Checkpointing models for tasks with widely different processing times

Paul Ezhilchelvan and Isi Mitrani

School of Computing, Newcastle University, NE4 5TG, UK
e-mail: [paul.ezhilchelvan,isi.mitrani]@ncl.ac.uk

Abstract. A server subject to random breakdowns and repairs offers services to incoming jobs whose lengths are highly variable. A checkpointing policy aiming to protect against possibly lengthy recovery periods is in operation. The problem of how to choose a checkpointing interval in order to optimize performance is addressed by analysing a general queueing model which includes breakdowns, repairs, back-ups and recoveries. Exact solutions are obtained under both Markovian and non-Markovian assumptions. Numerical experiments illustrate the conditions where checkpoints are useful and where they are not, and in the former case, quantify the achievable benefits.

Keywords: Breakdowns, Repairs, Recovery, M/G/1 queue, Embedded Markov chains, Server vacations, Checkpoint optimization

1 Introduction

Checkpointing is an important and useful crash-tolerant technique that involves storing process state during normal operation and restoring the recorded state to speed up recovery after a failure. It is cost-effective compared to hardware redundancy techniques, especially when the storage system for checkpointing data is reliable (Elnozahy et al. [12]). Not surprisingly, many commercial systems and public libraries, such as BlueGene/L (Adiga et al. [1]), IRIX OS (Tuthill et al. [26]) and Unix (Wang et al. [27]), have emerged to provide convenient APIs to facilitate its implementation.

The history of checkpointing goes back more than four decades, to the early days of transaction processing. Traditionally, the technique has involved keeping an 'audit trail' of transactions executed since the last checkpoint. Those transactions would be re-run in the event of a breakdown. The main question of interest would be how to choose the checkpoint frequency so as to minimize some appropriate cost function. In answering that question, the actual lengths of individual transactions were either ignored, or they were all assumed to have the same characteristics (e.g., exponentially distributed with the same mean).

We are interested in studying a checkpointing policy under a more realistic scenario where job processing times are random variables with a large coefficient of variation. That is, most of the jobs requiring service are short, but a few are

very long. Exactly such a pattern of demand has been observed by monitoring a real-life cluster, see Chen et al. [5]. Under those conditions, the purpose of the policy would be to shorten the execution time of the long jobs by reducing the recovery period following a breakdown, without at the same time adding significantly to the processing of the short jobs. These considerations would govern the choice of the checkpoint interval.

The contribution of this paper is to analyse a queue served by a single unreliable server, operating a checkpoint policy in a mixed workload environment. That server is likely to be part of some distributed system. The incoming jobs are typically submitted by a web server. Processing a request may involve one or more database accesses whereby data is cached locally. All checkpoint back-ups take place on a database or possibly on the local disk.

A server breakdown may occur while (i) a job is being served, (ii) a checkpoint is being established, (iii) a recovery from a previous breakdown is in progress, or (iv) the queue is empty and the server is idle. Following a breakdown event, the server does nothing for a random interval which will be referred to as the 'repair period'. After that, in cases (i), (ii), or (iii), it performs a recovery operation consisting of going back to the last checkpoint if there is one, or to the beginning of the job if not, and redoing the work done since then. In case (iv), after being repaired the server takes the first waiting job, if any, and starts a new service.

Note that the repair period may not in fact involve an actual repair or reboot of the server. It may consist of disconnecting the primary server and replacing it with a secondary one that had been kept in reserve and in receipt of checkpoints from the primary (see Güler and Özkasap, [18], Oliveira et al. [23]). As far as the model is concerned, the exact nature of the operation is immaterial; of importance is only the distribution of the resulting inoperative interval.

The start of a job's execution plays the role of an initial checkpoint. Further checkpoints may be inserted at intervals as the run progresses. When the execution is completed, the job departs from the queue and a waiting job, if any, starts service. Thus, the checkpoint policy can be designed so as to leave short jobs largely unaffected, while reducing the run time of long jobs by shortening their recovery periods following a server breakdown.

Such a model has not, to our knowledge, been analyzed before. The objective of the analysis is to determine a performance measure such as the average response time or the average number of jobs present. This will enable the evaluation of the trade-offs between the costs incurred in backing-up the current process state, and the benefits derived from faster recovery operations. We start by solving the model under Markovian assumptions, but later generalize it to allow non-exponential distributions and also multiple servers.

1.1 Related work

Models of checkpointing policies have been studied quite extensively over the years, under a variety of assumptions and application contexts. Research advances on checkpointing have also been 'checkpointed' by surveys at regular intervals. Worth mentioning are the surveys by Chandy [4], Nicola [22], Elnozahy

et al. [12] and Marzouk and Jmaiel [20]. The Elnozahy et. al. survey focusses exclusively on long-running computations, while Chandy mainly surveys checkpointing of streams of short transactions.

