($Z = 1$) or not ($Z = 0$) and by W the phase of the job in service. Note that we have $W \in \{1, \ldots, n_p\}$ when $Z = 1$ and $W \in \{0, \ldots, n_c\}$ when $Z = 0$, where $W = 0$ if the queue is idle. The possible transitions of the QBD Markov chain are listed in Table 1, corresponding to: 1. a batch of j child jobs arriving

Table 1. Transitions for the QBD in Section 3

	From	To	Rate	For
1.	$(0,0,0,0)$	$(0,j,0,k)$	$\lambda_{c,j}(r)\alpha^c_k$	$j=1,\ldots,m,\ k=1,\ldots,n_c$
2.	$(0,0,0,0)$	$(0,j,1,k)$	$(\lambda+\lambda_p(r))p_j\alpha^p_k$	$j=0,1,\ldots,m,\ k=1,\ldots,n_p,$
3.	(X,Y,Z,W)	$(X+1,Y,Z,W)$	λ	$X+Y+Z\geq 1,\ W\geq 1$
4.	$(X,Y,1,k)$	$(X,Y,0,\ell)$	$s^p_k\alpha^c_\ell$	$X\geq 0, Y\geq 1, k=1,\ldots,n_p,\ \ell=1,\ldots,n_c,$
5.	$(X,Y,0,k)$	$(X,Y-1,0,\ell)$	$s^c_k\alpha^c_\ell$	$X\geq 0, Y\geq 2, k,\ell=1,\ldots,n_c,$
6.	$(0,0,1,k)$	$(0,0,0,0)$	s^p_k	$k=1,\ldots,n_p,$
7.	$(0,1,0,k)$	$(0,0,0,0)$	s^c_k	$k=1,\ldots,n_c,$
8.	$(X,0,1,k)$	$(X-1,j,1,\ell)$	$s^p_k p_j\alpha^p_\ell$	$X\geq 1, j=0,1,\ldots,m,k,\ell=1,\ldots,n_p,$
9.	$(X,1,0,k)$	$(X-1,j,1,\ell)$	$s^c_k p_j\alpha^p_\ell$	$X\geq 1, j=1,\ldots,m,k=1,\ldots,n_c,\ell=1,\ldots,n_p,$
10.	$(X,Y,1,k)$	$(X,Y,1,\ell)$	$S^p_{k,\ell}$	$X,Y\geq 0,k,\ell=1,\ldots,n_p,$ with $k\neq\ell,$
11.	$(X,Y,0,k)$	$(X,Y,0,\ell)$	$S^c_{k,\ell}$	$X\geq 0, Y\geq 1, k,\ell=1,\ldots,n_c,$ with $k\neq\ell,$
12.	(X,Y,Z,W)	$(X-1,Y,Z,W)$	rq	$X,W\geq 1, Y+Z=1,$
13.	$(X,Y,1,W)$	$(X,Y-j,1,W)$	$rq\phi_{Y,j}$	$X\geq 0, Y\geq j, j=1,\ldots,m, W=1,\ldots,n_p,$
14.	$(X,Y,0,W)$	$(X,Y-j,0,W)$	$rq\psi_{Y-1,j}$	$X\geq 0, Y\geq j+1, j=1,\ldots,m-1, W=1,\ldots,n_c.$

at an idle queue and the first child job proceeding directly into service, 2. a parent job arriving at an idle queue and proceeding directly into service, spawning j child jobs, 3. a parent arriving to a non-idle queue, 4. completion of a parent in service, succeeded by a child job, 5. child service completion, succeeded by another child job, 6. completion of a parent in service, not succeeded by any job, 7. child service completion, not succeeded by another job, 8. parent service completion, succeeded by a parent job that enters service and spawns j child jobs, 9. child service completion, succeeded by a parent job that enters service and spawns j child jobs, 10. a phase change occurs in the service of a parent, 11. a phase change occurs in the service of a child,12. negative parent job arrival, 13. a parent is in service and a batch of negative child job arrivals occurs, 14. a child job is in service and a batch of negative child job arrivals occurs.

The four dimensional process $\{X_t(r), Y_t(r), Z_t(r), W_t(r) : t \geq 0\}$ is an irreducible, aperiodic Quasi-Birth-Death process, where the *level* $\ell = *$ when the chain is in state $(0,0,0,0)$ and equals $\ell \geq 0$ when the chain is in a state with $X = \ell$ (different from $(0,0,0,0)$). When the level $\ell \geq 0$, the *phase* of the QBD is three dimensional and given by (Y,Z,W). The $mn_c + (m+1)n_p$ phases of level $\ell \geq 0$ are ordered such that the j-th phase corresponds to $(Y,Z,W) = (\lceil j/n_c\rceil, 0, (j-1) \mod n_c{+}1)$, for $j = 1,\ldots,mn_c$ and phase $mn_c{+}j$ to $(Y,Z,W) = (\lceil j/n_p\rceil{-}1, 1, (j{-}1) \mod n_p{+}1)$ for $j = 1,\ldots,(m{+}1)n_p$.

As explained below, the generator of the process is

$$Q(r) = \begin{bmatrix} -\lambda_0(r) & \sum_{j=1}^m \lambda_{c,j}(r)\kappa_j + (\lambda+\lambda_p(r))\alpha & & & \\ \mu & B_0(r) & A_1 & & \\ & A_{-1}(r) & A_0(r) & A_1 & \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

with $\lambda_0(r) = \sum_{j=1}^m \lambda_{c,j}(r) + \lambda + \lambda_p(r)$. The row vector κ_j is defined as $\kappa_j = \left[0'_{(j-1)n_c}\ \alpha^c\ 0'_{(m-j)n_c+(m+1)n_p}\right]$, where 0_i is a column vector of zeroes of length i. The initial probability vector α records the distribution of child jobs upon a parent job entering service and the initial phase of that parent. It is given by $\alpha = \left[0'_{mn_c} \quad p_0\alpha^p \quad p_1\alpha^p \quad \ldots \quad p_m\alpha^p\right]$. Indeed, at rate $\lambda_{c,j}(r)\alpha^c_k$ a batch of j child jobs arrives in an idle server, causing a jump to level 1 and phase $(j,0,k)$,

while at rate $(\lambda+\lambda_p(r))\alpha_k^p$ a parent job arrives that spawns j child jobs with probability p_j causing a jump to phase $(j,1,k)$ of level 1.

For further use, define

$$S(r)=\begin{bmatrix} S_{00}(r) & 0 \\ S_{10} & S_{11}(r)\end{bmatrix},$$

where $S_{00}(r)$ is an $mn_c\times mn_c$ matrix and $S_{11}(r)$ is an $(m+1)n_p\times(m+1)n_p$ matrix,

$$S_{00}(r)=rq\begin{bmatrix} & & \\ \psi_{1,1} & & \\ \vdots & \ddots & \\ \psi_{m-1,m-1} & \cdots & \psi_{m-1,1}\end{bmatrix}\otimes I_{n_c}+\begin{bmatrix} S^c & & & \\ s^c\alpha^c & \ddots & & \\ & \ddots & \ddots & \\ & & s^c\alpha^c & S^c\end{bmatrix},$$

$$S_{10}=\begin{bmatrix} 0_{n_p} & \cdots & \\ s^p\alpha^c & & \\ & s^p\alpha^c & \\ & & \ddots\end{bmatrix},\ S_{11}(r)=rq\begin{bmatrix} & & \\ \phi_{1,1} & & \\ \vdots & \ddots & \\ \phi_{m,m} & \cdots & \phi_{m,1}\end{bmatrix}\otimes I_{n_p}+\begin{bmatrix} S^p & & & \\ & \ddots & & \\ & & \ddots & \\ & & & S^p\end{bmatrix},$$

where $\otimes$ denotes the Kronecker product and I_k denotes the identity matrix of size $k\times k$. The matrix $A_0(r)$ contains the possible transitions for which the level $\ell>0$ remains unchanged, this is when child jobs are stolen, or when a waiting child moves into service, or when phase of the job in service changes. Hence

$$A_0(r)=S(r)-\lambda I-rqI.$$

Here, $I=I_{mn_c+(m+1)n_p}$. Whenever it is clear what dimensions an identity matrix should have, we simply write I for the identity matrix of the appropriate size. Note that even when there are no child jobs waiting, the rate rq appears on the main diagonal of $A_0(r)$ due to the negative parent arrivals. When $\ell=0$ there are no parent jobs waiting and therefore the negative parent arrivals that occur in phases $(1,0,k)$ and $(m+1,1,k')$, for $k=1,\ldots,n_c$ and $k'=1,\ldots,n_p$, have no impact. This implies that

$$B_0(r)=A_0(r)+rqV_0=S(r)-\lambda I-rq(I-V_0),$$

where $V_0=\operatorname{diag}\left(\begin{bmatrix}1'_{n_c} & 0'_{(m-1)n_c} & 1'_{n_p} & 0'_{mn_p}\end{bmatrix}\right)$.The level ℓ can only decrease by one due to a service completion from a phase with no pending child jobs, that is, from phases $(1,0,k)$ and $(m+1,1,k')$, for $k=1,\ldots,n_c$ and $k'=1,\ldots,n_p$. To capture these events define $\mu=\begin{bmatrix}(s^c)' & 0'_{(m-1)n_c} & (s^p)' & 0'_{mn_p}\end{bmatrix}'$. The level can also decrease due to a negative parent arrival when $\ell>0$. The matrix $A_{-1}(r)$ records the transitions for which the level decreases and therefore equals

$$A_{-1}(r)=\mu\alpha+rqV_0.$$

Finally, parent job arrivals always increase the level by one:

$$A_1=\lambda I.$$

Denote by $A(r)=A_{-1}(r)+A_0(r)+A_1$, the generator of the phase process, then

$$A(r)=S(r)+\mu\alpha-rq(I-V_0).$$

Define

$$\pi_*(r) = \lim_{t\to\infty} P[X_t(r) = 0, Y_t(r) = 0, Z_t(r) = 0, W_t(r) = 0],$$

and for $\ell \geq 0$,

$$\pi_\ell(r) = (\pi_{\ell,1,0}(r), \ldots \pi_{\ell,m,0}(r), \pi_{\ell,0,1}(r), \ldots, \pi_{\ell,m,1}(r)),$$

where

$$\pi_{\ell,j,0}(r) = (\pi_{\ell,j,0,1}(r), \ldots, \pi_{\ell,j,0,n_c}(r))$$
$$\pi_{\ell,j,1}(r) = (\pi_{\ell,j,1,1}(r), \ldots, \pi_{\ell,j,1,n_p}(r))$$

and where

$$\pi_{\ell,j,k,w}(r) = \lim_{t\to\infty} P[X_t(r) = \ell, Y_t(r) = j, Z_t(r) = k, W_t(r) = w].$$

