Reduction of Variability in Split–Merge Systems

Iryna Tsimashenka and William Knottenbelt

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

Abstract. We consider an optimisation problem applicable to systems that can be represented as split–merge queueing networks with a limited buffer space for processed subtasks. We assume Poisson arrivals and generally distributed service times. The proposition is to reduce variability in terms of the difference in the times of arrival of the first and last subtasks in systems where the release times of the subtasks can be controlled. This stands in contrast to the overwhelming majority of research which is focused on reduction of mean response time or percentiles of response time. We formally define our notion of variability in split–merge systems and construct an associated cost function and optimisation problem. For two case studies we use simulation to explore the optimisation landscape and to solve the associated optimisation problem.

1 Introduction

Performance analysis has acquired increased importance due to the growing complexity of automated systems. Performance modelling enables an understanding of the relationships between system workload, control parameters and key metrics such as customer response time, system utilisation and buffer occupancy. For systems that involve the flow and processing of customers and resources, queueing models are an appropriate formalism. Optimisation of control parameters allows to minimise, for example, mean response time within given constraints [2].

It has been observed in a recent paper related to the scheduling of Map-Reduce jobs in clusters that delayed scheduling of jobs can counterintuitively lead to greater fairness and a higher level of data locality [8]. In another research, delay scheduling was applied in the context of Quality of Service in networks [6]. Specifically, adding delays to input packets results in shaping the traffic such that packet interarrival times follow an exponential distribution. This construction permits the analysis and optimisation of the network with mathematically tractable Markovian models.

In this paper we show that adding judiciously chosen deterministic delays to subtask processing in split–merge systems can result in a reduction of variability in terms of time difference between the completion of the first and last subtasks in a job. At the same time, corresponding beneficial effects on output buffer occupancy are observed.

A major application area of our approach is automated warehouse systems [7], where partially completed subtasks need to be held in a physical buffer

Andrew V. Jones (Ed.): Proceedings of the 1st Imperial College Student Workshop (ICCSW '11), pp. 101–107. September 29th–30th 2011, London UK.

space. Another application field of this technique is parallel computing where it is sometimes desirable to minimise mean synchronisation time between tasks [5]. In healthcare systems, we can minimise the time patients wait for results following treatment [1]. Lastly, in project scheduling we can reduce mean slack time [9].

The remainder of this paper is organised as follows. Section 2 presents background material relating to split-merge systems. Section 3 presents a formal definition of variability, an associated cost function, and a simple simulation-based optimisation methodology. Section 4 illustrates the application of the methodology in the context of two case studies. Section 5 concludes and considers avenues for future work.

2 Background

As shown in Fig. 1, a split–merge system consists of a queue of waiting tasks (assumed to arrive according to a Poisson process with mean rate λ), a *split* point at which tasks split into subtasks, several (potentially) heterogeneous servers (assumed to process subtasks according to a general service time distribution $F_i(t)$ with mean service time $1/\mu_i$), a buffer for completed subtasks (the merge buffer) and a *merge* point.

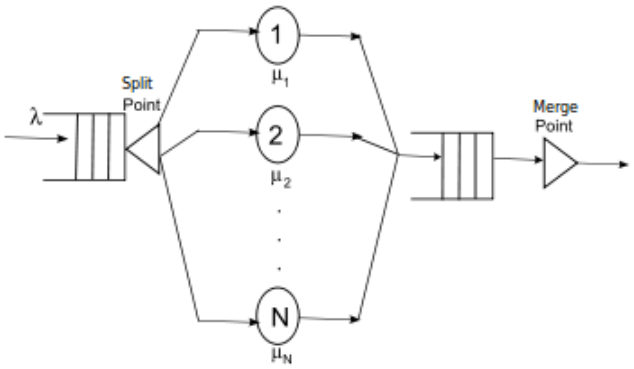

Fig. 1. Split–Merge queueing model.

