Split-Merge Model of Workunit Replication in Distributed Computing

Alexander Rumyantsev
Institute of Appiled Mathematical Research, Karelian Research Centre of RAS
11 Pushkinskaya Str., Petrozavodsk, 185910, Russia
Petrozavodsk State University
33 Lenina Pr., Petrozavodsk, 185910, Russia
ar0@krc.karelia.ru

Srinivas R. Chakravarthy
Department of Industrial and Manufacturing Engineering
Kettering University, Flint, MI 48504, USA

1 Introduction

A special class of tasks that demand huge computational resources is known as *embarrassingly parallel*. Such tasks may be split into a large amount of small independent subtasks, known as *workunits*. The *results* of the processed workunits are later assembled. Note that the independence of the workunits makes it inappropriate to run embarrassingly parallel applications on a supercomputer. This is due to the fact that the key feature of a supercomputer (the high speed interconnect) is not utilized during the computation. Thus, one needs to use general purpose instruments like Map–Reduce framework, Beowulf-type clusters for parallel computing (having modern implementations, such as Simple Network of Workstations `SNOW` package for R language), and Desktop Grids (DG), to get results for such embarassingly parallel tasks. It should be pointed out that DG is a specifically designed, inexpensive, and very powerful option.

A common mechanism used in the DG environment (as well as in other modern distributed computing approaches, such as Cloud-based computing), is *replication*. Replication requires each workunit to be processed by multiple hosts, and when the required number of results from these hosts is obtained, the workunit is said to be completed. The replication is known to reduce latency in a high-throughput systems [JSW15a], and to increase redundancy in RAID storage systems [Tho14]. In DG, the replication mechanism is used together with obtaining the *quorum* of results, which allows to reduce the probability of malicious activity, increase redundancy and reduce the response time [BYSS+12, HAH09]. The quorum is the number of results (from one to the number of replicas) that are required to conclude that the workunit is complete. Moreover, when the result is obtained only approximately, or there is no easy algorithm to validate the result (e.g. in prime number searching problem), the quorum mechanism is the ultimate solution. Below we use the term *workunit* to indicate a single subtask of a DG project before replication (and after obtaining a quorum), and the term *result* to denote the replica of the workunit (it will allow to adopt the more common term customer or task to the DG context). We stress that the replication and quorum settings dramatically affect the performance of the distributed computing systems. Thus, a model is required to study the effect of the aforementioned settings on the key performance metrics of the DG system.

The class of queueing models that describes the replication and quorum mechanisms is known as Fork-Join queues. In the Fork-Join n-server queueing system, a single input of workunits is replicated to r results and

In: E. Ivashko, A. Rumyantsev (eds.): Proceedings of the Third International Conference BOINC:FAST 2017, Petrozavodsk, Russia, August 28 - September 01, 2017, published at http://ceur-ws.org

dispatched to r servers, each having its own queue, according to the dispatching discipline (e.g. to the first r servers with least residual workload). The r servers act independently and serve the results according to some queueing discipline (e.g. First-Come-First-Served). After being processed, each result is routed to a single unlimited *join queue*. When the quorum q of results of the same workunit is obtained (and waits in the join queue), the workunit is marked as completed, and the unused results, which are either waiting in the queues, or being served by corresponding servers, are abandoned. Such models, known as (n, r, q) Fork-Join systems, have been studied extensively in [Jos16]. Note however, that, despite the simplicity of the model, only few analytic results are available and that too under stringent conditions (e.g. for the (n, n, n) M/M-type Fork-Join system [Jos16]), and in most cases only the asymptotic bounds or approximations are obtained [BDVD98, BM97]. In particular, the upper bound on the performance of the (n, r, r) Fork-Join system in [JSW15b] is done with the help of another closely related type of model, known as *Split-Merge*, or Split and Match queues. The key property of the Split-Merge system is that the service of the results of a particular workunit is started only after the previous workunit is completed (note, that in particular if $q = 1$ and r divides n, then Split-Merge and Fork-Join models coincide). The Split-Merge queues have been extensively studied since three decades ago. Two-server Split-Merge system was considered in [RP85], the departure process from Split-Merge was studied in [Rao90]. An emerging interest to the Split-Merge systems is related to new applications in Cloud and distributed systems [FL15].

