

# Examining 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 concern, 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. In earlier work, the scalability [1], robustness [2], and resilience [3] of the DLS techniques were investigated using the MSG interface of the SimGrid simulation framework [4]. The present work complements the earlier work and concentrates on the verification via reproducibility of the implementation of the DLS techniques in SimGrid-MSG. This work describes the challenges of verifying the performance of using DLS techniques in earlier implementations of scientific applications. The verification is performed via reproducibility of simulations based on SimGrid-MSG. To simulate experiments selected from earlier literature, the reproducibility process begins by extracting the information needed from the earlier literature and converting it into the input required by SimGrid-MSG. The reproducibility study is carried out by comparing the performance of SimGrid-MSG-based experiments with those reported in two selected publications in which the DLS techniques were originally proposed. While the reproducibility was not successful for experiments from one of the selected publications, it was successful for experiments from the other. This successful reproducibility implies the verification of the DLS implementation in SimGrid-MSG for the considered applications and systems, and thus, it allows well-founded future research on the DLS techniques.

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

## I. INTRODUCTION

Scientific applications are often large in time and/or space, computationally intensive, data parallel, and irregular. A dominant performance degradation factor is load imbalance. Load imbalance occurs due to application, algorithmic,

and/or systemic variability. Dynamic load balancing can be used to achieve load balanced execution of applications with unpredictable changing of workload, when processing elements (PEs) differ in performance, or when perturbations in the system or in the network occur.

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

Scientific applications often consist of iterative computations in the form of computationally intensive loops, which may require a large amount of time steps or comprise a large amount of data points of the computational domain. These loops are a rich source of parallelism. The loop iterations represent the underlying numerical model and can be also considered as independent or dependent tasks.

The research area of dynamic loop scheduling (DLS) addresses algorithmic and systemic sources of load imbalance by dynamically assigning tasks to PEs. Over the years, different loop scheduling techniques were developed, and it has been proved that these techniques are highly successful in balancing applications' workload. The use of DLS techniques is not restricted to loops and they can be applied on any collection of independent tasks. Throughout the present work, a task refers to a loop iteration and both terms are used interchangeably.

DLS techniques have been exhaustively analyzed and applied in real scientific applications on real machines, for instance, in Monte Carlo simulations, radar signal processing, N-body simulations, computational fluid dynamics on unstructured grids, or in wave packet simulations ([5]-[9]). They have shown very good results in reducing the load imbalance caused by algorithmic and systemic variances arising over the course of the execution. Extending upon real experiments, simulations provide the capabilities to

demonstrate the strengths of the DLS techniques for any probability distributions of the task execution times and availability of PEs. In earlier work, the scalability [1], robustness [2], and resilience [3] of the DLS techniques were investigated for various such distributions using the MSG interface of the SimGrid [4] simulation framework (denoted SimGrid-MSG).

The present work complements the previous work ([1]-[3]) and concentrates on the verification via *reproducibility* of the implementation of the DLS techniques in SimGrid-MSG. To the best of our knowledge, there is no work published where the implementation in SimGrid-MSG of the non-adaptive DLS techniques described in Section III is verified. Reproducibility is a form of verification. Successful reproducibility is essential for the trustworthiness of large scale scientific applications using DLS techniques. The performance reproducibility study is carried out by comparing the SimGrid-MSG-based scheduling experiments with those reported in earlier literature. 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 discussion of the performance results obtained via reproducibility of scheduling experiments using DLS techniques published in earlier literature. A short introduction of the DLS techniques, their implementation and incorporation into the simulation framework SimGrid and its interface MSG are presented in Section II. In Section III, information is given about the process of verification via reproducibility of scheduling experiments. The reproducibility results are presented and analyzed 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

The increase in the numerical complexity of simulation models in conjunction with the rapid increase of parallelism in supercomputers<sup>1</sup> lead to the need of efficient communication methods and adequate techniques for assigning the workload to the PEs with regards to existing and future architectures and their scalability. The unit of hardware considered to be a PE depends on the context. A PE can be a functional block of a processor (e.g. FPU), a core, a CPU, a workstation, or another type of processing component. Throughout the present work, a processing element refers to a single computing core. One challenge in achieving optimal performance of the application and the system is mitigating the impact of performance degradation factors, such as overhead caused by load imbalance. Over the years,

the DLS techniques have successfully been used to achieve a load balanced execution of scientific applications.