When all subtask servers are idle and the task queue is not empty, a task is taken from the head of the task queue. This task splits into N subtasks at the *split* point. Each subtask server then processes its allocated subtask. Outgoing subtasks join the merge buffer. When all subtasks belonging a task are present in the merge buffer, the task exits the system via the merge point.

3 Variability in Split–Merge systems

We define the variability of a split–merge system as the mean difference in time between the arrival of the first and last subtasks (belonging to each task) in the merge buffer. Our challenge is to control this variability via the introduction of a vector of delays:

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

Here element d_i of the vector represents the deterministic delay that will be applied before a subtask is sent to server i for processing.

We further define the *cost function* of a split-merge system for a given delay vector $\mathbf{d}$ as:

$$C(\mathbf{d}) = E(X) - E(Y) \tag{2}$$

where X is the random variable denoting the maximum completion time across all subtasks (arising from a particular task), and Y is the random variable denoting the minimum completion time across all subtasks.

Assuming that subtasks at server i are served independently with service time sampled from a distribution function $F_i(t)$, then, taking into account the delay that is applied before each subtask begins processing, X will have cumulative distribution function:

$$F_X(t) \sim \prod_{i=1}^{n} F_i(t - d_i) \tag{3}$$

Here it is assumed, for all i, that $F_i(t - d_i) = 0$ for all $t < d_i$. Similarly, Y has cumulative distribution function:

$$F_Y(t) \sim \overline{\prod_{i=1}^{n} \overline{F_i(t - d_i)}} \tag{4}$$

For a given split-merge system, our challenge is to find that vector $\mathbf{d}$ which minimises $C(\mathbf{d})$. To constrain the solution space while avoiding unnecessary delays to overall mean task processing time, we set $d_i = 0$ for the subtask server(s) with the largest mean service time. We will denote the resulting *vector of optimal delays* as:

$$\widetilde{\mathbf{d}} = (\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_{i-1}}, 0, \widetilde{d_{i+1}}, ..., \widetilde{d_{n-1}}, \widetilde{d_n}) \tag{5}$$

We note that minimising C results in minimum merge buffer utilisation in the split–merge system. This property is particularly relevant in physical systems (e.g. warehouses of major online retailers), which are often constrained in terms of the amount of physical output buffer space available.

Although it is our ultimate goal to establish an efficient analytical procedure for determining $\widetilde{\mathbf{d}}$, in the present paper we apply a simple simulation-based methodology – based on extensions to the JINQS queueing network simulation package [3] – to explore the shape of the cost function landscape and hence to find (near-)optimal solutions for $\widetilde{\mathbf{d}}$.

4 Numerical Results

Case Study 1

Consider a split–merge system with 3 service nodes having the following service time distributions: Uniform(2,3), Pareto(3,1) and Det(5). The latter has the highest expectation, so its optimal delay is set to 0 in $\widetilde{\mathbf{d}}$ from Eq. 5. By the simulation-based algorithm outlined at the end of the previous section, we find the *vector of optimal delays* to be:

$$\widetilde{\mathbf{d}} = (2.5317, 3.7154, 0.0) \tag{6}$$

Fig. 2 displays the CDFs of the service times of the servers before and after application of the optimal delays. Fig. 3 shows how $C(\mathbf{d})$ depends on *delay1* added to the Uniform distribution and *delay2* added to the Pareto distribution.

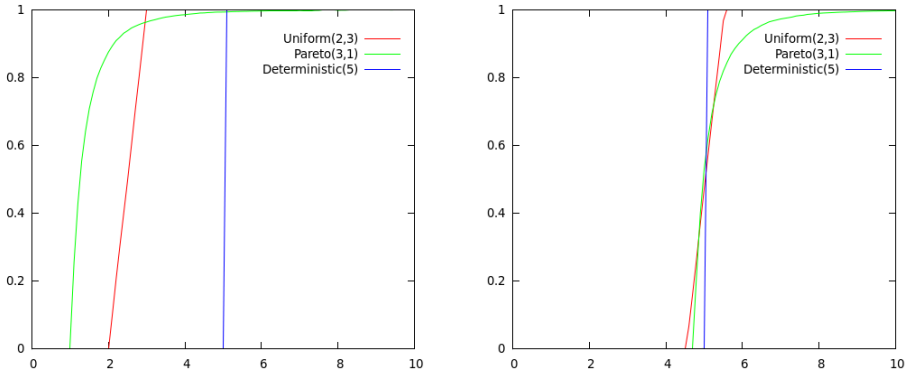

Fig. 2. CDFs of server service times before and after adding the optimal delays.

