

# Controlling variability in split–merge systems and its impact on performance

Iryna Tsimashenka · William J. Knottenbelt · Peter G. Harrison

Published online: 16 April 2014

© The Author(s) 2014. This article is published with open access at Springerlink.com

**Abstract** We consider split–merge systems with heterogeneous subtask service times and limited output buffer space in which to hold completed but as yet unmerged subtasks. An important practical problem in such systems is to limit utilisation of the output buffer. This can be achieved by judiciously delaying the processing of subtasks in order to cluster subtask completion times. In this paper we present a methodology to find those deterministic subtask processing delays which minimise any given percentile of the difference in times of appearance of the first and the last subtasks in the output buffer. Technically this is achieved in three main steps: firstly, we define an expression for the distribution of the range of samples drawn from  $n$  independent heterogeneous service time distributions. This is a generalisation of the well-known order statistic result for the distribution of the range of  $n$  samples taken from the same distribution. Secondly, we extend our model to incorporate deterministic delays applied to the processing of subtasks. Finally, we present an optimisation scheme to find that vector of delays which minimises a given percentile of the range of arrival times of subtasks in the output buffer. We show the impact of applying the optimal delays on system stability and task response time. Two case studies illustrate the applicability of our approach.

**Keywords** Split–merge system · Subtask variability · Optimal subtask delays

**CR Subject Classification** C.4 Performance

---

I. Tsimashenka (✉) · W. J. Knottenbelt · P. G. Harrison  
Department of Computing, Imperial College London, South Kensington Campus,  
London SW7 2AZ, UK  
e-mail: it09@doc.ic.ac.uk

W. J. Knottenbelt  
e-mail: wjk@doc.ic.ac.uk

P. G. Harrison  
e-mail: pgh@doc.ic.ac.uk

## 1 Introduction

Numerous physical systems of practical interest feature a queue of incoming tasks which split into synchronised subtasks that are processed in parallel at a set of (potentially heterogeneous) servers. Subtasks that complete service are held in an output buffer until all of its siblings have completed service. Examples of such systems include the processing of logical I/O requests by a RAID enclosure housing several physical disk drives (Lebrecht et al. 2011), parallel job processing in MapReduce environments comprising several compute nodes (Zaharia 2010), and the assembly of customer orders made up of multiple items in the highly-automated warehouses of large online retailers (Serfozo 2009).

The importance of performance prediction in such systems has been long appreciated by performance modellers who have devised abstractions for their representation, most notably *split-merge* queueing systems and their less synchronised—but analytically much less tractable—counterparts, *fork-join* queueing systems (Bolch 2006).

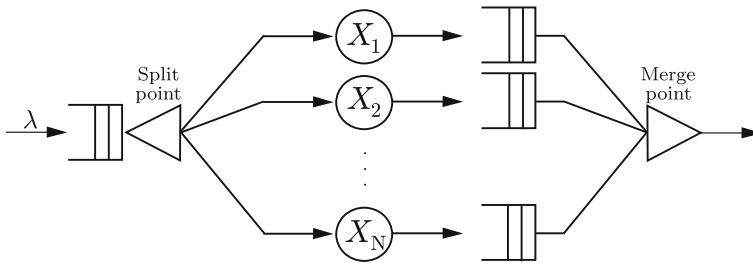
Understandably, for both kinds of model, the primary focus of research work to date has been on the computation of the stationary distribution of the number of subtasks queued at each server and on moments of task response time, most especially the mean. Flatto et al. (Flatto 1985; Flatto and Hahn 1984) derive exact analytical solutions for the stationary distribution of the number of subtasks in each queue in a two-node fork-join queueing systems with exponential task arrivals and heterogeneous exponential service time distributions. Heidelberg and Trivedi (1983) develop two highly accurate approximation methods for mean queue length and mean response time prediction in closed parallel queueing systems based on M/M/1-queues where primary tasks fork into two or more secondary subtasks. For fork-join systems with homogeneous exponential service time distributions, Nelson and Tantawi (1988) describe a technique which yields approximate lower and upper bounds on mean task response time as a function of the number of servers. Kim and Agrawala (1989) derive an approach which approximates mean task response time and state-occupancy probabilities in multi-server fork-join systems with Erlang service time distributions. Baccelli et al. (1989, 1989) derive bounds on various transient and steady-state performance measures for (predominantly homogeneous) fork-join queueing systems by stochastically comparing a given system with constructed queueing systems with simpler structure but identical stability characteristics. Towsley et al. (1990) develop mathematical models for mean task response time of fork-join parallel programs executing on a shared memory multiprocessor under Poisson tasks arrivals and two different scheduling policies. Under the *task scheduling* policy, the authors derive lower and upper bounds for mean task response time, while under the *job scheduling* policy, standard birth-death theory leads to an exact expression for mean task response time. Varma and Makowski (1994) use interpolation between light and heavy traffic modes to approximate the mean response time for a homogeneous fork-join system of M/M/1 queues. The same fork-join system was considered in Lebrecht and Knottenbelt (2007), where a maximum order statistic provides an easily-computable upper bound on response time. Varki et al. (unpublished) present bounds on mean response time in a fork-join queueing system with Poisson arrivals and exponential subtask service time distributions. Harrison and Zertal (2007) present an approximation for moments of the maximum of response times in a split-merge queueing system with Poisson task arrivals and general heterogeneous subtask service times; this gives an exact result in the case of exponential subtask service time distributions. Another recent approach by Sun and Peterson (2012) presented in the context of parallel program execution time analysis—but with ready application to the analysis of split-merge systems—approximates the

expectation of the maximum value of a set of random variables drawn from certain distribution classes by solving for the domain value at which the inverse cdf of the maximum order statistic is equal to a constant (0.570376002).

By contrast, the focus of the present paper is not response time computation; rather it concerns ways to control the variability of subtask completion time (that is the difference in time between the arrival of the first and last subtasks of a task in the output buffer) in split–merge systems. The idea is to try to cluster the arrival of subtasks in the output buffer by applying judiciously-chosen deterministic delays to subtasks before they are dispatched to the parallel servers. This has especial relevance for systems that involve the retrieval of orders comprising multiple items from automated warehouses (Serfozo 2009), since partially completed subtasks must be held in a physical buffer space that is often limited and highly utilised; consequently it is difficult to manage. Despite this, to the best of our knowledge, this problem has not received significant attention in the literature. Our previous work (Tsimashenka and Knottenbelt 2011) presented a simple mean-based methodology for computing the vector of deterministic subtask delays that minimises a cost function given by the difference between the expected maximum and expected minimum subtask completion times (across all subtasks arising from a particular task). However, an expected value does not always satisfy service level objectives; in addition there is a dependence between the maximum and minimum subtask completion times which must be taken into account for any distributional analysis. The methodology we present here yields the set of subtask delays which minimises any given percentile of the distribution of the difference in the time of appearance of the first and last subtasks in the output buffer.

The present paper is an extended version of our work presented in Tsimashenka et al. (2012). The technical contribution of this work begins with a generalisation of the well-known order statistics result for the distribution of the range when  $n$  samples are taken from a given distribution  $F(t)$ . In particular, we obtain the distribution of the range of  $n$  samples taken from heterogeneous distributions  $F_i(t)$  ( $i = 1, \dots, n$ ). Having extended this theory to incorporate deterministic subtask processing delays, we show how an optimisation procedure can be applied to a split–merge system to find that vector of subtask delays which minimises a given percentile of the range of subtask completion times. Our further contributions over Tsimashenka et al. (2012) include: (a) quantification of the adverse impact of minimising subtask variability on system stability and expected task sojourn time in the system, (b) lowering of the computational complexity of our optimisation procedure through the use of Brent’s method rather than the Bisection method, (c) implementing a split–merge system simulator and performing mutual validation between the analysis and simulation, and (d) an additional case study.

