Towards the Reproducibility of Using Dynamic Loop Scheduling Techniques in Scientific Applications

Franziska Hoffeins
Technische Universität Dresden
Center for Information Services and High Performance Computing
01062 Dresden, Germany
Email: franziska.hoffeins@tu-dresden.de

Florina M. Ciorba
University of Basel
Department of Mathematics and Computer Science
4051 Basel, Switzerland
Email: florina.ciorba@unibas.ch

Ioana Banicescu
Mississippi State University
Department of Computer Science and Engineering
Mississippi State, MS 39762, USA
Email: ioana@cse.msstate.edu

***Abstract*—Reproducibility of the execution of scientific applications on parallel and distributed systems is a growing interest, underlying the trustworthiness of the experiments and the conclusions derived from experiments. Dynamic loop scheduling (DLS) techniques are an effective approach towards performance improvement of scientific applications via load balancing. These techniques address algorithmic and systemic sources of load imbalance by dynamically assigning tasks to processing elements. The DLS techniques have demonstrated their effectiveness when applied in real applications. Complementing native experiments, simulation is a powerful tool for studying the behavior of parallel and distributed applications. This work is a comprehensive reproducibility study of experiments using DLS techniques published in earlier literature to verify their implementations into SimGrid-MSG [1]. The reproducibility study is carried out by comparing the performance of the SimGrid-MSG-based experiments with those reported in [2]. In earlier work [3] it was shown that a very detailed degree of information regarding the experiments to be reproduced is essential for successful reproducibilty. This work concentrates on the reproducibility of experiments with *variable application behaviour* and *high degree of parallelism*. It is shown that reproducing measurements of applications with high variance is challenging, albeit feasible and useful. The success of the present reproducibility study denotes the fact that the implementation of the DLS techniques in SimGrid-MSG is verified for the considered applications and systems. Thus, it enables well-founded future research using the DLS techniques in simulation.**

***Keywords*-Verification via Reproducibility; Simulation; Scheduling; Dynamic Loop Scheduling; Scientific Applications; SimGrid-MSG; high variance**

I. Introduction

Contributions presented in scientific publications are often based on and/or supported by experiments. In particular, in parallel and distributed computing, the theoretical analysis of algorithms and applications is often complemented by experimental analyses due to the hardware and software complexities, which are challenging to model. Reproducing experiments of such analyses increases the trustworthiness of the reported results, and therefore, of the derived conclusions.

The increase in the numerical complexity of simulation models in conjunction with the rapid increase in computing resources leads, among others, to the need of efficient methods for assigning the work load to processing elements (PE) to achieve a load balanced execution. A PE can be a functional block of a processor (e.g. FPU), a core, a CPU, a workstation, or another type of a processing component. Throughout the present work, a *processing element* refers to a single compute core. Load imbalance between PEs occurs due to application, algorithmic, and/or systemic variability. Dynamic load balancing can be used to achieve a load balanced execution of applications with unpredictably changing of workload, when PEs differ in performance, or when perturbations in the system or in the network occur. Dynamic loop scheduling (DLS) techniques address algorithmic and systemic sources of load imbalance by dynamically assigning tasks to PEs. Over the years, different loop scheduling techniques have been developed. It has been proven that these techniques are highly successful in balancing applications' workload. The use of DLS techniques is not restricted to parallel loops as they can be applied on any collection of independent parallel tasks. Throughout the present work, a *task* refers to a loop iteration, both terms being used interchangeably.

The DLS techniques have exhaustively been analyzed and applied in real scientific applications on real machines. Examples include Monte Carlo simulations, radar signal processing, N-body simulations, computational fluid dynamics on unstructured grids, or wave packet simulations ([4], [5], [6], [7], [8]). The DLS techniques have shown very good results in reducing the load imbalance caused by algorithmic and systemic variances arising over the course of the application execution. Extending upon real experiments, simulations provide the capabilities to demonstrate the strengths of the DLS techniques for *any* probability distribution of the task execution times and availability of PEs. In earlier work, the scalability [9], robustness [10], and resilience [11] of the DLS techniques were investigated for various such distributions using the MSG interface of the SimGrid [1] simulation framework (denoted SimGrid-MSG).

TABLE I: Notation

Notation	Definition
n	number of tasks
p	number of PEs
m	ratio of number of tasks to number of PEs
D	probability distribution of task execution times
h	scheduling overhead
μ	mean of the task execution times
σ	variance of the task execution times
r	number of runs
μ_r	mean of the results of r runs
σ_r	standard deviation of the results of r runs

The present work is an extension of earlier recent work [3] and concentrates on increasing the trustworthiness of the verification via reproducibility of the implementations of the DLS techniques in SimGrid-MSG. The performance reproducibility study is carried out by comparing the SimGrid-MSG-based scheduling experiments with those reported in [2]. Once the SimGrid-MSG implementation is verified, the impact of the overhead of DLS techniques on the performance of scientific applications in heterogeneous computing systems can be assessed.

The current work presents an analysis and a discussion of the performance results obtained via reproducibility of scheduling experiments using DLS techniques published in earlier literature. A short introduction into the DLS techniques and their implementation and incorporation into the SimGrid simulation framework and its MSG interface are presented in Section II. In Section III, an overview of the experiments reported in [2] and the experiments selected for reproducibility in this work are given. The reproducibility results are presented, analyzed, and discussed in Section IV. A description of the reproducibility of this work is outlined in Section V, while conclusions and future directions are given in Section VI.

II. Simulation of Dynamic Loop Scheduling Using SimGrid-MSG

A dominant performance degradation factor in scientific applications is load imbalance. Load imbalance can be induced by different factors. The application itself can have a variable behaviour resulting in imbalance in terms of execution time between the PEs. The underlying numerical model can have variable computational load in different areas of the computational domain, e.g the computational load at the boundaries of the domain can be much higher or much lower than the computational load inside the domain. In addition, load imbalance can occur due to systemic variability caused, for instance, by heterogeneous PEs, or by contention on shared resources.

A powerful tool for achieving a load balanced execution of scientific applications is DLS in which chunks of tasks are dynamically assigned to the available PEs. The two extremes of allocating n tasks to p PEs are static chunking (STAT) and self scheduling (SS). While with STAT the number of tasks is evenly distributed to the PEs, by assigning $\frac{n}{p}$ tasks to each PE before the computation starts, with SS each of the n tasks is assigned to an available PE. The limitation of the one is the strength of the other and vice versa. On one hand, STAT has minimum scheduling overhead due to performing only one scheduling operation per PE. However, this DLS technique is not able to compensate for the variability in the task execution times, thus, resulting in high load imbalance. On the other hand, SS has very high scheduling overhead due to the n scheduling operations accommodated by a good load balancing. The compromise between these two is to dynamically assign chunks of tasks to available PEs using DLS techniques. Over the years, different DLS techniques were developed with the goal to achieve load balanced execution with low scheduling overhead. A comprehensive review of the DLS techniques may be found in [12]. In Table II, an overview of the techniques, their corresponding references and a statement regarding the adaptivity at execution time of each technique are given. Although, not every of the DLS techniques listed in Table II is adaptive at execution time, all listed DLS techniques are dynamic, due to the dynamic assignment of chunks of tasks to the available PEs.