In this work we suggest a Split-Merge model of the replication and quorum mechanism of a DG, which we refer below as (n, r, q) Split-Merge model. Our work extends the (n, n, n) Split-Merge model studied in [FL15], and elaborates the Fork-Join type models studied in [Jos16]. We give exact solutions for particular cases of the model that are analytically tractable, and study the effect of parameters, r and q, on the performance of the model by means of simulation.

The work is organized as follows. In section 2 we give a brief description of the DG technology, providing necessary details for the BOINC-based DG (the standard software used for distributed computing), and discuss the limitations of our approach. In section 3 we present the model, providing the necessary notation. In section 4 we validate the model and discuss the results of numerical experiments. The conclusion and possible extensions of the model are given in section 5.

2 BOINC-Based Desktop Grid

DG technology is a distributed computing technology, that allows to utilize the idle resources (central processor unit time, memory and disk space) of a *host* in order to complete the computational project that consists of loosely coupled workunits. The hosts are either provided by volunteers (the Volunteer Computing, VC [NDS14]), or are a part of some controlled environment, e.g., the computational resources of a company (the Enterprise Desktop Grid, EDG [Iva15]). The project management, workunit generation, replication and result validation via quorum mechanism is performed by the management server of the project. During the project evaluation, the server produces workunits, then generates the necessary number of replicas of a workunit, and dispatches them to the hosts, either on their requests (VC), or impassively (EDG). The server receives and validates the results of computation, and assimilates them. However, it also has to deal with several key difficulties of VC: host availability, trust, malicious activity etc. [NDS14]. We also note, that the VC project requires a lot of social work with the volunteers, however, discussion of these issues is beyond the scope of this paper.

The EDG is a far less studied concept, which, however, has several simplifications compared to the VC. Namely, in most cases the hosts of an EDG may be treated as trusted, reliable, controllable and available. Moreover, since the hosts are directly under control of, say, the enterprise owner, the workunits may be PUSHed to the hosts (i.e. immediately dispatched), rather than PULLed by them (i.e. requested by clients in asynchronous regime, the basic technology used in the VC), and moreover may be canceled on demand [YJKA11]. In the present paper we mostly consider the model of EDG performance controlled by replication and quorum. However, some results of the study may be related to the VC model as well.

We briefly describe the key performance metrics, which are basically studied for the DG [ETA09]:

throughput — the number of workunits completed per unit time;

latency — the time from workunit generation (arrival) to workunit completion (departure);

delay — the time from workunit generation to the beginning of results computation;

starvation — the fraction of the idle time of the hosts, which is not used for computation;

completion time — the average time it takes to complete a fixed number of workunits.

Moreover, the centralized structure of the DG network in some cases makes the project management server become the bottleneck of the system. Indeed, all the tasks related to dispatching, scheduling, result validation and assimilation, are done on a single server. In the current paper we mainly focus on the throughput, latency and delay of the DG model.

3 Stochastic Model of Desktop Grid Performance

We consider an EDG project, that has a fixed number, say n, of identical hosts processing the workunits, generated by management server at instants of a point process with intensity λ. The workunit is immediately replicated and dispatched upon arrival to the hosts that have the least work left to be finished. Let $r \leqslant n$ be the number of results corresponding to a single workunit requested by the management server from the hosts, and let $q \leqslant r$ results be required to constitute a quorum. We assume, that the hosts start evaluating the results of a single workunit simultaneously, and the service time of results on each host is iid with distribution function F_S. Immediately when q results are obtained, computation of $r-q$ incomplete results (if any) is canceled by the management server. We assume that once received, the result is correct, i.e., the earliest q results received are valid. We also assume an unbounded time for result computation, that is, q results are eventually received. We are interested in the performance metrics of such a computation.

