Controlling Variability in Split-Merge Systems

Iryna Tsimashenka William Knottenbelt Peter Harrison

Imperial College London, 180 Queen's Gate,
London SW7 2AZ, United Kingdom,
Email: {it09,wjk,pgh}@doc.ic.ac.uk

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. A case study illustrates the applicability of our proposed approach.

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 [10], parallel job processing in MapReduce environments comprising several compute nodes [19], and the assembly of customer orders made up of multiple items in the highly-automated warehouses of large online retailers [15].

The importance of performance prediction in such systems has been long appreciated by performance modellers who have devised appropriate abstractions for their representation, most notably *split-merge* queueing systems and their less synchronised – but analytically much less tractable – counterparts, *fork-join* queueing systems [2]. Understandably, for both kinds of model, the primary focus of research work to date has been on the computation of moments of response

time, most especially the mean. For example, Harrison and Zertal present an approximation for moments of the maximum of service times in a split-merge queueing system with general heterogeneous service times [8]; this gives an exact result in the case of exponential queues. For fork-join systems with homogeneous Markovian service time distributions, Nelson and Tantawi describe a technique which yields approximate upper and lower bounds on the mean response time as a function of the number of servers [13]. For the same system, Varki et al. [17] present approximate bounds on mean response time. Varma and Makowski [18] use interpolation between light and heavy traffic modes to approximate the mean response time for a homogeneous system of fork-join M/M/1 queues. The same fork-join system was considered in [9], where the maximum order statistic provides an easily-computable upper bound on response time.

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 [15], 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 [16] 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 technical contribution of this paper 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 present an exact analytical expression for the distribution of the range of n samples taken from heterogeneous distributions $F_i(t)$ $(i = 1, \ldots, n)$. 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.

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 presents a case study which demonstrates the applicability of our work. Section 6 concludes and considers appropriate directions for future work.

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 λ), 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 [2]. 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

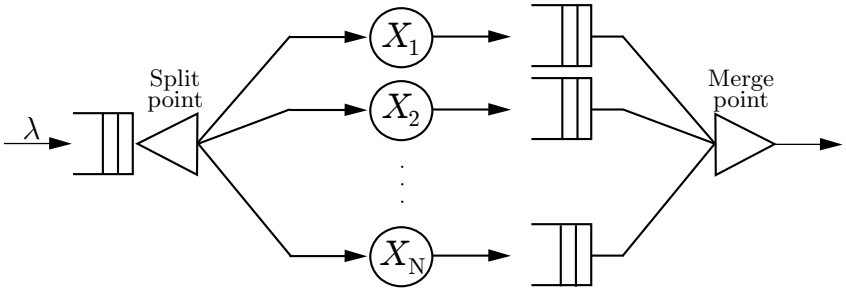

Fig. 1: Split–Merge queueing model.

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. Split-merge systems can also be said to be a more conservative type of fork-join system in the sense that analysis of task response time in a split-merge system yields an upper bound on task response time in a fork-join system [9].

2.2 Theory of Order Statistics [6].

Definition: Let the increasing sequence $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ be a permutation of the real valued random variables $X_1, X_2, \ldots, X_n$, i.e. the X_i arranged in ascending order $X_{(1)} \leqslant X_{(2)} \leqslant \ldots \leqslant X_{(n)}$. Then $X_{(i)}$ is called the *ith order statistic*, for $i = 1, 2, \ldots, n$. 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.

Distribution of the kth-Order Statistic (iid case)

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

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

Likewise, the cdf of the minimum statistic is:

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

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

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

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" ϵ, 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 $n!/((r-1)!1!(n-r)!)$ is the number of ways of doing this, given that the X_i are stochastically indistinguishable.

The joint density function of the rth and sth order statistics $X_{(r)}, X_{(s)}$, where $(1 \leqslant r < s \leqslant n)$, 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, for $x < y$ we have:

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

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

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

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

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 rth and sth 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:

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

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

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

As noted in [6], this equation follows intuitively by noting that the integrand (multiplied by an infinitesimal quantity $\mathrm{d}x$) 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 $X_1, \ldots, X_n$ 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 [7]. This includes the work of Sen [14], 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 ith heterogeneous order statistic are considered in [5], with special consideration of recurrence relations among distribution functions of order statistics.