Due to the QBD structure [19], we have

$$\pi_0(r) = \pi_*(r)R_0(r), \tag{2}$$

where $R_0(r)$ is a row vector of size $mn_c + (m+1)n_p$ and for $\ell \geq 1$,

$$\pi_\ell(r) = \pi_0(r)R(r)^\ell, \tag{3}$$

where $R(r)$ is a $(mn_c + (m+1)n_p) \times (mn_c + (m+1)n_p)$ matrix and by [13] the smallest nonnegative solution to

$$A_1 + R(r)A_0(r) + R(r)^2A_{-1}(r) = 0.$$

Also, due to the balance equations with $\ell = 0$, we have

$$\sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + (\lambda + \lambda_p(r))\alpha + R_0(r)B_0(r) + R_0(r)R(r)A_{-1}(r) = 0$$

and due to [13, Chapter 6]

$$A_1G(r) = R(r)A_{-1}(r),$$

where $G(r)$ is the smallest nonnegative solution to

$$A_{-1}(r) + A_0(r)G(r) + A_1G(r)^2 = 0.$$

Combining the above yields the following expression:

$$R_0(r) = -\Big(\sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + (\lambda + \lambda_p(r))\alpha\Big)(B_0(r) + \lambda IG(r))^{-1}, \tag{4}$$

where $B_0(r) + \lambda IG(r)$ is a subgenerator matrix and is therefore invertible. We note that $R(r)$ and $G(r)$ are independent of $\lambda_{c,1}(r), \ldots, \lambda_{c,m}(r)$ and $\lambda_p(r)$ and can be computed easily using the toolbox presented in [1]. To fully characterize the QBD in terms of $\lambda, \alpha^c, S^c, \alpha^p, S^p$ and the probabilities $p_i, \phi_{i,j}$ and $\psi_{i,j}$, we need to specify $\lambda_{c,1}(r), \ldots, \lambda_{c,m}(r)$ and $\lambda_p(r)$.

To determine these rates we use the following observation: $q = 1 - \rho$ should be the probability that the QBD is in state $(0, 0, 0, 0)$ and in this state batches of j child jobs arrive at rate $\lambda_{c,j}(r)$. Therefore $q\lambda_{c,j}(r)$ should equal the parent arrival rate λ times the expected number of times that a batch of j child jobs is stolen per parent job. The main difficulty in using this equality lies in the fact that we must also take into account that a child job can be stolen several times before it is executed.

To this end and as a preparation for Proposition 3.1, we define recursively the row vector $p_{i,j}(r)$ such that the k-th entry of $p_{i,j}(r)$ is the probability that the phase (i, j, k) is visited by the queue during the service of a job just after an arrival, a steal or a completion.

By conditioning on whether we first have a service completion or steal event, we have

$$p_{1,m}(r) = p_m \alpha^p$$
$$p_{1,i}(r) = p_i \alpha^p + rq \sum_{j>i} \phi_{j,j-i} p_{1,j}(r)(rqI - S^p)^{-1},$$

for $i \in \{0, \ldots, m-1\}$. For $i \in \{1, \ldots, m\}$, with $p_{0,m+1} = 0$, we further have

$$p_{0,i}(r) = p_{1,i}(r)(rqI - S^p)^{-1} s^p \alpha^c + p_{0,i+1}(r)(rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{j>i} \psi_{j-1,j-i} p_{0,j}(r)(rqI - S^c)^{-1}.$$

Note that

$$p_{1,0}(r)1_{n_p} + p_{0,1}(r)1_{n_c} = 1, \tag{5}$$

as every queue eventually visits phase $(0, 1, k)$ or $(1, 0, k)$ for some k just after a completion or a steal.

We also define the row vector $p_i^j(r)$ recursively, where k-th entry is the probability that the queue visits phase $(0, i, k)$ just after a completion or a steal has occurred, given that the queue started with j child jobs. We have

$$p_j^j(r) = \alpha^c,$$
$$p_i^j(r) = p_{i+1}^j(r)(rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{k=i+1}^{j} \psi_{k-1,k-i} p_k^j(r)(rqI - S^c)^{-1},$$

for $i \in \{1, \ldots, j-1\}$. Note that we have $p_1^j(r)1_{n_c} = 1$, for $1 \le j \le m$, as the QBD eventually visits phase $(0, 1, k)$ for some k just after a completion or a steal. We are now in a position to define $\lambda_{c,i}(r)$ recursively as:

$$\lambda_{c,m}(r) = \frac{\lambda}{q} rq \phi_{m,m} p_{1,m}(r)(rqI - S^p)^{-1} 1_{n_p}$$
$$\lambda_{c,i}(r) = \frac{\lambda}{q} rq \sum_{j \ge i} \phi_{j,i} p_{1,j}(r)(rqI - S^p)^{-1} 1_{n_p} + \frac{\lambda}{q} rq \sum_{j>i} \psi_{j-1,i} p_{0,j}(r)(rqI - S^c)^{-1} 1_{n_c} + rq \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} \psi_{k-1,i} p_k^j(r)(rqI - S^c)^{-1} 1_{n_c} \tag{6}$$

for $i \in \{1, \ldots, m-1\}$. Note that $rq \sum_{j=1}^{n_p} [(rqI - S^p)^{-1}]_{i,j}$ is the probability that a steal happens before the completion of the parent, given that the parent started service in phase i. It then follows that $rq\phi_{m,m} p_{1,m}(r)(rqI - S^p)^{-1} 1_{n_p}$ indeed equals the expected number of batches of size m that are stolen per parent job (as the job must spawn m child jobs and these must be stolen as a batch before the parent completes service). For $i < m$, the first two sums of (6) represent the expected number of size i batches that are stolen from the original server, while the double sum counts the expected number of such steals that occur on a server different from the original server.

It remains to define $\lambda_p(r)$, for this we demand that $\pi_*(r) = q$ and that

$$\pi_*(r) + \sum_{\ell \ge 0} \pi_\ell(r)\mathbf{1} = 1,$$

where we denote $1_{mn_c+(m+1)n_p}$ by $\mathbf{1}$ for ease of notation.Then from equations (2) and (3),

$$q\Big(1 + R_0(r)(I - R(r))^{-1}\mathbf{1}\Big) = 1, \tag{7}$$

where the inverse of $I - R(r)$ exists due to Proposition 3.1. Using (4) and (7) we get:

$$\lambda_p(r) = \frac{(1-q) - q(\sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + \lambda\alpha)w}{q\alpha w}, \tag{8}$$

with $w = -(B_0(r) + \lambda IG(r))^{-1}(I - R(r))^{-1}\mathbf{1}$. Note that $\lambda_p(r)$ is well-defined for $q > 0$, i.e. $\rho < 1$. This completes the description of the QBD Markov chain.

Proposition 3.1. *The QBD process $\{X_t(r), Y_t(r), Z_t(r), W_t(r) : t \geq 0\}$ has a unique stationary distribution for any $r \geq 0$ if $\rho < 1$.*

Proof. The proof is given in Appendix A. □

4 RESPONSE TIME DISTRIBUTION

We define $T(r)$ as the response time of a job in a system with probe rate r. The response time is defined as the length of the time interval between the arrival of a parent job and the completion of this parent job and all of its spawned child jobs. $T(r)$ can be expressed as the sum of the waiting time $W(r)$ and the service time $J(r)$. The waiting time is defined as the amount of time that the parent job waits in the queue before its service starts. Clearly, the waiting and the service time of a job are independent in our QBD model.

By repeating the arguments of the proof of [22, Theorem 6.1], we find that $W(r)$ can be expressed as:

Theorem 4.1. *The distribution of the waiting time is given by*

$$P[W(r) > t] = (\mathbf{1}' \otimes \pi_0(I - R(r))^{-1})e^{\mathbb{W}t}vec\langle I\rangle$$

with $\mathbb{W} = ((A_0(r) + A_1)' \otimes I) + ((A_{-1}(r))' \otimes R(r))$ and where $vec\langle\cdot\rangle$ is the column stacking operator. The mean waiting time is

$$E[W(r)] = \int_0^\infty P[W(r) > t]\, dt = (\mathbf{1}' \otimes \pi_0(I - R(r))^{-1})(-\mathbb{W})^{-1}vec\langle I\rangle.$$

We now focus on the service time $J(r)$. We can derive recursive formulas for $P[J(r) < t]$ which turn out to not be very suitable for numerical calculations. As these formulas are similar to those from [11, Section 4], we omit them in this paper. We can also find a formula for the mean service time for general PH distributed child and parent jobs, however, to improve readability, we opt to present a scheme for calculating the mean service time in case of hyper-exponential parent and child job service requirements.