A large body of literature deals with long-running computations (sometimes referred to as 'infinite horizon'), motivated by scientific workloads which might typically take hours or days to complete. Those papers are not interested in performance metrics related to customers (e.g., average latency). The optimization criterion is the fraction of time that the server is doing useful work. Examples of such studies are Coffman and Gilbert [6], Liu et al. [19], Grassi et al. [17], Bruno and Coffman [3], Plank and Thomason [24], Subasi et al. [25] and Gelenbe et al. [16] (the last paper also aims to minimize the energy used).

More recently, Dimitriou [9] analysed a model where jobs finding a busy server are not queued, but retry after a random period. Such a policy would not be implemented in a transaction processing system because of its inefficient use of service capacity: the server may remain idle while jobs requiring service are present.

Models involving a queue of jobs have also been studied. Gelenbe [15] derived an expression for the optimal checkpoint interval. Baccelli [2] developed a numerical procedure for computing the average response time, while Dohi et al. [10] generalized the checkpoint policy by making it age-dependent. All those authors obtained their results by assuming that during operative periods the system behaves like an M/M/1 queue. Instead of an implicit checkpoint at the start of each job, an audit trail is maintained, keeping track of the jobs that would have to be re-run in the event of a breakdown. The consequence of that trail should be that results cannot be released to users, and jobs must be kept in the queue, until the next checkpoint is successfully established.

However, that is not what happens in the above models. Jobs are assumed to depart from the queue as soon as their service is completed. The recovery following a breakdown is simply a period during which jobs continue to arrive but none are served. The duration of that period is a linear function of the operative time elapsed since the last checkpoint.

In our model, departures upon service completion are justified by the fact that a breakdown only affects the job currently served, not the ones already completed.

The analysis of the Markovian model is described in Section 2, and its exact solution is presented in Section 3. The exact solutions when the back-up, checkpoint and repair intervals have general distributions, and the approximation for non-exponential intervals between breakdowns are described in Section 4. Several numerical experiments exploring the behaviour of the system for different parameter settings, under both Markovian and non-Markovian assumptions are shown in Section 5. These include an evaluation of the maximum achievable benefit of checkpointing.

2 Analysis under Markovian assumptions

The server goes through alternating periods of being operative and broken (or available and unavailable). These are distributed exponentially with means $1/\xi$ and $1/\eta$, respectively. Jobs arrive in a Poisson stream with rate λ. The required service times have a Hyperexponential distribution with K exponential phases, where phase k is entered with probability q_k and has an average of $1/\mu_k$ ($k = 1, 2, \ldots, K$). After completing the chosen exponential phase, the job departs.

A Hyperexponential distribution with K phases can be used to model jobs of K different types of the kind observed in [5]. Its coefficient of variation is always greater than or equal to 1, and can be arbitrarily large. For example, using just two Hyperexponential phases, with q_2 and μ_2 much smaller than q_1 and μ_1, respectively, one can model patterns of demand where most of the jobs are short and a few are very long.

While being served, a job sets up periodic checkpoints at random intervals. At the start of its service phase, or after a checkpoint has been established, a timer distributed exponentially with mean $1/\alpha$ is started. If that timer expires before the phase completes, a new checkpoint is attempted. The establishment of a checkpoint is not instantaneous but requires an exponentially distributed interval of time with mean $1/\beta$. That interval will be referred to as the 'back-up' operation.

Both the service intervals and the back-up operations may be interrupted by a server breakdown. Bearing in mind that the shortest of several exponentially distributed random variables is distributed exponentially with parameter equal to the sum of the parameters of the participating variables, we conclude that any service interval during phase k is distributed exponentially with parameter ν_k, given by

$$\nu_k = \mu_k + \alpha + \xi \; . \tag{1}$$

The end point of such an interval is either a service completion, with probability μ_k/ν_k, or a checkpointing attempt, with probability α/ν_k, or a server breakdown, with probability ξ/ν_k.

Similarly, any back-up operation in any phase is distributed exponentially with parameter σ, given by

$$\sigma = \beta + \xi \; . \tag{2}$$

Such an operation terminates with either the successful establishment of a checkpoint, with probability β/σ, or a server breakdown, with probability ξ/σ.

If the server breaks down during a service interval or during a following back-up operation, the work performed since the last checkpoint, or in the absence of a checkpoint since the beginning of the phase, must be repeated when the server is repaired. This is referred to as the 'recovery' operation. According to the above observations, the recovery operations in phase k are distributed exponentially with parameter ν_k. Of course, the server may break down again during a recovery, in which case another recovery (depending on the phase) is started after the repair.

We assume that each recovery operation is a new sample from the appropriate distribution. That assumption is motivated by the fact that different runs of the same task never take exactly the same time, particularly in a multi-core environment.

Note that the action taken after the server is repaired following a breakdown depends on whether the breakdown occurred while the server was idle, or whether it occurred while the server was active (i.e., serving a job, backing-up or recovering). In the former case there is no need for a recovery: either the server is again idle, or a job has arrived in the meantime and a new service begins. In the latter case, a new recovery starts, whose duration depends on the phase that was in progress when the breakdown occurred.