Let $S_1, \dots, S_r$ be iid random variables corresponding to the (potential) times of computation of the results of a single workunit. Since the results are started simultaneously, then the time to complete evaluation of the workunit is distributed as the q^{th} order statistics of r r.v., which we denote by $S_{q:r}$. Recall, that

$$P(S_{q:r} \leqslant x) = \sum_{i=q}^{r} \binom{r}{i} F_S^i(x) \overline{F}_S^{r-i}(x),$$

where $\overline{F}_S(x) := 1 - F_S(x)$. In particular, the minimum of $S_1, \dots, S_r$ is distributed as

$$P(S_{1:n} \leqslant x) = 1 - \overline{F}_S^r(x).$$

Thus, the system under study is equivalent to an $M/G/\frac{n}{r}$ queueing system with general service time $S_{q:r}$.

Note however, that it is not necessary, that $F_{S_{q:r}}(x) := P(S_{q:r} \leqslant x)$ belongs to the same class of distributions, as F_S. Obtaining the distribution of q^{th} order statistics analytically is difficult in general. Note that if the service times are exponentially distributed with $F_S(x) = 1 - e^{-\mu x},\ \mu > 0$, then $S_{q:r}$ is exponentially distributed only for $q = 1$, which provides $F_{S_{q:r}}(x) = 1 - e^{-r\mu x}, r \geqslant 1$.

The class of distributions, that is closed under the operation of obtaining order statistics, is phase-type (PH) distribution. A continuous-time PH distribution of order m with representation (β, B) is the time to absorption of the continuous-time Markov chain with a fixed number m of transient states (phases), initial distribution of states $\beta = (\beta_1, \dots, \beta_m)$ and transition subintensity matrix B. The vector $b_0 = -B\mathbf{1} \geqslant 0$ gives the intensity of transitions to absorbing state $m+1$ (for more details on this type of distributions see e.g. [He14, Neu81]). Recall also, that if S has a (β, B) PH distribution, then $\mathrm{E}S = -\beta B^{-1}\mathbf{1}$ and $\mathrm{E}S^2 = 2\beta B^{-2}\mathbf{1}$ [BKF14]. Let F_S be a (β, B) PH distribution.

Now we give a procedure to obtain PH representation of the distribution $F_{S_{q:r}}(x)$ (a detailed discussion on the procedure in case of non-identical PH distributions may be found in [BN17]).

First, we need to extend the phase space. Denote $M_k := \{1, \dots, m\}^k$ the set of m^k lexicographically ordered k-tuples corresponding to the states of k processes that have not yet reached absorbing state. Now we construct the process

$$\{\mathcal{X}(t) := \left(X_1(t), \dots, X_{rq-q(q-1)/2}(t)\right), t \geqslant 0\} \in M := M_r \cup \dots \cup M_{r-q+1},$$

that corresponds to evolution of the r processes until q absorptions, where the number of components equals $r + (r-1) + \dots + (r-q+1)$. Below we explain the evolution of the process in detail. Initially the first r components of the process are independent copies of r.v. with PH(β, B) distribution. Thus, the evolution of the process $(X_1(t), \dots, X_r(t)) \in M_r$ without absorptions is governed by the $m^r \times m^r$ subintensity matrix $B^{\oplus r}$, where

$$B^{\oplus r} = \underbrace{B \oplus \dots \oplus B}_{r}.$$

Intuitively the notation above means, that only one of the r.v. $X_1, \dots, X_r$ makes its own independent transition, that does not end into absorption.