Consider a set of j servers, where the k-th server contains i_k child jobs, with the child job in service being in phase f_k, where $j \geq 1$, $0 \leq i_1 + \cdots + i_j \leq m$ and $i_k \geq 0$ for $k = 1, \ldots, j$. Let $E^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$ be the expected time until all these child jobs have completed service. Define similarly $\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$, except that the first server contains i_1 pending child jobs and a parent job that is in service and is in phase f_1. By definition, we can drop i_k's that are zero (except i_1 in $\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$) as long as we drop the corresponding upper indices. We also can permute the indices of $E^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$ except the first one of $\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$, as long as we permute upper and lower indices in the same way. We have $E^k_1(r) = 1/s^c_k$ and $\bar{E}^k_0(r) = 1/s^p_k$, as $E^k_1(r)$ (resp. $\bar{E}^k_0(r)$) simply denotes the mean time until completion of a single child (resp. parent)

job in phase k. Consider a configuration with $j \geq 1$ queues, with queue $k = 1, \ldots, j$ having i_k children and the child in service being in phase f_k. A service completion in the k-th queue therefore occurs at rate $s^c_{f_k}$, which decreases i_k by 1. If $i_k > 1$, then the next child job starts service in phase s with probability α^c_s. A steal can only occur in the k-th queue if $i_k > 1$, in this case, at rate rq, ν jobs get transferred with probability $\psi_{i_k-1,\nu}$ to a new queue and the first of these children start service in phase s with probability α^c_s. The total rate at which jobs get completed and stolen equals $\sum_{k=1}^{j} s^c_{f_k}$ and $rq \sum_{k=1}^{j} 1[i_k > 1]$ respectively. As such it takes on average $1/\sum_{k=1}^{j}(s^c_{f_k} + rq1[i_k > 1])$ units of time until a completion or a steal occurs and the probability that a completion (resp. a steal) occurs in k-th server is given by $s^c_{f_k}/\sum_{k=1}^{j}(s^c_{f_k} + rq1[i_k > 1])$ (resp. $rq1[i_k > 1]/\sum_{k=1}^{j}(s^c_{f_k} + rq1[i_k > 1])$). We therefore get the following recursive relations:

$$E^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r) = \frac{1}{\sum_{k=1}^{j}(s^c_{f_k} + rq1[i_k > 1])}\Big(1 + \sum_{k=1}^{j} s^c_{f_k} \sum_{s=1}^{n_c} \alpha^c_s E^{f_1,\ldots,f_{k-1},s,f_{k+1},\ldots,f_j}_{i_1,\ldots,i_{k-1},i_k-1,i_{k+1},\ldots,i_j}(r)$$
$$+ rq \sum_{k=1}^{j} 1[i_k > 1] \sum_{\nu=1}^{i_k-1} \psi_{i_k-1,\nu} \sum_{s=1}^{n_c} \alpha^c_s E^{f_1,\ldots,f_j,s}_{i_1,\ldots,i_{k-1},i_k-\nu,i_{k+1},\ldots,i_j,k}(r)\Big).$$

Similarly, we can derive a recursive formula for $\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$, except that now we have a parent job being served in the first queue:

$$\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r) = \frac{1}{s^p_{f_1} + rq1[i_1 > 0] + \sum_{k=2}^{j}(s^c_{f_k} + rq1[i_k > 1])}\Big(1$$
$$+ s^p_{f_1} \sum_{s=1}^{n_c} \alpha^c_s E^{s,f_2,\ldots,f_j}_{i_1,\ldots,i_j}(r) + rq1[i_1 > 0] \sum_{k=1}^{i_1} \phi_{i_1,k} \sum_{s=1}^{n_c} \alpha^c_s \bar{E}^{f_1,\ldots,f_j,s}_{i_1-k,i_2,\ldots,i_j,k}(r)$$
$$+ \sum_{\nu=2}^{j} s^c_{f_\nu} \sum_{s=1}^{n_c} \alpha^c_s \bar{E}^{f_1,\ldots,f_{\nu-1},s,f_{\nu+1},\ldots,f_j}_{i_1,\ldots,i_{\nu-1},i_\nu-1,i_{\nu+1},\ldots,i_j}(r) + rq \sum_{\nu=2}^{j} 1[i_\nu > 1] \sum_{k=1}^{i_\nu-1} \psi_{i_\nu-1,k} \sum_{s=1}^{n_c} \alpha^c_s \bar{E}^{f_1,\ldots,f_j,s}_{i_1,\ldots,i_{\nu-1},i_\nu-k,i_{\nu+1},\ldots,i_j,k}(r)\Big).$$

We then have

$$E[J(r)] = \sum_{k=0}^{m} p_k \sum_{i=1}^{n_p} \alpha^p_i \bar{E}^i_k(r).$$

Hence, for hyperexponential job sizes we have a recursive formula for the mean service time. The idea of this formula can be extended to the case where parent and child jobs have acyclic phase type[1] distributed job requirements, where we condition not only on whether we have a steal or a service completion, but also on whether we have a phase change.

We end this section with an explanation on how to implement the recursive formulas $E^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$ and $\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$. We first explain how to compute $E^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$, where we assume without loss of generality that $i_1 \geq i_2 \geq \cdots \geq i_j$. For $k = 1, 2, \ldots, m$, let $\mathfrak{p}_k$ denote the number of unique partitions of integer k and let the $\mathfrak{p}_k \times k$ matrix P_k be a list of the unique partitions of integer k, for example

$$P_4 = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

[1] A phase type distribution characterized by (α, S) is acyclic if the rows and columns of S can be permuted to make S upper triangular.

and $\mathfrak{p}_4 = 5$. If $i_1 + i_2 + \cdots + i_j = k$, then if we ignore the zeros of P_k, its rows contain all possible tuples $(i_1, \ldots, i_j)$. Note, that the number of queues in a given configuration that can have their jobs stolen is simply given by the number of integers greater than 1 in the corresponding row of P_k. Similarly, the number of busy servers in a configuration is given by the number of non-zero entries in the corresponding row of P_k.

For $k = 1, \ldots, m$ set E_k as the zero matrix of size $\mathfrak{p}_k \times n_c^k$. If $i_1 + \cdots + i_j = k$ and if $(i_1, \ldots, i_j)$ can be found in the g-th row of P_k, then we would like the (g, h)-th entry of E_k to be equal to $E^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$, where $h = 1 + \sum_{s=1}^{j}(f_s - 1)n_c^{s-1}$. To calculate the entries of the g-th row of E_k we only need to know the lower rows of E_k (due to steals) and the matrix E_{k-1} (due to completions). As $E_1 = [1/s_1^c, \ldots, 1/s_{n_c}^c]$, we can calculate the entries of E_k inductively, where, for every k, the rows of E_k are calculated from the bottom up.

$\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$ can be computed similarly, although we now have to account for the parent in the first server. W.l.o.g. assume that $i_2 \geq i_3 \geq \cdots \geq i_j$, $i_1 \geq 0$ and $1 + i_1 + i_2 + \cdots + i_j = k$. Let $\bar{P}_k$ be the matrix containing all the tuples $(i_1, \ldots, i_j)$. Let $\bar{\mathfrak{p}}_k$ denote the number of rows of $\bar{P}_k$. We denote the zero matrix of dimension $k \times \ell$ as $0_{k,\ell}$, i.e. $0_{k,\ell} = 0_k \otimes 0'_\ell$. Then $\bar{P}_k$ can be constructed as follows:

$$\bar{P}_k = \begin{bmatrix} k & & 0_{1,k-1} \\ (k-1)1_{\mathfrak{p}_1} & P_1 & 0_{\mathfrak{p}_1,k-2} \\ (k-2)1_{\mathfrak{p}_2} & P_2 & 0_{\mathfrak{p}_2,k-3} \\ \vdots & \vdots & \vdots \\ 2 \cdot 1_{\mathfrak{p}_{k-2}} & P_{k-2} & 0_{\mathfrak{p}_{k-2},1} \\ 1_{\mathfrak{p}_{k-1}} & & P_{k-1} \end{bmatrix}.$$

Note that the first column of $\bar{P}_k$ represents the number of jobs in the first queue (including the parent job).

For $k = 1, \ldots, m+1$ set $\bar{E}_k$ as the zero matrix of size $\bar{\mathfrak{p}}_k \times n_p n_c^{k-1}$. If $1 + i_1 + \cdots + i_j = k$ and if $(i_1, \ldots, i_j)$ corresponds to g-th row of $\bar{P}_k$, then we would like the (g, h)-th entry of $\bar{E}_k$ to be equal to $\bar{E}^{f_1,\ldots,f_j}_{i_1,\ldots,i_j}(r)$, where $h = 1 + \sum_{s=2}^{j}(f_s - 1)n_c^{s-2} + (f_1 - 1)n_c^{j-1}$. To calculate the entries of the g-th row of $\bar{E}_k$ we only need to know the lower rows of $\bar{E}_k$ (due to steals) and the matrices $\bar{E}_{k-1}$ and E_{k-1} (due to child and parent completions respectively). As $\bar{E}_1 = [1/s_1^p, \ldots, 1/s_{n_p}^p]$, we can calculate the entries of $\bar{E}_k$ inductively, where, for every k, the rows of $\bar{E}_k$ are calculated from the bottom up.

5 NUMERICAL EXPERIMENTS

In [11], we defined the class of monotone deterministic (MD) strategies and we tested in different settings the strategies of stealing a single child job, half of the waiting children and all waiting children against the optimal MD strategy for those settings in case of exponential parent/child job sizes. We concluded that the stealing policy where the half of child jobs gets stolen is in general a good stealing policy for higher values of r and moderate system loads ρ, while the strategy of stealing all children performs best for low values of r and higher values of ρ. We concluded further that stealing only one child performs the worst in most of the cases. In this section we examine whether these conclusions remain valid for systems with hyperexponential parent and child job sizes with two phases ($hexp(2)$).