Denote by T the random variable representing the total period between the start of a job's service and its completion. That period includes service intervals and back-up operations, as well as repair times and recovery operations following any breakdowns. The interval T will be referred to as the 'effective service time'. We shall need the first and second moments of that interval. In particular, the necessary and sufficient condition for stability of the system is that the offered load generated by the effective service times of the incoming jobs should be less than 1:

$$\lambda E(T) < 1 \ . \tag{3}$$

Let X_k be the time a job takes to complete phase k. The moments of the effective service time are simply expressed in terms of the moments of X_k:

$$E(T) = \sum_{k=1}^{K} q_k E(X_k) \ , \tag{4}$$

and

$$E(T^2) = \sum_{k=1}^{K} q_k E(X_k^2) \ . \tag{5}$$

3 Exact solution

To determine the moments of the effective service time, we shall introduce two sets of auxiliary random variables. Define Y_k as the interval between attempting to establish a checkpoint during phase k, and resuming phase k service. That interval may include repairs and recovery operations resulting from breakdowns during the back-up operation. Also define V_k as the interval between a breakdown in phase k and resuming phase k service. It includes the repair and the recovery operation, plus any additional repairs and recoveries caused by further breakdowns. Denote by $y_k(s)$ and $v_k(s)$ the Laplace transforms of the Y_k and V_k probability density functions, respectively.

The Laplace transform of an exponential p.d.f. with some parameter, γ, is equal to $\gamma/(\gamma+s)$. Also, the transform of a sum of independent random variables

is equal to the product of their transforms. Hence, we can write the following equation for $v_k(s)$:

$$v_k(s) = \frac{\eta}{\eta + s} \frac{\nu_k + \xi}{\nu_k + \xi + s} \left[\frac{\nu_k}{\nu_k + \xi} + \frac{\xi}{\nu_k + \xi} v_k(s) \right] , \tag{6}$$

where ν_k is given by (1). The first term in the square brackets is the probability that the recovery operation completes without interruption; the second term contains the probability that another breakdown occurs and a new random V_k is started.

The first and second moments of V_k are obtained from $E(V_k) = -v_k'(0)$ and $E(V_k^2) = v_k''(0)$. Differentiating (6) twice at $s = 0$ yields, after some algebra,

$$E(V_k) = \frac{\nu_k + \xi + \eta}{\nu_k \eta} , \tag{7}$$

and

$$E(V_k^2) = 2 \left[E(V_k)^2 - \frac{1}{\nu_k \eta} \right] . \tag{8}$$

Since the interval Y_k terminates either as a successful back-up, or is interrupted by a breakdown and is followed by a recovery operation V_k, we can express $y_k(s)$ in terms of $v_k(s)$:

$$y_k(s) = \frac{\sigma}{\sigma + s} \left[\frac{\beta}{\sigma} + \frac{\xi}{\sigma} v_k(s) \right] , \tag{9}$$

where σ is given by (2). The first and second moments of Y_k are given by

$$E(Y_k) = \frac{1}{\sigma} [1 + \xi E(V_k)] , \tag{10}$$

and

$$E(Y_k^2) = \frac{2}{\sigma^2} [1 + \xi E(V_k)] + \frac{\xi}{\sigma} E(V_k^2) . \tag{11}$$

Now remember that the phase k execution, X_k, consists of a service interval which either terminates uninterrupted, or is interrupted by a back-up operation, Y_k, and later resumed, or is interrupted by a breakdown interval, V_k, and later resumed. This leads to the following equaton for the Laplace transform of X_k, $x_k(s)$.

$$x_k(s) = \frac{\nu_k}{\nu_k + s} \left[\frac{\mu_k}{\nu_k} + \frac{\alpha}{\nu_k} y_k(s) x_k(s) + \frac{\xi}{\nu_k} v_k(s) x_k(s) \right] . \tag{12}$$

Differentiating twice at $s = 0$ and substituting the moments of Y_k and V_k already derived, we obtain the first and second moments of X_k:

$$E(X_k) = \frac{1}{\mu_k} [1 + \alpha E(Y_k) + \xi E(V_k)] , \tag{13}$$

and

$$E(X_k^2) = 2E(X_k)^2 + \frac{1}{\mu_k} [1 + \alpha E(Y_k^2) + \xi E(V_k^2)] . \tag{14}$$

Using expressions (10) and (7), the average execution time of phase k can be rewritten as

$$E(X_k) = \frac{1}{\mu_k} \frac{\xi+\eta}{\eta} \frac{\alpha+\sigma}{\sigma} \frac{\nu_k+\xi}{\nu_k} \ , \tag{15}$$

The ergodicity condition (3) can now be stated explicitly:

$$\sum_{k=1}^{K} q_k \rho_k \frac{\nu_k+\xi}{\nu_k} < \frac{\eta}{\xi+\eta} \frac{\sigma}{\alpha+\sigma} \ , \tag{16}$$

where $\rho_k = \lambda/\mu_k$. Note that when $\alpha = 0$ and $\xi = 0$, i.e. when there are no checkpoints and no breakdowns, this reduces to the usual stability condition for the M/G/1 queue: the offered load must be less than 1.