The rest of the paper is organised as follows. Section 2 describes essential preliminaries including a definition of split–merge systems and selected results from the theory of order statistics. Section 3 presents various heterogeneous order statistic results, including the distribution of the range. Section 4 shows how the basic split–merge model can be enhanced to support deterministic delays, defines an appropriate objective function, and presents a related optimisation procedure. Section 5 describes the impact of applying subtask delays on system stability and expected task response time. Section 6 presents two case studies which demonstrate the applicability of our work. The first case study considers a split–merge system with just three parallel servers, which allows for convenient visual representation of the optimisation landscape. The second case study considers a larger system of eight parallel servers. Section 7 concludes and considers avenues for future work.



**Fig. 1** Split–merge queueing model

## 2 Essential preliminaries

### 2.1 Parallel systems

A split–merge system (see Fig. 1) is a composition of a queue of waiting tasks (assumed to arrive according to a Poisson process with mean rate  $\lambda$ ), a split point, several heterogeneous servers (which serve their allocated subtask with general service time distribution with mean service rate  $1/\mu_i$ ), buffers for completed subtasks (merge buffers) and a merge point (Bolch 2006). We note that in practice in physical systems it is not uncommon for the merge buffers to share the same physical space which is managed as a single logical output buffer. When the queue of waiting tasks is not empty and the parallel servers are idle, a task is injected into the system from the head of the queue. The task is split into  $n$  subtasks at the *split* point and the subtasks arrive simultaneously at the  $n$  parallel servers to receive service. Completed subtasks join a merge buffer. Only after all subtasks (belonging to a particular task) are present in the merge buffers does the original task depart the system via the merge point. We note that this split–merge system is a more synchronised type of fork–join system. In split–merge systems parallel servers are blocked after they have served a subtask while the original task is in the system, whereas in fork–join systems there is no queue of waiting tasks, but there is a queue of subtasks at each parallel server. Task response time in a split–merge system yields an upper bound on task response time in a fork–join system having the same set of parallel servers and task arrival rate (Lebrecht and Knottenbelt 2007).

### 2.2 Theory of order statistics (David 1980)

**Definition:** Let the increasing sequence  $X_{(1)}, X_{(2)}, \dots, X_{(n)}$  be a permutation of the real valued random variables  $X_1, X_2, \dots, X_n$ , i.e. the  $X_i$  arranged in ascending order  $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n)}$ . Then  $X_{(i)}$  is called the  $i$ th order statistic, for  $i = 1, 2, \dots, n$ . The first and last order statistics,  $X_{(1)}$  and  $X_{(n)}$ , are the minimum and maximum respectively, which are also called the *extremes*.  $T = X_{(n)} - X_{(1)}$  is the *range*.

We assume initially that the random variables  $X_i$  are identically distributed as well as independent (iid), but of course the  $X_{(i)}$  are dependent because of the ordering.

#### 2.2.1 Distribution of the $k$ th-order statistic (iid case)

If  $X_1, X_2, \dots, X_n$  are  $n$  independent random variables, the cumulative distribution function (cdf) of the maximum order statistic (the maximum) is simply given by

$$F_n(x) = Pr\{X_{(n)} \leq x\} = Pr\{X_i \leq x, 1 \leq i \leq n\} = F^n(x)$$

Likewise, the cdf of the minimum order statistic is:

$$F_1(x) = Pr\{X_{(1)} \leq x\} = 1 - Pr\{X_{(1)} > x\} = 1 - Pr\{X_i > x, 1 \leq i \leq n\} = 1 - [1 - F(x)]^n$$

These are special cases of the general cdf of the  $r$ th order statistic,  $F_r(x)$ , which can be expressed as:

$$F_r(x) = Pr\{X_{(r)} \leq x\} = Pr\{\text{at least } r \text{ of the } X_i \leq x\} \\ = \sum_{i=r}^n \binom{n}{i} F(x)^i [1 - F(x)]^{n-i} \tag{1}$$

The pdf of  $X_r$ ,  $f_r(x) = F'_r(x)$ , where the prime denotes the derivative with respect to  $x$ , when it exists, is then:

$$f_r(x) = \frac{n!}{(r-1)!(n-r)!} F^{r-1}(x) f(x) [1 - F(x)]^{n-r}.$$

Multiplying both sides by “small”  $\epsilon$ , this result follows intuitively from noting that we require one of the  $X_i$  to take a value in the interval  $(x, x + \epsilon]$ , exactly  $r - 1$  of the  $X_i$  to be less than or equal to  $x$  and exactly  $n - r$  of them to be greater than  $x$ . The coefficient  $n!/((r - 1)!(n - r)!)$  is the number of ways of doing this, given that the  $X_i$  are stochastically indistinguishable.

The joint density function of the  $r$ th and  $s$ th order statistics  $X_{(r)}, X_{(s)}$ , where for  $1 \leq r < s \leq n$  and  $x \leq y$ , is:

$$f_{rs}(x, y) = S_{rs} F^{r-1}(x) f(x) [F(y) - F(x)]^{s-r-1} f(y) [1 - F(y)]^{n-s} \tag{2}$$

where  $S_{rs} = \frac{n!}{(r-1)!(s-r-1)!(n-s)!}$ , by similar reasoning. The corresponding joint cdf  $F_{rs}(x, y)$  of  $X_{(r)}$  and  $X_{(s)}$  may be obtained by integration of the pdf or, alternatively, following the same conditions, we have:

$$F_{rs}(x, y) = Pr\{\text{at least } r \text{ of the } X_i \leq x, \text{ at most } n - s \text{ of the } X_i > y\} \\ = \sum_{j=s}^n \sum_{i=r}^j Pr\{\text{exactly } i \text{ of the } X_i \leq x, \text{ exactly } n - j \text{ of the } X_i > y\} \\ = \sum_{j=s}^n \sum_{i=r}^j \frac{n!}{i!(j-i)!(n-j)!} F^i(x) [F(y) - F(x)]^{j-i} [1 - F(y)]^{n-j}$$

Finally, the joint pdf for the  $k$  order statistics  $X_{(n_1)}, \dots, X_{(n_k)}$ ,  $1 \leq n_1 < \dots < n_k \leq n$ , is similarly, for  $x_1 \leq \dots \leq x_k$ :

$$f_{n_1, \dots, n_k}(x_1, \dots, x_k) = S_{n_1, \dots, n_k} F^{n_1-1}(x_1) f(x_1) [F(x_2) - F(x_1)]^{n_2-n_1-1} f(x_2) \dots \\ [F(x_k) - F(x_{k-1})]^{n_k-n_{k-1}-1} f(x_k) [1 - F(x_k)]^{n-n_k}$$

where  $S_{n_1, \dots, n_k} = \frac{n!}{(n_1-1)!(n_2-n_1-1)! \dots (n_k-n_{k-1}-1)!(n-n_k)!}$ .