Distribution of the rth Heterogeneous Order Statistic

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

$$\begin{aligned} F_{(r)}(x) &= Pr\{X_{(r)} \leqslant x\} = Pr\{\text{at least } r \text{ of the } X_i \leqslant x\} \\ &= \sum_{i=r}^{n} \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_i} \prod_{k=1}^{i} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] \end{aligned} \tag{4}$$

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

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

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

and

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

Differentiating Eq. 4 and simplifying yields the pdf:

$$\begin{aligned} f_{(r)}(x) = \sum_{i=r}^{n} \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_i} \Bigg[\sum_{j=1}^{i} \prod_{k=1, k \neq j}^{i} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] f_{\ell_{1j}}(x) - \\ \sum_{j=1}^{n-i} \prod_{k=1}^{i} F_{\ell_{1k}}(x) \prod_{k=1, k \neq j}^{n-i} [1 - F_{\ell_{2k}}(x)] f_{\ell_{2j}}(x) \Bigg] \end{aligned}$$

$$= \sum_{i=r}^{n} \sum_{h=1}^{n} \left[\sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{i-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] f_h(x) - \right.$$

$$\left. I_{i<n} \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_{i}^{h-}} \prod_{k=1}^{i} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i-1} [1 - F_{\ell_{2k}}(x)] f_h(x) \right]$$

$$= \sum_{h=1}^{n} f_h(x) \left[\sum_{i=r}^{n} \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{i-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] - \right.$$

$$\left. \sum_{i=r+1}^{n} \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_{i-1}^{h-}} \prod_{k=1}^{i-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-i} [1 - F_{\ell_{2k}}(x)] \right]$$

$$= \sum_{h=1}^{n} f_h(x) \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\} \in \mathcal{P}_{r-1}^{h-}} \prod_{k=1}^{r-1} F_{\ell_{1k}}(x) \prod_{k=1}^{n-r} [1 - F_{\ell_{2k}}(x)]$$

where $I_\bullet$ is the indicator function and $\mathcal{P}_i^{h-}$ is the set of all 2-set partitions of $\{1, 2, \ldots, n\} \setminus \{h\}$ 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 $1 \leqslant r < s \leqslant n$, follows similarly as:

$$f_{(r)(s)}(x, y) = \sum_{1 \leq h_1 \neq h_2 \leq n} f_{h_1}(x) f_{h_2}(y) \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3\} \in \mathcal{P}_{r-1, s-r-1}^{h_1-, h_2-}} \prod_{k=1}^{r-1} F_{\ell_{1k}}(x) \times \quad (5)$$

$$\prod_{k=1}^{s-r-1} [F_{\ell_{2k}}(y) - F_{\ell_{2k}}(x)] \prod_{k=1}^{n-s} [1 - F_{\ell_{3k}}(y)]$$

where $\mathcal{P}_{i_1, i_2}^{h_1-, h_2-}$ is the set of all 3-set partitions of $\{1, 2, \ldots, n\} \setminus \{h_1, h_2\}$ 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 $1 \leq h_1 \neq h_2 \leq n$.

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 $t_{rs} = y - x$:

$$f_{(r:s)}(t_{rs}) = \sum_{1 \leq h_1 \neq h_2 \leq n} \int_{-\infty}^{\infty} f_{h_1}(x) f_{h_2}(x + t_{rs}) \quad (6)$$

$$\sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3\} \in \mathcal{P}_{r-1, s-r-1}^{h_1-, h_2-}} \prod_{k=1}^{r-1} F_{\ell_{1k}}(x) \prod_{k=1}^{s-r-1} [F_{\ell_{2k}}(x + t_{rs}) - F_{\ell_{2k}}(x)] \prod_{k=1}^{n-s} [1 - F_{\ell_{3k}}(x + t_{rs})] dx$$

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

$$f_{(1:n)}(t) = \sum_{1 \le h_1 \ne h_2 \le n} \int_{-\infty}^{\infty} f_{h_1}(x) f_{h_2}(x+t) \sum_{\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3\} \in \mathcal{P}_{0,n-2}^{h_1-, h_2-}} \prod_{k=1}^{n-2} [F_{\ell_{2k}}(x+t) - F_{\ell_{2k}}(x)] dx$$

$$= \sum_{1 \le h_1 \ne h_2 \le n} \int_{-\infty}^{\infty} f_{h_1}(x) f_{h_2}(x+t) \prod_{k \ne h_1, h_2} [F_k(x+t) - F_k(x)] dx \tag{7}$$