To this end we describe a $hexp(2)$ distribution using the parameters EX, SCV, f where EX is the mean of the distribution, where SCV is the squared coefficient of variation and where f is the fraction of the workload contributed by phase 1 jobs (f is sometimes called the shape parameter). Using these parameters we can generate a $hexp(2)$ distribution with parameters $([\beta_1, 1-\beta_1], [\mu_1, \mu_2])$, where

$$\mu_1 = \frac{SCV + (4f-1) + \sqrt{(SCV-1)(SCV-1+8f(1-f))}}{2EX \cdot f(SCV+1)},$$

$$\mu_2 = \frac{SCV + (4(1-f)-1) - \sqrt{(SCV-1)(SCV-1+8f(1-f))}}{2EX \cdot (1-f)(SCV+1)}$$

and $\beta_1 = EX \cdot \mu_1 f$. In the remainder of the paper we use the same values of SCV and f for parent and child jobs. We assume that parent and child jobs have service requirements with mean 2 and mean 1 respectively.

Recall that we defined the matrix Ψ as the matrix where $[\Psi]_{i,j} = \psi_{i,j}$ and Φ similarly. Note that a strategy is fully characterized by the matrices Ψ and Φ. The strategies of stealing a single child job, half of child jobs and all children are defined as follows:

(1) **Steal one:** The strategy of always stealing one child job, that is $\phi_{i,1} = \psi_{i,1} = 1$ for every i.
(2) **Steal half:** The strategy of always stealing half of the pending child jobs. If n, the number of pending child jobs, is uneven, there is a fifty percent chance that $\lfloor n/2 \rfloor$ child jobs get stolen and $\lceil n/2 \rceil$ jobs otherwise.
(3) **Steal all:** The strategy of stealing all of the pending child jobs, that is $\phi_{i,i} = \psi_{i,i} = 1$ for every i.

Note that these strategies do not rely on any knowledge on the (mean) job sizes or system load. In [11] a strategy was called monotone deterministic (MD) if, for every i, $\psi_{i,j} = 1$ implies $\psi_{i+1,j'} = 1$ for some $j' \geq j$ and the same holds for Φ. The optimal MD strategy is determined using brute force and its response time is denoted as $T_{MD}(r)$. The mean response time of other strategies is always normalized by $T_{MD}(r)$ in the subsequent experiments.

We now present a selection of performed numerical experiments (due to the lack of space). The main conclusions in the omitted experiments are in agreement with the results presented here. Let $\mathbf{p} = [p_0, p_1, \ldots, p_m]$.

Example 5.1. In Figure 1 we illustrate the effect of increasing the load ρ on the on performance of the three strategies for different values of SCV. We do this for $\rho \in [0.05, 0.95]$, $SCV \in \{2, 5, 20\}$, $f = 1/3$, $\mathbf{p} = [5, 4, 3, 2, 1]/15$ and $r = 2$. These results (and other results omitted here) confirm that stealing all is best when the load is sufficiently high, while stealing half of the child jobs is a good strategy for systems with a moderate load.

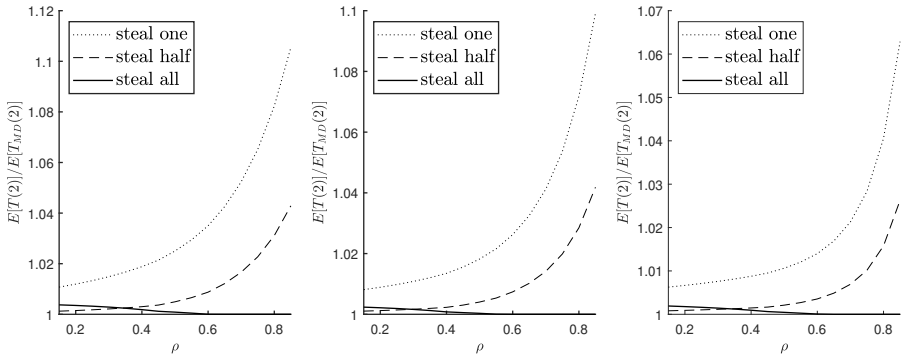

Fig. 1. Example 5.1 with $SCV = 2$ (left), $SCV = 5$ (mid) and $SCV = 20$ (right).

Example 5.2. In Figure 2 we consider the system with $\rho = 0.85$, $SCV \in [1, 20]$, $f = 1/2$, $\mathbf{p} = 1'_6/6$ and $r \in \{1, 5, 10\}$. We examine the effect of increasing the value of the SCV on the performance of the three stealing strategies against the performance of the system with no stealing. Clearly, as the SCV increases the ratio $E[T(r)]/E[T(0)]$ decreases for each of the three strategies. In fact, for every stealing strategy we have that as the $SCV \to \infty$, the ratio $E[T(r)]/E[T(0)] \to 0$. This implies that as the SCV increases it is more and more worthwhile to steal.

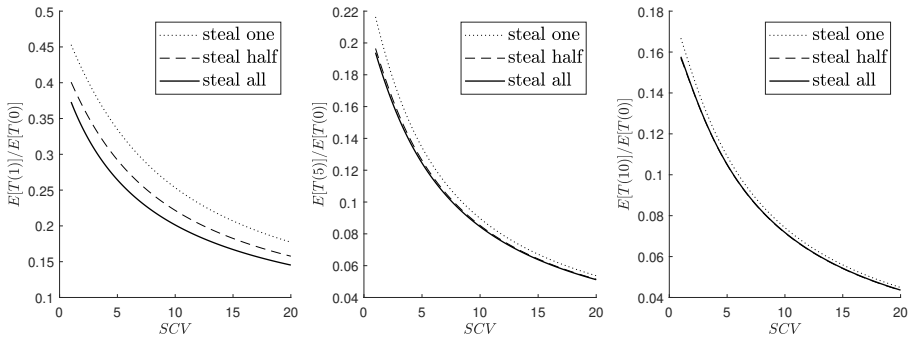

Fig. 2. Example 5.2 with $r = 1$ (left), $r = 5$ (mid) and $r = 10$ (right).

Example 5.3. Example 5.2 shows that the mean response time of the strategy of not stealing grows quicker than those of the strategies where stealing occurs. In this example we examine this growth in more detail. In Figure 3, we therefore plot $E[T(r)]$ in function of SCV for the three strategies and for the system where no stealing occurs. We do this for $\rho = 0.85$, $SCV \in [1, 40]$, $f = 1/2$, $\mathbf{p} = 1'_6/6$ and $r = 5$. Clearly, the growth of $E[T(0)]$ (no stealing) is linear in function of SCV. Further as $SCV \to \infty$, $E[T(r)]$ seems to converge for any stealing strategy, which is equivalent to saying that from the moment that SCV is large enough, there is not much difference in the performance of a stealing strategy when the SCV is increased further.

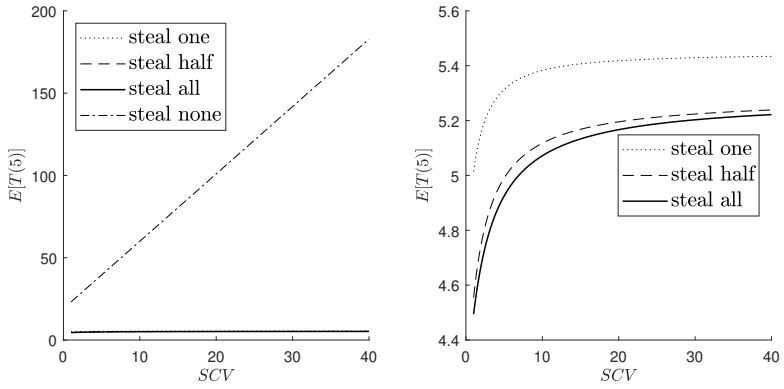

Fig. 3. Example 5.3 with the system where no stealing occurs (left) and without that system (right).

6 MODEL VALIDATION

Based on numerical experiments in the previous section, we see that stealing all or half of the children are good stealing policies, stealing all works best for low values of r, while stealing half of the children works well for higher values. Therefore we validate the model for these two policies by means of simulation. We run all simulations for $T = 10^5$ with a warm up period of 33% of T, always starting from an empty system.

In Figure 4, we compare the simulated waiting and service time distributions and those of the QBD model. We do this for $\rho = 0.85, SCV = 2, f = 1/2, \mathbf{p} = 1'_5/5$ and $r \in \{1, 5\}$. The simulated waiting and service times were calculated based on 5 runs, with the number of queues $N = 2000$. We see that there is a good match between the simulated waiting and service time distributions and those of the QBD model. Also, that the match is less good for $r = 5$ than for $r = 1$. Note that having 2000 or more CPU-cores is not uncommon in an HPC cluster.

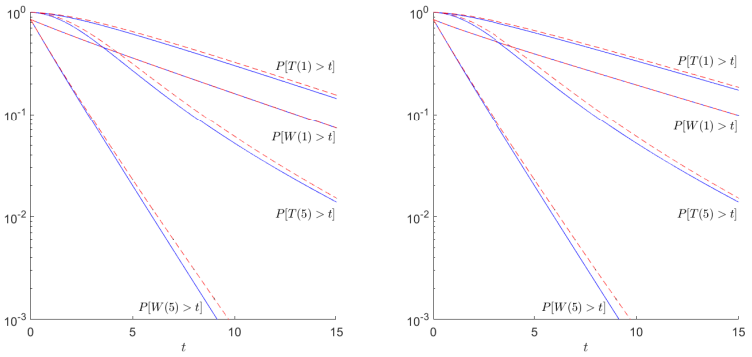

Fig. 4. Waiting and response times from the QBD (blue) and simulations (red) for the strategies of stealing all children (left) and half of the children (right).

In Table 2 we compare the relative error of the simulated mean response time, based on 20 runs, to the one obtained using formula from Section 4. In We do this for the policy of stealing half of the available children. Similar results were obtained when all children are stolen. We simulate the systems for $f = 1/2$, $SCV \in \{2, 20\}$, $\mathbf{p} = 1'_5/5$, $\rho \in \{0.75, 0.85\}$, $r = 1$ and $N \in \{250, 500, 1000, 2000, 4000\}$.