The exact solution for our model cannot be obtained by treating it as a simple M/G/1 queue. This is because of the possibility that the server may break down while the queue is empty. If that happens, a job may arrive into the system, find an empty queue, yet be unable to start service immediately. This situation is similar to what is known in the literature as a 'server of walking type', or 'server with vacations', where a server encountering an empty queue goes away for a random period (e.g., see [13]). In those studies, vacations are always taken, and are independent of the arrival process. Our model is different, in that the server only takes a vacation (a repair period) if a breakdown occurs while the queue is empty.

For an exact analysis, we shall consider the number of jobs present at (just after) consecutive departure instants. This is a discrete time Markov chain. The following notation will be used:

π_i is the steady-state probability that there are i jobs left in the system after a departure ($i = 0, 1, \ldots$). This is also the probability that an incoming job would see i jobs present. Hence, by the PASTA property, it is also the probability that a random observer would see i jobs present.

a_i is the probability that i jobs arrive during an effective service time, T;

b_i is the probability that i jobs arrive during a repair period;

We shall also introduce the generating functions

$$\pi(z) = \sum_{i=0}^{\infty} \pi_i z^i \ ; \ \ a(z) = \sum_{i=0}^{\infty} a_i z^i \ ; \ \ b(z) = \sum_{i=0}^{\infty} b_i z^i \ . \tag{17}$$

If $i > 0$ and k jobs arrive during the ensuing service time ($k = 0, 1, \ldots$), then the next state will be $i + k - 1$. If $i = 0$, then the next state will also depend on whether the idle period is interrupted by a breakdown or not. Transforming the relevant probabilities and balance equations into generating functions, we obtain, after some work, the following equation for $\pi(z)$ in terms of $a(z)$ and $b(z)$:

$$\pi(z) = \frac{(1-\rho)a(z)(z-1)}{z-a(z)} \left[\frac{\eta}{\xi+\eta} + \frac{\xi}{\xi+\eta} \frac{\eta}{\lambda+\eta} \frac{zb(z)-1}{z-1} \right] \ . \tag{18}$$

This representation is significant, because it expresses $\pi(z)$ as a product of two generating functions. The first fraction in the right-hand side of (18) is the well-known generating function of the steady-state distribution of the M/G/1 queue (e.g., see [7]). The term in the square brackets is the generating function of a random variable which we shall call U; it is related to the number of arrivals during a repair period. That product form implies the following result:

Lemma. *The number of jobs present in our system in the steady state is distributed as the sum of the number of jobs in the corresponding M/G/1 queue (arrival rate λ and service time T), plus the random variable U.*

Similar results can be found in the literature related to models of servers with vacations.

In particular, denoting by L and $L_{M/G/1}$ the steady-state average numbers of jobs in our system and in the corresponding M/G/1 queue respectively, we can write

$$L = L_{M/G/1} + E(U) \; . \tag{19}$$

When the repair periods are distributed exponentially, the distribution b_i is geometric with parameter $\lambda/(\lambda+\eta)$. The generating function $b(z)$ is

$$b(z) = \frac{\eta}{\lambda(1-z)+\eta} \; . \tag{20}$$

Substituting this into the generating function of U and differentiating at $z = 1$, we obtain

$$E(U) = \frac{\lambda\xi}{\eta(\xi+\eta)} \; . \tag{21}$$

This is quite an intuitive expression: it represents the average number of arrivals during a repair period (λ/η), multiplied by the fraction of time during which the server is broken.

Equation (19), together with Pollaczek-Khinchin's result (e.g., see [21]), now provides the average L in terms of quantities that have already been derived:

$$L = \lambda E(T) + \frac{\lambda^2 E(T^2)}{2(1-\lambda E(T))} + \frac{\lambda\xi}{\eta(\xi+\eta)} \; , \tag{22}$$

where $E(T)$ and $E(T^2)$ are given by (4) and (5).

This completes the exact solution of the model under Markovian assumptions. It is worth mentioning that other phase-type distributions of the required service times can be handled by the same methods. For example, in some applications it may be more convenient to assume a Coxian distribution with K exponential phases, where phase k has an average of $1/\mu_k$ and is followed by phase $k+1$ with probability q_k. After completing phase K, the job departs. A three-phase Coxian distribution is illustrated in Figure 1.

The class of Coxian distributions is, for most purposes, general (see [8]), and can also model mixed job populations.

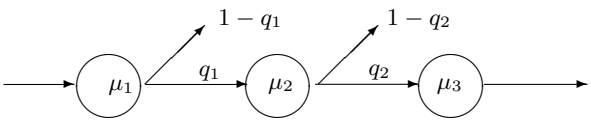

Fig. 1. A Coxian distribution with three phases

Let r_k be the probability that a job reaches phase k during the course of its service:

$$r_1 = 1 \;\; ; \;\; r_k = \prod_{i=1}^{k-1} q_i \;\; ; \;\; k = 2, 3, \ldots, K \; . \tag{23}$$

Then the moments of the effective service time, T, are obtained from

$$E(T) = \sum_{k=1}^{K} r_k E(X_k) \; , \tag{24}$$

and

$$E(T^2) = \sum_{k=1}^{K} r_k E(X_k^2) + 2 \sum_{k=2}^{K} r_k \sum_{i=1}^{k} E(X_i) E(X_k) \; . \tag{25}$$

The analysis presented above would apply, provided that each new phase begins with a checkpoint.