There are two naive approaches of allocating  $n$  tasks to  $p$  PEs. The very fine grained approach is self scheduling (SS), where each of the  $n$  tasks is dynamically assigned to an available PE. The coarse grained approach is static chunking (STAT), where  $\frac{n}{p}$  chunks of tasks are assigned to each PE before computation starts. The asset of the one is the drawback of the other. In detail, STAT has negligible scheduling overhead but high load imbalance, while SS has very high overhead but good load balancing. The compromise between these two is to dynamically assign tasks in variable size chunks to available PEs. This is accomplished via DLS techniques. The first DLS technique, published in 1985, was fixed size chunking (FSC) [10]. It was developed for load balanced execution of applications with algorithmic variances by taking the variance of the task execution times into account when computing the chunk sizes of the tasks to be scheduled. Systemic variances are taken into account by guided self scheduling (GSS) [11] and trapezoid self scheduling (TSS) [12], which were originally developed for addressing the problem of uneven PE starting times. Factoring (FAC) [5] addresses both, the algorithmic and systemic variances, by scheduling chunks in batches of decreasing chunk sizes. The computation of the chunk sizes considers the mean and the variance of the task execution times. In the case in which the mean and the variance of the task execution times are not known in advance, the authors suggest to chose a decreasing factor for reducing the chunk size of  $x_i = 2$  (FAC2), which works well in practice. The taper (TAP) [13] and the bold (BOLD) [14] strategies are further developments of FAC. For load balanced execution on heterogeneous systems, weighted factoring (WF) [6] has been developed. This DLS technique takes into account the different speeds of the PEs. Adaptive weighted factoring (AWF) [15] has originally been developed for time-stepping applications. It is adaptive at execution time against algorithmic and systemic variances by dynamically assigning new weight values to PEs at execution time, by closely following the rate of change in PE speed after each time-step. More fine grained variations of AWF are AWF-B and AWF-C [16], where the weights are adjusted after each batch or chunk, respectively. A more complex and generalized DLS technique is the adaptive factoring (AF) [17]. It is adaptive at execution time against algorithmic variances as well as to systemic variances, by dynamically estimating for each PE, the new mean and the new variance of the task execution times after the execution of each chunk. A comprehensive review of the DLS techniques may be found in [18].

The DLS techniques have demonstrated their effectiveness when applied in real applications ([5]-[9], [13], [15]-[16]). Complementing native experiments, simulation is a powerful tool for studying the behavior of parallel and distributed ap-

<sup>1</sup><http://www.top500.org>

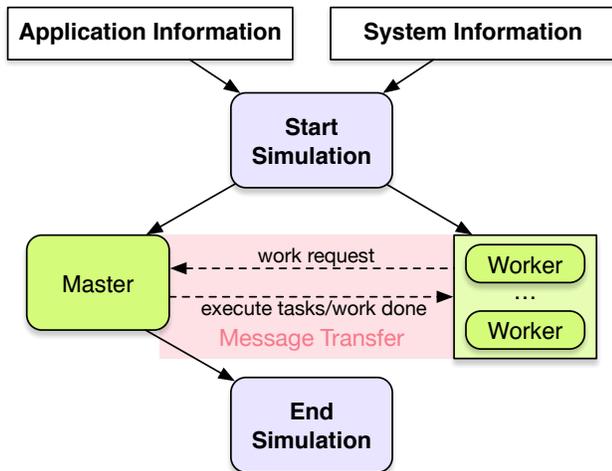


Figure 1: SimGrid-MSG architecture

plications. It allows controllable and repeatable experiments, and the use of a wider range of application and system parameters than measurements of real applications on real machines can offer. Simulations provide the opportunity to capture any probability distribution of the task execution times and availabilities of PEs. For experimental studies, the DLS techniques have been implemented in the MSG interface of the simulation framework SimGrid version 3.13 [4]. SimGrid is a simulation framework “to study the behavior of large-scale distributed systems (...). It can be used to evaluate heuristics, prototype applications or even assess legacy MPI applications.”<sup>2</sup> In addition, it provides an adequate level of abstraction and simulation scalability. SimGrid contains four modules: SimDag, MetaSimGrid (MSG), GRAS, and SMPI. The MSG module was developed for studying scheduling algorithms and, therefore, perfectly satisfies the requirements of the present reproducibility study.