### 2.2.2 Distribution of the range

The pdf  $f_{T_{rs}}(x)$  of the interval  $T_{rs} = X_{(s)} - X_{(r)}$  follows from the joint pdf of the  $r$ th and  $s$ th order statistics in Eq. 2 by setting  $y = x + t_{rs}$  and integrating over  $x$ , giving:

$$f_{T_{rs}}(t_{rs}) = S_{rs} \int_{-\infty}^{\infty} F^{r-1}(x)f(x)[F(x + t_{rs}) - F(x)]^{s-r-1}f(x + t_{rs})[1 - F(x + t_{rs})]^{n-s} dx$$

In the special case when  $r = 1$  and  $s = n$ ,  $T_{rs}$  is the range  $T = X_{(n)} - X_{(1)}$  and the pdf simplifies to:

$$f_T(t) = n(n - 1) \int_{-\infty}^{\infty} f(x)[F(x + t) - F(x)]^{n-2}f(x + t)dx.$$

The cdf of  $T$  then follows by integrating inside the integral with respect to  $x$ , giving:

$$\begin{aligned} F_T(t) &= n \int_{-\infty}^{\infty} f(x) \int_0^t (n - 1)f(x + t')[F(x + t') - F(x)]^{n-2} dt' dx \\ &= n \int_{-\infty}^{\infty} f(x) [[F(x + t') - F(x)]^{n-1}]_{t'=0}^{t'=t} dx \\ &= n \int_{-\infty}^{\infty} f(x)[F(x + t) - F(x)]^{n-1} dx. \end{aligned} \tag{3}$$

As noted in David (1980), this equation follows intuitively by noting that the integrand (multiplied by an infinitesimal quantity  $dx$ ) is the probability that  $X_i$  falls into the interval  $(x, x + dx]$  (for some  $i$ ) and the remaining  $n - 1$  of the  $X_j, j \neq i$  fall into  $(x, x + t]$ . There are  $n$  ways of choosing  $i$ , giving the factor  $n$ .

### 3 Heterogeneous order statistics

We now consider  $n$  independent, real-valued random variables  $X_1, \dots, X_n$  where each  $X_i$  has an arbitrary probability distribution  $F_i(x)$  and probability density function  $f_i(x) = F'_i(x)$ . In this case of “heterogeneous” (or independent, but not necessarily identically distributed) random variables, we call the order statistics *heterogeneous order statistics* to distinguish them from the better known results where the random variables are implicitly assumed to be identically distributed.

Recent decades have seen increasing consideration given to the heterogeneous case in the literature. Key theoretical results for the distribution and density functions of heterogeneous order statistics are summarised in David and Nagaraja (2005). This includes the work of Sen (1970), who derived bounds on the median and the tails of the distribution of heterogeneous order statistics. Practical issues related to the numerical computation of the  $i$ th heterogeneous order statistic are considered in Cao and West (1997), with special consideration of recurrence relations among distribution functions of order statistics.

### 3.1 Distribution of the $r$ th heterogeneous order statistic

The  $r$ th heterogeneous order statistic, derived similarly to Eq. 1, has the following cdf:

$$\begin{aligned}
 F_{(r)}(x) &= Pr\{X_{(r)} \leq x\} = Pr\{\text{at least } r \text{ of the } X_i \leq x\} \\
 &= \sum_{i=r}^n \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_i} \prod_{k=1}^i F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)]
 \end{aligned} \tag{4}$$

where  $\mathcal{P}_i$  is the set of all two-set partitions  $\{D, E\}$  of  $\{1, 2, \dots, n\}$  with  $|D| = i$  and  $|E| = n - i$ , and  $\ell_{hk}$  is the  $k$ th component of the vector  $\ell_h$  for  $h = 1, 2$ .

Similarly to the homogeneous case, the minimum and maximum order statistics are respectively given by:

$$\begin{aligned}
 F_{(1)}(x) &= Pr\{X_{(1)} \leq x\} = 1 - Pr\{X_{(1)} > x\} \\
 &= 1 - Pr\{X_i > x \forall i : 1 \leq i \leq n\} = 1 - \prod_{i=1}^n [1 - F_i(x)],
 \end{aligned}$$

and

$$F_{(n)}(x) = Pr\{X_{(n)} \leq x\} = Pr\{X_i \leq x \forall i : 1 \leq i \leq n\} = \prod_{i=1}^n F_i(x).$$

Differentiating Eq. 4 and simplifying yields the pdf:

$$\begin{aligned}
 f_{(r)}(x) &= \sum_{i=r}^n \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_i} \left[ \sum_{j=1}^i \prod_{k=1, k \neq j}^i F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] f_{\ell_{1j}}(x) \right. \\
 &\quad \left. - \sum_{j=1}^{n-i} \prod_{k=1}^i F_{\ell_{1k}}(x) \prod_{k=1, k \neq j}^{n-i} [1 - F_{\ell_{2k}}(x)] f_{\ell_{2j}}(x) \right] \\
 &= \sum_{i=r}^n \sum_{h=1}^n \left[ \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{i-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] f_h(x) \right. \\
 &\quad \left. - I_{i < n} \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^i F_{\ell_{1k}}(x) \prod_{k=1}^{n-i-1} [1 - F_{\ell_{2k}}(x)] f_h(x) \right] \\
 &= \sum_{h=1}^n f_h(x) \left[ \sum_{i=r}^n \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{i-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] \right. \\
 &\quad \left. - \sum_{i=r+1}^n \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{i-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] \right] \\
 &= \sum_{h=1}^n f_h(x) \sum_{\{\ell_1, \ell_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{r-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-r} [1 - F_{\ell_{2k}}(x)]
 \end{aligned}$$

where  $I_{\bullet}$  is the indicator function and  $\mathcal{P}_i^{h-}$  is the set of all 2-set partitions of  $\{1, 2, \dots, n\} \setminus \{h\}$  with  $i$  elements in the first set and  $1 \leq h \leq n$ . In fact this result also follows from an intuitive argument using the infinitesimal interval  $(x, x + \epsilon]$ , as in the homogeneous case.

The joint density function  $f_{rs}(x, y)$  of two order statistics,  $X_{(r)}$  and  $X_{(s)}$ , for  $1 \leq r < s \leq n$  and  $x \leq y$ , follows similarly as:

$$\begin{aligned}
 f_{(r)(s)}(x, y) &= \sum_{1 \leq h_1 \neq h_2 \leq n} f_{h_1}(x)f_{h_2}(y) \sum_{\{\ell_1, \ell_2, \ell_3\} \in \mathcal{P}_{r-1, s-r-1}^{h_1-h_2-}} \prod_{k=1}^{r-1} F_{\ell_{1k}}(x) \\
 &\times \prod_{k=1}^{s-r-1} [F_{\ell_{2k}}(y) - F_{\ell_{2k}}(x)] \prod_{k=1}^{n-s} [1 - F_{\ell_{3k}}(y)]
 \end{aligned}
 \tag{5}$$

where  $\mathcal{P}_{i_1, i_2}^{h_1-h_2-}$  is the set of all 3-set partitions of  $\{1, 2, \dots, n\} \setminus \{h_1, h_2\}$  with  $i_1$  elements in the first set,  $i_2$  elements in the second set, and so  $n - i_1 - i_2 - 2$  in the third, and  $1 \leq h_1 \neq h_2 \leq n$ .

### 3.2 Distribution of the range for heterogeneous order statistics

From the joint pdf of two heterogeneous order statistics in Eq. 5, we obtain the pdf of the interval  $T_{rs} = X_{(r)} - X_{(s)}$  by setting  $t_{rs} = y - x$ :

$$\begin{aligned}
 f_{(r)(s)}(t_{rs}) &= \sum_{1 \leq h_1 \neq h_2 \leq n} \int_{-\infty}^{\infty} f_{h_1}(x)f_{h_2}(x + t_{rs}) \\
 &\sum_{\{\ell_1, \ell_2, \ell_3\} \in \mathcal{P}_{r-1, s-r-1}^{h_1-h_2-}} \prod_{k=1}^{r-1} F_{\ell_{1k}}(x) \prod_{k=1}^{s-r-1} [F_{\ell_{2k}}(x + t_{rs}) \\
 &- F_{\ell_{2k}}(x)] \prod_{k=1}^{n-s} [1 - F_{\ell_{3k}}(x + t_{rs})] dx.
 \end{aligned}
 \tag{6}$$