TABLE II: Abbreviations of DLS techniques, reference to publication which first introduced the DLS technique and whether it is adaptive at execution time

Notation	Definition and Reference	Adaptive
FSC	Fixed size chunking [13]	
GSS	Guided self scheduling [14]	
TSS	Trapezoid self scheduling [15]	
FAC	Factoring [4]	
FAC2	Factoring with decreasing factor 2 [4]	
TAP	Taper strategy [16]	
BOLD	Bold strategy [2]	
WF	Weighted factoring [5]	
AWF	Adaptive weighted factoring [17]	✓
AWF-B/C	Variants of adaptive weighted factoring [18]	✓
AF	Adaptive factoring [19]	✓

In earlier work, the scalability [9], robustness [10], and resilience [11] of DLS techniques applied in scientific applications were investigated using the SimGrid-MSG simulation framework [1]. To support the conclusions derived from DLS experiments that use SimGrid-MSG, this work is an extension of earlier work [3] and concentrates on the *verification via reproducibility*. A successful performance reproducibility of experiments supports the conclusions derived from this experiments. Furthermore, the increase in the number of successfully reproduced experiments highlights the trustworthiness of the verification.

The work in [3] constitutes first performance reproducibility study of experiments using DLS techniques published in earlier literature through their implementation into SimGrid-MSG, for the purpose of verifying their performance results. It was shown that the information needed for performance reproducibility of experiments published in earlier literature is, in certain circumstances, insufficient. Nevertheless, the information given in [2] regarding the application, the system (in this case a simulated system) and the execution are very detailed. The performance reproducibility of a set of experiments from [2] (experiment parameters given in Table Va) was successful for the eight DLS techniques considered: STAT, SS, GSS, TSS,

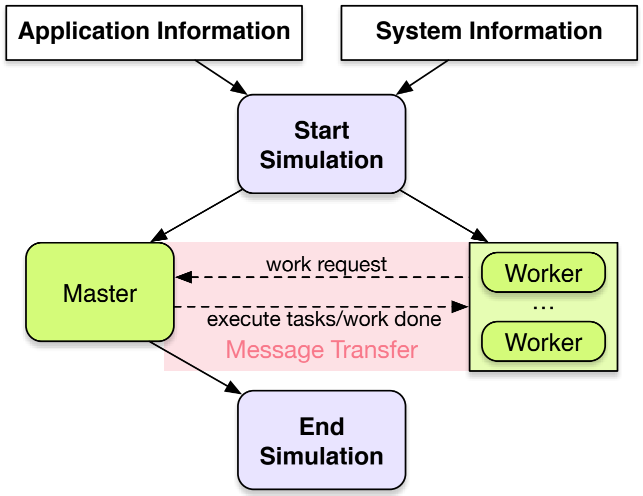

Fig. 1: SimGrid-MSG execution architecture

FAC, FAC2, and BOLD. This work is an extension of the earlier work by increasing the trustworthiness of the results presented in [3] via performance reproducibility of additional experiments from [2] using the simulation framework SimGrid-MSG.

Using the MSG interface of SimGrid, a master-worker execution model is implemented as illustrated in Figure 1. The master and workers are each mapped to an individual PE. Before starting the simulation, information regarding the application and the system needs to be provided. While the application information is given in the SimGrid-MSG deployment file or can be directly implemented in the user code via functions provided by the SimGrid-MSG interface, the system information is specified in the SimGrid-MSG platform file. At the beginning of a simulation and every time a worker finishes the execution of tasks, it sends work request messages to the master. Depending on the chosen DLS technique, when receiving a work request message, the master computes the chunk size and sends this chunk of tasks to the requesting worker. The worker simulates the execution of the tasks, and upon finishing the execution, it sends again a work request message. On completion of all tasks, the master sends final messages to the worker, and the simulation ends. SimGrid-MSG allows to send a specified amount of data with each message transfer. However, in the current work, the assumption is made that the application data is *replicated* and no data transfer is necessary for executing the tasks.

Dynamic scheduling is associated with scheduling overhead. The computation of the chunk size and the allocation of the chunk of tasks to a PE requires a certain amount of time. The authors of [2] assume a fixed delay for computing the chunk size and allocating a chunk of tasks to a PE. They have implemented their own simulator for performing experimental analyses of the DLS techniques, consisting of a central monitor and of $p \geq 2$ initially idle PEs. The central monitor maintains a set of unassigned tasks, initially containing n tasks. At the beginning of the simulation and whenever a PE becomes idle, it accesses the monitor. A chunk of previously unassigned tasks is assigned to the PE, and the PE starts the execution of the tasks in the chunk. The time needed for any PE to process a chunk of tasks is $h + T_1 + \ldots + T_k$, where $h > 0$ is the scheduling overhead and $T_1, \ldots, T_k$ are independent non-negative random variables with a common probability distribution D.

Transferring this information to SimGrid-MSG implies that the master in SimGrid-MSG corresponds to the central monitor of the simulator implemented in [2]. Modelling the fixed scheduling overhead h assumed in [2] is challenging in SimGrid-MSG. Sure, for every scheduling operation there is communication over the network between the master and the requesting worker. The worker sends a work request message and the master replies with a chunk of tasks. This communication needs a certain time, and cannot be fixed, due to potential contention on the network and the potential bottleneck at the master. Therefore, the network connection in SimGrid-MSG is characterized by a very high bandwidth and a very low latency to simulate that the message transfer needs no time.

A visualization[1] of a SimGrid-MSG simulation with $n = 1024$ tasks and $p = 33$ PEs (32 PEs mapped to worker processes and 1 PE mapped to the master process) using the DLS technique FSC is shown in Figure 2. The probability distribution of the task execution times is exponential with mean and standard deviation $\mu = \sigma = 1$. The x-axis is the amount of time in seconds needed for execution of the chunks (marked with different colors) for every worker on the y-axis. It is obvious that there is no time between the execution of chunks for each worker due to the network settings described above.

The authors of [2] were interested in the relative performance of the different DLS techniques and not in the random fluctuations in task execution times. Therefore, the wasted time and not the finishing time was measured. During a run, each of the p PEs is in one of three states. A PE is in *computing* state during the execution of tasks. Between computing states, a PE is in the *coordination* state, waiting for new tasks to execute. Toward the end of the simulation, a PE may be in *idle* state waiting for the last PE to finish its work. The *wasted time* of a PE is defined as the sum of the coordination time and the idle time. The wasted time was measured for a given number of runs r of the same experiment with p PEs. Then, the mean was computed by using the sum of the average wasted times of the r runs divided by the number of runs. In the SimGrid-MSG simulation, the wasted time of a PE is computed by subtracting the time a PE is in computing state from the total simulation time. This value represents the idle time (knowing that the message transfer between the master and worker takes no time, due to very high bandwidth and very low latency of the links between the PEs). The average idle time is the ratio of the summation of the idle times of all PEs to the number PEs. The coordination time is the time needed for sending the chunks of tasks to the PE. This is defined by the scheduling overhead h.