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

$$F_{(1:n)}(t) = \sum_{1 \le h_1 \ne h_2 \le n} \int_{-\infty}^{\infty} f_{h_1}(x) \int_0^t f_{h_2}(x+t') \prod_{k \ne h_1, h_2} [F_k(x+t') - F_k(x)] dx\ dt'$$

$$= \sum_{1 \le h_1 \le n} \int_{-\infty}^{\infty} f_{h_1}(x) \prod_{k \ne h_1} [F_k(x+t) - F_k(x)] dx \tag{8}$$

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 $i = 1, 2, \ldots, n$, the integrand (multiplied by an infinitesimal quantity $\mathrm{d}x$) is the probability that X_i falls into the interval $(x, x + dx]$ and the other $X_j, j \ne i$ 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:

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

This is a useful result, which requires a sum of only n terms. It will form the basis for range-optimisation in split-merge systems with heterogeneous service time distributions as considered in the next section.

4 Controlling Variability in Split-Merge systems

Introducing Delays

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

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

Here, element d_i of the vector $\mathbf{d}$ denotes the deterministic delay that will be applied before a subtask is sent to server i for processing. 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.

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

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

We assume that, $\forall i$, $f_i(t-d_i) = 0$, $\forall t < d_i$. Similarly, $\forall j$, $F_j(t-d_j) = 0$, $\forall t < d_j$.

Optimisation procedure

In this section we move away from our previous mean-based technique [16] towards a more sophisticated framework for finding delay vectors which provide soft (probabilistic) guarantees on variability. More specifically, for a given probability α, we aim to minimise the 100αth percentile of variability with respect to $\mathbf{d}$. That is, we aim to solve for $\mathbf{d}$ in:

$$\min_{\mathbf{d}} F^{-1}_{range}(\alpha, \mathbf{d}) \quad (12)$$

Put another way, we aim to find that vector $\mathbf{d}$ which yields the lowest value for the 100αth 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 α and $\mathbf{d}$ is performed by means of straightforward numerical integration using the trapezoidal rule. While this is adequate and accurate for almost all continuous service time density and distribution functions, complications arise in the case of the pdf of deterministic service time density functions because of their infinitely thin, infinitely high impulse. We choose to resolve this by replacing the deterministic pdf with delay parameter a by the Gaussian approximation:

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

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

In order to invert Eq. 11 for a given α and $\mathbf{d}$, we make use of the well-known Bisection method [4] which in turn exploits Bolzano's Intermediate Value Theorem. Although it is more computationally expensive than the Newton-Rhapson method, we choose the Bisection method because its gradient-free nature makes it considerably more robust. In circumstances where computational efficiency is a critical concern, we note that it is possible to apply more efficient gradient-free algorithms such as Brent's method [3].

Finally, we explore the optimisation surface of $F^{-1}_{range}(\alpha, \mathbf{d})$ with the initial $\mathbf{d} = \{0, \ldots, 0\}$ using a numerical optimization 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 [12], although we note that a range of more sophisticated (and correspondingly considerably more complex to implement) gradient-free optimisation techniques are also available, e.g. [1,11].

5 Case Study

Consider a split-merge system with 3 parallel servers having heterogeneous service time density functions:

$$f_1(t) = \text{Pareto}(\alpha = 3, b = 3.5) \quad (E[X_1] = 5.25,\ \text{Med}[X_1] = 4.40972,\ \text{Var}[X_1] = 9.1875)$$
$$f_2(t) = \text{Erlang}(n = 2, \lambda = 1) \quad (E[X_2] = 2,\ \text{Med}[\text{X}_2] = 1.67835,\ \text{Var}[X_2] = 2)$$
$$f_3(t) = \text{Det}(5) \quad (E[X_3] = 5,\ \text{Med}[X_3] = 5,\ \text{Var}[X_3] = 0)$$

Without adding any extra delays, it is straightforward to apply Eq. 9 in a simple root finding algorithm (e.g. the Bisection method) to compute the 50th ($\alpha = 0.5$) and 90th ($\alpha = 0.9$) percentile of the range of subtask arrival times as $t = 3.629$ time units and $t = 5.52998$ time units respectively.

Incorporating delays into the distribution of the range of subtask merge 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 $\alpha = 0.5$ for $\mathbf{d}$ yields

$$\mathbf{d} = (0.79335, 3.47083, 0)$$

as shown in Figure 2. We note that in this case the "bottleneck" server is server 3, despite the fact that the server 1 has a higher mean service time than server 3. With the incorporation of the optimal delays, the 50th percentile of the range of subtask arrival times becomes $t = 1.32592$, representing an improvement of 63.4% over the original system configuration without delays.