In earlier work, the scalability [1], robustness [2], and resilience [3] of the DLS techniques were investigated using SimGrid-MSG. In the present work, the verification of the SimGrid-MSG implementation via reproducibility of scheduling experiments using DLS techniques published in earlier literature is investigated. With this verification, well founded research on the impact of the overhead of the DLS techniques applied in a scientific application becomes possible. The simulation framework SimGrid is used for two purposes. The systems where the measurements were published in earlier literature are often no longer available and can not be reproduced by current systems. In addition, it is challenging to perform controlled and repeated experiments of scheduling scientific applications on real computer systems for a wide range of application and system characteristics.

<sup>2</sup><http://simgrid.gforge.inria.fr>

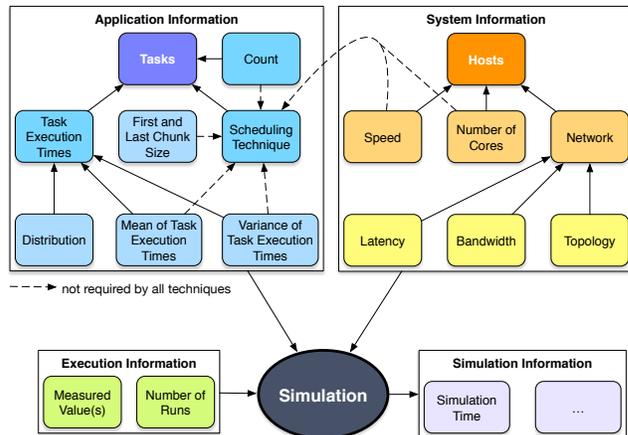


Figure 2: Information required for performing a DLS simulation with SimGrid-MSG

The MSG interface implements a master-worker execution model, illustrated in Figure 1. Before the master and workers are launched, descriptions of the application and the system need to be provided. The application information is given in the SimGrid-MSG deployment file or can be directly implemented in the user code via provided functions of the SimGrid-MSG interface. In the SimGrid-MSG platform file, the system information is specified. When starting the simulation, all workers are in idle state, and send work request messages to the master. When the master receives a work request message, it computes the chunk size for the chosen DLS technique and sends the computed number of tasks to the requesting worker. The worker simulates executing the tasks, and when it finishes, it sends again a work request message to the master. On completion of all tasks, the master sends finalization messages to the workers, 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.

### III. SELECTION OF REPRODUCIBILITY CANDIDATES

To reproduce the scheduling experiments using DLS techniques and SimGrid-MSG, a certain amount of information is needed. An overview of this information is shown in Figure 2.

The application description including the mapping of the master process and the worker processes to hosts is specified in a deployment file or directly in the user code. The number of tasks to be scheduled and the task execution time for each task needs to be maintained. The required parameters for computing the chunk sizes with DLS techniques are listed in Table II, while Table I contains the notation used.

Finally, for reproducing the measurements presented in earlier publications, the information regarding which values

Table I: Notation

Notation	Definition
$p$	number of PEs
$n$	number of tasks
$r$	number of remaining tasks
$h$	scheduling overhead
$\mu$	mean of the task execution times
$\sigma$	variance of the task execution times
$f$	first chunk size
$l$	last chunk size
$m$	number of remaining and under execution tasks

Table II: Required parameters for the DLS techniques

DLS	$p$	$n$	$r$	$h$	$\mu$	$\sigma$	$f$	$l$	$m$
STAT	✓	✓							
SS									
FSC	✓	✓		✓		✓			
GSS	✓		✓						
TSS	✓	✓					✓	✓	
FAC	✓		✓		✓	✓			
FAC2	✓		✓		✓	✓			
BOLD	✓		✓	✓	✓	✓			✓

are measured is needed, and, where appropriate, the number of runs as well.

Throughout this and the following sections “XYZ publication” refers to the work which first introduced the DLS technique XYZ, while it may also contain the description of other DLS techniques used for a comparative performance evaluation.