Case Study 2

Similarly, we show results for optimal delays in a split–merge system with service time distribution functions: Det(5), Erlang(6,1/3), Exp(2). Here the *vector of optimal delays* is as follows:

$$\widetilde{\mathbf{d}} = (11.9386, 0.0, 16.5880) \tag{7}$$

Fig. 4 displays the CDFs of the service times of the servers before and after application of the optimal delays.

Fig. 5 shows how $C(\mathbf{d})$ depends on *delay1* added to the Deterministic distribution and *delay2* added to the Exponential distribution.

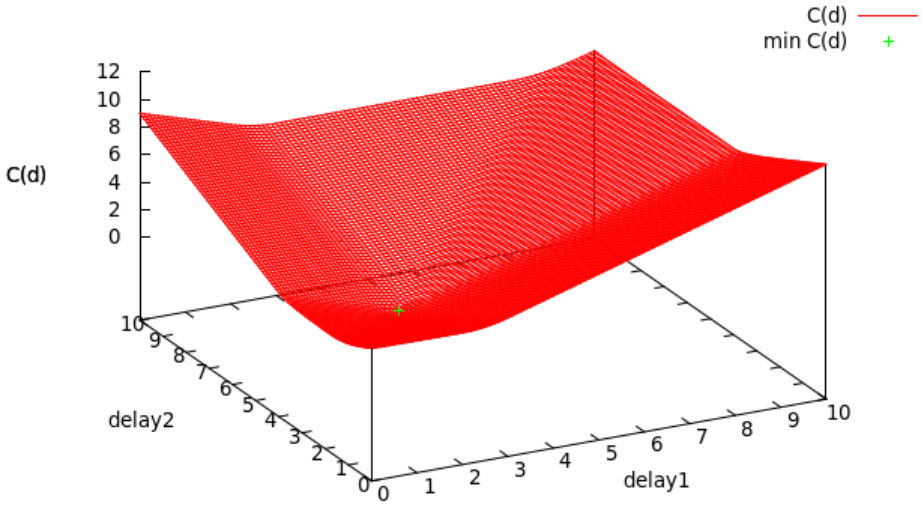

Fig. 3. Surface plot of cost function against delays.

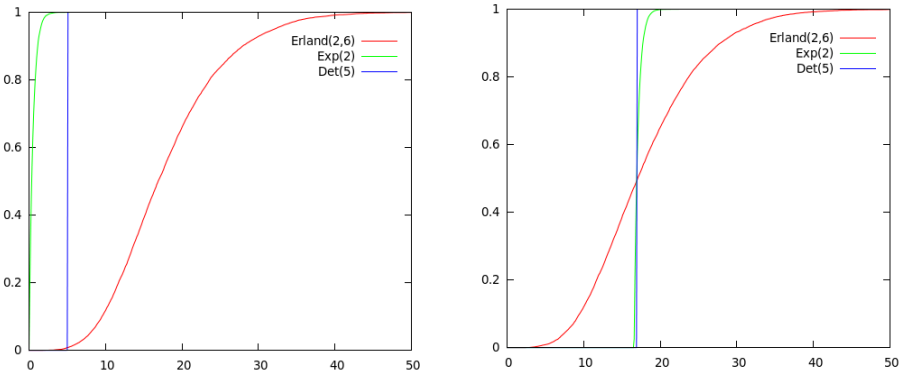

Fig. 4. CDFs of server service times before and after adding the optimal delays.