For the range, we want the special case in which  $r = 1, s = n$  and  $T = X_{(n)} - X_{(1)}$ , giving the pdf:

$$\begin{aligned}
 f_{(1:n)}(t) &= \sum_{1 \leq h_1 \neq h_2 \leq n} \int_{-\infty}^{\infty} f_{h_1}(x)f_{h_2}(x + t) \sum_{\{\ell_1, \ell_2, \ell_3\} \in \mathcal{P}_{0, n-2}^{h_1-h_2-}} \prod_{k=1}^{n-2} [F_{\ell_{2k}}(x + t) - F_{\ell_{2k}}(x)] dx \\
 &= \sum_{1 \leq h_1 \neq h_2 \leq n} \int_{-\infty}^{\infty} f_{h_1}(x)f_{h_2}(x + t) \prod_{k \neq h_1, h_2} [F_k(x + t) - F_k(x)] dx.
 \end{aligned}
 \tag{7}$$

The cdf now follows by integration (inside the sum and integral with respect to  $x$ ):

$$\begin{aligned}
 F_{(1:n)}(t) &= \sum_{1 \leq h_1 \neq h_2 \leq n} \int_{-\infty}^{\infty} f_{h_1}(x) \int_0^t f_{h_2}(x + t') \prod_{k \neq h_1, h_2} [F_k(x + t') - F_k(x)] dt' dx \\
 &= \sum_{1 \leq h_1 \leq n} \int_{-\infty}^{\infty} f_{h_1}(x) \prod_{k \neq h_1} [F_k(x + t) - F_k(x)] dx.
 \end{aligned}
 \tag{8}$$

In fact, the same result can be obtained by noting that Eq. 3 generalises using the argument given immediately following it. This is that, given a particular choice of  $i = 1, 2, \dots, n$ , the integrand (multiplied by an infinitesimal quantity  $dx$ ) is the probability that  $X_i$  falls into the



interval  $(x, x + dx]$  and the other  $X_j, j \neq i$  fall into  $(x, x + t]$ . Of course there are  $n$  ways of choosing  $i$ , and so we have to sum over  $n$  terms; in the homogeneous case, all these terms are the same, which gave the factor  $n$ . For heterogeneous order statistics, we therefore obtain:

$$F_{\text{range}}(t) = F_{(1:n)}(t) = \sum_{i=1}^n \int_{-\infty}^{\infty} f_i(x) \prod_{j=1, j \neq i}^n [F_j(x + t) - F_j(x)] dx \tag{9}$$

This is a useful result, which requires a sum of only  $n$  terms. It can be directly applied to determine the distribution of the variability of subtask completion times in any split–merge system with heterogeneous service time distributions, and will form the basis for range-optimisation as considered in the next section.

### 4 Controlling variability in split–merge systems

#### 4.1 Introducing subtask delays

Our aim is to control the variability of subtask completion (equivalently output buffer arrival) times by introducing a vector of delays:

$$\mathbf{d} = (d_1, d_2, \dots, d_i, \dots, d_{n-1}, d_n) \tag{10}$$

Here,  $d_i$  denotes the deterministic delay that will be applied before a subtask is sent to server  $i$  for processing. From  $X_i \sim F_i(t)$  we define the random variables of parallel service time with applied delays as  $X_i^{\mathbf{d}} \sim F_i(t - d_i)$ . The order statistics of parallel service time with applied delays are  $X_{(1)}^{\mathbf{d}}, X_{(2)}^{\mathbf{d}}, \dots, X_{(n)}^{\mathbf{d}}$ .

After applying the delays from Eq. 10, the distribution of the range from Eq. 9 becomes:

$$F_{\text{range}}(t, \mathbf{d}) = \sum_{i=1}^n \int_{-\infty}^{\infty} f_i(x - d_i) \prod_{j=1, j \neq i}^n [F_j(x + t - d_j) - F_j(x - d_j)] dx \tag{11}$$

We assume that,  $\forall i, f_i(t - d_i) = 0, \forall t < d_i$ . Similarly,  $\forall j, F_j(t - d_j) = 0, \forall t < d_j$ . We note that in order to avoid unnecessarily delaying all subtasks we require that the subtask delay for at least one server (the “bottleneck” server) be set to 0.

#### 4.2 Optimisation procedure

In this section we move away from our previous mean-based technique (Tsimashenka and Knottenbelt 2011) towards a more sophisticated framework for finding delay vectors which provide soft (probabilistic) guarantees on variability. More specifically, for a given probability  $\alpha, \alpha \in (0, 1]$  we aim to minimise the 100 $\alpha$ th percentile of variability with respect to  $\mathbf{d}$ . That is, we aim to solve for  $\mathbf{d}$  in:

$$\mathbf{d}_\alpha = \min_{\mathbf{d}} F_{\text{range}}^{-1}(\alpha, \mathbf{d}) \tag{12}$$

Put another way, we aim to find that vector  $\mathbf{d}_\alpha$  which yields the lowest value for the 100 $\alpha$ th percentile of the difference in the completion times of the first and the last subtasks (belonging to each task).

Practically, we developed a numerical optimisation procedure by prototyping it in Mathematica and subsequently implementing a full version of it in C++ for efficiency reasons. Evaluation of Eq. 11 for a given  $\alpha$  and  $\mathbf{d}$  is performed by means of numerical integration using the trapezoidal rule. While this is adequate for almost all continuous service time density functions, complications arise in the case of the pdf of deterministic service time density functions because of its infinitely thin, infinitely high impulse. We resolve this by replacing the deterministic pdf with delay parameter  $a$  by the Gaussian approximation:

$$f_{\text{Det}}(a)(x) \approx \frac{1}{c\sqrt{\pi}} e^{-\frac{(x-a)^2}{c^2}}$$

which becomes exact as  $c \rightarrow 0$ ; in practice we set  $c = 0.01$ .

In order to invert Eq. 11 for a given  $\alpha$  and  $\mathbf{d}$ , we have applied the Bisection method (Burden and Burden 2006) in our previous work (Tsimashenka et al. 2012) because of its excellent robustness characteristics, but it might be not computationally optimal for all classes of functions, particularly convex functions. In the present work, we apply Brent's method (1971, 2002) which combines bisection, secant and inverse quadratic interpolation. On each iteration the proposed algorithm chooses which method to use based on (a) the method used for the previous iterate and (b) the trends observed in the most recent iterates. For “well-behaved” functions, this has the potential to deliver a considerably higher convergence rate while guaranteeing robustness (Brent 1971).

Finally, we explore the optimisation surface of  $F_{\text{range}}^{-1}(\alpha, \mathbf{d})$  with the initial  $\mathbf{d} = \{0, \dots, 0\}$  using a numerical optimisation procedure. We constrain the search such that  $d_i \geq 0$  for all  $i$  and  $\prod_i d_i = 0$  (that is, the “bottleneck” server(s) should have no unnecessary additional delay). In our implementation, we have used a simple Nelder–Mead optimisation technique (Nelder and Mead 1965), which is based on the simplex method. We note that a range of more sophisticated (and correspondingly considerably more complex to implement) gradient-free optimisation techniques are also available e.g. Ali and Gabere (2010), Lewis et al. (2007).

## 5 Impact of applied delays on system capacity and performance

In Tsimashenka et al. (2012) our concern was solely the minimisation of a given percentile of subtask dispersion time by introducing delays at the parallel servers. However, the introduction of such delays has negative implications for system stability and task response time, as quantified below.