When an absorption occurs, we exclude the absorbing component, switching to the process $(X_{r+1}, \dots, X_{2r-1}) \in M_{r-1}$. Thus, the transitions of the process related to absorption of one of the components are governed by $m^r \times m^{r-1}$ matrix $B_0^{(r)}$. We stress, that the rate $[B_0^{(r)}]_{i,i'}$ of transition from a state $i = (i_1, \dots, i_r) \in M_r$ to the state $i' = (i'_1, \dots, i'_{r-1}) \in M_{r-1}$ depends on the number of possibilities to remove exactly one component of i and arrive at i' as follows

$$[B_0^{(r)}]_{i,i'} = \sum_{t \in N(i,i')} b_{0,i_t}, \quad i \in M_r, i' \in M_{r-1},$$

where $N(i,i') = \{t \leqslant r-k+1 : i_j = i'_j, 1 \leqslant j \leqslant t, i_{j+1} = i'_j, j > t\}$ is the set of indices of such components of i, that, being removed from i, convert it to i'. We note, however, that, by exploiting the lexicographical order of M_r and M_{r-1}, a more straightforward expression for the transition rate matrix appears

$$B_0^{(r)} = b_0^{\oplus r} := b_0 \otimes I_{m^{r-1}} + I_m \otimes b_0 \otimes I_{m^{r-2}} + \dots + I_{m^{r-1}} \otimes b_0, \tag{1}$$

(with obvious conventions if $r < 2$) where I_j is the identity matrix of size $j \geqslant 1$.

Then the transitions of the process $\mathcal{X}(t)$ are governed by the following bidiagonal matrix (where zeroes are the zero matrices of the corresponding dimension)

$$B_{q:r} = \begin{pmatrix} B^{\oplus r} & b_0^{\oplus r} & 0 & 0 & \dots & 0 \\ 0 & B^{\oplus r-1} & b_0^{\oplus r-1} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & B^{\oplus r-q+1} \end{pmatrix}. \tag{2}$$

It remains to note, that the initial distribution of the process is given by

$$\beta_{q:r} = (\beta^{\otimes r}, \mathbf{0}), \tag{3}$$

and the transition intensity to the super-absorbing state corresponding to exactly q absorptions of r independent processes is given by $-B_{q:r}\mathbf{1}$. It may be seen, that row sums of the matrix $B_{q:r}$ are zero for all rows, except the last m^{r-q+1}, which correspond to the phases of process with $q-1$ absorptions.

It may be now seen, that the r.v. $S_{q:r}$ has a $\mathrm{PH}(\beta_{q:r}, B_{q:r})$ distribution.

The PH representation allows to obtain immediately the stability condition and performance measures of the system as follows.

Stability condition of the system

$$\lambda \mathrm{E} S_{q:r} < \frac{n}{r}, \tag{4}$$

where, recall,

$$\mathrm{E} S_{q:r} = -\beta_{q:r} B_{q:r}^{-1} \mathbf{1}.$$

Thus, the maximal throughput of the system equals

$$\theta = \frac{n}{r} [\mathrm{E} S_{q:r}]^{-1}.$$

The explicit forms for the mean stationary delay ($\mathrm{E}D$), as well as the mean latency ($\mathrm{E}W$), are available for $r = n$ (for more details see Appendix E in [Jos16], case $q = r = n$ is considered in [FL15]), since the system in this case is equivalent to an $M/PH/1$ system:

$$\mathrm{E} D = \frac{\lambda \beta_{q:n} B_{q:n}^{-2} \mathbf{1}}{1 + \lambda \beta_{q:n} B_{q:n}^{-1} \mathbf{1}} \tag{5}$$

$$\mathrm{E} W = \mathrm{E} S_{q:r} + \mathrm{E} D. \tag{6}$$

For $r < n$ it is possible either to use approximation [Jos16], or rely on simulation. Alternatively, the solution of continuous-time QBD process corresponding to the $M/PH/\frac{n}{r}$ system may be obtained in terms of the intensity

matrix R, which (except some special cases) is derived only numerically. However, discussion of this approach is beyond the scope of this paper and will be presented elsewhere.