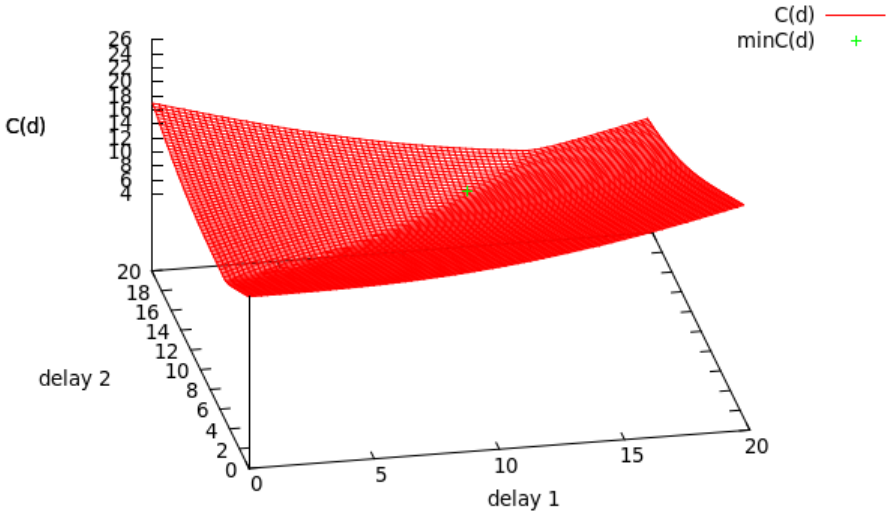

Fig. 5. Surface plot of cost function against delays.

In both cases we note the optimal delays take on different values to those one might intuitively expect (e.g. by subtracting the mean response time of each server from the maximum mean service time).

5 Conclusions and Future Work

This paper has considered the problem of introducing delays into the processing of subtasks in split-merge systems in order to reduce the variability of overall task processing time (and thereby merge buffer occupancy). We have illustrated the use of a simple simulation-based methodology for finding optimal delays in the context of two case studies.

There are several directions in which this work can be extended. Firstly, concurrent parallel generation of the cost function landscape across several computers – with a good corresponding speedup – should be simple to achieve given the embarrassingly parallel nature of the problem. Secondly, it would be interesting to explore if it is possible to find an efficient analytical procedure for determining the vector of optimal delays. It may be that it is necessary to restrict the form of service time distribution functions that can be supported (e.g. requiring that they be continuous). Finally, we intend to investigate the analogous optimisation of a type of less synchronised parallel processing system, namely fork-join systems [4]. The latter have ready application to the modelling of RAID and other computing systems.

References

1. S. W. M. Au-Yeung. *Response Times in Healthcare Systems.* PhD thesis, Imperial College London, January 2008.
2. G. Bolch. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* John Wiley & Sons, Inc., 2006.
3. A. J. Field. *JINQS: An Extensible Library for Simulating Multiclass Queueing Networks.* Imperial College London, August 2006.
4. P. G. Harrison and S. Zertal. Queueing Models with Maxima of Service Times. In *Computer Performance Evaluations, Modelling Techniques and Tools. 13th International Conference, TOOLS 2003, Urbana, IL, USA, September 2-5, 2003*, volume 2794, pages 152–168, September 2003.
5. J. C. S. Lui, R. R. Muntz, and D. Towsley. Computing performance bounds of fork-join parallel programs under a multiprocessing environment. *IEEE Transactions on Parallel and Distributed Systems*, 9(3):295 –311, 1998.
6. D. C. Reeve. *A New Blueprint for Network QoS.* PhD thesis, Computing Laboratory, University of Kent, Canterbury, Kent, UK, August 2003.
7. R. Serfozo. *Basics of Applied Stochastic Processes.* Springer, 2009.
8. M. Zaharia, D. Borthakur, S. Sen, K. Elmeleegy, S. Shenker, and I. Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 265–278, New York, NY, USA, 2010. ACM.
9. J. Zhao, M. M. Dessouky, and S. T. S. Bukkapatnam. Optimal slack time for schedule-based transit operations. *Transportation Science*, 40(4):529–539, 2006.