	$\rho = 0.75$		$\rho = 0.85$	
N	sim. ± conf.	rel.err.%	sim. ± conf.	rel.err.%
$SCV = 2$				
250	6.4925 ± 6.67e-03	0.4706	9.5338 ± 1.72e-02	0.7855
500	6.4788 ± 3.88e-03	0.2586	9.4893 ± 1.48e-02	0.3156
1000	6.4683 ± 3.94e-03	0.0963	9.4691 ± 7.66e-03	0.1021
2000	6.4638 ± 2.13e-03	0.0260	9.4647 ± 7.38e-03	0.0547
4000	6.4635 ± 9.64e-04	0.0214	9.4597 ± 4.51e-03	0.0024
QBD	6.4621		9.4595	
$SCV = 20$				
250	8.1792 ± 2.88e-02	2.0152	17.1200 ± 1.18e-01	2.3903
500	8.0953 ± 1.77e-02	0.9686	16.8921 ± 7.14e-02	1.0272
1000	8.0473 ± 8.81e-03	0.3701	16.8081 ± 6.00e-02	0.5248
2000	8.0347 ± 8.64e-03	0.2127	16.7477 ± 3.80e-02	0.1637
4000	8.0226 ± 7.66e-03	0.0619	16.7388 ± 3.72e-02	0.1104
QBD	8.0176		16.7204	

Table 2. Relative error of simulation results for $E[T(r)]$ for the policy of stealing half of the children, based on 20 runs.

The relative error in all cases is below 2.5% and tends to increase with the value of the *SCV*. We note that based on simulations omitted here the relative error tends to increase with the steal rate r, which is in agreement with Figure 4. More importantly the relative error seems roughly to halve when doubling N (which is in agreement with the results in [8]). This suggests that the approximation error tends to zero as the number of servers tends to infinity.

7 MEAN FIELD MODEL

In this section we present a mean field model for the work stealing system considered in this paper and show that it has a unique fixed point that coincides with the steady state vector of the QBD Markov chain. In this manner we provide additional support for the claim that the approximation error of the QBD model tends to zero as the number of servers becomes large. Note that this result is not sufficient to formally prove this. One of the main challenges in coming up with such a formal proof is to establish global attractor of the fixed point, which is often done using monotonicity arguments. This however is not feasible for the system considered in this paper, as the system is clearly not monotone in some cases (e.g. a system where out of 5 child jobs always 5 get stolen, whereas out of 4 children only one is transferred upon a successful steal attempt).

We start by writing down the set of ODEs that capture the evolution of the mean field model (i.e., the so-called drift equations). We denote by $f_{\ell,j,k,i}(t)$ the fraction of queues at time t with ℓ parent jobs in waiting in the queue, $j \in \{1,\dots,m\}$ child jobs in the queue, $k \in \{0,1\}$ describing whether a parent job is in service ($k=1$) or not ($k=0$) and i being the phase of the job currently in service (if $k=1$, then $i \in \{1,\dots,n_p\}$ and if $k=0$, then $i \in \{1,\dots,n_c\}$). When there is no job in service we set $i=0$. Note that ℓ does not count parent jobs in service, whereas j counts child jobs waiting and in service. In particular for $\ell=0$ and $j+k \geq 1$ the server is busy and there may be child jobs waiting, which can be transferred. We denote $f_{0,0,0,0}(t)$ as $f_*(t)$, the fraction of idle queues. For a statement A we set $1[A]$ to be 1 if A is true and 0 if A is false.

Let

$$\vec{f}_\ell(t) = (\vec{f}_{\ell,1,0}(t),\dots,\vec{f}_{\ell,m,0}(t),\vec{f}_{\ell,0,1}(t),\dots,\vec{f}_{\ell,m,1}(t))$$

for every $\ell \geq 0$, where $\vec{f}_{\ell,j,0}(t) = (\vec{f}_{\ell,j,0,1}(t),\dots,\vec{f}_{\ell,j,0,n_c}(t))$ and $\vec{f}_{\ell,j,1}(t) = (\vec{f}_{\ell,j,1,1}(t),\dots,\vec{f}_{\ell,j,1,n_p}(t))$. Then, for $\ell > 0$ we have

$$\frac{d}{dt}\vec{f}_\ell(t) = \lambda\vec{f}_{\ell-1}(t) + \vec{f}_\ell(t)\tilde{A}_0(t) + \vec{f}_{\ell+1}(t)\mu\alpha + rf_*(t)\vec{f}_{\ell+1}(t)V_0, \tag{9}$$

for $\ell=0$ and $j+k \geq 1$ we have

$$\frac{d}{dt}\vec{f}_0(t) = \lambda f_*(t)\alpha + \vec{f}_0(t)\tilde{B}_0(t) + \vec{f}_1(t)\mu\alpha + rf_*(t)\vec{f}_1(t)V_0 + rf_*(t)\sum_{\ell'\geq 0}\vec{f}_{\ell'}(t)T + rf_*(t)\sum_{\ell'\geq 1}\vec{f}_{\ell'}(t)v_0\alpha, \tag{10}$$

and for $\ell, j, k = 0$

$$\frac{d}{dt}f_*(t) = -\lambda f_*(t) + \vec{f}_0(t)\mu - rf_*(t)\big(1 - f_*(t) - \vec{f}_0(t)v_0\big). \tag{11}$$

The first term of (9) and (10) is due to arrivals. The matrices $\tilde{A}_0(t)$ and $\tilde{B}_0(t)$ are the same matrices as $A_0(r)$ and $B_0(r)$ respectively, except with every instance of q changed to $f_*(t)$. The second term of (9) and (10) therefore denotes the drift due to transitions for which the level remains unchanged, due to arrivals to and due to parent steals from queues of length ℓ. The third and the fourth term are due, respectively, to service completions and parent steals in queues of length $\ell+1$. We denote $v_0 = \begin{bmatrix} 1'_{n_c} & 0'_{(m-1)n_c} & 1'_{n_p} & 0'_{mn_p} \end{bmatrix}'$, where the entries are non-zero when $j+k=1$ (i.e. $V_0 = \mathrm{diag}(v_0)$). We define the $(m-1)\times(m-1)$ matrix Ψ as $[\Psi]_{i,j} = \psi_{i,j}$ and similarly $m \times m$ matrix Φ. Recall that we denote the zero matrix of dimension $k \times \ell$ as $0_{k,\ell}$. The matrix $T = T_\psi + T_\phi$ records the distribution of the number of

child jobs transferred when a probe is successful:

$$T_\psi = \begin{bmatrix} 0_{n_c,(m-1)n_c} & 0_{n_c,n_c+(m+1)n_p} \\ \Psi \otimes (1_{n_c}\alpha^c) & 0_{(m-1)n_c,n_c+(m+1)n_p} \\ 0_{(m+1)n_p,(m-1)n_c} & 0_{(m+1)n_p,n_c+(m+1)n_p} \end{bmatrix}, T_\phi = \begin{bmatrix} 0_{mn_c+n_p,mn_c} & 0_{mn_c+n_p,(m+1)n_p} \\ \Phi \otimes (1_{n_p}\alpha^c) & 0_{mn_p,(m+1)n_p} \end{bmatrix}.$$

The final two terms of (10) are thus due to transfers to empty queues of child jobs and of parents respectively. Similarly, for $\frac{d}{dt}f_*(t)$ the first term is due to job arrivals, the next is due to service completions and the last is due to job transfers. Note that if $\phi_{i,1} = 1$ for every $i \in \{1,\ldots,m\}$ and if $\psi_{j,1} = 1$ for every $j \in \{1,\ldots,m-1\}$, then $T = (1-v_0)\kappa_1$.

We show that the stationary distribution of the QBD corresponds to the unique fixed point ζ of the set of ODEs in Equations (9)-(11). The following lemma says that, in equilibrium, the rate at which the level in non-empty queues increases, that is $\lambda(1-\zeta_*)$, is exactly the rate at which the level decreases in such queues (which can only happen due to a service completion or a steal in queues with no pending child jobs).

Lemma 7.1. *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \ldots)$ with $\zeta_* + \sum_{\ell\geq 0}\vec{\zeta}_\ell \mathbf{1} = 1$ of the set of ODEs in Equations (9)-(11) we have*

$$\lambda = \lambda\zeta_* + \sum_{\ell\geq 1}\vec{\zeta}_\ell\mu + r\zeta_*\sum_{\ell\geq 1}\vec{\zeta}_\ell v_0.$$

Proof. As $\frac{d}{dt}\vec{f}_\ell(t) = 0$ in a fixed point we get using $\sum_{\ell\geq 0}(\ell+1)\frac{d}{dt}\vec{f}_\ell(t) = 0$ and $\sum_{\ell\geq 0}\vec{\zeta}_\ell\mathbf{1} = 1-\zeta_*$ that

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell\mu = \lambda + r\zeta_*\left(1-\zeta_* - \sum_{\ell\geq 0}\vec{\zeta}_\ell v_0\right). \tag{12}$$

The claim now follows by using (12) and (11) in a fixed point. □

Define recursively the row vector

$$\xi_{1,m} = \lambda p_m \alpha^p$$

and

$$\xi_{1,k} = \lambda p_k \alpha^p + r\zeta_* \sum_{d=k+1}^{m}\phi_{d,d-k}\xi_{1,d}(r\zeta_* I - S^p)^{-1},$$

for $k = 0,\ldots,m-1$, and

$$\xi_{0,k'} = \xi_{1,k'}(r\zeta_* I - S^p)^{-1}s^p\alpha^c + r\zeta_*\sum_{d=k'}^{m}\phi_{d,k'}\xi_{1,d}(r\zeta_* I - S^p)^{-1}1_{n_p}\alpha^c$$
$$+ 1[k' < m]\xi_{0,k'+1}(r\zeta_* I - S^c)^{-1}s^c\alpha^c + r\zeta_*\sum_{d=k'+1}^{m}\xi_{0,d}(r\zeta_* I - S^c)^{-1}(\psi_{d-1,d-k'} + \psi_{d-1,k'}1_{n_c}\alpha^c),$$

for $k' = 1,\ldots,m$. $\xi_{i,j,k}$ is the rate at which servers enter into phase (i,j,k) due to arrivals, completions and steals.

