On redundancy-d with cancel-on-start a.k.a Join-shortest-work (d)

Urtzi Ayesta[1], Tejas Bodas [2] and Ina Maria Verloop[1]

[1]CNRS, IRIT, 2 rue C. Camichel, 31071 Toulouse, France

[2]CNRS, LAAS, 7 avenue du colonel Roche, 31400 Toulouse, France

1. INTRODUCTION

Using redundancy to minimize latency in parallel server systems has become very popular in recent years [1–6]. While there are several variants of a redundancy-based system, the general notion of redundancy is to create multiple copies of the same job that will be sent to a subset of servers. By allowing for redundant copies, the aim is to minimize the system latency by exploiting the variability in the queue lengths of the different queues. Several recent works, both empirically [1,2] and theoretically [3–6], have provided indications that redundancy can help in reducing the response time of a system.

Broadly speaking, depending on when replicas are deleted, we can consider two classes of redundancy systems: cancel-on-start (*c.o.s*) and cancel-on-completion (*c.o.c*). In redundancy systems with *c.o.c*, once one of the copies has completed service, the other copies are deleted and the job is said to have received service. On the other hand, in redundancy systems with *c.o.s*, copies are deleted as soon as one copy starts being served. In a recent series of papers, Gardner et al. [5, 6] have provided a thorough analysis of redundancy systems with *c.o.c* in a system with K servers each with their own queue. In the redundancy-d model of [6], redundant copies of an arriving job are sent to $d \leq K$ homogeneous servers chosen uniformly at random. Most of the recent literatures have focused on systems with *c.o.c*, and *c.o.s*. has remained largely elusive to exact analysis. From a practical point of view, if servers have similar computing capacity, the *c.o.s* system is preferable. Both configurations require the same amount of signaling among servers, but the *c.o.s* system does not waste any computation resources.

In this paper, we provide, to the best of our knowledge, the first analysis on a redundancy-d system with *c.o.s*. for $1 \leq d \leq K$. We assume exponentially distributed service times, but copies do not need to be i.i.d. To analyze redundancy-d system with *c.o.s.*, we use the same state space representation as that of the multi-type job and server model of Visschers et al. [7]. This allows us to conclude that the steady-state distribution of *c.o.s* is of product-form. We then obtain an expression for the probability generating function (PGF) of the number of waiting jobs in the system and for the mean number of jobs in the system. We finally show that redundancy-d with *c.o.s* is equivalent to Join-Shortest-Work queue with redundancy (JSW(d)). As a result, performance measures obtained for the *c.o.s* model carry over to JSW(d).

*Corresponding author: tejaspbodas@gmail.com.

Research partially supported by the French "Agence Nationale de la Recherche (ANR)" through the project ANR-15-CE25-0004 (ANR JCJC RACON)

Copyright is held by author/owner(s).

2. MODEL DESCRIPTION

The redundancy-d model with *c.o.s* consists of K homogeneous servers each with a first-in first-out (FIFO) queue. The service rate of each server is denoted by μ. Let $\mathcal{M} = \{m_1, \ldots, m_K\}$ denote the set of servers. Jobs arrive according to a Poisson process with rate λ and have an exponentially distributed service requirement with unit mean. An arriving job chooses d of K servers uniformly at random and sends d copies of the same job to these *feasible* servers. We will say that all jobs that choose the same d servers are said to be of the same type. In all, there are $\binom{K}{d}$ job types and the arrival rate of any job type is $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$. Let $\mathcal{C}$ denote the set of job types. Once any of the copies is taken for service, the remaining copies are canceled immediately. Further, on arrival of a job, if more than one of its d *feasible* servers are idle, then the job is served immediately at one of the idle *feasible* servers based on a *uniform assignment rule* (choose an idle *feasible* server uniformly at random) and the remaining copies are canceled. The total system load is defined as $\rho = \frac{\lambda}{K\mu}$ and we assume $\rho < 1$. For both *c.o.s.* and *c.o.c.*, $\rho < 1$ is also the stability condition.

An important point to note is that the *c.o.s.* redundancy-d model has a *central queue architecture* (*c.q.a.* for short). This means that the parallel server system can equivalently be represented by a single central queue with multiple servers choosing (multi-type) jobs from the central queue. Refer to [5] where the *c.q.a.* was first noted for the *c.o.c.* model. In order to analyze the redundancy-d model with *c.o.s.*, we use the *c.q.a.* which now allows us to view our model as a special case of the multi-type job and server model [7]. See technical report [8] for details.