4 Generalizations

It is possible, and desirable, to relax a number of the assumptions that have been made so far. In the majority of cases this can be done while retaining the exact nature of the solution. However, one of the generalizations appears to be intractable and requires an approximation.

4.1 General repair, backup and checkpoint intervals

Suppose that the repair period, R, has a general distribution, with Laplace transform $\eta(s)$. The exact analysis can proceed largely as before, subject to two important changes. First, equation (6) becomes

$$v_k(s) = \eta(s) \frac{\nu_k + \xi}{\nu_k + \xi + s} \left[\frac{\nu_k}{\nu_k + \xi} + \frac{\xi}{\nu_k + \xi} v_k(s) \right] \; . \tag{26}$$

Given the first two moments of the repair time, $E(R)$ and $E(R^2)$, this expression enables us to evaluate $E(V_k)$ and $E(V_k^2)$, and hence proceed to determine $E(X_k)$ and $E(X_k^2)$.

The second change is that the argument leading to (22) should now be generalized to include random vacations with general distribution. The analysis of

the previous section can be modified to produce a generalized form of expression (18), involving the Laplace Transform of the Random Modification of the repair period (i.e. the remaining duration of the repair period, as seen by a random observer). The resulting quantity $E(U)$, which appears in (19) and (21) now contains the first and second moments of the repair time:

$$E(U) = \lambda \frac{E(R^2)}{2} \frac{\xi}{1 + \xi E(R)} \ . \tag{27}$$

Similar arguments can be employed when the back-up times and the checkpoint intervals have general distributions with Laplace transforms $b(s)$ and $a(s)$, respectively. They lead to a generalized form of equation (12), from which the moments $E(X_k)$ and $E(X_k^2)$ are determined by differentiation at $s = 0$. The details of those developments are omitted for lack of space.

4.2 Approximation for general breakdown intervals

Relaxing the Markovian character of the breakdown process, while still computing an exact solution, appears to be considerably more difficult. We are therefore proposing a simple approximation that can be readily justified in a practical situation. The argument is based on the fact that breakdowns are rare events.

Assume that the operative periods of the server, i.e. the intervals between completing a repair and the next breakdown, are i.i.d. random variables with an m-phase Coxian distribution. That is sufficiently general for practical purposes. By choosing the number of phases and their parameters appropriately, one can approximate most commonly used distributions, such as Erlang, Weibull, Hyperexponential, Lognormal, Normal, etc.

Denote by ξ_i the parameter of phase i ($i = 1, 2, \ldots, m$), by $\bar{q}_i$ the probability of moving from phase $i-1$ to phase i, and by $\bar{r}_i = \bar{q}_1 \bar{q}_2 \cdots \bar{q}_{i-1}$ the probability of reaching phase i.

It is reasonable to assume that the parameters ξ_i are small compared to the other parameters (i.e. the average lengths of all phases are large). In that case, the queueing process can be assumed to be stable and reach steady state during each phase. The approximate solution of the generalized model would then consist of the following steps.

1. Compute the steady-state probabilities, γ_i, that an operative server is in phase i of its breakdown interval. These probabilities satisfy the balance equations

$$\gamma_i \xi_i = \gamma_{i-1} \xi_{i-1} \bar{q}_{i-1} \ \ ; \ \ i > 1 \ . \tag{28}$$

They can therefore be expressed, after normalization, as

$$\gamma_i = \frac{\xi_1 \bar{r}_i}{\xi_i} \left[\sum_{j=1}^{m} \frac{\xi_1 \bar{r}_j}{\xi_j} \right]^{-1} \ \ ; \ \ i = 1, 2, \ldots, m \ . \tag{29}$$

2. Apply the existing exact solution to phase i, using $\xi = (1 - \bar{q}_i)\xi_i$ as the breakdown rate, and compute the average number of jobs present, L_i.
3. Compute the overall average number of jobs present, L, as a weighted average over all phases:

$$L = \sum_{i=1}^{m} \gamma_i L_i \ . \tag{30}$$

Remark. The above approximation, known as 'decomposition', has been used in a variety of other contexts. A quantitative evaluation of its accuracy for our system may be provided by simulations. However, as the breakdowns are rare events, that would be a non-trivial exercise, possibly requiring special techniques. Such an undertaking is outside the scope of the present work.