### 5.1 Impact on system stability

We observe first that, at a high-level, any given split–merge system with a set of parallel servers is conceptually equivalent to a single M/G/1 queue whose service time is equal to the maximum of the service times of the replaced parallel servers, i.e.  $\max\{X_1, X_2, \dots, X_n\} = X_{(n)}$ . This is of course applicable for a split–merge system with delays as well, i.e.  $\max\{X_1^{\mathbf{d}}, X_2^{\mathbf{d}}, \dots, X_n^{\mathbf{d}}\} = X_{(n)}^{\mathbf{d}}$ . Secondly, due to the fact that cdfs are monotonically increasing functions:

$$F_i(x - d_i) \leq F_i(x), \forall i \text{ (given } d_i \geq 0 \forall i)$$

It follows that

$$\prod_{i=1}^n F_i(x - d_i) \leq \prod_{i=1}^n F_i(x)$$

and

$$\int_0^\infty 1 - \prod_{i=1}^n F_i(x - d_i) dx \geq \int_0^\infty 1 - \prod_{i=1}^n F_i(x) dx.$$

Thus:

$$\mathbb{E}[X_{(n)}^{\mathbf{d}}] \geq \mathbb{E}[X_{(n)}]$$

which proves the intuition that applying *any* delay to any subtask maintains or increases the mean task service time. A similar argument can be used to show that the same property applies to any percentile of task service time.

Exploiting the well-known stability condition for an M/G/1 queue, and denoting the maximum arrival rate at which the system with and without delays remains stable as  $\lambda_{\max}^{\mathbf{d}}$  and  $\lambda_{\max}$  respectively, we have:

$$\lambda_{\max}^{\mathbf{d}} = \frac{1}{\mathbb{E}[X_{(n)}^{\mathbf{d}}]} \leq \lambda_{\max} = \frac{1}{\mathbb{E}[X_{(n)}]}. \tag{13}$$

Thus, adding any delay to any subtask adversely impacts the maximum arrival rate at which the system remains stable.

### 5.2 Impact on response time

In order to quantify the impact of subtask delays on mean task response time we use the following metric:

$$\text{Response Penalty}_{\lambda} = \frac{\mathbb{E}[R_{\mathbf{d}=\mathbf{d}_i,\lambda}] - \mathbb{E}[R_{\mathbf{d}=\mathbf{0},\lambda}]}{\mathbb{E}[R_{\mathbf{d}=\mathbf{0},\lambda}]} \times 100 \tag{14}$$

where  $\mathbb{E}[R_{\mathbf{d}=\mathbf{d}_i,\lambda}]$  and  $\mathbb{E}[R_{\mathbf{d}=\mathbf{0},\lambda}]$  correspond for a given arrival rate  $\lambda$  to mean task response time with optimal delays and without any delays respectively. *Response Penalty* corresponds to the percentage by which task response time increases after application of optimal delays.

The expected task response time in a split–merge system is conceptually equivalent to the expected task response time in a single M/G/1 queue, whose service time is given by the maximum of the service times of the parallel servers. Consequently, we utilise the Pollaczek–Khinchine formula for mean task response time in an M/G/1 queue:

$$\mathbb{E}[R_{\mathbf{d}=\mathbf{0},\lambda}] = \frac{\rho + \mu \lambda \text{Var}[X_{(n)}]}{2(\mu - \lambda)} + \mu^{-1} \tag{15}$$

where  $\lambda$  is arrival rate,  $1/\mu$  is the mean service time (with  $\mu = \mathbb{E}[X_{(n)}]$ ), and  $\text{Var}[X_{(n)}]$  is the variance of service time. The latter can be calculated as:

$$\text{Var}[X_{(n)}] = 2 \int_0^\infty x \left(1 - \prod_{i=1}^n F_i(x)\right) dx - \left( \int_0^\infty \left(1 - \prod_{i=1}^n F_i(x)\right) dx \right)^2$$

Straightforward modification of the above formulae yields the expected task response time with applied delays ( $\mathbb{E}[R_{\mathbf{d}=\mathbf{d}_x,\lambda}]$ ).

### 6 Numerical results

Below we consider two case studies, which are analysed using a combination of numerical results from the methodology above and simulation results, with mutual validation as appropriate. The simulator is written in C++ and performs an event-driven simulation of a split–merge queueing system using component classes for task queues, split/merge points, parallel servers, merge/output buffers and links. For split–merge systems with up to 10 parallel servers, the simulator processes approximately 300,000 tasks per second on a 3.5GHz Intel Core-i5 workstation with 8GB RAM, and permits computation and validation of various quantities, both in scenarios with and without application of optimal delays, including (a) distribution of the range (cf. Eq. 11) of subtask processing times, (b) expected task response time (cf. Eq. 15) and (c) mean output buffer population. In the following, each simulation is replicated three times. Each replication starts with a warm-up phase involving the processing of 1,000,000 tasks; this is followed by a measurement phase involving the processing of 10,000,000 tasks. Because of the very large number of processed tasks, the resulting confidence intervals are extremely narrow and are consequently not reported. Results, whether derived from numerical analysis or simulation, are reported to three significant figures.

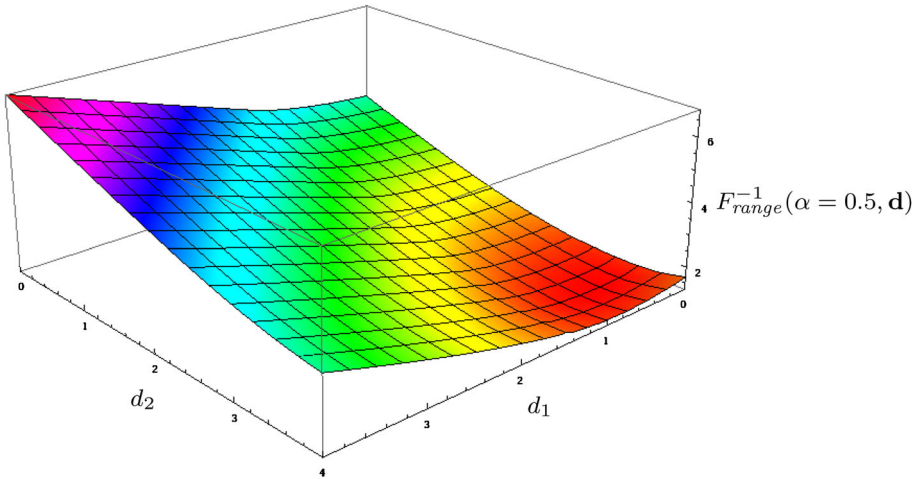
#### 6.1 Case study 1

Consider a split–merge system with task arrival rate  $\lambda = 0.1$  (tasks/time unit) and three parallel servers having heterogeneous service time density functions:

$$\begin{aligned} X_1 &\sim \text{Pareto}(\alpha = 3, l = 3.5, h = 10) \\ &\quad (\mathbb{E}[X_1] = 4.81, \text{Med}[X_1] = 1.2, \text{Var}[X_1] = 24.96) \\ X_2 &\sim \text{Erlang}(n = 2, \lambda = 1) \\ &\quad (\mathbb{E}[X_2] = 2, \text{Med}[X_2] = 1.68, \text{Var}[X_2] = 2) \\ X_3 &\sim \text{Det}(5) \\ &\quad (\mathbb{E}[X_3] = 5, \text{Med}[X_3] = 5, \text{Var}[X_3] = 0) \end{aligned}$$

Without adding any extra delays, it is straightforward to apply Eq. 9 in a root-finding algorithm (e.g. Brent’s method) to compute the 50th ( $\alpha = 0.5$ ) and 90th ( $\alpha = 0.9$ ) percentiles of the range of subtask completion times as  $t_{0.5} = 3.62$  and  $t_{0.9} = 5.21$  time units respectively. Simulations of the split–merge system show that the mean output buffer population is 0.560 subtasks.