We note, that it is easy to sample exactly from q^{th} order statistics of r independent PH(β, B)-type variables by sampling from a single PH$(\beta_{q:r}, B_{q:r})$ r.v. The drawback of this approach is the growing size of the number of phases that requires to process square matrices of order $\sum_{i=0}^{q-1} m^{r-i} = (m^{r+1} - m^{r-q+1})/(m-1)$. Note however, that it is mostly a technical limitation.

Now we focus on the latency in some particular cases, that are most analytically tractable.

An important particular case is the $M/M/\frac{n}{r}$-type system with exponential interarrival times. This corresponds to an $(n, r, 1)$-type replication with exponentially distributed service times (with intensity μ). In this case we have $(n, r, 1)$ split-merge model, i.e., we send r results and wait for the fastest, we obtain the $Exp(r\mu)$ service distribution of the minimum of r exponentials, and obtain the $M/M/\lfloor\frac{n}{r}\rfloor$-type model. That is, the sojourn time equals

$$\mathrm{E}W = \frac{1}{r\mu} + \frac{C(\lfloor n/r\rfloor, \lambda/(r\mu))}{\lfloor n/r\rfloor r\mu - \lambda}, \tag{7}$$

and in particular, if $r = 1$ (no replication at all), the sojourn time equals

$$\mathrm{E}W = \frac{1}{\mu} + \frac{C(n, \lambda/\mu)}{n\mu - \lambda}, \tag{8}$$

where

$$C(n, \lambda/\mu) = \left[1 + (1-\rho)\frac{n!}{(n\rho)^n}\sum_{k=0}^{n-1}\frac{(n\rho)^k}{k!}\right]^{-1} \tag{9}$$

is the Erlang C-formula. On the other hand, if $r = n$ (full replication), we obtain the $M/M/1$ system with service intensity $n\mu$, and the mean stationary sojourn time is $\mathrm{E}W = \frac{1}{n\mu} + \frac{1}{n\mu-\lambda}$.

4 Numerical Experiments

In this section we discuss the results of several numerical experiments performed for illustrative purposes and quantitative analysis.

First, we study the proximity between the exact value (6) of average stationary sojourn time of the DG model with (n, n, q) replication, and a simple mean estimator obtained by stochastic modeling of the $M/PH/1$-type system with general service time $S_{q:r}$. First we define a 3-phase PH(β, B) distribution by

$$B = \begin{pmatrix} -4 & 2 & 0 \\ 2 & -5 & 1 \\ 1 & 0 & -1 \end{pmatrix}, \quad b_0 = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}, \quad \beta = (0.2, 0.4, 0.4). \tag{10}$$

We set $n = r = 3$ and $q = 2$, and derive $\beta_{q:r}$ from (3), and $B_{q:r}$ from (2). We consider $N = 5 \cdot 10^5$ tasks arriving at epochs of a Poisson process (with intensity λ), with service times generated according to PH$(\beta_{q:r}, B_{q:r})$ distribution. We fix $\lambda = 0.9\mu$ and $\mu = [\mathrm{E}S_{q:r}]^{-1}$. Note, that the average stationary latency in the system obtained from (6) is 7.42. The results of the experiment shown on Fig. 1 show the proximity between the theoretical result and practical estimaion.

Next, we study the dependence of the theoretical value of sojourn time (6) in a system with (n, n, q) replication. We fix the PH(β, B) distribution used in the first experiment. We vary $r = 1, \ldots, 5$ and $q = 1, \ldots, r$. We depict the theoretical value obtained from (6) for each pair (q, r) by the size of a circle on the graph. It may be easily seen form Fig. 2, that the sojourn time is increasing both for increasing q and decreasing r.