State space representation for *c.o.s.*

A Markovian descriptor introduced in [7] which is appropriate for the redundancy-d *c.o.s.* system is of the type $(n_i, M_i, n_{i-1}, M_{i-1}, \ldots, n_1, M_1)$ which denotes states with i busy servers (denoted by $M_1, \ldots, M_i$) and n_j waiting jobs between servers M_j and M_{j+1} for $1 \leq j \leq i-1$. In this state

space representation, waiting jobs and active servers are arranged in a FIFO basis from right to left. Therefore all n_j jobs have arrived before n_k jobs where $1 \leq j < k \leq i$. Note that a job is waiting in the central queue only if all of its *feasible* servers are busy (serving jobs that came before it). Therefore n_1 denotes the number of those jobs (who have arrived before n_j jobs where $j > 1$) that have to wait since they have server M_1 as their only *feasible* server which happens to be busy. Similarly, n_j represent jobs (that arrived after n_{j-1} but before n_{j+1} jobs) that have to wait because their feasible servers are busy. The feasible servers for these n_j jobs must clearly be a subset of the active servers $\{M_1, M_2, \ldots, M_j\}$ ahead of it (otherwise the job would not have been waiting if any of its feasible server was idle). (See [7] for precise details.) An important point to note is that for redundancy-d *c.o.s.*, since each job type has d feasible servers, it can never happen that there are jobs in the central queue waiting to be served and that there are less than d busy servers in the system. Therefore, $n_1 = \ldots = n_{d-1} = 0$.

Now denote the state space of the Markov chain by $\mathcal{S}$ and let any generic state $s \in \mathcal{S}$ be of the type $(n_i, M_i, \ldots, n_1, M_1)$. Define $\lambda_M(\{M_1, \ldots, M_i\})$ as the activation rate of server M, when the set of active servers is $\{M_1, \ldots, M_i\}$. In state $s = (n_i, M_i, \ldots, n_1, M_1)$, this is the transition rate from state s to state $(0, M, s)$. This activation rate $\lambda_M(\{M_1, \ldots, M_i\})$ depends on the *assignment rule* defined for the model, which determines to which idle feasible machine (if any) a job is routed. By considering *assignment rules* that satisfy a certain *assignment condition*, Visschers et al. [7] are able to obtain a product form stationary distribution (Theorem 2, [7]). Now to apply Theorem 2, [7] to the *c.o.s.* model, we need to verify that the *uniform assignment rule* required for the redundancy-d *c.o.s.* model also satisfies the *assignment condition* given in [7]. We prove in the following lemma that this is indeed the case. (See [8] for the proof.) This will subsequently let us obtain a product form stationary distribution for *c.o.s* model as in [7].

Lemma 1. *The uniform assignment rule satisfies the assignment condition given in [7].*

3. REDUNDANCY-d WITH *C.O.S.*: AN EXACT ANALYSIS

In this section, we provide the first exact analysis for the redundancy-d model with *c.o.s.*.

3.1 Product form stationary distribution

We first provide the steady state distribution for the *c.o.s* system with the Markovian descriptor $s = (n_i, M_i, \ldots, n_1, M_1)$, which has a product form. We obtain this by applying Theorem 2, [7] to the *c.o.s.* model. (See the technical report [8] for a proof.)

Proposition 1. *The steady state distribution for any state $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$ with $n_1 = \ldots = n_{d-1} = 0$ is given by*

$$\pi(s) = \begin{cases} \bar{G}^i(K,d)\frac{\rho^i \pi(0)}{i!} & \text{for } i < d \\ \bar{r}_i^{n_i} \ldots \bar{r}_d^{n_d} \bar{G}^i(K,d) \frac{\pi(0)}{i!} \rho^{\left(i+\sum_{j=d}^{i} n_j\right)} & \text{for } i \geq d. \end{cases} \quad (1)$$

where $\pi(0)$ is the probability of an empty system and where

$$\bar{r}_i = \frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}}, \quad \bar{G}^i(K,d) = \prod_{j=1}^{i} \bar{G}_j(K,d) \text{ and}$$

$$\bar{G}_j(K,d) = \frac{d}{\binom{K-1}{d-1}} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha}.$$

For any state $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$ where $n_1 = \ldots = n_{d-1} = 0$ is not true, we have $\pi(s) = 0$. □

It is interesting to point out that the stationary distribution does not depend on the identity of the servers that are active since the servers are assumed to be homogeneous. Hence the stationary probabilities for states with the same number of active servers (i) and same number of waiting (n_i) jobs between servers are the same.