The intuition behind the next two lemmas is that if the system is in equilibrium, the rate at which queues enter phase (i,j,k) should equal to the rate at which queues leave phase (i,j,k).

Lemma 7.2. *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \ldots)$ with $\zeta_* + \sum_{\ell\geq 0}\vec{\zeta}_\ell\mathbf{1} = 1$ of the set of ODEs in Equations (9)-(11) we have for $1 \leq k \leq m$:*

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_p} \\ I_{n_p} \\ 0_{(m-k)n_p,n_p} \end{pmatrix}(r\zeta_* I - S^p) = \xi_{1,k}. \tag{13}$$

Proof. We prove the lemma using complete backward induction on k. By demanding that $\sum_{\ell\geq 0}\vec{f}_\ell(t)\left[0'_{mn_c+kn_p,n_p}, I_{n_p}, 0'_{(m-k)n_p,n_p}\right]' = 0$ for any $k\in\{1,\ldots,m\}$, we find due to Lemma 7.1 that

$$0 = \lambda p_k \alpha^p - \sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_p} \\ I_{n_p} \\ 0_{(m-k)n_p,n_p}\end{pmatrix}\left(r\zeta_* I - S^p\right) + r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell\left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1}I_{n_p},\ldots,\phi_{m,m-k}I_{n_p}\right]'.$$

This is equivalent to

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c+kn_p,n_p} \\ I_{n_p} \\ 0_{(m-k)n_p,n_p}\end{pmatrix}\left(r\zeta_* I - S^p\right) = \lambda p_k\alpha^p + r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell\left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1}I_{n_p},\ldots,\phi_{m,m-k}I_{n_p}\right]'. \tag{14}$$

(14) is equivalent to (13) for $k=m$. Suppose now that $k<m$ and that (13) holds for all $k'\in\{k+1,\ldots,m\}$. Due to (14), it suffices to show that

$$r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell\left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1}I_{n_p},\ldots,\phi_{m,m-k}I_{n_p}\right]' = r\zeta_*\sum_{d=k+1}^{m}\phi_{d,d-k}\xi_{1,d}\left(r\zeta_* I - S^p\right)^{-1}.$$

This is equivalent to

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell\left[0'_{mn_c+(k+1)n_p,n_p}, \phi_{k+1,1}I_{n_p},\ldots,\phi_{m,m-k}I_{n_p}\right]'\left(r\zeta_* I - S^p\right) = \sum_{d=k+1}^{m}\phi_{d,d-k}\xi_{1,d},$$

which holds due to induction hypothesis. □

Lemma 7.3. *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \ldots)$ with $\zeta_* + \sum_{\ell\geq 0}\vec{\zeta}_\ell\mathbf{1} = 1$ of the set of ODEs in Equations (9)-(11) we have for $2\leq k\leq m$:*

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{(k-1)n_c,n_c} \\ I_{n_c} \\ 0_{(m-k-2)n_c+(m+1)n_p,n_c}\end{pmatrix}\left(r\zeta_* I - S^c\right) = \xi_{0,k}. \tag{15}$$

Proof. The proof is analogous to that of Lemma 7.2, except we also rely on (13). □

Proposition 7.4. *For any fixed point $\zeta = (\zeta_*, \vec{\zeta}_0, \vec{\zeta}_1, \ldots)$ with $\zeta_* + \sum_{\ell\geq 0}\vec{\zeta}_\ell\mathbf{1} = 1$ of the set of ODEs in Equations (9)-(11) we have*

$$\zeta_* = q, \tag{16}$$

$$r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T = \zeta_*\sum_{j=1}^{m}\lambda_{c,j}(r)\kappa_j, \tag{17}$$

where $\lambda_{c,j}(r)$ was defined in (6).

Proof. Denote by $(1{:}k)$ the column vector $[1,\ldots,k]'$ for $k\geq 1$. To prove (16) it suffices to show

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell \begin{pmatrix} 0_{mn_c} \\ 1_{(m+1)n_p}\end{pmatrix} = \lambda\alpha^p\left(-S^p\right)^{-1}1_{n_p}, \tag{18}$$

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell\begin{pmatrix}1_{mn_c}\\0_{(m+1)n_p}\end{pmatrix}=\lambda\left(\sum_{i=1}^{m}ip_i\right)\alpha^c(-S^c)^{-1}1_{n_c}. \tag{19}$$

By demanding $\sum_{\ell\geq 0}\frac{d}{dt}\vec{f}_\ell(t)=0$ and by using Lemma 7.1, we find

$$0=\lambda\alpha+\sum_{\ell\geq 0}\vec{\zeta}_\ell\tilde{S}+r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell V_0-r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell+r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T, \tag{20}$$

Where $\tilde{S}$ is the same matrix as $S(r)$ except with all instances of q changed to ζ_*. By multiplying (20) with $[0'_{mn_c,n_p},(1_{m+1}\otimes I_{n_p})']'$, we get

$$0=\lambda\alpha^p+\sum_{\ell\geq 0}\vec{\zeta}_\ell\begin{pmatrix}0_{mn_c,n_p}\\1_{m+1}\otimes S^p\end{pmatrix}, \tag{21}$$

which yields (18). By demanding $\sum_{\ell\geq 0}\frac{d}{dt}\vec{f}_\ell(t)[((1{:}m)\otimes 1_{n_c})'\ 0\ ((1{:}m)\otimes 1_{n_p})']'=0$ and by using Lemma 7.1, one can show that

$$\sum_{\ell\geq 0}\vec{\zeta}_\ell\begin{pmatrix}1_m\otimes s^c\\0_{(m+1)n_p}\end{pmatrix}=\lambda\sum_{i=1}^{m}ip_i. \tag{22}$$

By multiplying (20) with $[(1_m\otimes I_{n_c})',0'_{(m+1)n_p,n_c}]'$ and by using (12) on the last sum we get

$$\lambda\alpha^c=\sum_{\ell\geq 0}\vec{\zeta}_\ell\begin{pmatrix}1_m\otimes s^c\\1_{m+1}\otimes s^p\end{pmatrix}\alpha^c+\sum_{\ell\geq 0}\vec{\zeta}_\ell\begin{pmatrix}1_m\otimes S^c\\0_{(m+1)n_p,n_c}\end{pmatrix}.$$

Due to (21) and (22) this is equivalent to

$$0=\lambda\left(\sum_{i=1}^{m}ip_i\right)\alpha^c+\sum_{\ell\geq 0}\vec{\zeta}_\ell\begin{pmatrix}1_m\otimes S^c\\0_{(m+1)n_p,n_c}\end{pmatrix},$$

which gives (19). To prove the second claim it suffices to show, due to the definition of T, that for $i=1,\ldots,m$ we have:

$$r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T\begin{pmatrix}0_{(i-1)n_c}\\1_{n_c}\\0_{(m-i)n_c+(m+1)n_p}\end{pmatrix}=\zeta_*\lambda_{c,i}(r).$$

This is equivalent to showing the following two equalities:

$$\begin{aligned}r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T_\psi\begin{pmatrix}0_{(i-1)n_c}\\1_{n_c}\\0_{(m-i)n_c+(m+1)n_p}\end{pmatrix}&=\lambda rq\sum_{j>i}\psi_{j-1,i}p_{0,j}(r)(rqI-S^c)^{-1}1_{n_c}\\&\quad+rq^2\sum_{j=i+1}^{m}\lambda_{c,j}(r)\sum_{k=i+1}^{j}\psi_{k-1,i}p_k^j(r)(rqI-S^c)^{-1}1_{n_c}\end{aligned} \tag{23}$$

for $i=1,\ldots,m-1$, and

$$r\zeta_*\sum_{\ell\geq 0}\vec{\zeta}_\ell T_\phi\begin{pmatrix}0_{(i-1)n_c}\\1_{n_c}\\0_{(m-i)n_c+(m+1)n_p}\end{pmatrix}=\lambda rq\sum_{j\geq i}\phi_{j,i}p_{1,j}(r)(rqI-S^p)^{-1}1_{n_p}, \tag{24}$$

for $i = 1, \dots, m$. Due to (13), we have

$$r\zeta_* \sum_{\ell \geq 0} \vec{\zeta}_\ell T_\phi \begin{pmatrix} 0_{(i-1)n_c} \\ 1_{n_c} \\ 0_{(m-i)n_c+(m+1)n_p} \end{pmatrix} = r\zeta_* \sum_{d \geq i} \phi_{d,i} \xi_{1,d} (rqI - S^p)^{-1} 1_{n_p}.$$

As

$$\xi_{1,d} = \lambda p_{1,d}(r), \tag{25}$$

where all instances of q in the formula of $p_{1,d}(r)$ have been changed to ζ_*, equation (24) follows from (16). Equation (23) requires more work to prove. Due to (15), it suffices to show that

$$r\zeta_* \sum_{k=i}^{m-1} \psi_{k,i} \xi_{0,k+1} (r\zeta_* I - S^c)^{-1} 1_{n_c} = \lambda rq \sum_{j>i} \psi_{j-1,i} p_{0,j}(r) (rqI - S^c)^{-1} 1_{n_c} + rq^2 \sum_{j=i+1}^{m} \lambda_{c,j}(r) \sum_{k=i+1}^{j} \psi_{k-1,i} p_k^j(r) (rqI - S^c)^{-1} 1_{n_c}.$$