The information given in earlier publications differs in the degree of detail. In the FSC publication [10] it is not defined which values are measured, and there is no experiment described with the analytically determined optimal chunk size. In addition, the system where the measurements were performed is not named or even described. In this case, the reproducibility is very challenging, and the verification via reproducibility of this DLS technique is not possible, due to the missing experiment with the optimal chunk size. However, in the FAC [5] and the following publications ([6], [15], [16]) real applications are measured, and the systems are named. For reproducibility, task execution times need to be known. Therefore, a trace file or similar information describing the behavior of the measured application needs to be maintained. The AF publication [17] does not contain measurements in support of the correctness of the analytical evaluation. However, experiments underlying the effectiveness of AF have been described in [8]. The information regarding the measurements in the TSS publication [12] is very detailed. The experiment description is explicit, and the system is named, yet not fully described. This seems to be a good candidate for reproducibility. Therefore, in Section III-A the reproducibility efforts of [12] are presented, and in Section IV-A the results are shown.

To verify the analytical results from the GSS [11] and the BOLD [14] publications, the authors of both publications

implemented their own simulator. In both publications, the experiments are described in fine-grained details. In [11], GSS was measured against SS, while in [14] eight DLS techniques were experimentally in simulations analyzed. For verification of a larger number of DLS techniques via reproducibility, the experiments in [14] are reproduced in the present work. The reproducibility efforts are described in Section III-B, while the results are presented in Section IV-B.

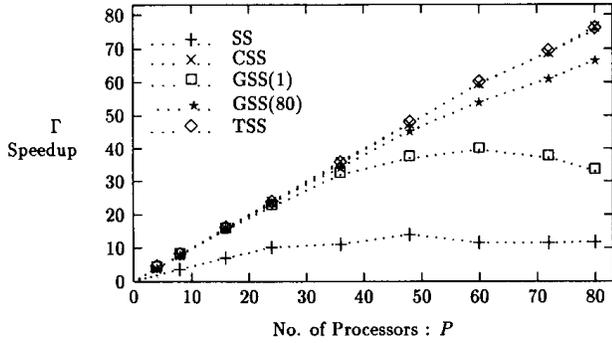
#### A. Reproducibility of the TSS publication [12]

In this publication, task execution times and their distributions (constant, random, decreasing, and increasing) are given. The speedups of the DLS techniques SS, CSS (chunk self scheduling, where the chunk size is chosen by the programmer), GSS, and TSS are measured. In experiments 1 and 2 in [12], the speedup is measured for 100,000 tasks with constant workload of  $110\mu s$ , and for 10,000 tasks with constant workload of  $2ms$ , respectively. The system where the measurements were taken is a 96-node BBN GP-1000 with physically distributed memory. The network is a multistage interconnection network (a slight variant of the OMEGA network). It is not necessary to transform the whole network to networks that can be represented by the SimGrid-MSG platform file because communication takes place only between the master and the workers for work request messages, and messages containing the work, or finalization messages. The results of the reproducibility of these experiments are presented and analyzed in Section IV-A.

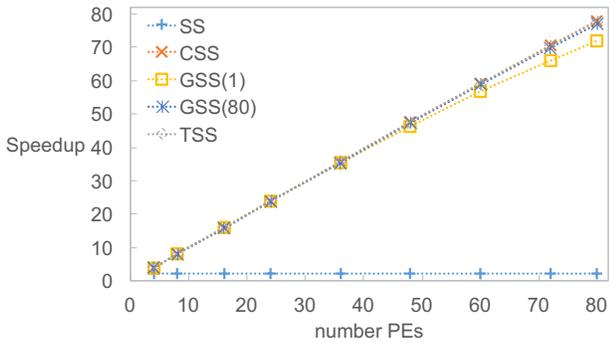
#### B. Reproducibility of the BOLD publication [14]

In the BOLD publication [14], all non-adaptive DLS techniques, except TAP, are measured. Exact values are given and the experiment description is very detailed. Task execution times are generated with the aid of the random number generators *erand48* and *nrand48* with given distribution and parameters. The authors measure the *average wasted time* for a given number of runs. The *wasted time of a single worker* in one run is the sum of the idle time and of the scheduling overhead of this worker. The *average wasted time of a single run* is the sum of the wasted times of all workers divided by the number of workers. The information given regarding the application and execution is very detailed. However, the system description is not given. The authors implemented their own simulator for measuring the DLS techniques.