3.2 Normalization constant

We denote by $p(i)$ the stationary probability that the redundancy-d model with *c.o.s.* has i busy servers, for $1 \leq i \leq K$. This metric would be useful in obtaining the normalizing constant $\pi(0)$, which is also the probability that the system is empty, i.e., there are no busy servers ($p(0) = \pi(0)$). Noting that $p(i) = \sum_{s \in \mathcal{S}_i} \pi(s)$ where $\mathcal{S}_i = \{s \in \mathcal{S} :$ exactly i servers are busy$\}$, we have the following lemma. The proof can be found in [8].

Lemma 2. *The probability that i servers are busy is given by $p(i) = \bar{p}(i)\pi(0)$ where*

$$\bar{p}(i) = \begin{cases} \binom{K}{i}\bar{G}^i(K,d)\rho^i, & \text{for } i < d \\ \left(\frac{1}{1-r_i}\right) \ldots \left(\frac{1}{1-r_d}\right) \binom{K}{i}\bar{G}^i(K,d)\rho^i, & \text{for } i \geq d \end{cases}$$

and $r_i = \rho \bar{r}_i$. Further, $\pi(0)$ is given by $\pi(0) = \left(1 + \sum_{i=1}^{K} \bar{p}(i)\right)^{-1}$. □

3.3 PGF of number of waiting jobs

We next obtain the expression for the probability-generating function for the number of waiting jobs in the system. Define $p(i,m)$ as the probability that the system has i busy servers and m waiting jobs in the system. When $i < d$, we have $p(i,m) = p(i)$ for $m = 0$ and $p(i,m) = 0$ elsewhere. For $i \geq d$, it follows from Proposition 1 that

$$p(i,m) = \pi(0)\binom{K}{i}\bar{G}^i(K,d)\rho^i l_i(m)$$

$$\text{where } l_i(m) = \sum_{\substack{\{n_i \ldots n_d: \\ \sum_{l=d}^{i} n_l = m\}}} r_i^{n_i} \ldots r_d^{n_d}.$$

Let Q denote the random variable corresponding to the number of waiting jobs in the *c.o.s.* system. The probability that there are m waiting jobs ($Q = m$) is given by $\hat{p}(m) = \sum_{i=1}^{K} p(i,m)$. Using the above expressions for $p(i,m)$, we have the following result (proof in [8]).

Proposition 2. *The P.G.F. for the number of waiting jobs is given by*

$$E(z^Q) = \sum_{i=0}^{d-1} p(i) + \sum_{i=d}^{K} p(i) \left(\prod_{j=d}^{i} Geom_{r_j}(z)\right) \quad (2)$$

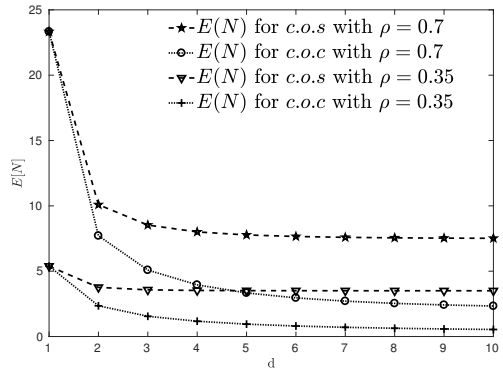

Figure 1: $E(N)$ for $c.o.c.$ and $c.o.s.$ for $K = 10$.

where $Geom_{r_j}(z) = \frac{1-r_j}{1-r_j z}$ *and* $r_i = \rho \bar{r}_i$. *The expected number of waiting jobs in the system is given by*

$$E(Q) = \sum_{i=d}^{K} p(i) \left(\sum_{j=d}^{i} \frac{r_j}{1-r_j} \right) \tag{3}$$

and the expected number of jobs in the system is given by $E(N) = E(Q) + \rho K$. □

4. COMPARING *C.O.S* AND *C.O.C*

We compare $E(N)$ under *c.o.s* and *c.o.c* systems (the mean number of customers $E(N)$ in the *c.o.c.* system is derived in [6] assuming copies are i.i.d.). Consider *c.o.c.* and *c.o.s.* systems with parameters $K = 10, \rho = 0.35$ and 0.7. For varying values of d, Fig. 1 compares $E(N)$ for the two systems.