[1]Vampir [20] is used for the visualization of a trace-file generated with the otf2-trace-file-generator [21] from the SimGrid-MSG output

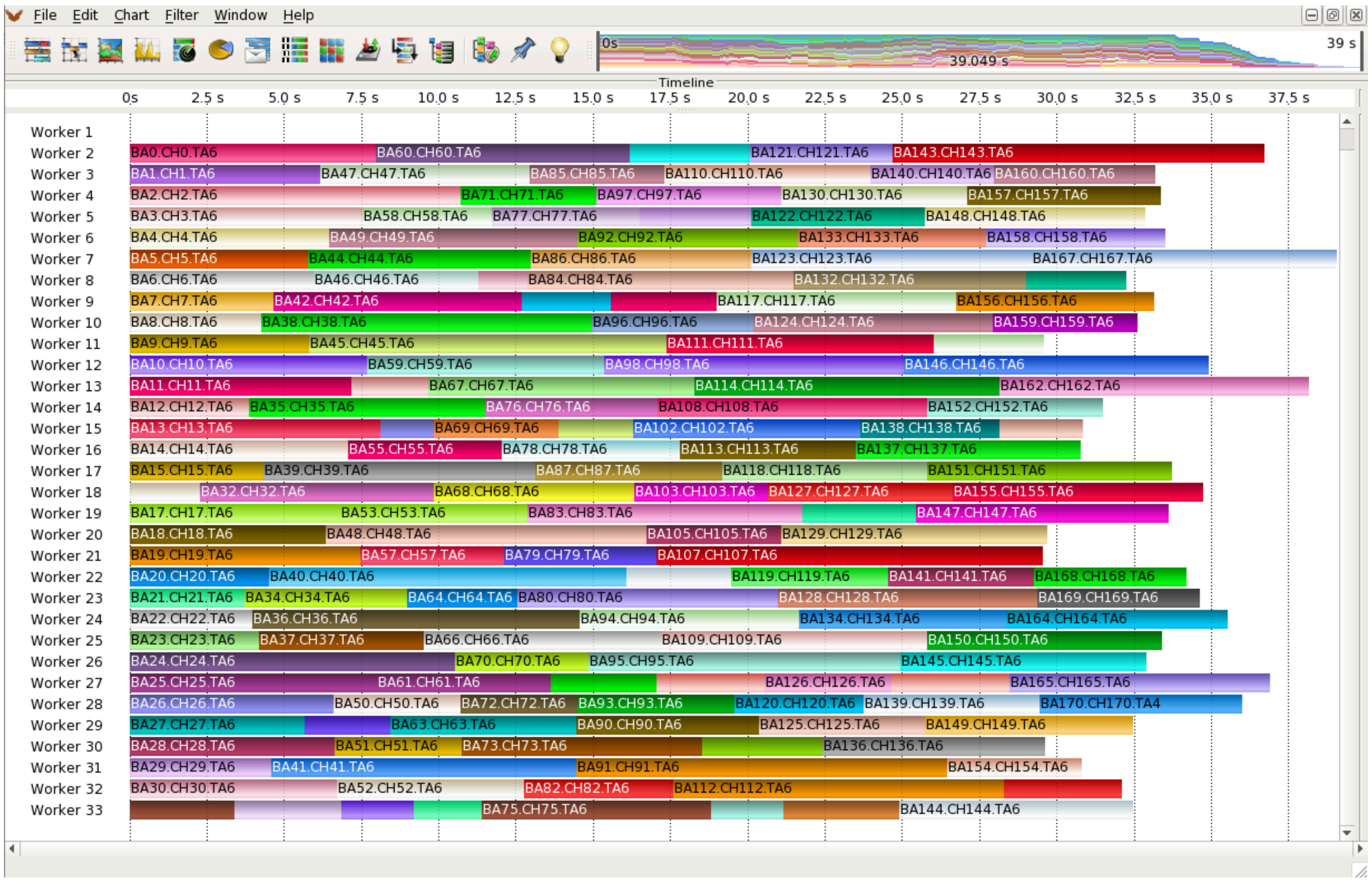

Fig. 2: Visualization of a SimGrid-MSG simulation using the DLS technique FSC with $n = 1024$ tasks scheduled by the master to 32 worker processes, where BAxx.CHyy.TAzz denotes the batch number xx, chunk number yy, and zz is the number of tasks in the chunk.

Thus, the average coordination time is the number of chunks multiplied with h divided by the number of PEs. As the result, the average wasted time of a run can be computed by adding the average idle time and the average coordination time.

The value of interest in this work is the wasted time of a run represented by a random quantity X. To obtain a sample of the t values $X_1, \ldots, X_t$ for X consider a number of t successive runs with identical parameter settings. The mean μ_r can be computed by $\mu_r = \frac{1}{t}\sum_{i=1}^{t} X_i$ and the standard deviation by

$$\sigma_r = \sqrt{\frac{1}{t-1}\sum_{i=1}^{t}(X_i - \mu_r)^2}$$

(following the authors of [2] by dividing by $t-1$ rather than t in the definition of σ_r). The mean μ_r is the reported experiment result in [2] and constitutes the reference value for the reproducibility study. The standard deviation σ_r is not reported in the earlier work. Nevertheless, it is part of this work to explain experiment results presented in Section IV.

III. Experiments in Original BOLD Publication

In the original BOLD publication [2] the analytical analysis including the derivation of the introduced BOLD technique is accompanied by experimental studies of the BOLD technique in comparison to the DLS techniques STAT, SS, FSC, GSS, TSS, FAC, and FAC2. The results of different sets of experiments are reported in six tables. The parameter settings of this six sets of experiments are represented in Tables Va-Vf. The performance of the DLS techniques is analyzed for varying number of PEs p, number of tasks n, probability distribution of the task execution times D, scheduling overhead h, and number of runs r. For Tables Vb and Vd, instead of the number tasks the ratio m of tasks per PE is given. Therefore, the set with the number tasks n is defined as follows:

$$\{n_{jk} | n_{jk} = p_j \cdot m_k \quad \forall j = 1, \ldots, 5; k = 1, \ldots, 4\}.$$

For simulating different applications behaviour, the task execution times were modelled with the aid of the standard Unix random number generators *erand48* and *nrand48* in [2]. For the probability distributions of the task execution times, the uniform (flat), the exponential, and the two-point distributions were used. The notations and definitions of the probability distributions used in this work are given in Table III. The uniform distribution is abbreviated $U(a, b)$, it has constant probability on a given interval $[a, b]$. The mean (or expected value) μ of an uniformly distributed random variable $X \sim U(a, b)$ is given by $\mu = \frac{a+b}{2}$, the standard deviation is $\sigma = \sqrt{\frac{(b-a)^2}{12}}$. The exponential

TABLE III: Probability Distributions

Notation	Definition
$U(a,b)$	Uniform distribution over the interval $[a,b]$
$Exp(\lambda)$	Exponential distribution with rate parameter λ
$B(p)$	Two-point distribution with $P(X=\frac{1}{p})=p$ and $P(X=0)=1-p$

TABLE IV: Probability distributions D for task execution times with mean μ and standard deviation σ considered herein

Notation	Definition	μ	σ
D_1	$U(1.0-0.2\sqrt{3}, 1.0+0.2\sqrt{3})$	1.0	0.2
D_2	$U(1.0-0.5\sqrt{3}, 1.0+0.5\sqrt{3})$	1.0	0.5
D_3	$Exp(1.0)$	1.0	1.0
D_4	$B(\frac{1}{10})$	1.0	3.0
D_5	$B(\frac{1}{101})$	1.0	10.0

distribution is noted $Exp(\lambda)$, the mean μ and the standard deviation σ of an exponentially distributed random variable $X \sim Exp(\lambda)$ is given by $\mu=\sigma=\frac{1}{\lambda}$. The two-point distribution is noted $B(p)$ where p is the success probability. A two-point distributed random variable X on $\{0,\frac{1}{p}\}$ ($X \sim B(p)$) takes the value $\frac{1}{p}$ with success probability p, and the value 0 with failure probability $1-p$. The mean μ is given by $\mu = (1-p)\cdot 0 + p \cdot \frac{1}{p}$, and the standard deviation is $\sigma=\sqrt{p(1-p)\cdot(\frac{1}{p}-0)^2}$. The probability distributions used in this work are shown in Table IV, including the associated parameters.