The missing system description in [14] leads to the use of typical parameters of systems of the late 90s in the SimGrid-MSG platform definition. The reproducibility of the measurements failed by using this fictitious system description. Therefore, the implemented simulator of the authors of [14] was replicated. Their simulator did not measure the network traffic needed for every scheduling



(a) Values from original publication [12]



(b) Values from SimGrid-MSG simulation

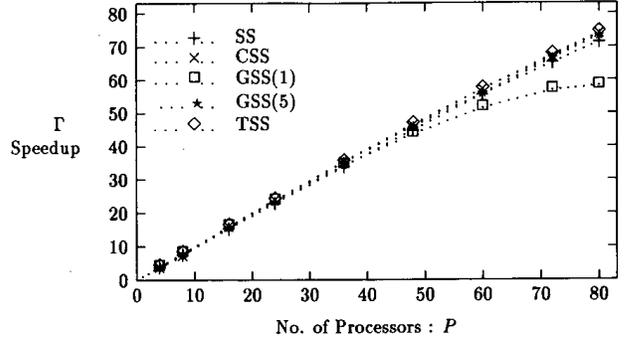
Figure 3: Results of reproducibility of experiment 1 from [12] (100,000 tasks, constant workload of  $110\mu s$ )

operation. It was assumed that every scheduling operation takes a fixed amount of time (parameter  $h$ ). This scheduling overhead for each scheduling operation was added directly to the simulation times. However, the SimGrid-MSG interface considers the network traffic due to the communication between the master and the workers. Therefore, the amount of time for message exchange needs to be excluded. This is reproduced by setting the network parameters bandwidth to a very high value and the latency to a very low value. This simulates no costs for communication. Like the authors of [14] the scheduling overhead  $h$  is added for each scheduling operation directly. The results of the reproducibility of experiment 1 from [14] are shown in Section IV-B.

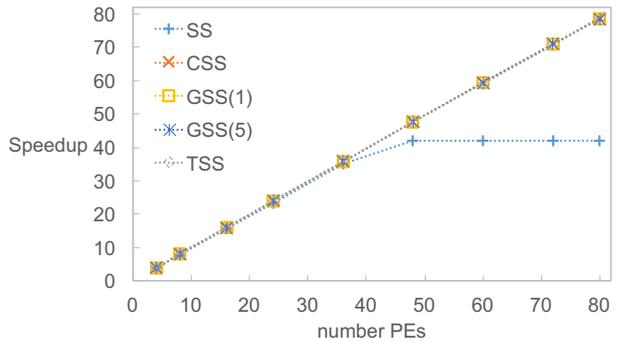
#### IV. REPRODUCIBILITY RESULTS

##### A. Results of reproducing selected scheduling experiments from the TSS publication [12]

This section presents the results of the reproducibility efforts of measurements presented in the original TSS publication [12]. In this original publication, the measurements were obtained on a real parallel computing system. In Section IV-A, the SimGrid-MSG experiments are described and the reproducibility results are analyzed. In Section IV-B the results of the reproducibility of measurements in the



(a) Values from original publication [12]



(b) Values from SimGrid-MSG simulation

Figure 4: Results of reproducibility of experiment 2 from [12] (10,000 tasks, constant workload of  $2ms$ )

original BOLD publication [14] are described and analyzed. The behaviour of the DLS techniques was measured by a simulator.

In experiment 1 from [12], the speedup for a variable number of PEs is reported. The measured DLS techniques are SS, CSS, GSS(1), GSS(80), and TSS, where for GSS, the values in the parentheses represent the smallest to be scheduled chunk size. The chosen chunk size for CSS is the number of tasks divided by the number of PEs. The workload is constant at  $110\mu s$  for each of the 100,000 tasks. Figure 3a shows the results presented in the original publication, while in Figure 3b the values of the present SimGrid-MSG experiments are depicted. A comparison of the values in the publication to the values of the SimGrid-MSG simulations indicates a very similar performance of CSS and TSS. The SS and GSS plots have almost the same tendency, yet the values differ strongly. In experiment 2 from [12], the number of tasks is decreased by an order of magnitude and the workload is increased to  $2ms$  for each of the 10,000 tasks. Figure 4 shows that CSS, GSS(5), and TSS perform similarly, while the performance of SS and GSS(1) does not reproduce the one in the original publication [12].