Finally, we illustrate the system with exponential service times. We consider (n, r, q) replication with $n = 100$, $q = 1, 2, 3$ and $r = 1, \ldots, 10$. We perform a parameter sweep experiment for all (q, r) pairs s.t. $q \leqslant r$. We also consider the theoretical value (7) for the case $q = 1$. We depict the dependency of the sample mean latency obtained by simulation of the $M/G/\lfloor\frac{n}{r}\rfloor$ system with $S_{q:r}$ obtained as an q-th order statistics of r exponentially distributed r.v. and exponentially distributed interarrival times with $\lambda = 0.9\lfloor\frac{n}{r}\rfloor(\mathrm{E}S_{q:r})^{-1}$. For a fixed pair (q, r) we generate the input of $5 \cdot 10^5$ tasks and evaluate the delay of each task at arrival by means of multiserver system simulation using the `hpcwld` package for R language. The results of the simulation are presented on Fig. 3, that coincide with the previous experiment.

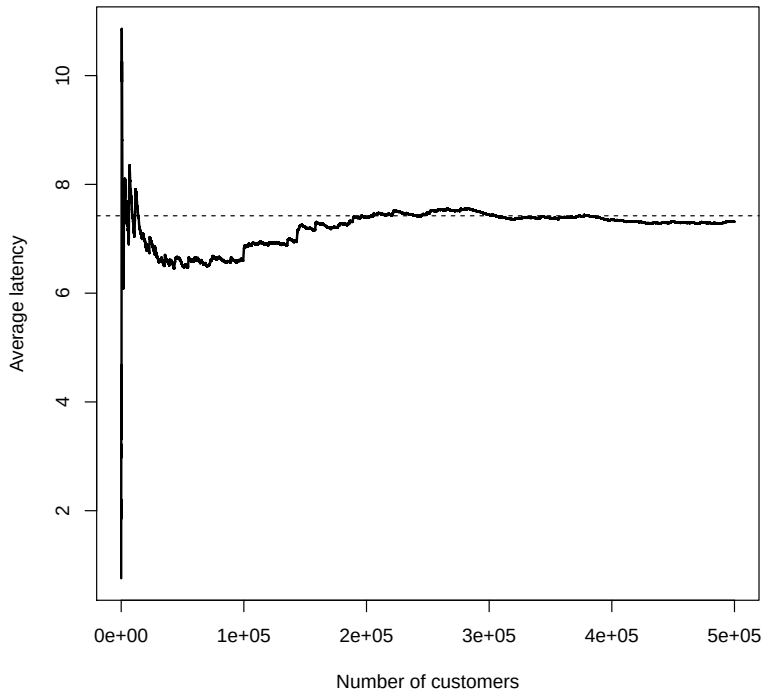

Figure 1: Comparison of the theoretical value of average stationary latency EW obtained from (6) and sample mean for $5 \cdot 10^5$ tasks

5 Final Remarks

In this paper we have presented a model of the EDG system with (n, r, q)-type replication and quorum mechanism. The results of numerical experiments illustrate the practical applicability of the model and show good correspondence to the obtained, as well as earlier known, theoretical results.

Note however, that the numerical experiments mostly cover the so-called log-convex distributions. It is known, that the monotonicity of dependence of the mean stationary latency on the values q, r may the opposite, compared to the log-concave distribution (for a detailed discussion on the impact of log-concavity/convexity on latency for the Fork-Join type models see [Jos16]). Moreover, the dependence may not be monotonic for the distributions with the so-called heavy tails (e.g. Pareto distribution). Thus, a more intensive numerical study is required, and we leave this for future research.

We also note, that an extension to heterogeneous servers may be done taking into accounts results of [AM01], and order statistics of heterogeneous PH distributions are discussed in [BN17].

Acknowledgements

The work of AR is partially supported by RFBR, projects 15-07-02341, 15-07-02354, 15-29-07974, 16-07-00622, and by President RF's grant No.MK-1641.2017.1.

References

[AM01] Søren Asmussen and Jakob R. Møller. Calculation of the steady state waiting time distribution in GI/PH/c and MAP/PH/c queues. *Queueing systems*, 37(1-3):9–29, 2001.