Incorporating delays into the distribution of the range of subtask output buffer arrival times as per Eq. 11, and executing a Nelder–Mead optimisation (suitably constrained so that  $\prod_i d_i = 0$ ) to solve Eq. 12 given  $\alpha = 0.5$  yields



**Fig. 2** 50th percentile of the range of subtask output buffer arrival times for various deterministic processing delays. The optimal delay vector is  $\mathbf{d}_{0.5} = (0.533, 3.50, 0)$

$$\mathbf{d}_{0.5} = (0.533, 3.50, 0)$$

as shown in Fig. 2. The improved optimisation procedure’s run time (using Brent’s method) is 13 s in comparison with 46 s for the previous optimisation procedure from Tsimashenka et al. (2012). We note that in this case the “bottleneck” server is server 3. With the incorporation of the optimal delays, the 50th percentile of the range of subtask arrival times becomes  $t_{0.5} = 1.29$  time units, representing an improvement of 64 % over the original system configuration without delays. Mean output buffer population is 0.372, a 33 % reduction.

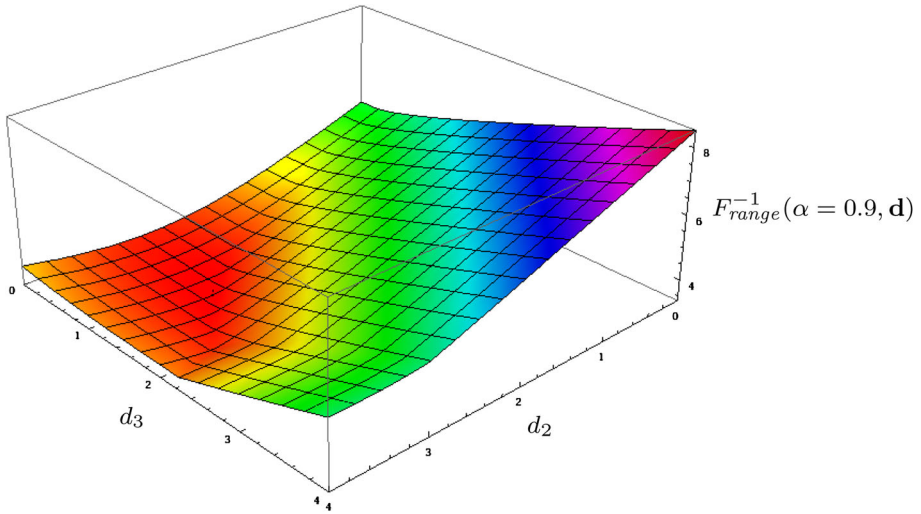
For  $\alpha = 0.9$  we obtain

$$\mathbf{d}_{0.9} = (0, 2.82, 1.16)$$

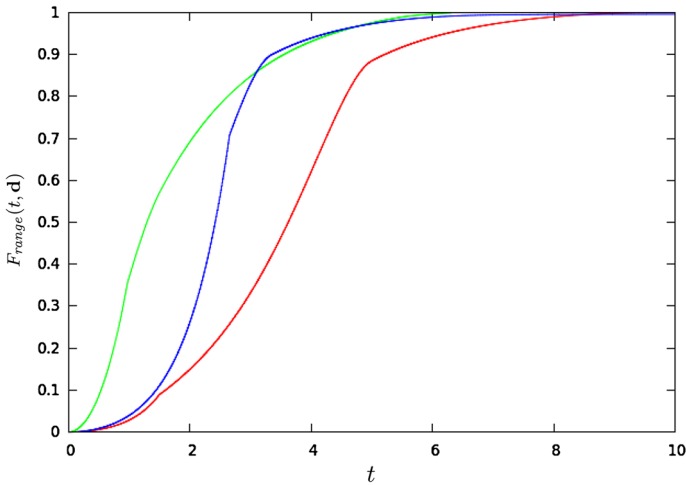
as shown in Fig. 3. We note that for this percentile the “bottleneck” switches from server 3 to server 1, despite the fact that server 3 has a higher mean service time than server 1. With the incorporation of the optimal delays, the 90th percentile of the range of subtask arrival times becomes  $t_{0.9} = 3.35$  time units, representing an improvement of 36 % over the original system configuration without delays. Mean output buffer population is 0.392 subtasks, a 29 % reduction.

Figure 4 shows how the distribution of the range of subtask output buffer arrival times changes according to the value of  $\alpha$ . We note that a change of  $\alpha$  can have a significant impact on the quantiles of  $F_{range}(t, \mathbf{d})$ , and may result in the shifting of the “bottleneck” server.

Without delays, the expected task response time is  $\mathbb{E}[R_{\mathbf{d}=0,0.1}] = 8.93$  time units. After introducing optimal subtask delays, the expected task response time becomes  $\mathbb{E}[R_{\mathbf{d}=\mathbf{d}_{0.5},0.1}] = 11.7$  time units for  $\alpha = 0.5$  and  $\mathbb{E}[R_{\mathbf{d}=\mathbf{d}_{0.9},0.1}] = 12.8$  time units for  $\alpha = 0.9$ . The percentage increases in expected task response time from Eq. 14 are 31 and 43 % respectively. Fig. 5 presents the corresponding distributions of task response time.

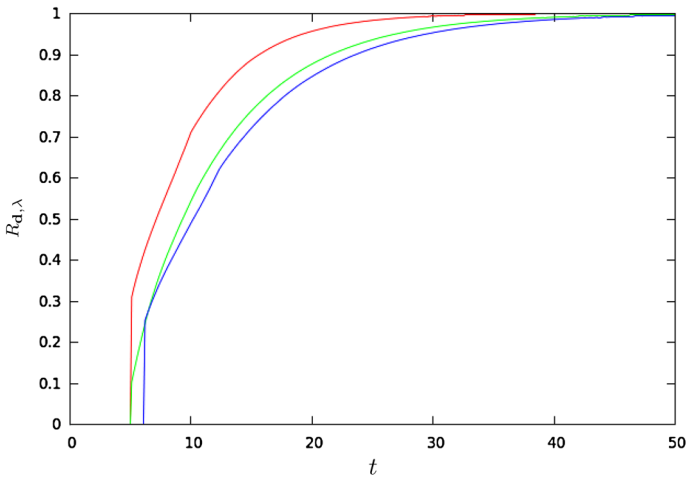


**Fig. 3** 90th percentile of the range of subtask output buffer arrival times for various deterministic processing delays. The optimal delay vector is  $\mathbf{d}_{0,9} = (0, 2.82, 1.16)$

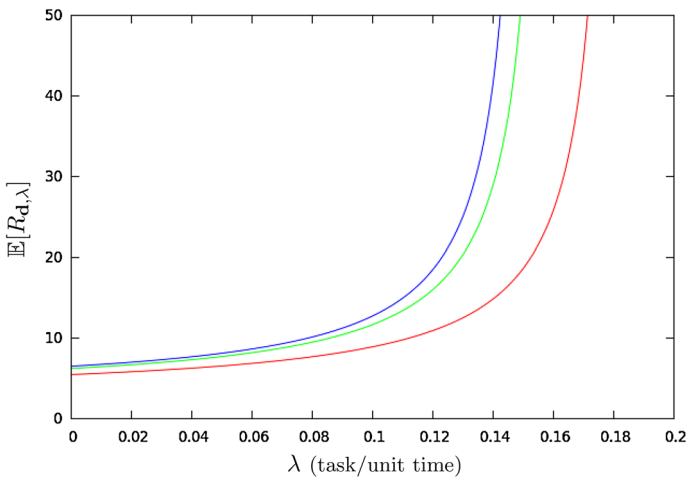


**Fig. 4** Distributions of the range of subtask output buffer arrival times without any delays (*red line*), with delays optimised under  $\alpha = 0.5$  (*green line*) and delays optimised under  $\alpha = 0.9$  (*blue line*). (Color figure online)

For the system without delays the maximum sustainable arrival rate from Eq. 13 is  $\lambda_{\max} < 0.182$  tasks/time unit. After introducing subtask delays to minimise the 50th and 90th percentiles of the range of subtask completion times, this drops to 0.160 and 0.153 tasks/time unit respectively. This is illustrated in Fig. 6.