In general, the constant workload is the simplest form of the distribution of the task execution times. Nevertheless, the

reproducibility was unsuccessful. In [12] implicit parallelism is used. The parallelization of loops is done by primitives in shared memory systems; this is accompanied by contention on the shared loop index. Furthermore, the used system has physically distributed memory. This is associated with higher latencies when accessing other CPU’s memory during the execution of tasks. In addition, the chunk calculation seems to have a strong influence for GSS. For SS, CSS, and TSS atomic instructions are used, while GSS is implemented using lock mechanisms. In SimGrid-MSG these aspects do not arise, due to the explicit parallelism of the master-worker execution model.

### B. Results of reproducing a selected scheduling experiment from the BOLD publication [14]

The reproducibility of the first experiment presented in [14] (results are shown in Table I) is examined in this subsection. In this experiment, eight DLS techniques (STAT, SS, FSC, GSS, TSS, FAC, FAC2, and BOLD) are employed for scheduling a variable number of tasks ( $n = \{1, 024; 8, 192; 65, 536; 524, 288\}$ ) onto a variable number of PEs ( $p = \{2; 8; 64; 256; 1, 024\}$ ). Exact values are given for the sample means of the average wasted time over 1,000 runs with scheduling overhead  $h = 0.5s$ , and an exponential distribution of the task execution times with mean  $\mu = 1s$ .

The SimGrid-MSG simulation in the present work uses the same parameters as the authors in [14]. It is not expected to arrive at exact the same results, given that the task execution times are generated with a random number generator (with a non-reported seed). However, the simulation results obtained in the present work are expected to be close to the original values.

For each run of the SimGrid-MSG simulation, the *simulation time* is measured, and for each worker, the time it spends in computation (executing the tasks) is retained. The *wasted time* (described earlier in Section III-B) of a single worker in one run is computed by subtracting the time it spends in computation from the overall simulation time. The *average wasted time* for a single run is the sum of the *wasted time* of all workers, divided by the number of workers. The *sample mean of the average wasted time* of 1,000 runs is then computed by summation of the *average wasted times* of the runs, divided by the number of runs. The *wasted time* in the simulation is the average of the *idle times* of the workers. The *scheduling overhead time*  $h$  is multiplied with the number of chunks (this is also the number of scheduling operations), and this value is added to the *average wasted time* of all runs.

The following subsections describe the results of the reproducibility experiments, while an overview of the experiments is given in Table III. In every experiment, the eight DLS techniques STAT, SS, FSC, GSS, TSS, FAC, FAC2, and BOLD are measured over 1,000 runs, the distribution

Table III: Overview of reproducibility experiments

Number of tasks	Number of PEs= {2; 8; 64; 256; 1, 024}
1,024	Sec. IV-B1; Figure 5
8,192	Sec. IV-B2; Figure 6
65,536	Sec. IV-B3; Figure 7
524,288	Sec. IV-B4; Figure 8

of the task execution times is exponential with  $\mu = 1s$ , the standard deviation is  $\sigma = 1s$ , and the scheduling overhead is  $h = 0.5s$ .

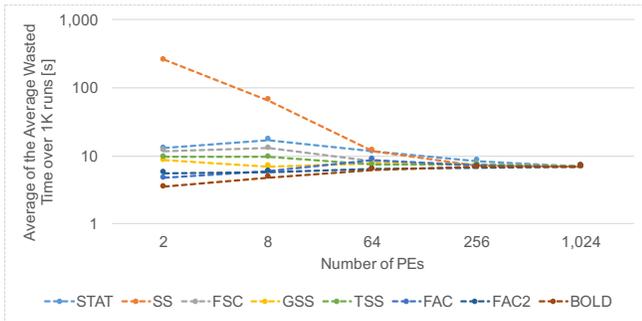
1) 1,024 tasks: Figure 5a shows a graphical illustration of the values published in [14] for 1,000 runs with exponential distribution of the task execution times with  $\mu = 1s$  for 1,024 tasks. The number of PEs is plotted on the x-axis and the average wasted time over 1,000 runs on the logarithmic y-axis. In Figure 5b the simulation results are shown for the same experiments with the same number of runs obtained for the present work with SimGrid-MSG.

The graphs look very similarly, which is supported by Figures 5c and 5d. They show the discrepancy between the values from the original publication and the SimGrid-MSG simulation values, and the relative discrepancy to the value in the original publication, respectively. A positive difference indicates that the present simulation runs slower. The absolute discrepancy is less than  $1.1s$  for all techniques, which translates into an absolute relative discrepancy not higher than  $15\%$  for all techniques. This is an acceptable reproducibility result, given the available information.