[BDVD98] S. Balsamo, L. Donatiello, and N.M. Van Dijk. Bound performance models of heterogeneous parallel processing systems. *IEEE Transactions on Parallel and Distributed Systems*, 9(10):1041–1056, October 1998.

[BKF14] Peter Buchholz, Jan Kriege, and Iryna Felko. *Input Modeling with Phase-Type Distributions and Markov Models.* SpringerBriefs in Mathematics. Springer International Publishing, Cham, 2014.

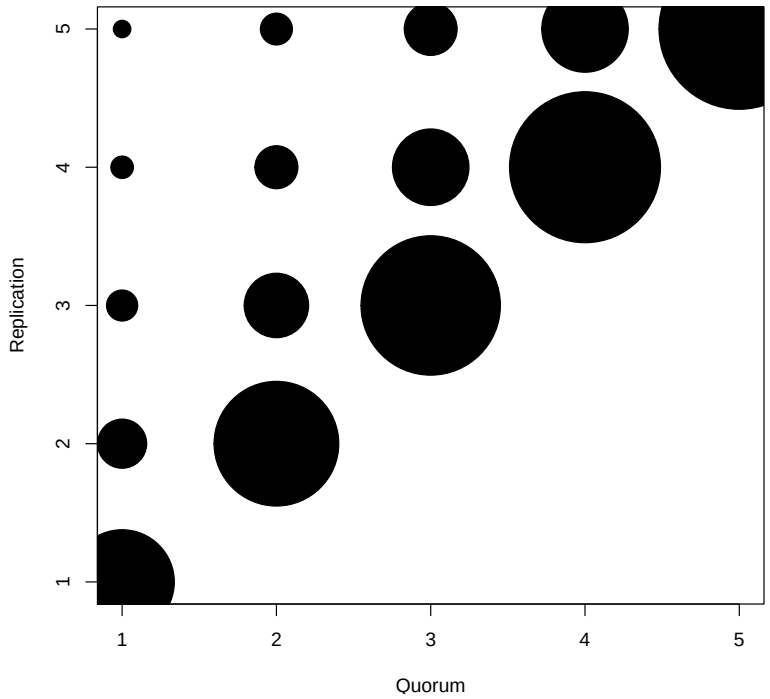

Figure 2: Comparison of the theoretical value of average stationary latency EW obtained from (6) for parameter sweep experiment, with $r = 1, \ldots, 5$ and $q = 1, \ldots, r$.

[BM97] Simonetta Balsamo and Ivan Mura. *On queue length moments in fork and join queuing networks with general service times*, pages 218–231. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[BN17] Mogens Bladt and Bo Friis Nielsen. *Matrix-Exponential Distributions in Applied Probability*, volume 81 of *Probability Theory and Stochastic Modelling*. Springer US, Boston, MA, 2017. DOI: 10.1007/978-1-4939-7049-0.

[BYSS+12] O.A. Ben-Yehuda, A. Schuster, A. Sharov, M. Silberstein, and A. Iosup. ExPERT: Pareto-Efficient Task Replication on Grids and a Cloud. *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 167–178, May 2012.

[ETA09] Trilce Estrada, Michela Taufer, and David P. Anderson. Performance Prediction and Analysis of BOINC Projects: An Empirical Study with EmBOINC. *Journal of Grid Computing*, 7(4):537–554, December 2009.

[FL15] Pierre M. Fiorini and Lester Lipsky. Exact analysis of some split-merge queues. *ACM SIGMETRICS Performance Evaluation Review*, 43(2):51–53, 2015.

[HAH09] Eric Martin Heien, David P. Anderson, and Kenichi Hagihara. Computing Low Latency Batches with Unreliable Workers in Volunteer Computing Environments. *Journal of Grid Computing*, 7(4):501–518, December 2009.

[He14] Qi-Ming He. *Fundamentals of Matrix-Analytic Methods*. Springer New York, 2014.

[Iva15] Evgeny Ivashko. Enterprise desktop grids. In *Proceedings of the Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2015)*, pages 16–21. CEUR Workshop Proceedings, Vol-1502, 2015.