IV. Reproducibility Results

This section presents the results of the reproducibility efforts of measurements presented in the original BOLD publication [2]. While in earlier work the reproducibility of experiments reported in Table Va was analyzed, this work is an extension of [3] and concentrates on the performance reproducibility of additional measurements from [2]. In the original BOLD publication [2], experiments using eight DLS techniques were performed. The DLS techniques STAT and SS were only measured in the experiments noted in Table Va. Reproducing these experiments was subject of the earlier work [3]. The percentage relative to the original value of the discrepancy between the original value from the publication and the SimGrid-MSG simulation ranged from -1.29% to 0.53% for both techniques. This implies a successful reproducibility, and therefore, a verification of the implementation of these DLS techniques in SimGrid-MSG. Furthermore, STAT and SS are basic DLS techniques and leave little room for errors in their implementation. Therefore, the implementation of the master-worker execution model in SimGrid-MSG leaves little room for errors, as well.

For six of the eight DLS techniques, additional experiments were reported in the original BOLD publication. In the following subsections, the performance reproducibility results of experiments using the techniques FSC (Section IV-A), GSS (Section IV-B), TSS (Section IV-C), FAC (Section IV-D), FAC2 (Section IV-E), and BOLD (Section IV-F) are analyzed. Note

TABLE V: Parameters used in the experiments in original BOLD publication [2]

(a) Experiments in Table I in [2] (reproduced in [3])

Var.	Value(s)
n	$\{2^{10}, 2^{13}; 2^{16}, 2^{19}\}$
p	$\{2^1, 2^3, 2^6, 2^8, 2^{10}\}$
D	D_3
h	0.5
r	$1,000$
DLS	{STAT, SS, FSC, GSS, TSS, FAC, FAC2, BOLD}

(b) Experiments in Table II in [2] (reproduced herein)

Var.	Value(s)
m	$\{2^2; 2^4; 2^6; 2^8\}$
p	$\{2^1, 2^3, 2^6, 2^8, 2^{10}\}$
D	D_3
h	0.5
r	$10,000$
DLS	{FSC, GSS, TSS, BOLD}

(c) Experiments in Table III in [2] (left for further studies)

Var.	Value(s)
n	2^7
p	2^3
D	$\{D_1, D_2, D_3, D_4, D_5\}$
h	$\{0.1, 0.5, 2.0, 5.0, 10.0\}$
r	$100,000$
DLS	{FSC, GSS, TSS, BOLD}

(d) Experiments in Table IV in [2] (reproduced herein)

Var.	Value(s)
m	$\{2^2, 2^4, 2^6, 2^8\}$
p	$\{2^1, 2^3, 2^6, 2^8, 2^{10}\}$
D	D_4
h	2.0
r	$10,000$
DLS	{FSC, GSS, TSS, BOLD}

(e) Experiments in Table V in [2] (reproduced herein)

Var.	Value(s)
n	$\{2^{22}, 2^{25}, 2^{28}, 2^{31}\}$
p	$\{2^1, 2^3, 2^6, 2^8, 2^{10}\}$
D	D_3
h	0.5
r	$1,000$
DLS	{FAC, FAC2, BOLD}

(f) Experiments in Table VI in [2] (left for further studies)

Var.	Value(s)
n	2^{19}
p	2^6
D	$\{D_1, D_2, D_3, D_4, D_5\}$
h	$\{0.1, 0.5, 2.0, 5.0, 10.0\}$
r	$100,000$
DLS	{FAC, FAC2, BOLD}

that the experiments with parameters reported in Tables Vc and Vf are left for further studies, and that this work concentrates on the reproducibility of experiments with parameters given in Tables Vb, Vd and Ve. This additional performance reproducibility study increases the trustworthiness of the results presented in [3], and lays the foundation for further studies of the DLS techniques.

A. FSC

In the original BOLD publication [2], the DLS technique FSC has been analyzed in four sets of experiments . The parameters and the values of those experiments are reported in Tables Va, Vb, Vc, and Vd. While the performance reproducibility of the FSC experiments from Table Va was analyzed in earlier work [3], this work concentrates on the reproducibility of the experiments from Tables Vb and Vd. The experiments in Table Vc are left for further studies.

The experiments from the original BOLD publication reported in Tables Va and Vb are very similar. While in Table Va the number of tasks is $\{2^{10}, 2^{13}, 2^{16}, 2^{19}\}$ and the number of runs is $1,000$, in Table Vb the number of runs is increased by an order of magnitude to $10,000$ and the number of tasks is, in general, decreased to $\{2^2, 2^4, 2^6, 2^8\}$ per PE. The probability distribution of the task execution times is $Exp(1.0)$ with

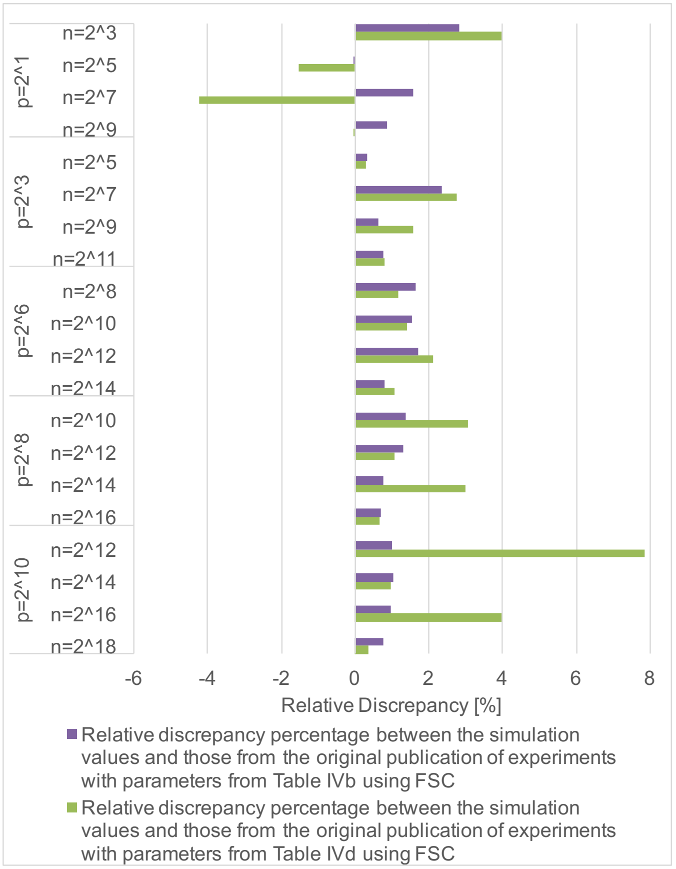

Fig. 3: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using FSC performed for parameters given in Tables Vb and Vd