The case $d = 1$ for both models is equivalent to Bernoulli routing to K servers in which case we have $E(N) = \frac{K\rho}{1-\rho}$. The case, $d = K$ for the *c.o.c.* model coincides with an $M/M/1$ server with arrival rate of λ and service rate of $K\mu$ and hence $E(N) = \frac{\rho}{1-\rho}$. On the other hand, $d = K$ for *c.o.s.* corresponds to an $M/M/K$ system with arrival rate λ and K servers each with a service rate 1. Hence $E(N)$ for the $M/M/K$ system is larger than that of the $M/M/1$ system associated with $d = K$ for *c.o.c.* For $1 < d < K$, we also see that $E(N)$ for *c.o.s.* is larger than that of the *c.o.c.* model and this is true for any value of ρ. This is an unexpected conclusion (since *c.o.s.* does not waste any resources) that *c.o.s.* is worse in terms of mean number of jobs. It can be argued (see [8]) that this is primarily due to the assumption of i.i.d copies in case of *c.o.c.* (together with exponential requirements). Clearly, such assumptions can lead to conclusions that are qualitatively different from what one intuitively expects.

5. JSW(d) & redundancy-d with *C.O.S.*

In this section, we will provide our main proposition on the equivalence of the redundancy-d *c.o.s.* model with that of the join the shortest work among d servers (JSW(d)) policy. While this is trivial to note for the case when $d = 1$ and $d = K$, the proposition formalizes this fact for any d. The reasoning is based on a sample-path argument, hence, the equivalence holds for generally distributed service requirements and heterogeneous servers. The proof can be found in the technical report [8].

Proposition 3. *Assume generally distributed service requirements and heterogeneous servers. For any given sample-path realization, a given job will be served under both JSW(d) and redundancy-d with c.o.s. in the same server. As a result, the following performance measures of joint probability of servers being busy or idle, delay distribution of a job and total number of (waiting) jobs in the system, are the same under both models.* □

Remark 1. *With the above proposition, redundancy-d with c.o.s. can be perceived as a method of implementing JSW(d) without requiring knowledge of residual work in each queue.*

From the equivalence between JSW(d) and redundancy-d with *c.o.s.*, we have the following corollary that to the best of our knowledge provides performance metrics for JSW(d) that have not been obtained before.

Corollary 1. *For a JSW(d) system with Poisson arrivals and exponential service requirements, the P.G.F. for the number of waiting jobs is given by Eq. (2) and the expected number of waiting jobs in JSW(d) is given by Eq. (3). The expected number of jobs in JSW(d) is given by* $E(N) = E(Q) + \rho K$. □

See [9] for a very recent analysis of JSW(d) in the mean-field regime.

6. REFERENCES

[1] Ganesh Ananthanarayanan et al., "Reining in the outliers in map-reduce clusters using mantri.," in *OSDI*, 2010, vol. 10, p. 24.

[2] Ashish Vulimiri et al., "Low latency via redundancy," in *Proceedings of the ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 283–294.

[3] Gauri Joshi, Emina Soljanin, and Gregory Wornell, "Queues with redundancy: Latency-cost analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 2, pp. 54–56, 2015.

[4] Nihar B Shah, Kangwook Lee, and Kannan Ramchandran, "When do redundant requests reduce latency?," *IEEE Transactions on Communications*, vol. 64, no. 2, pp. 715–722, 2016.

[5] Kristen Gardneret et al., "Queueing with redundant requests: exact analysis," *Queueing Systems*, vol. 83, no. 3-4, pp. 227–259, 2016.

[6] Kristen Gardner et al., "Redundancy-d: The power of d choices for redundancy," *Operations Research*, 2017.

[7] Jeremy Visschers, Ivo Adan, and Gideon Weiss, "A product form solution to a system with multi-type jobs and multi-type servers," *Queueing Systems*, vol. 70, no. 3, pp. 269–298, 2012.

[8] Urtzi Ayesta, Tejas Bodas, and Ina Maria Verloop, "On a unifying product form framework for redundancy models," *Rapport LAAS n 18047*, 2018.

[9] Tim Hellemans and Benny Van Houdt, "On the power-of-d-choices with least loaded server selection," *To appear in ACM Sigmetrics 2018. Also available as arXiv preprint arXiv:1802.05420*, 2018.