[Jos16] Gauri Joshi. *Efficient redundancy techniques to reduce delay in Cloud systems*. PhD thesis, Massachusetts Institute of Technology, 2016.

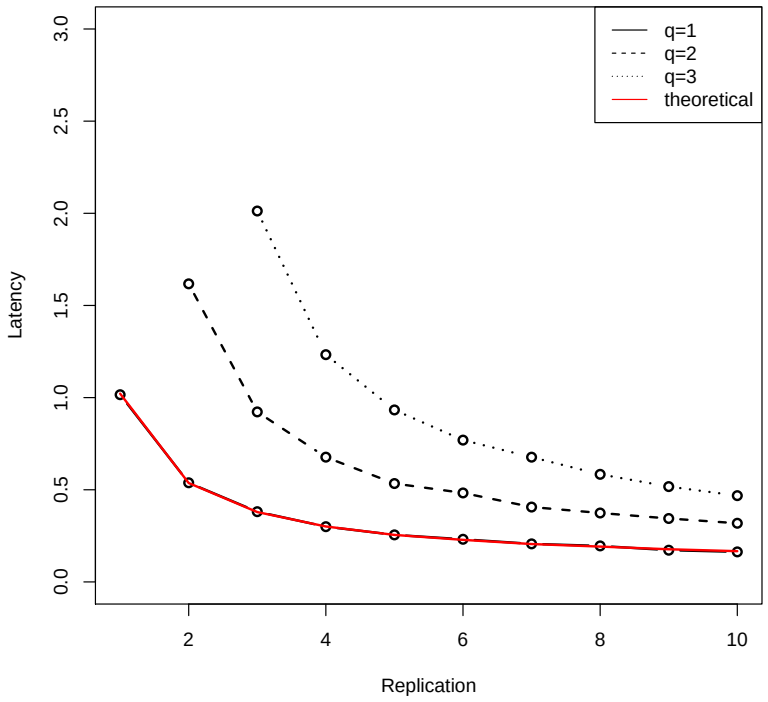

Figure 3: Comparison of the sample mean latency for $q = 1, 2, 3$, $r = 1, \ldots, 10$ and $n = 100$ as a result of parameter sweep experiment with $M/G/\lfloor \frac{n}{r} \rfloor$ queueing system with general service times $S_{q:r}$ observed serving $5 \cdot 10^5$ tasks, compared to the theoretical value of average stationary latency $\mathrm{E}W$ obtained from (7).

[JSW15a] G. Joshi, E. Soljanin, and G. Wornell. Efficient replication of queued tasks for latency reduction in cloud systems. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 107–114, Sept 2015.

[JSW15b] Gauri Joshi, Emina Soljanin, and Gregory Wornell. Queues with redundancy: Latency-cost analysis. *ACM SIGMETRICS Performance Evaluation Review*, 43(2):54–56, 2015.

[NDS14] Muhammad Nouman Durrani and Jawwad A. Shamsi. Volunteer computing: requirements, challenges, and solutions. *Journal of Network and Computer Applications*, 39:369–380, March 2014.

[Neu81] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models*. Johns Hopkins University Press, Baltimore, 1981.

[Rao90] B.M. Rao. On the departure process of the split and match queue. *Computers & Operations Research*, 17(4):349 – 357, 1990.

[RP85] B. M. Rao and M. J. M. Posner. Algorithmic and approximation analyses of the split and match queue. *Communications in Statistics. Stochastic Models*, 1(3):433–456, 1985.

[Tho14] Alexander Thomasian. Analysis of fork/join and related queueing systems. *ACM Comput. Surv.*, 47(2):17:1–17:71, August 2014.

[YJKA11] Sangho Yi, Emmanuel Jeannot, Derrick Kondo, and David P. Anderson. Towards real-time, volunteer distributed computing. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 154–163. IEEE Computer Society, 2011.