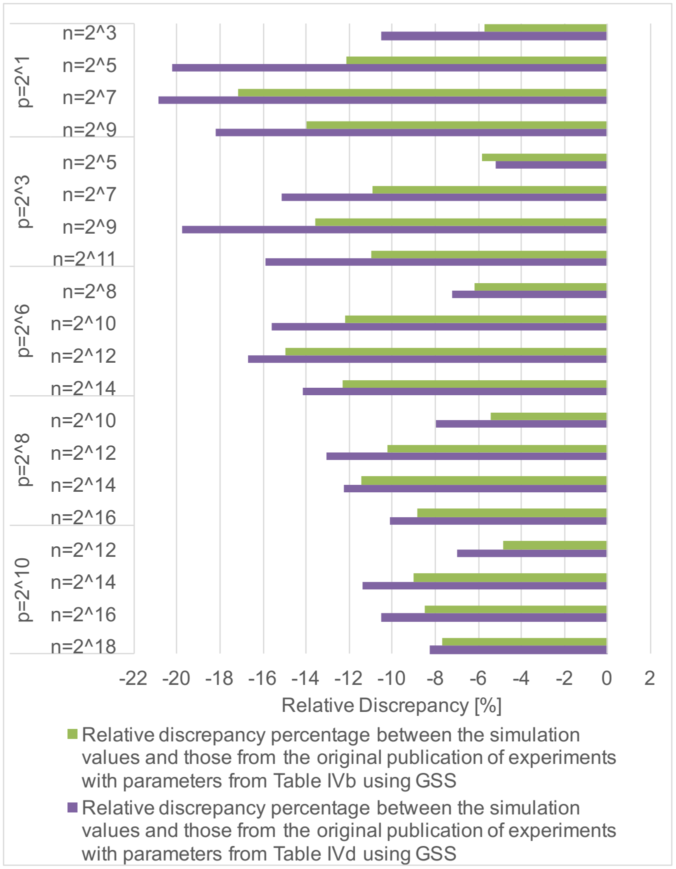

Fig. 4: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using GSS performed for parameters given in Tables Vb and Vd

mean and standard deviation $\mu = \sigma = 1.0$. The scheduling overhead is $h = 0.5$. The experiments reported in Table Vd are characterized by an increased scheduling overhead to $h = 2.0$ and a more variable application behaviour using the two-point probability distribution $B(10)$ to generate the task execution times with a random number generator. This implies a standard deviation of $\sigma = 10$. The number of tasks, PEs, and runs is equal to the values in Table Vb. In general, it could be assumed that the reproducibility results of Table Vb are closer to the original values compared to the reproducibility results of Table Vd. The higher standard deviation of the task execution times leads to more fluctuations of the single runs.

The results of the performance reproducibility of all those 40 experiments (Tables Vb and Vd) are very promising. In Figure 3, the percentage discrepancy relative to the value in the original publication between the values from the original publication and those obtained from the SimGrid-MSG simulation are shown. A negative discrepancy indicates that the average wasted time obtained by the SimGrid-MSG simulation is higher. The discrepancy is bounded by -4.2% and 7.9% which is an acceptable result for reproducibility. As expected, the discrepancy percentage differs with the probability distribution of the task execution times. While the discrepancy of the original value and the SimGrid-MSG value is in the interval $[-0.04\%, 2.8\%]$ for the experiments from Table Vb (exponential distribution with standard deviation $\sigma = 1.0$), this interval increases more than fourfold to $[-4.8\%, 7.9\%]$ for experiments from Table Vd (two-point distribution with standard deviation $\sigma = 10$). However, the results imply the successful performance reproducibility of the FSC experiments.

B. GSS

In the original BOLD publication [2], the experiments using GSS equal the FSC experiments (Tables Va, Vb, Vc, and Vd). The relative discrepancy percentage between the original values and the SimGrid-MSG results of experiments from Tables Vb and Vd are plotted in Figure 4.

The discrepancy percentage is negative for all 40 experiments due to the higher SimGrid-MSG value of the average wasted time. Considering the GSS experiments with parameters reported in Table Vb, only for the experiment with $n = 2^7$ and $p = 2^1$ the relative discrepancy percentage between the original value and the SimGrid-MSG value is higher than $\pm 15\%$ (see Table VIa). The fluctuation of the average wasted times of the different runs of this experiments is very high, the values range from 2.00 to 20.61, the standard deviation is $\sigma_r = 3.1$, which corresponds to 70% of the mean of all runs. The reason

TABLE VI: Mean, minimum, maximum, and standard deviation σ_r of average wasted time of SimGrid-MSG simulations of experiments with parameters given in Tables Vb and Vd using GSS with a relative percentage discrepancy between the original and simulation value higher than 15% (corresponding results are included in Figure 4)

(a) Results of experiments with parameters from Table Vb (1/20 experiments)

		average wasted time (SimGrid-MSG) [s]			
% discr	**$[p, n]$**	**μ_r**	**minimum**	**maximum**	**σ_r**
-17.15%	$[2^1, 2^7]$	4.58	2.00	20.61	3.10

(b) Results of experiments with parameters from Table Vd (8/20 experiments)

		average wasted time (SimGrid-MSG) [s]			
% discr	**$[p, n]$**	**μ_r**	**minimum**	**maximum**	**σ_r**
-20.24%	$[2^1, 2^5]$	10.73	6.00	36.00	4.65
-20.84%	$[2^1, 2^7]$	16.33	8.00	73.00	8.84
-18.22%	$[2^1, 2^9]$	25.57	10.00	120.00	18.22
-15.15%	$[2^3, 2^7]$	18.46	6.50	60.25	7.19
-19.80%	$[2^3, 2^9]$	27.52	9.00	130.25	12.93
-15.92%	$[2^3, 2^{11}]$	39.27	11.50	171.5	24.50
-15.63%	$[2^6, 2^{10}]$	30.99	11.88	69.84	7.73
-16.72%	$[2^6, 2^{12}]$	49.36	17.25	125.69	13.82

my be found in the low number of tasks scheduled to a low number of PEs. Outliers of the task execution times cannot be balanced by the other task execution times. This results in fluctuations of the runs, and both extremes can occur, either load balanced execution or high load imbalance. However, by excluding the outlier of -17.15% discrepancy between the original and the simulation value for the experiments from Table Vb, the discrepancy percentage becomes below 15% which is an adequate result for reproducibility.

For the experiments from Table Vd, the relative discrepancy percentage between the original and SimGrid-MSG results is bounded by -20.8% and -5.2%. The outliers of the experiments where the discrepancy percentage is higher than 15% are listed in Table VIb. Especially for low numbers of PEs in combination with high numbers of tasks, the discrepancy percentage between the original and simulation results is increasing. The reason may be found in the probability distribution of the task execution times which has a standard deviation $\sigma = 10$. The low number of PEs can not compensate the variability in the task execution times. This results in uneven outcome of the different runs of the experiments, which decreases the confidence in the computed mean of the runs, and explains the discrepancy of the mean value to the one from the original publication.

C. TSS

The parameters of the experiments in the original BOLD publication [2] using the DLS technique TSS are equal to the parameters of the experiments using GSS and FSC (Tables Va, Vb, Vc, and Vd). The relative discrepancy percentage between the original values and the SimGrid-MSG results of the experiments from Tables Vb and Vd are plotted in Figure 5.