**Fig. 5** Distributions of task response time given  $\lambda = 0.1$ , without any delays (*red line*), with delays optimised under  $\alpha = 0.5$  (*green line*) and delays optimised under  $\alpha = 0.9$  (*blue line*). (Color figure online)



**Fig. 6** Expected response time of split–merge system for various customer arrival rates without any delays (*red line*), with delays optimised under  $\alpha = 0.5$  (*green line*) and delays optimised under  $\alpha = 0.9$  (*blue line*). (Color figure online)

### 6.2 Case study 2

We consider a split–merge system with arrival rate  $\lambda = 0.1$  tasks/time unit and eight service nodes with following heterogeneous service time distributions:

$$\begin{aligned}
 X_1 &\sim \text{Exponential}(\lambda = 1) \\
 &\quad (\mathbb{E}[X_1] = 1, \text{Med}[X_1] = 0.693, \text{Var}[X_1] = 1) \\
 X_2 &\sim \text{Erlang}(n = 2, \lambda = 5) \\
 &\quad (\mathbb{E}[X_2] = 0.4, \text{Med}[X_2] = 0.336, \text{Var}[X_2] = 0.08) \\
 X_3 &\sim \text{Det}(2) \\
 &\quad (\mathbb{E}[X_3] = 2, \text{Med}[X_3] = 2, \text{Var}[X_3] = 0) \\
 X_4 &\sim \text{Pareto}(n = 2.1, a = 3.5, b = 10) \\
 &\quad (\mathbb{E}[X_4] = 4.89, \text{Med}[X_4] = 4.63, \text{Var}[X_4] = 28.81) \\
 X_5 &\sim \text{Exponential}(\lambda = 4) \\
 &\quad (\mathbb{E}[X_5] = 0.25, \text{Med}[X_5] = 0.173, \text{Var}[X_5] = 0.0625) \\
 X_6 &\sim \text{Erlang}(n = 3, \lambda = 3) \\
 &\quad (\mathbb{E}[X_6] = 1, \text{Med}[X_6] = 0.891, \text{Var}[X_6] = 0.333) \\
 X_7 &\sim \text{Exponential}(\lambda = 8) \\
 &\quad (\mathbb{E}[X_7] = 0.125, \text{Med}[X_7] = 0.0866, \text{Var}[X_7] = 0.0156) \\
 X_8 &\sim \text{Pareto}(n = 2.5, a = 3.5, b = 10) \\
 &\quad (\mathbb{E}[X_8] = 4.98, \text{Med}[X_8] = 4.49, \text{Var}[X_8] = 26.97)
 \end{aligned}$$

Applying Eq. 9 in a root-finding algorithm, we compute the 50th ( $\alpha = 0.5$ ) and 90th ( $\alpha = 0.9$ ) percentiles of the range of subtask completion times as  $t_{0.5} = 5.43$  time units and  $t_{0.9} = 8.26$  time units respectively. Simulations show that the mean output buffer population for the system is 3.20 subtasks.

Applying our methodology for determining optimal subtasks delays under  $\alpha = 0.5$  yields:

$$\mathbf{d}_{0.5} = (3.84, 4.13, 2.32, 0, 4.71, 3.55, 4.88, 0.279)$$

For this case the improved optimisation procedure (using Brent’s method) takes 2 min 5 s, whereas the procedure from Tsimashenka et al. (2012) takes 29 min. We see the “bottleneck” server is server 4. With the incorporation of the optimal delays, the 50th percentile of the range of subtask arrival times becomes  $t_{0.5} = 2.02$  time units, representing an improvement of 63 % over the original system configuration without delays. Mean output buffer population is 1.28 subtasks, a 60 % reduction.

For  $\alpha = 0.9$  we obtain:

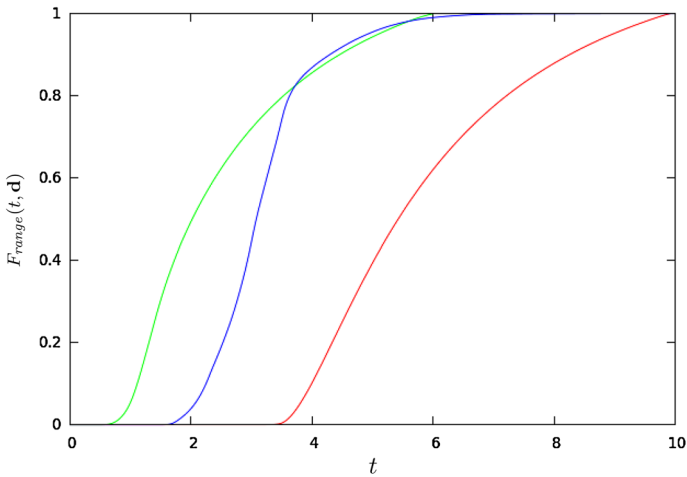
$$\mathbf{d}_{0.9} = (4.92, 5.97, 3.16, 0, 5.02, 4.37, 7.04, 0.758)$$

The “bottleneck” remains server 4. After adding optimal delays, the 90th percentile of the range of subtask arrival times becomes  $t_{0.9} = 4.29$  time units, representing an improvement of 48 % over the original system configuration without delays. Mean output buffer population is 1.36, a 58 % reduction.

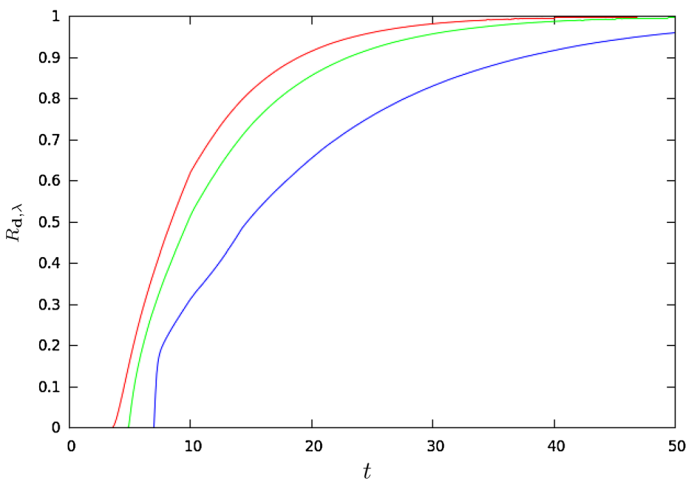
Figure 7 shows how the distribution of the range of subtask output buffer arrival times changes according to the value of  $\alpha$ .