5 Numerical results

The exact solutions of Sections 3 and 4 were applied to several example systems, with the aim of examining the trade-offs between the costs and benefits of checkpointing. In order to reduce the parameter space to be explored, some of the parameters are kept fixed. The required service times are assumed to have a two-phase Hyperexponential distribution with quite a large coefficient of variation: $1/\mu_1 = 40$; $1/\mu_2 = 400$; $q = 0.2$. In other words, the average requirement of 80% of the incoming jobs is 40, and for the other 20% it is 400. These parameters are chosen to conform, as far as it was possible to extract average values from the reported statistics, with the data collected in [5]. The average repair period is assumed to be relatively short, $1/\eta = 15$. The arrival rate λ, the checkpointing rate, α, the average back-up interval. $1/\beta$ and the breakdown rate, ξ, are varied. The performance measure in all cases is the average number of jobs present, L. If we wished to evaluate the the average response tme, W, we would use Little's result to compute $W = L/\lambda$.

In the first example, the arrival rate is set to $\lambda = 0.0065$, and L is plotted against α, for three different values of the breakdown rate: $\xi = 10^{-4.5}$, $\xi = 10^{-4}$ and $\xi = 10^{-3.5}$. These rates are comparable to the ones reported in Garraghan et al. [14].

With these parameters, the system load, as measured by $\lambda E(T)$, is on the order of 75% or higher.

The results are shown in Figure 2. All three plots exhibit a steep initial decline in occupancy, followed by a slow increase. As the breakdown rate increases, the optimal α also increases slightly; it is about 0.1 when $\xi = 10^{-4.5}$, between 0.1 and 0.2 when $\xi = 10^{-4}$ and close to 0.3 when $\xi = 10^{-3.5}$. Such an increase could have been expected, since a more likely breakdown during service makes the backing-up of the current state more advantageous. The low gradient of the plateau following the optimal point is explained by the low cost of the back-up operation. Of course, if α carries on increasing, so that the 'non-productive' back-up operations push the queue closer to saturation, the increase in occupancy would accelerate without bound.

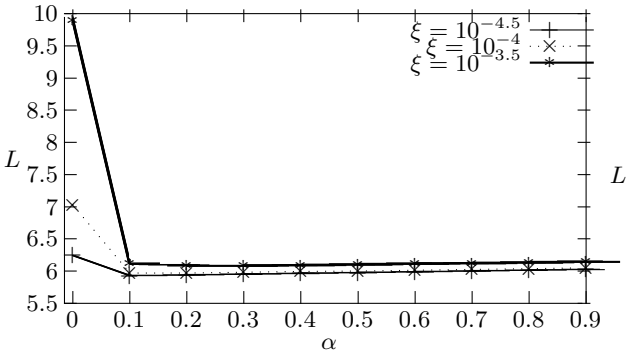

Fig. 2. $\lambda = 0.0065$, different ξ

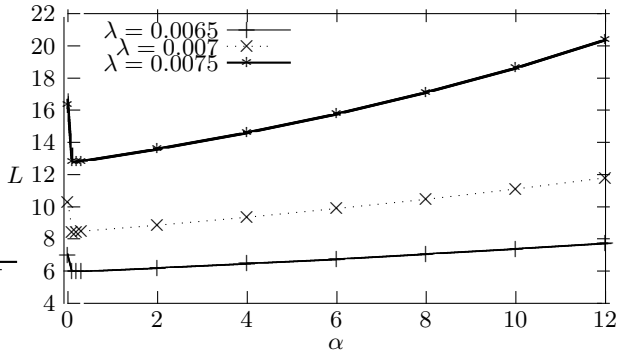

Fig. 3. $\xi = 10^{-4}$, different λ.

This phenomenon is illustrated in the second experiment, where the breakdown rate is kept fixed at $\xi = 10^{-4}$, while the arrival rate takes three different values: $\lambda = 0.0065$, $\lambda = 0.007$ and $\lambda = 0.0075$. The results are shown in Figure 3. Again, the plots of L against α display a steeply decreasing portion, followed by an increasing one.

For all three values of λ, the optimal checkpointing rate is between $\alpha = 0.1$ and $\alpha = 0.2$. However, having extended the range of α considerably beyond the optimum, we observe the non-linear increase in L caused by the back-up operations. This is particularly noticeable in the case of $\lambda = 0.0075$, where the system was quite heavily loaded to start with.

It is intuitively clear that the more reliable the server, the less frequent need be the checkpoints. On the other hand, the less time it takes to perform a back-up operation, the more checkpoints can be afforded. In order to quantify these observations, we have evaluated the optimal checkpointing rate, α^*, as a function of the average interval between breakdowns, $1/\xi$. This was done for three different values of the back-up rate, $\beta = 100$, $\beta = 200$ and $\beta = 400$. The results are presented in Figure 4, where $1/\xi$ is scaled exponentially. At each point, the optimal α^* was found by carrying out a search.

As expected, the optimal checkpointing rate eventually becomes zero when the server becomes sufficiently reliable. However, that point is not reached quickly: when $\beta = 100$ or $\beta = 200$, the mean time between failures needs to be about 10^7 before checkpoints become counterproductive. That interval increases to 10^8 for $\beta = 400$. At the other end of the range, when breakdowns are relatively frequent, the optimal checkpointing rate is higher and increases with the speed of the back-up operations.