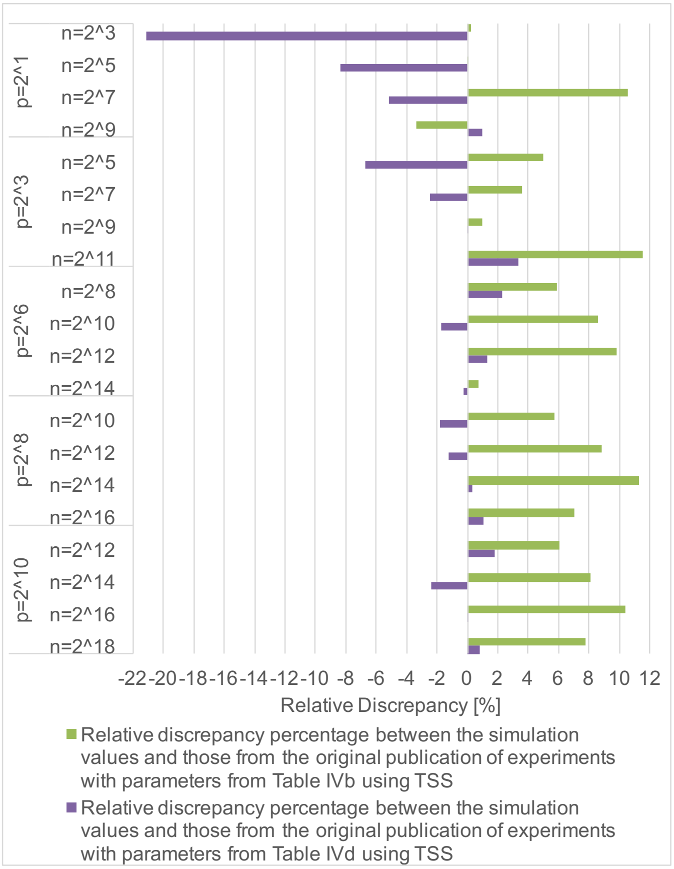

Fig. 5: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using TSS performed for parameters given in Tables Vb and Vd

The discrepancy percentage for both sets of experiments is bounded by 15% excluding the outlier for the experiment from Table Vd with $n = 2^3$ and $p = 2^1$. Using TSS for scheduling a low number of tasks to a low number of PEs where the single task execution times have a high variability leads to variability of the different runs. In addition, for the experiment from Table Vd with $n = 2^3$ and $p = 2^1$ the number of scheduling operations is five. This implies an average scheduling overhead per PE of $5s$ ($h = 2.0s$). This high scheduling overhead compared to the overall original value of the average wasted time of $5.99s$ in conjunction with the variability of the runs due to the standard deviation of $\sigma = 10$ of the task execution time distribution leads to this outlier. Excluding the outlier of -21.14% discrepancy between the original and the simulation value for the experiments from Tables Vb and Vd, the discrepancy percentage ranges from -8.32% to 11.53% which is an acceptable result for reproducibility.

D. FAC

The DLS technique FAC was used in the original BOLD publication in the experiments with reported parameters in

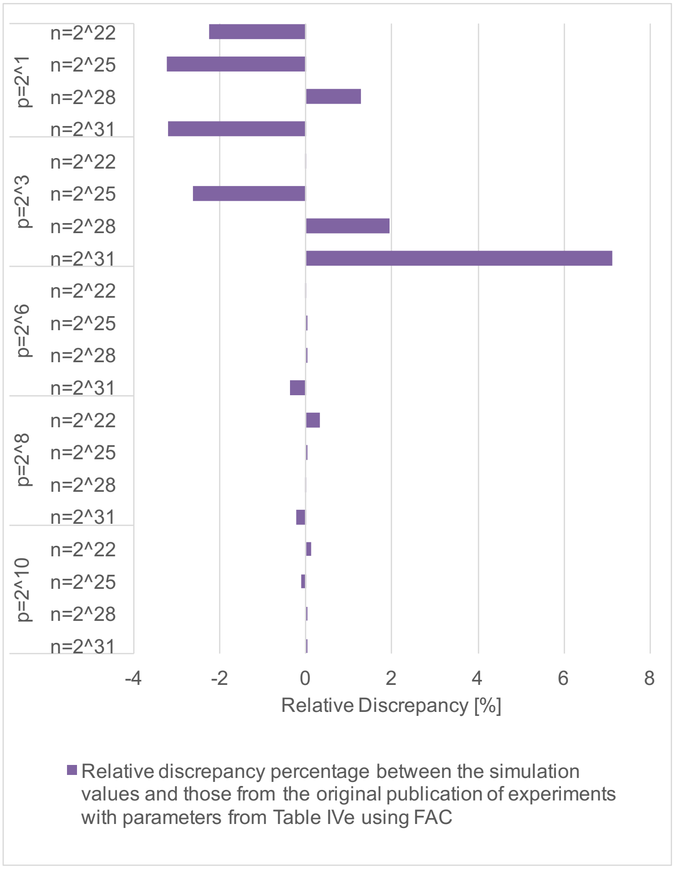

Fig. 6: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using FAC performed for parameters given in Table Ve

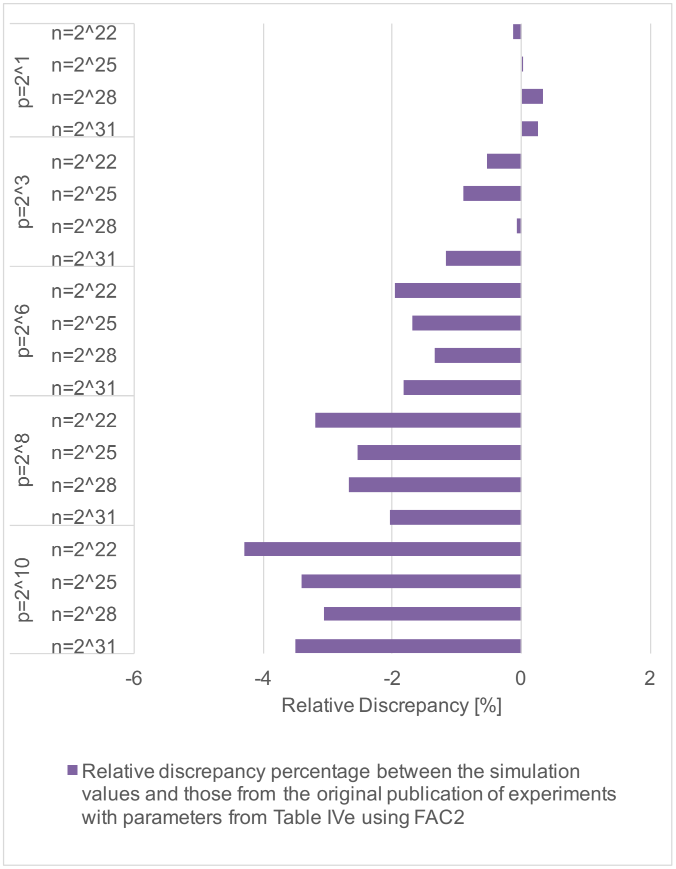

Fig. 7: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using FAC2 performed for parameters given in Table Ve

Tables Va, Ve, and Vf. While the reproducibility study of experiments in Table Va was subject in earlier work [3], this work extends the earlier work by concentrating on the performance reproducibility of experiments from Table Ve. The experiments reported in Table Vf are characterized by using different probability distributions for generating the task execution times. Reproducing these experiments is left for future work.