Without delays, the expected task response time is  $\mathbb{E}[R_{\mathbf{d}=0,\lambda}] = 10.3$  time units. After introducing optimal subtask delays, the expected task response time becomes  $\mathbb{E}[R_{\mathbf{d}_{0.5},\lambda}] = 12.4$  time units for  $\alpha = 0.5$  and  $\mathbb{E}[R_{\mathbf{d}_{0.9},\lambda}] = 19.2$  time units for  $\alpha = 0.9$ . The percentage increases in expected task response time are 20 and 86 % respectively. Figure 8 presents



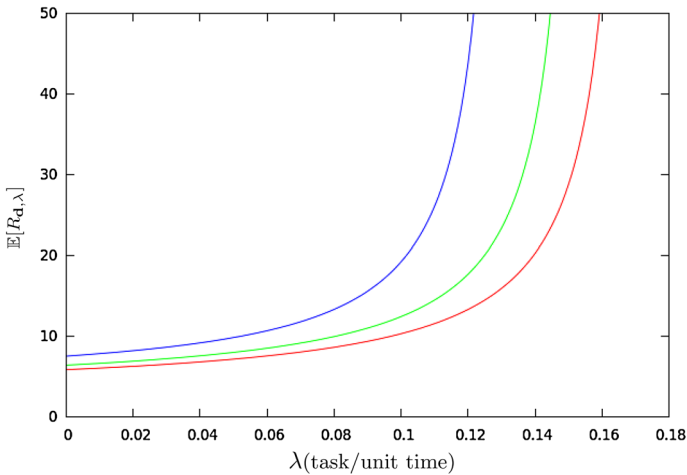


**Fig. 7** Distributions of the range of subtask output buffer arrival times without any delays (*red line*), with delays optimised under  $\alpha = 0.5$  (*green line*) and delays optimised under  $\alpha = 0.9$  (*blue line*). (Color figure online)



**Fig. 8** Distributions of task response time given  $\lambda = 0.1$  without any delays (*red line*), with delays optimised under  $\alpha = 0.5$  (*green line*) and delays optimised under  $\alpha = 0.9$  (*blue line*). (Color figure online)

the corresponding distributions of task response time. For the system without delays the maximum sustainable arrival rate from Eq. 13 is  $\lambda_{\max} < 0.171$  tasks/time unit. After introducing subtask delays to minimise 50th and 90th percentile of the range of subtask completion times this drops to 0.156 and 0.133 tasks/time unit respectively. This is illustrated in Fig. 9.



**Fig. 9** Expected response time of split–merge system for various customer arrival rates without any delays (red line), with delays optimised under  $\alpha = 0.5$  (green line) and delays optimised under  $\alpha = 0.9$  (blue line). (Color figure online)

## 7 Conclusions and future work

This paper has presented a methodology for controlling variability in split–merge systems. Here variability is defined in terms of a given percentile of the range of arrival times of subtasks in the output buffer, and is controlled through the application of judiciously chosen deterministic delays to subtask service times. The methodology has three main building blocks. The first is an exact analytical expression for the distribution of the range of subtask output buffer arrival times over  $n$  heterogeneous servers in a split–merge system. This is a natural generalisation of the well-known order statistics result for the distribution of the range taken over  $n$  homogeneous servers. The second is the introduction of deterministic subtask delays into the aforementioned expression. The third is an optimisation procedure which yields the vector of subtask delays which minimises a given percentile of the range of subtask output buffer arrival times. We also quantified the impact of our optimisation on the system stability and mean task response time. We presented two case studies which showed that the choice of percentile can have a significant impact on the optimal delay vector and the “bottleneck” server.

As previously mentioned fork–join systems are significantly less analytically tractable than split–merge systems. However, they are more realistic abstractions of many real world systems on account of their less-constrained task synchronisation. Consequently a natural future direction of this work is to try and generalise our results to fork–join systems. In line with previous research we believe we are unlikely to find an exact analytical expression for the distribution of the range of join buffer arrival times. However, a numerical approach and/or an analytical approximation may be possible.

Finally, the scalability of our methodology to very large split–merge systems with 100+ service nodes is currently an open question. However, large-scale problems are sometimes encountered when modelling real-life systems. Consequently we will conduct experiments to assess the scaling behaviour of our methodology. It may be beneficial to devise an approach that makes use of parallel computations using for example Message Passing Interface (MPI).

**Open Access** This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

## References

- Ali, M. M., & Gabere, M. N. (2010). A simulated annealing driven multi-start algorithm for bound constrained global optimization. *Journal of Computational and Applied Mathematics*, 223(10), 2661–2674.
- Baccelli, F., Makowski, A. M., & Shwartz, A. (1989). The fork–join queue and related systems with synchronization constraints: Stochastic ordering and computable bounds. *Advances in Applied Probability*, 21(3), 629–660.
- Baccelli, F., Massey, W. A., & Towsley, D. (1989). Acyclic fork–join queuing networks. *Journal of the ACM*, 36(3), 615–642.
- Bolch, G., et al. (2006). *Queueing networks and Markov chains*. Hoboken, NJ: Wiley.
- Brent, R. P. (2002). *An algorithm with guaranteed convergence for finding a zero of a function. Algorithms for minimization without derivatives*, Dover Books on Mathematics. Mineola, NY: Dover Publications.
- Brent, R. P. (1971). An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14(4), 422–425.
- Burden, E. F., & Burden, R. L. (2006). *Numerical methods. Cram101 textbook outlines* (3rd ed.). Ventura, CA: Academic Internet Publishers.
- Cao, G., & West, M. (1997). Computing distributions of order statistics. *Communications in Statistics*, 26(3), 755–764.
- David, H. A. (1980). *Order statistics. Wiley series in probability and mathematical statistics*. New York: Wiley.
- David, H. A., & Nagaraja, H. N. (2005). *The non-IID case. Order statistics* (3rd ed., pp. 95–120). New York: Wiley.
- Flatto, L. (1985). Two parallel queues created by arrivals with two demands II. *SIAM Journal on Applied Mathematics*, 45(5), 861–878.
- Flatto, L., & Hahn, S. (1984). Two parallel queues created by arrivals with two demands I. *SIAM Journal on Applied Mathematics*, 44(5), 1041–1053.
- Harrison, P. G., & Zertal, S. (2007). Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation*, 64(7–8), 664–689.
- Heidelberg, P., & Trivedi, K. S. (1983). Analytic queueing models for programs with internal concurrency. *IEEE Transactions on Computers*, 32(1), 73–83.
- Kim, C., & Agrawala, A. (1989). Analysis of the fork–join queue. *IEEE Transactions on Computers*, 38(2), 250–255.
- Lebrecht, A., & Knottenbelt, W. J. (2007). *Response time approximations in fork–join queues*. 23rd Annual UK Performance Engineering Workshop (UKPEW).
- Lebrecht, A. S., Dingle, N. J., & Knottenbelt, W. J. (2011). Analytical and simulation modelling of zoned RAID systems. *The Computer Journal*, 54(5), 691–707.
- Lewis, R. M., Shepherd, A., & Torczon, V. (2007). Implementing generating set search methods for linearly constrained minimization. *SIAM Journal on Scientific Computing*, 29(6), 2507–2530.
- Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4), 308–313.
- Nelson, R., & Tantawi, A. N. (1988). Approximate analysis of fork/join synchronization in parallel queues. *IEEE Transactions on Computers*, 37(6), 739–743.
- Sen, P. K. (1970). A note on order statistics for heterogeneous distributions. *The Annals of Mathematical Statistics*, 41(6), 2137–2139.
- Serfozo, R. (2009). *Basics of applied stochastic processes*. Berlin: Springer.
- Sun, J., & Peterson, G. (2012). An effective execution time approximation method for parallel computing. *IEEE Transactions on Parallel and Distributed Systems*, 23(11), 2024–2032.
- Towsley, D., Rommel, C. G., & Stankovic, J. A. (1990). Analysis of fork–join program response times on multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 1(3), 286–303.
- Tsimashenka, I., Knottenbelt, W. (2011). Reduction of variability in split–merge systems. *Imperial College Computing Student Workshop (ICCSW 2011)*, (pp. 101–107).
- Tsimashenka, I., Knottenbelt, W., & Harrison, P. (2012). *Controlling variability in split–merge systems. Analytical and stochastic modeling techniques and applications* (pp. 165–177). Berlin: Springer.

- Varki, E., Merchant, A., Chen, H. The M/M/1 fork-join queue with variable sub-tasks (unpublished).
- Varma, S., & Makowski, A. M. (1994). Interpolation approximations for symmetric fork-join queues. *Performance Evaluation*, 20(13), 245–265.
- Zaharia, M., et al. (2010). *Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling*. Proceedings of the 5th European Conference on Computer Systems (EuroSys '10), (pp. 265–278).