Due to (16), it suffices to show that

$$\sum_{k=i+1}^{m} \psi_{k-1,i} \xi_{0,k} = \lambda \sum_{k=i+1}^{m} p_{0,k}(r) \psi_{k-1,i} + q \sum_{k=i+1}^{m} \sum_{j=k}^{m} \lambda_{c,j}(r) p_k^j(r) \psi_{k-1,i}. \tag{26}$$

We show that for $k = 2, \dots, m$, we have

$$\xi_{0,k} = \lambda p_{0,k}(r) + q \sum_{j=k}^{m} \lambda_{c,j}(r) p_k^j(r) \tag{27}$$

and (26) then follows. We prove (27) by complete backward induction on k. By definition and (16), we have for $k = m$

$$\xi_{0,m} = \xi_{1,m} (r\zeta_* I - S^p)^{-1} s^p \alpha^c + r\zeta_* \phi_{m,m} \xi_{1,m} (r\zeta_* I - S^p)^{-1} 1_{n_p} \alpha^c = \lambda p_{0,m}(r) + q\lambda_{c,m}(r) p_m^m(r).$$

Suppose now that $k < m$ and that (27) holds for all $k' \in \{k+1, \dots, m\}$. We have by definition

$$\xi_{0,k} = \xi_{1,k} (r\zeta_* I - S^p)^{-1} s^p \alpha^c + r\zeta_* \sum_{d=k}^{m} \phi_{d,k} \xi_{1,d} (r\zeta_* I - S^p)^{-1} 1_{n_p} \alpha^c + \xi_{0,k+1} (r\zeta_* I - S^c)^{-1} s^c \alpha^c + r\zeta_* \sum_{d=k+1}^{m} \xi_{0,d} (r\zeta_* I - S^c)^{-1} (\psi_{d-1,d-k} + \psi_{d-1,k} 1_{n_c} \alpha^c).$$

By induction hypothesis, (16) and (25) this is equal to

$$\lambda p_{1,k}(r) (rqI - S^p)^{-1} s^p \alpha^c + \lambda rq \sum_{d=k}^{m} \phi_{d,k} p_{1,d}(r) (rqI - S^p)^{-1} 1_{n_p} \alpha^c + \left(\lambda p_{0,k+1}(r) + q \sum_{j=k+1}^{m} \lambda_{c,j}(r) p_{k+1}^j(r) \right) (rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{d=k+1}^{m} \left(\lambda p_{0,d}(r) + q \sum_{i=d}^{m} \lambda_{c,i}(r) p_d^i(r) \right) (rqI - S^c)^{-1} (\psi_{d-1,d-k} + \psi_{d-1,k} 1_{n_c} \alpha^c).$$

By first using the formula for $p_{0,k}(r)$ and then for $\lambda_{c,k}(r)$ (6) this is further equal to

$$\begin{aligned}
&\lambda p_{0,k}(r) + \lambda rq \sum_{d=k}^{m} \phi_{d,k} p_{1,d}(r)(rqI - S^p)^{-1} 1_{n_p} \alpha^c \\
&+ q \sum_{j=k+1}^{m} \lambda_{c,j}(r) p^j_{k+1}(r)(rqI - S^c)^{-1} s^c \alpha^c + \lambda rq \sum_{d=k+1}^{m} \psi_{d-1,k} p_{0,d}(r)(rqI - S^c)^{-1} 1_{n_c} \alpha^c \\
&+ rq^2 \sum_{i=k+1}^{m} \lambda_{c,i}(r) \sum_{d=k+1}^{i} p^i_d(r)(rqI - S^c)^{-1}(\psi_{d-1,d-k} + \psi_{d-1,k} 1_{n_c} \alpha^c) \\
&= \lambda p_{0,k}(r) + q\lambda_{c,k}(r)\alpha^c + q \sum_{j=k+1}^{m} \lambda_{c,j}(r) p^j_{k+1}(r)(rqI - S^c)^{-1} s^c \alpha^c \\
&+ rq^2 \sum_{i=k+1}^{m} \lambda_{c,i}(r) \sum_{d=k+1}^{i} \psi_{d-1,d-k} p^i_d(r)(rqI - S^c)^{-1}.
\end{aligned}$$

By rearranging the terms and by using the formula for $p^j_k(r)$ this equals

$$\begin{aligned}
&\lambda p_{0,k}(r) + q\lambda_{c,k}(r) p^k_k(r) + q \sum_{j=k+1}^{m} \lambda_{c,j}(r) \left(p^j_{k+1}(r)(rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{d=k+1}^{j} \psi_{d-1,d-k} p^j_d(r)(rqI - S^c)^{-1} \right) \\
&= \lambda p_{0,k}(r) + q \sum_{j=k}^{m} \lambda_{c,j}(r) p^j_k(r),
\end{aligned}$$

which shows (27), thus finishing the proof. □

Theorem 7.5. *The stationary distribution $\pi(r)$ of the QBD Markov chain characterized by $Q(r)$ is the unique fixed point ζ of the set of ODEs in Equations (9)-(11).*

Proof. Using Proposition 3.1 we show that the fixed point equations $\frac{d}{dt}\vec{f}_\ell(t) = 0$ are equivalent to the balance equations of the QBD Markov chain characterized by $Q(r)$. The uniqueness of the fixed point follows from the uniqueness of the stationary distribution of the Markov chain.

For $\ell \geq 1$, $\frac{d}{dt}\vec{f}_\ell(t) = 0$ can be written as

$$0 = \vec{\zeta}_{\ell-1}(\lambda I) + \vec{\zeta}_\ell \tilde{A}_0 + \vec{\zeta}_{\ell+1}(\mu\alpha + r\zeta_* V_0),$$

where $\tilde{A}_0$ is the same matrix as $A_0(r)$ except with every instance of q changed to ζ_*. This is exactly the balance equations of $Q(r)$ for $\ell \geq 1$ as $\zeta_* = q$ due to Proposition 7.4. This implies that $\vec{\zeta}_\ell = \vec{\zeta}_0 R(r)^\ell$, for all $\ell \geq 1$ for any fixed point.

For $\ell = 0$, $\frac{d}{dt}\vec{f}_\ell(t) = 0$ implies

$$0 = \vec{\zeta}_0 \tilde{B}_0 + \vec{\zeta}_1(\mu\alpha + r\zeta_* V_0) + \lambda\zeta_*\alpha + r\zeta_* \sum_{\ell' \geq 0} \vec{\zeta}_{\ell'} T + r\zeta_* \sum_{\ell' \geq 1} \vec{\zeta}_{\ell'} v_0 \alpha,$$

where $\tilde{B}_0$ is the same matrix as $B_0(r)$ except with every instance of q changed to ζ_*. Due to Proposition 7.4 we can rewrite this as

$$0 = \vec{\zeta}_0 B_0(r) + \vec{\zeta}_1 A_{-1}(r) + q\left(\sum_{j=1}^{m} \lambda_{c,j}(r)\kappa_j + \lambda\alpha + r \sum_{\ell \geq 1} \vec{\zeta}_\ell v_0 \alpha \right).$$

This implies that

$$\vec{\zeta}_0 = -q\left(\sum_{j=1}^{m}\lambda_{c,j}(r)\kappa_j + \lambda\alpha + r\sum_{\ell\geq 1}\vec{\zeta}_\ell v_0\alpha\right)(B_0(r) + \lambda IG(r))^{-1},$$

as $\lambda IG(r) = R(r)A_{-1}(r)$. As $\sum_{\ell\geq 0}\vec{\zeta}_\ell\mathbf{1} = 1 - q = \sum_{\ell\geq 0}\pi_\ell(r)\mathbf{1}$, we find that

$$r\sum_{\ell'\geq 1}\vec{\zeta}_{\ell'}v_0 = \lambda_p(r) \tag{28}$$

defined in (8). This indicates that $\frac{d}{dt}\vec{f}_\ell(t) = 0$ corresponds to the balance equation for $\ell = 0$. As $T\mathbf{1} = \mathbf{1} - v_0$, we have for $\frac{d}{dt}f_*(t) = 0$ that

$$\begin{aligned} 0 &= -\lambda\zeta_* + \vec{\zeta}_0\mu - r\zeta_*(1 - \zeta_* - \vec{\zeta}_0 v_0) \\ &= -\zeta_*\left(r\sum_{\ell'\geq 0}\vec{\zeta}_{\ell'}T\mathbf{1} + \lambda + r\sum_{\ell'\geq 1}\vec{\zeta}_{\ell'}v_0\right) + \vec{\zeta}_0\mu, \end{aligned}$$

which is exactly the first balance equation due to Proposition 7.4 and (28). □

8 CONCLUSIONS AND FUTURE WORK

We introduced a model for randomized work stealing in multithreaded large-scale systems, where parent jobs spawn child jobs and where any number of existing child jobs can be stolen from a queue by a single probe. We defined a Quasi-Birth-Death (QBD) Markov chain to approximate the system behaviour and showed, using simulation, that the approximation error tends to zero as the number of servers tends to infinity. To further support this observation we introduced a mean field model and showed that the stationary distribution of the QBD is the unique fixed point of the mean field model.

Using numerical experiments we examined the effect of changing the load ρ, the steal rate r and the variability of the job sizes. We studied the performance of some basic steal strategies and showed that stealing half of the child jobs is in general a good policy for higher steal rates r and/or lower loads ρ, while the strategy of stealing all children performs best for low steal rates r and/or higher loads. We further showed that stealing becomes more and more worthwhile when the job size variability increases.

Possible generalizations include stealing multiple parent jobs per probe and systems where offspring of a job can spawn further offspring (multigenerational multithreading).