The parameters of the experiments reported in Table Ve are very close to the parameters of the experiments given in Table Va. The single difference is the increased number of tasks to $\{2^{22}, 2^{25}, 2^{28}, 2^{31}\}$ of the experiments in Table Ve. The number of PEs is $\{2^1, 2^3, 2^6, 2^8, 2^{10}\}$, the probability distribution of the task execution times is $Exp(1.0)$ with mean and standard deviation $\mu = \sigma = 1.0$, the scheduling overhead is $h = 0.5$, and the number of runs is $1,000$ for both experiments. The relative discrepancy percentage between the original values and the SimGrid-MSG results is shown in Fig. 6.

The discrepancy between the values from the original publication and the SimGrid-MSG values ranges from $-43.3s$ to $48.5s$. This results in a discrepancy percentage between -3.2% and 7.1% for the FAC experiments from Table Ve, which is an adequate result for performance reproducibility. In earlier work, an outlier for FAC was detected for the experiment with 2^{19} tasks and 2^1 PEs with a relative discrepancy percentage between the original value and the SimGrid-MSG result of about 26% [3]. The tendency that the discrepancy percentage of experiments with a high number of tasks scheduled to a low number of PEs is higher compared to experiments with a higher number of PEs still holds. Though, the discrepancy percentage for the experiments with 2^1 and 2^8 PEs from Table Ve is bounded by -3.2% and 7.1%. However, the fact that the discrepancy percentage is in the interval $[-3.2\%, 7.1\%]$ implies a successful reproducibility.

E. FAC2

In the original BOLD publication [2], the experiments using FAC2 equal the FAC experiments (Tables Va, Ve, and Vf). The relative discrepancy percentage between the original values and the SimGrid-MSG results of experiments from Table Ve is shown in Fig. 7.

The percentage discrepancy between the original value and the value obtained by SimGrid-MSG simulation is decreasing for the experiments with parameters given in Table Ve with the increasing number of PEs. This DLS technique produces results contrary to the FAC results. Although, the techniques are very similar, the FAC takes the mean μ and the standard deviation σ of the task execution times into account when computing the

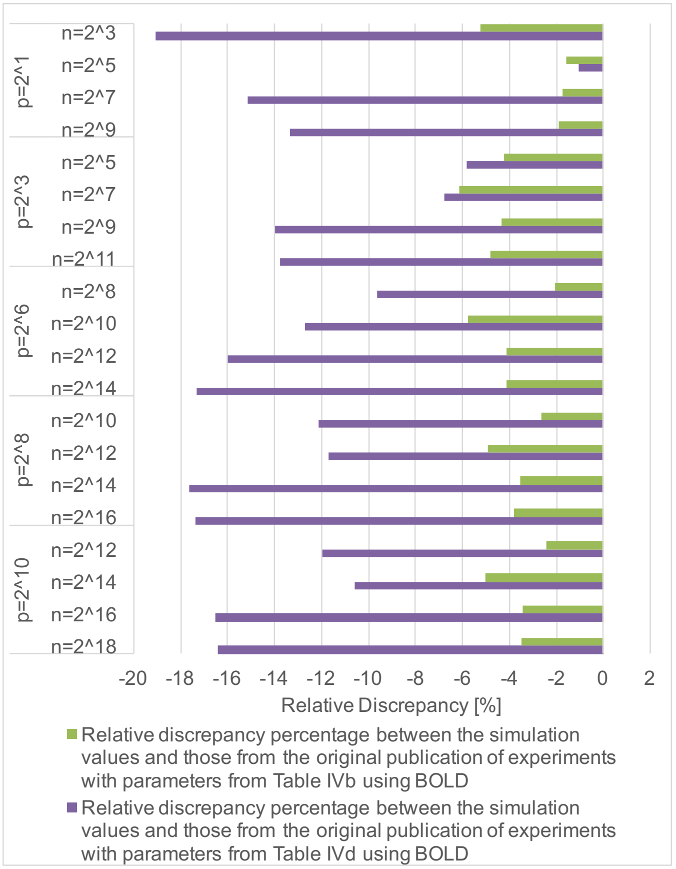

Fig. 8: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using BOLD performed for parameters given in Tables Vb and Vd

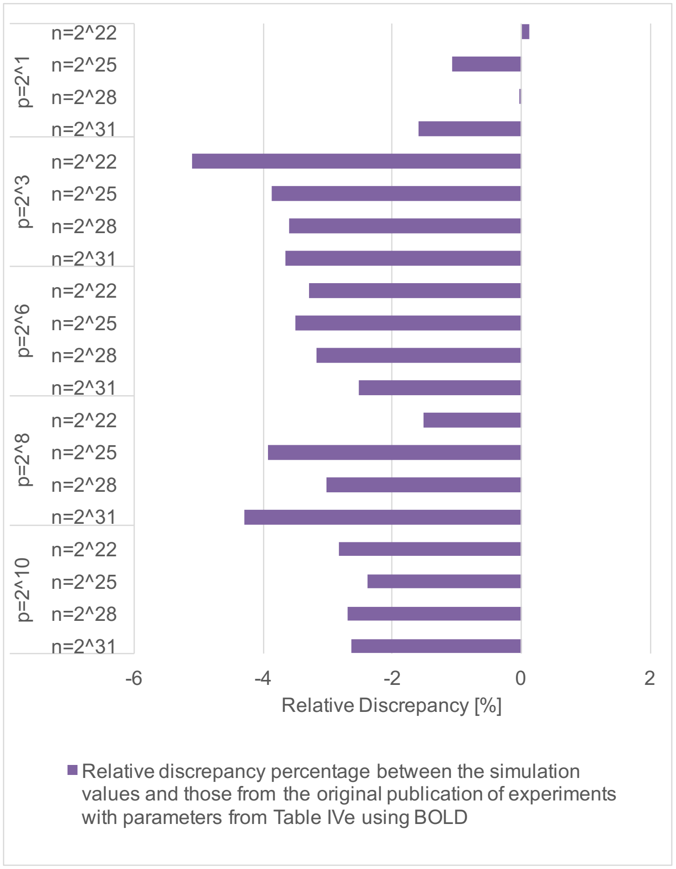

Fig. 9: Relative discrepancy percentage between the simulation values and those from the original publication [2] for experiments using BOLD performed for parameters given in Table Ve

number of tasks for a chunk to be scheduled. And this may be the reason. For a lower number of PEs, FAC may not be able to implement its strengths of balancing the load of the PEs by taking the variability of the task execution times into account. While FAC2 works not so effective for a higher number of PEs due to the unconsidered mean and standard deviation of the task execution times. Nevertheless, the small interval of the relative discrepancy percentage between the original values and the SimGrid-MSG results of $[-4.29\%, 0.33\%]$ implies a successful reproducibility.

F. BOLD

Since the original BOLD publication [2] should show the strengths of the BOLD technique, this technique is used in all reported experiments (Tables Va-Vf). The experiments with given parameters in Tables Vc and Vf are left for further studies.

For the experiments with parameters in Table Vb the discrepancy between the values from the original publication and the values obtained by SimGrid-MSG simulations is in the interval $[-0.47, -0.03]$, which implies a discrepancy percentage relative to the original value between -6.16% and -1.57% shown in Figure 8. Also in Figure 8, the discrepancy percentage for the experiments with parameters reported in Table Vd is plotted. An identical behaviour to the same experiments using FSC, GSS, and TSS is noticeable. The discrepancy percentage of this experiments with a two-point distribution with a standard deviation of $\sigma = 10$ is much higher than the discrepancy percentage of the experiments with exponential distribution with $\sigma = 1$ of the task execution times. The discrepancy percentage of experiments from Table Vd is bounded by -19.08% and -1.06%, which is a fourfold growth compared to the discrepancy percentage of the experiments from Vb. However, the results are acceptable taking into account the erratic behaviour of the individual runs of the experiments.