In order to examine the effect that a change of distributions has on performance, we have repeated the experiment of Figure 4, under the assumptions that the repair, back-up and checkpoint intervals are constant, keeping the means, $E(R)$, $E(B)$ and $E(A)$, as before. Clearly, such a change would reduce the variance of the effective service time and hence would reduce the average number L.

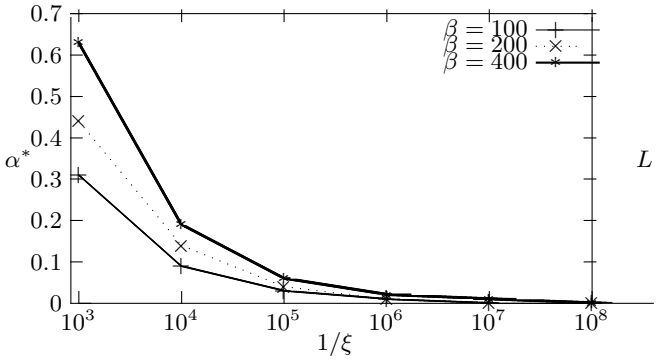

Fig. 4. Optimal α

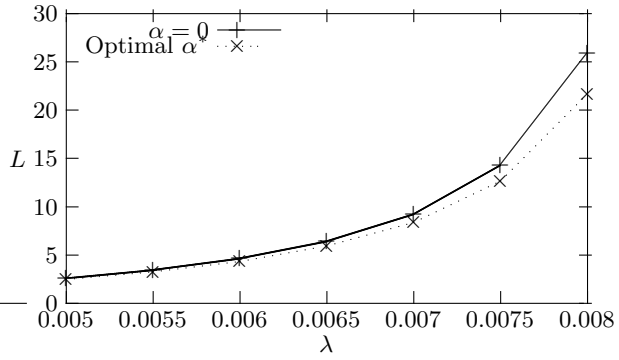

Fig. 5. Benefits of checkpointing

However, it is not obvious whether the optimal value of the checkpoint frequency, $\alpha = 1/E(A)$, is affected and if so, to what extent.

We observed a similar behaviour to the one exhibited in Figure 4: as the server becomes more reliable, the optimal checkpoint frequency decreases and eventually becomes 0 (i.e. checkpoints are no longer needed).

The notable difference in these observations is that the values of α^* were in all cases significantly lower than before. That outcome can be explained intuitively by arguing that a deterministic checkpointing policy is more effective than a random one with the same averages. Hence, one can achieve the desired result with fewer checkpoints.

Because of the similarity of behaviours, it was not deemed necessary to include another version of the figure.

The final experiment aims to quantify the gains achievable by a checkpointing policy. This is done by comparing the average number of jobs present, L, when no checkpoints are used (i.e. $\alpha = 0$ or $a = \infty$), with the number present when the policy uses the optimal checkpoint frequency α^*. The latter value is determined by a search. The other parameters are as before, with $\xi = 10^{-4}$. The repair, back-up and checkpoint intervals are constant.

In Figure 5, the unoptimized and optimized values of L are plotted against the arrival rate, which varies from $\lambda = 0.005$ to $\lambda = 0.008$. On that range, the offered load, λT, varies from about 55% to about 90%.

As might have been expected, when the system is lightly loaded, the gains of checkpointing are slight. Jobs affected by breakdowns do not tend to delay other jobs at light loads, because the queue is short. Hence, any savings in their effective service times show limited benefits. As the load increases, the queue gets longer and the savings are noticed by more waiting jobs. At the 90% load, the difference between no checkpointing and optimal checkpointing is about 25%.

We carried out the same comparison under the assumption that the A, B and R random variables are distributed exponentially. The results were very similar to the ones presented in Figure 5 and are therefore omitted.

6 Conclusions

We have examined the effects of checkpointing on performance by analysing a rather general queueing model involving breakdowns, repairs and back-up operations. A major objective of the study was to handle a job population with a large variability of required service times. Exact solutions were obtained under both Markovian and non-Markovian assumptions. These were used in order to determine the optimal checkpoint frequency for different parameter settings, and to quantify the benefits of checkpointing.

One of the proposed generalizations, to non-exponential intervals between breakdowns, involves an approximate solution. As indicated in the Remark at the end of subsection 4.2, assessing the accuracy of that approximations is outside the scope of the present paper but would be a worthy topic for future research.

Another interesting generalization that would be worth studying is to divide the mixed job population into several classes, with a separate queue for each class. A non-preemptive priority scheduling policy could be in operation among the classes. For example, short jobs might be given higher priority than long ones. It may be possible to generalize existing results on multi-class M/G/1 queues to such a system. To obtain an exact solution, it would be necessary to analyse the effect that breakdowns during idle periods have on the behaviour of queues. That too would be a worthy topic for future research.