REFERENCES

[1] Dario A. Bini, Beatrice Meini, Sergio Steffé, and Benny Van Houdt. 2006. Structured Markov chains solver: software tools. In *Proceeding from the 2006 workshop on Tools for solving structured Markov chains*. 1–14.

[2] M. Bladt, , and B.F. Nielsen. 2017. *Matrix-exponential distributions in applied probability*. Vol. 81. Springer.

[3] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou. 1996. Cilk: An efficient multithreaded runtime system. *Journal of parallel and distributed computing* 37, 1 (1996), 55–69.

[4] Robert D. Blumofe and Charles E. Leiserson. 1999. Scheduling multithreaded computations by work stealing. *Journal of the ACM (JACM)* 46, 5 (1999), 720–748.

[5] Maury Bramson, Yi Lu, and Balaji Prabhakar. 2010. Randomized load balancing with general service time distributions. In *ACM SIGMETRICS 2010*. 275–286. https://doi.org/10.1145/1811039.1811071

[6] Derek L. Eager, Edward D. Lazowska, and John Zahorjan. 1986. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Performance Evaluation* 6, 1 (1986), 53–68.

[7] Matteo Frigo, Charles E. Leiserson, and Keith H. Randall. 1998. The Implementation of the Cilk-5 Multithreaded Language. In *In Proceedings of the SIGPLAN '98 Conference on Program Language Design and Implementation*. 212–223.
[8] Nicolas Gast. 2017. Expected values estimated via mean-field approximation are 1/N-accurate. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 1 (2017), 17.
[9] Nicolas Gast and Bruno Gaujal. 2010. A mean field model of work stealing in large-scale systems. *ACM SIGMETRICS Performance Evaluation Review* 38, 1 (2010), 13–24.
[10] Thierry Gautier, Xavier Besseron, and Laurent Pigeon. 2007. Kaapi: A thread scheduling runtime system for data flow computations on cluster of multi-processors. In *Proceedings of the 2007 international workshop on Parallel symbolic computation*. 15–23.
[11] Grzegorz Kielanski and Benny Van Houdt. 2021. Performance Analysis of Work Stealing Strategies in Large Scale Multi-threaded Computing. In *Quantitative Evaluation of Systems*. Springer International Publishing, Cham, 329–348.
[12] J. Kriege and P. Buchholz. 2014. *PH and MAP Fitting with Aggregated Traffic Traces*. Springer International Publishing, Cham, 1–15. https://doi.org/10.1007/978-3-319-05359-2_1
[13] Guy Latouche and V. Ramaswami. 1999. *Introduction to matrix analytic methods in stochastic modeling*. Vol. 5. SIAM.
[14] Doug Lea. 2000. A Java Fork/Join Framework. In *Proceedings of the ACM 2000 Conference on Java Grande* (San Francisco, California, USA) *(JAVA '00)*. Association for Computing Machinery, New York, NY, USA, 36–43. https://doi.org/10.1145/337449.337465
[15] Daan Leijen, Wolfram Schulte, and Sebastian Burckhardt. 2009. The Design of a Task Parallel Library. In *Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications* (Orlando, Florida, USA) *(OOPSLA '09)*. Association for Computing Machinery, New York, NY, USA, 227–242. https://doi.org/10.1145/1640089.1640106
[16] Wouter Minnebo, Tim Hellemans, and Benny Van Houdt. 2017. On a class of push and pull strategies with single migrations and limited probe rate. *Performance Evaluation* 113 (2017), 42–67.
[17] Wouter Minnebo and Benny Van Houdt. 2014. A fair comparison of pull and push strategies in large distributed networks. *IEEE/ACM Transactions on Networking (TON)* 22, 3 (2014), 996–1006.
[18] Ravi Mirchandaney, Don Towsley, and John A. Stankovic. 1990. Adaptive load sharing in heterogeneous distributed systems. *Journal of parallel and distributed computing* 9, 4 (1990), 331–346.
[19] Marcel F. Neuts. 1981. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. John Hopkins University Press.
[20] A. Panchenko and A. Thümmler. 2007. Efficient Phase-type Fitting with Aggregated Traffic Traces. *Perform. Eval.* 64, 7-8 (Aug. 2007), 629–645. https://doi.org/10.1016/j.peva.2006.09.002
[21] Arch Robison, Michael Voss, and Alexey Kukanov. 2008. Optimization via reflection on work stealing in TBB. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE, 1–8.
[22] Nikki Sonenberg, Grzegorz Kielanski, and Benny Van Houdt. 2021. Performance Analysis of Work Stealing in Large-Scale Multithreaded Computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 6, 2, Article 6 (Sept. 2021), 28 pages. https://doi.org/10.1145/3470887
[23] Ignace Van Spilbeeck and Benny Van Houdt. 2015. Performance of rate-based pull and push strategies in heterogeneous networks. *Performance Evaluation* 91 (2015), 2–15.
[24] Mark S. Squillante and Randolph D. Nelson. 1991. Analysis of Task Migration in Shared-memory Multiprocessor Scheduling. *SIGMETRICS Perform. Eval. Rev.* 19, 1 (1991), 143–155. http://doi.acm.org/10.1145/107972.107987
[25] Benny Van Houdt. 2019. Randomized Work Stealing versus Sharing in Large-scale Systems with Non-exponential Job Sizes. *IEEE/ACM Transactions on Networking* 27 (2019), 2137–2149. Issue 5.
[26] Niklaus Wirth. 1996. Tasks versus Threads: An Alternative Multiprocessing Paradigm. *Software - Concepts and Tools* 17 (01 1996), 6–12.

A PROOF OF PROPOSITION 3.1

The positive recurrence of the QBD process only depends on the matrices $A_{-1}(r)$, $A_0(r)$ and A_1 [19]. These three matrices are the same three matrices as those of the QBD characterizing the M/MAP/1 queue where the MAP service process is characterized by $(S_0(r), S_1(r))$ with $S_0(r) = S(r) - rqI$ and $S_1(r) = \mu\alpha + rqV_0$. As such the QBD process is positive recurrent if and only if the arrival rate λ is less than the service completion intensity of the MAP $(S_0(r), S_1(r))$. This intensity equals $\theta^{(r)} S_1(r)\mathbf{1}/\theta^{(r)}\mathbf{1}$, where the vector $\theta^{(r)}$ is such that $\theta^{(r)}(S_0(r) + S_1(r)) = 0$.

We note that $S_0(r) + S_1(r) = A_{-1}(r) + A_0(r) + A_1 = A(r)$ and define

$$\theta^{(r)}_{(0,1)} = p_{0,1}(r)(-S^c)^{-1}, \quad \theta^{(r)}_{(0,i')} = p_{0,i'}(r)(rqI - S^c)^{-1},$$
$$\theta^{(r)}_{(1,0)} = p_{1,0}(r)(-S^p)^{-1}, \quad \theta^{(r)}_{(1,i)} = p_{1,i}(r)(rqI - S^p)^{-1},$$

for $i' = 2, \ldots, m$ and for $i = 1, \ldots, m$. Define $v^{(r)} = \theta^{(r)} A(r)$. Then

$$v^{(r)}_{(0,i')} = -p_{0,i'}(r) + p_{1,i'}(r)(rqI - S^p)^{-1} s^p \alpha^c + p_{0,i'+1}(r)(rqI - S^c)^{-1} s^c \alpha^c + rq \sum_{j>i'} \psi_{j-1,j-i'} p_{0,j}(r)(rqI - S^c)^{-1} = 0,$$

for $i' = 1, \ldots, m$. By using (5), we further get

$$\begin{aligned} v^{(r)}_{(1,i)} &= -p_{1,i}(r) + p_i (p_{0,1}(r)(-S^c)^{-1} s^c + p_{1,0}(r)(-S^p)^{-1} s^p)\alpha^p + rq \sum_{j>i} \phi_{j,j-i} p_{1,j}(r)(rqI - S^p)^{-1} \\ &= -p_{1,i}(r) + p_i \alpha^p + rq \sum_{j>i} \phi_{j,j-i} p_{1,j}(r)(rqI - S^p)^{-1} = 0, \end{aligned}$$

for $i = 0, \ldots, m$. Hence $\theta^{(r)} A(r) = \theta^{(r)}(S_0(r) + S_1(r)) = 0$. As

$$\begin{aligned} \frac{\theta^{(r)} S_1(r)\mathbf{1}}{\theta^{(r)}\mathbf{1}} &= \frac{1}{\theta^{(r)}\mathbf{1}} \left(p_{0,1}(r)(-S^c)^{-1} s^c + p_{1,0}(r)(-S^p)^{-1} s^p + rq p_{0,1}(r)(-S^c)^{-1} 1_{n_c} + rq p_{1,0}(r)(-S^p)^{-1} 1_{n_p} \right) \\ &\geq \frac{1}{\theta^{(r)}\mathbf{1}} (p_{0,1}(r)(-S^c)^{-1} s^c + p_{1,0}(r)(-S^p)^{-1} s^p) = \frac{1}{\theta^{(r)}\mathbf{1}}, \end{aligned}$$

it suffices that $\lambda < 1/\theta^{(r)}\mathbf{1}$ for the chain to be positive recurrent. For $r = 0$ we have $p_{1,i}(r) = p_i \alpha^p$ and $p_{0,i'}(r) = \sum_{j \geq i'} p_j \alpha^c$, which implies that $\theta^{(0)}\mathbf{1} = \rho/\lambda$. Therefore $\lambda < 1/\theta^{(0)}\mathbf{1}$ is equivalent to demanding that $\rho < 1$. As $\theta^{(r)}\mathbf{1}$ is the mean time between two service completions of the MAP process where the state is reset according to the vector α, we have that $\theta^{(r)}\mathbf{1}$ decreases in r. This completes the proof as $\rho < 1$ implies that $\lambda < 1/\theta^{(0)}\mathbf{1} \leq 1/\theta^{(r)}\mathbf{1}$.