The experiment setup reported in Table Ve is similar to the one reported in Table Vb. The only difference is the number of tasks which is increased for experiments from Table Ve to $\{2^{22}, 2^{25}, 2^{28}, 2^{31}\}$. The results of the SimGrid-MSG simulations are very promising. The discrepancy between the original values and the values obtained by simulation ranges from -0.451 to 0.006. This is equivalent to a discrepancy percentage between -5.1% and 0.13%. Such a low discrepancy percentage implies a successful reproducibility.

V. Reproducibility of this Work

This work concentrates on performance reproducibility of experiments using DLS techniques reported in earlier literature. It is natural to provide the information needed for reproducing this work. The individual measurements were performed in parallel on the HPC cluster *taurus* at the Centre for Information Services and High Performance Computing (ZIH) at Technische Universität Dresden, using SimGrid-MSG version 3.13, compiled with GCC version 5.3, with no additional flags. The library implementing the DLS techniques in SimGrid-MSG is under development and can be made available upon request. The raw data of the experiments can be made available upon request, as well.

VI. Conclusion and Future Work

This work presents reproducibility efforts with experiments using DLS techniques given in [2]. With the successful performance reproducibility obtained from this work in conjunction with the reproducibility results presented in earlier work [3], the trustworthiness in the correct SimGrid-MSG implementation of the DLS techniques STAT, SS, GSS, TSS, FAC, FAC2, and BOLD is increased. However, it was shown that the reproducibility is challenging if the fluctuations in the task executions times is very high. Note that the performance reproducibility of experiments using FSC, GSS, and TSS with high variance of the task execution times is, in general, made easier by the higher number of tasks and PEs. However, this observation is not valid for the BOLD technique. Nevertheless, the successful reproducibility implies the verification of the DLS implementation in SimGrid-MSG for the considered applications and systems. Future work remains for verifying the TAP and the adaptive techniques (AWF, AWF-B/C, and AF). Their verified implementations facilitate initially new, well-founded research concerning various properties of the DLS techniques. The scalability, flexibility, and resilience of the DLS techniques were investigated to a certain extent in earlier work. The present work lays the foundation for modeling the overhead of the DLS techniques, with the goal to identify the technique with lowest overhead and overall best performance for a given application and system, prior to execution.

Acknowledgment

This work was supported in part by the National Science Foundation under grant number NSF IIP-1034897.

References

[1] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899 – 2917, 2014.

[2] T. Hagerup, "Allocating independent tasks to parallel processors: An experimental study," *Journal of Parallel and Distributed Computing*, vol. 47, no. 2, pp. 185–197, December 1997.

[3] F. Hoffeins, F. M. Ciorba, and I. Banicescu, "Examining the reproducibility of using dynamic loop scheduling techniques in scientific applications," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2017) - Reproducibility in Parallel Computing*, under review.

[4] S. F. Hummel, E. Schonberg, and L. E. Flynn, "Factoring: A method for scheduling parallel loops," *Communications of the ACM*, vol. 35, no. 8, pp. 90–101, August 1992.

[5] S. F. Hummel, J. Schmidt, R. N. Uma, and J. Wein, "Load-sharing in heterogeneous systems via weighted factoring," in *Proceedings of the 8th annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1996)*. ACM, June 1996, pp. 318–328.

[6] I. Banicescu and S. F. Hummel, "Balancing processor loads and exploiting data locality in N-body simulations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 1995)*. ACM, December 1995, pp. 43–58.

[7] I. Banicescu and V. Velusamy, "Load balancing highly irregular computations with the Adaptive Factoring," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2002) - Heterogeneous Computing*. IEEE Computer Society Press, April 2002.

[8] R. L. Cariño, I. Banicescu, R. K. Vadapalli, C. A. Weatherford, and J. Zhu, "Parallel adaptive quantum trajectory method for wavepacket simulations," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2003) - Parallel and Distributed Scientific and Engineering Computing with Applications*. IEEE Computer Society Press, April 2003.

[9] M. Balasubramaniam, N. Sukhija, F. M. Ciorba, I. Banicescu, and S. Srivastava, "Towards the scalability of dynamic loop scheduling techniques via discrete event simulation," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2012) - Parallel and Distributed Scientific and Engineering Computing*. IEEE Computer Society Press, May 2012, pp. 1343–1351.

[10] N. Sukhija, I. Banicescu, S. Srivastava, and F. M. Ciorba, "Evaluating the flexibility of dynamic loop scheduling on heterogeneous systems in the presence of fluctuating load using SimGrid," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2013) - Parallel and Distributed Scientific and Engineering Computing*. IEEE Computer Society Press, May 2013, pp. 1429–1438.

[11] N. Sukhija, I. Banicescu, and F. M. Ciorba, "Investigating the resilience of dynamic loop scheduling in heterogeneous computing systems," in *Proceedings of the 14th International Symposium on Parallel and Distributed Computing (ISPDC 2015)*. IEEE Computer Society Press, June 2015, pp. 194–203.

[12] I. Banicescu and R. L. Cariño, "Addressing the stochastic nature of scientific computations via dynamic loop scheduling." *Electronic Transactions on Numerical Analysis*, vol. 21, pp. 66–80, 2005.

[13] C. P. Kruskal and A. Weiss, "Allocating independent subtasks on parallel processors," *IEEE Transactions on Software Engineering*, vol. 11, no. 10, pp. 1001–1016, October 1985.

[14] C. D. Polychronopoulos and D. J. Kuck, "Guided self-scheduling: A practical scheduling scheme for parallel supercomputers," *IEEE Transactions on Computers*, vol. 36, no. 12, pp. 1425–1439, December 1987.

[15] T. H. Tzen and L. M. Ni, "Trapezoid self-scheduling: A practical scheduling scheme for parallel compilers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 87–98, January 1993.

[16] S. Lucco, "A dynamic scheduling method for irregular parallel programs," in *Proceedings of the ACM SIGPLAN 1992 Conference on Programming Language Design and Implementation (PLDI 1992)*. ACM, June 1992, pp. 200–211.

[17] I. Banicescu, V. Velusamy, and J. Devaprasad, "On the scalability of dynamic scheduling scientific applications with adaptive weighted factoring," *Cluster Computing*, vol. 6, no. 3, pp. 215–226, July 2003.

[18] R. L. Cariño and I. Banicescu, "Dynamic load balancing with adaptive factoring methods in scientific applications," *Supercomputing*, vol. 44, no. 1, pp. 41–63, April 2008.

[19] I. Banicescu and Z. Liu, "Adaptive Factoring: A dynamic scheduling method tuned to the rate of weight changes," in *Proceedings of the High Performance Computing Symposium*, April 2000, pp. 122–129.

[20] A. Knuepfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Mueller, and W. W. E. Nagel, *The Vampir Performance Analysis Tool-Set*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 139–155. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68564-7_9

[21] A. Yesildeniz, "Visualisation of scheduling algorithms," bachelor project report, University of Basel, available online (http://hpc.dmi.unibas.ch/HPC/), 2016.