For $\alpha = 0.9$ we obtain

$$\mathbf{d} = (0, 2.68176, 1.45705)$$

as shown in Figure 3. We note that for this percentile the "bottleneck" switched from server 3 to server 1. With the incorporation of the optimal delays, the 90th percentile of the range of subtask arrival times becomes $t = 3.77626$, representing an improvement of 31.7% over the original system configuration without delays.

Figure 4 shows how the distribution of the range of subtask merge buffer arrival times changes according to the optimised percentile. We note that a change of the optimised percentile can have a significant impact on the quantiles of $F_{range}(t, \mathbf{d})$, according to how the "bottleneck" server shifts.

Although it is not our focus, it is interesting to consider the effect of the subtask delays on the expected task completion time. For a system without delays, the expected task completion time is $E[X_{(n)}] = 5.75712$ time units. After introducing subtask delays in order to minimise the 50th and 90th percentile of the range of subtask processing times, the expected task completion time becomes 6.57628 time units (14% increase) and 7.00894 time units (26%) increase respectively.

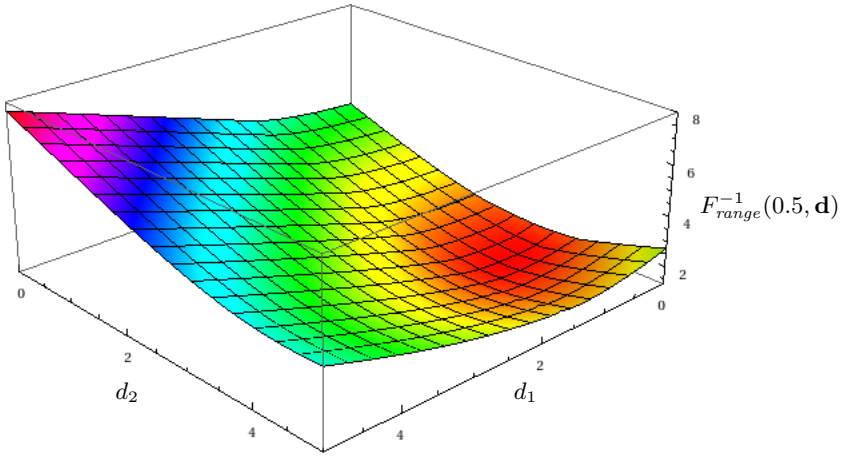

Fig. 2: 50th percentile of the range of subtask merge buffer arrival times for various deterministic processing delays. The optimal delay vector is $\mathbf{d} = (0.79335, 3.47083, 0)$.

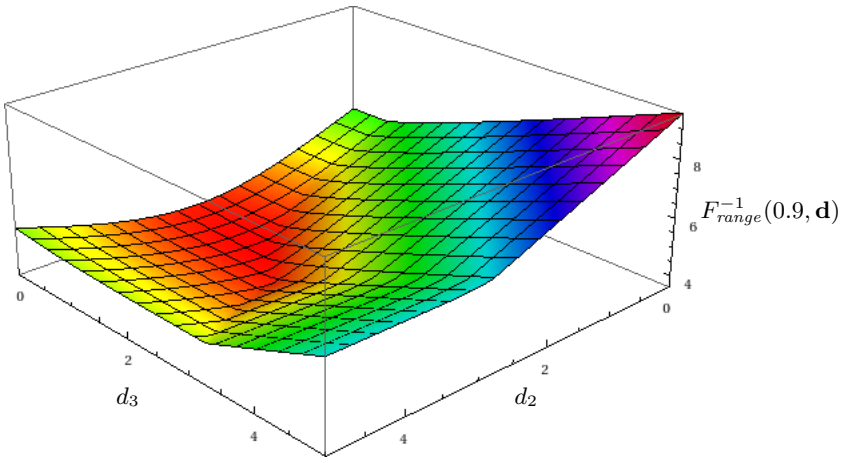

Fig. 3: 90th percentile of the range of subtask merge buffer arrival times for various deterministic processing delays. The optimal delay vector is $\mathbf{d} = (0, 2.68176, 1.45705)$.