References

1. N.R. Adiga et al., "An Overview of the BlueGene/L Supercomputer", Procs. ACM/IEEE Conf. on Supercomputing, pp. 60-60, 2002. doi: 10.1109/SC.2002.10017.
2. F. Baccelli, "Analysis of a Service Facility with Periodic Checkpointing", Acta lnformatica, 15, pp. 67-81, 1981.
3. J.L. Bruno and E.G. Coffman, "Optimal Fault-Tolerant Computing on Multi-Processor Systems", Acta Informatica, 34, pp. 881-904, 1997.
4. K.M. Chandy, "A Survey of Analytic Models of Rollback and Recovery Strategies", Computer, 8, 5, pp. 40-47, 1975.
5. Y. Chen, A.S. Ganapathi, R. Griffith and R.H. Katz, "Analysis and Lessons from a Publicly Available Google Cluster Trace", Technical Report No. UCB/EECS-2010-95, 2010.
6. E.G. Coffman and E.N. Gilbert, "Optimal strategies for scheduling checkpoints and preventive maintenance", IEEE Trans. on Reliability, 39, 1, pp. 9-18, 1990.
7. J.W. Cohen, *The Single Server Queue*, North-Holland, Amsterdam, 1969.
8. D.R. Cox, "A use of complex probabilities in the theory of stochastic processes", Mathematical Proceedings of the Cambridge Philosophical Society, 51, 2, pp. 313-319, 1955.
9. I. Dimitriou, "A retrial queue for modeling fault-tolerant systems with checkpointing and rollback recovery", Computers & Industrial Engineering, 79, pp. 156–167, 2015.
10. T. Dohi, N. Kaio and K.S. Trivedi, "Availability models with age-dependent checkpointing", 21st IEEE Symp. on Reliable Distributed Systems, pp. 130-139, 2002.

11. N. Duhan, A.K. Sharma and K.K. Bhatia, "Page Ranking Algorithms: A Survey", IEEE International Advance Computing Conference, pp. 1530-1537, 2009. doi: 10.1109/IADCC.2009.4809246.
12. E.N. Elnozahy, L. Alvisi, Y. Wang and D.B. Johnson, "A survey of rollback-recovery protocols in message-passing systems", ACM Computing Surveys, 34, 3, pp. 375–408, 2002.
13. S.W. Fuhrmann, "A Note on the M/G/1 Queue with Server Vacations", Operations Research, 32, 6 , pp. 1368-1373, 1984.
14. P. Garraghan, P. Townend and and J. Xu, "An Empirical Failure-Analysis of a Large-Scale Cloud Computing Environment", 15th Int. Symp. on High-Assurance Systems Engineering, pp.113-120, 2014.
15. E. Gelenbe. "On the Optimum Checkpoint Interval", Journal of the ACM, 26, 2, pp. 259-270, 1979.
16. E. Gelenbe, P. Boryszko, M. Siavvas and J. Domanska, "Optimum Checkpoints for Time and Energy", 28th IEEE Symp.on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp.1-8, 2020.
17. V. Grassi, L. Donatiello and S. Tucci, "On the optimal checkpointing of critical tasks and transaction-oriented systems", IEEE Trans. on Software Engineering, 18, 1, pp. 72-77, 1992.
18. B. Güler and Ö. Özkasap, "Efficient checkpointing mechanisms for primary-backup replication on the cloud", Concurrency and Computation: Practice and Experience, 30, 21, 2018.
19. Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun and S.L. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system", IEEE Symp. on Parallel and Distributed Processing, pp. 1-9, 2008.
20. S. Marzouk and M. Jmaiel, "A survey on Software Checkpointing and Mobility Techniques in Distributed Systems", Concurrency and Computation: Practice and Experience, 23, 11, pp. 1196-1212, 2011.
21. I. Mitrani, *Probabilistic Modelling*, Cambridge University Press, 1998.
22. V.F. Nicola, "Checkpointing and the Modelling of Program Execution Time", Software Fault Tolerance, M.R. Lyu, ed., pp. 167-188, John Wiley & Sons, 1995.
23. R. Oliveira, J. Pereira and A. Schiper, "Primary-backup replication: from a time-free protocol to a time-based implementation", Procs. 20th IEEE Symp. on Reliable Distributed Systems, pp. 14-23, 2001.
24. J.S. Plank and M.G. Thomason, "Processor allocation and checkpoint interval selection in cluster computing systems", Journal of Parallel and distributed Computing, 61, 11, pp. 1570-1590, 2001.
25. O. Subasi, G. Kestor and S. Krishnamoorthy, "Toward a General Theory of Optimal Checkpoint Placement", IEEE Conf. on Cluster Computing (CLUSTER), pp. 464-474, 2017.
26. B. Tuthill, K. Johnson and T. Schultz, "Irix checkpoint and restart operation guide", Document of Silicon Graphics, Inc, 1999.
27. Y.-M. Wang, Y. Huang, K.-Ph. Vo, P.-Y. Chung and C. Kintala, "Checkpointing and its applications" 25th Int. Symp. on Fault-Tolerant Computing. Digest of Papers, pp. 22-31, 1995.
28. W. Whitt, "Approximations for the GI/G/m queue", Production and Operations Management, 2, 2, pp. 114-161, 1993.