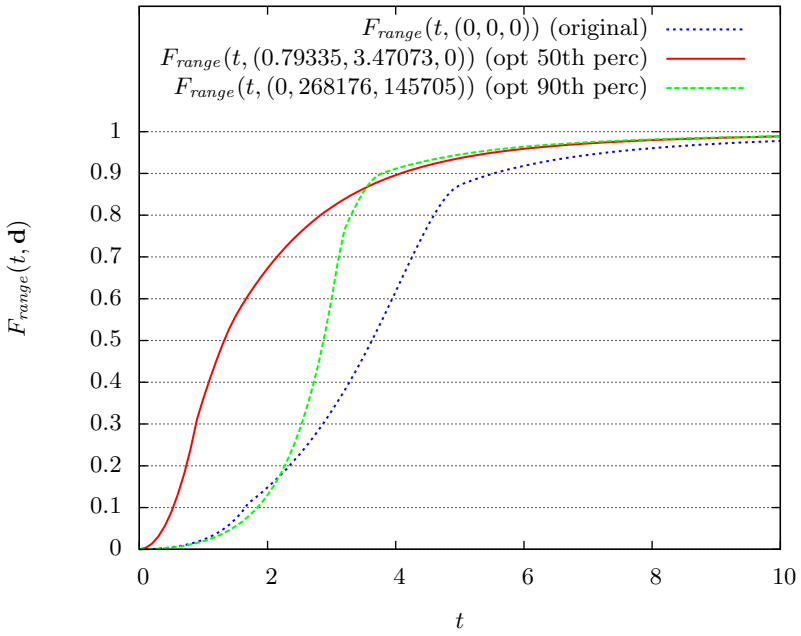

Fig. 4: Distributions of the range of subtask merge buffer arrival times given subtask delays optimised for various percentiles.

6 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 merge buffers, 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 merge 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 a optimisation procedure which yields the vector of subtask delays which minimises a given percentile of the range of subtask merge buffer arrival times. We presented a case study 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 behavior of our methodology. It may be beneficial to devise an approach that makes use of parallel computations using MPI (Message Passing Interface).

References

[1] M. M. Ali and M. N. Gabere. A simulated annealing driven multi-start algorithm for bound constrained global optimization. *Journal of Computational and Applied Mathematics*, 223(10):2661–2674, 2010.

[2] G. Bolch et al. *Queueing Networks and Markov Chains*. John Wiley, 2006.

[3] R.P. Brent. In *Algorithms for Minimization Without Derivatives*, Dover Books on Mathematics, chapter 4. Dover Publications, 2002.

[4] E. F. Burden and R. L. Burden. *Numerical Methods 3rd edition*. Cram101 Textbook Outlines. Academic Internet Publishers, 2006.

[5] G. Cao and M. West. Computing distributions of order statistics. *Communications in Statistics – Theory and Methods*, 26(3):755–764, 1997.

[6] H. A. David. *Order Statistics*. Wiley Series in Probability and Mathematical Statistics. John Wiley, 1980.

[7] H. A. David and H. N. Nagaraja. The non-IID case. In *Order Statistics*, chapter 5, pages 95–120. John Wiley & Sons, Inc., 3rd edition, 2005.

[8] P. G. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Perf. Evaluation*, 64(7–8):664–689, August 2007.

[9] A. Lebrecht and W. J. Knottenbelt. Response Time Approximations in Fork-Join Queues. In *23rd Annual UK Performance Engineering Workshop (UKPEW)*, July 2007.

[10] A. S. Lebrecht, N. J. Dingle, and W. J. Knottenbelt. Analytical and Simulation Modelling of Zoned RAID Systems. *The Computer Journal*, 54(5):691–707, May 2011.

[11] R. M. Lewis, A. Shepherd, and V. Torczon. Implementing generating set search methods for linearly constrained minimization. *SIAM Journal on Scientfic Computing*, 29(6):2507–2530, 2007.

[12] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[13] R. Nelson and A. N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Trans. on Computers*, 37(6):739 –743, 1988.

[14] P. K. Sen. A note on order statistics for heterogeneous distributions. *The Annals of Mathematical Statistics*, 41(6):pp. 2137–2139, 1970.

[15] R. Serfozo. *Basics of Applied Stochastic Processes*. Springer, 2009.

[16] I. Tsimashenka and W. J. Knottenbelt. Reduction of Variability in Split–Merge Systems. In *Imperial College Computing Student Workshop (ICCSW 2011)*, pages 101–107, 2011.

[17] E. Varki, A. Merchant, and H. Chen. The M/M/1 fork-join queue with variable sub-tasks.

[18] S. Varma and A. M. Makowski. Interpolation approximations for symmetric fork-join queues. *Performance Evaluation*, 20(1–3):245 – 265, 1994.

[19] M. Zaharia et al. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proc. 5th European Conference on Computer Systems (EuroSys '10)*, pages 265–278, 2010.