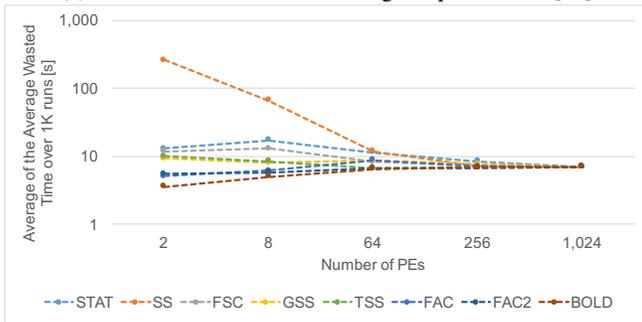
2) 8,192 tasks: Increasing the number of tasks to 8,192 delivers the results depicted in Figure 6. For these experiments, the plots of the values from the original publication and the SimGrid-MSG simulation values, Figures 6a and 6b, respectively, show again a very similar behavior of the DLS techniques, underlined by Figures 6c and 6d, where the discrepancy and the relative discrepancy are depicted. The absolute difference between the values in the publication and the simulation results increases to a maximum value of  $1.6s$ . Nevertheless, the maximum absolute relative discrepancy decreases to  $11.4\%$ .

3) 65,536 tasks: In Figure 7 the reproducibility results of experiments with 65,536 tasks are plotted. The results show the same tendency as in the experiments with 1,024 and 8,192 tasks, respectively. The absolute discrepancy increases to a value of  $2.2s$ , while the relative discrepancy decreases to a value below  $10\%$ .

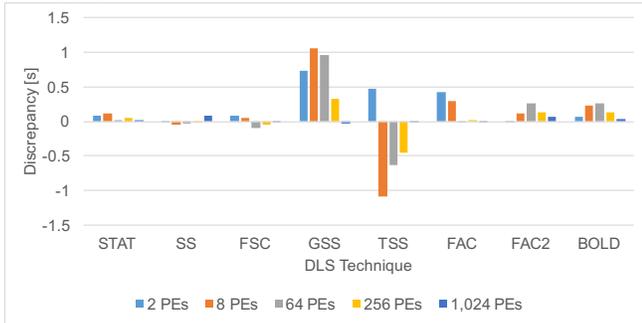
4) 524,288 tasks: In these experiments the number of tasks increases to 524,288. The results are shown in Figure 8. A comparison of the plot of the values in the original publication and the plot of the simulation values (Figures 8a and 8b) leads to the assumption that the DLS behavior in both cases is the same. This is supported by Figures 8c and 8d, where the discrepancy and the relative discrepancy is shown, excluding the outlier FAC with 2 PEs.



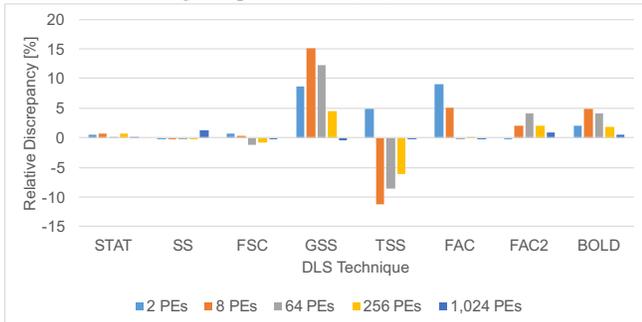
(a) 1,024 tasks - Values from original publication [14]



(b) 1,024 tasks - Values from SimGrid-MSG simulation

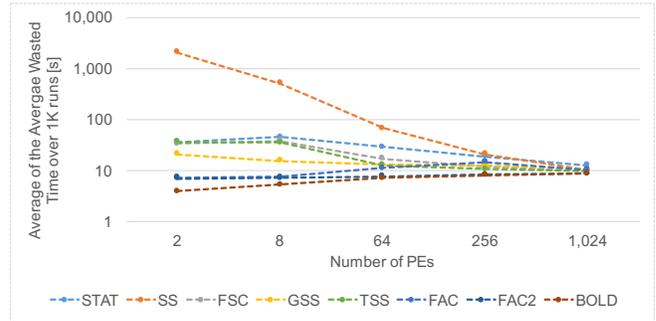


(c) 1,024 tasks - Discrepancy between the simulation values and those from the original publication [14]

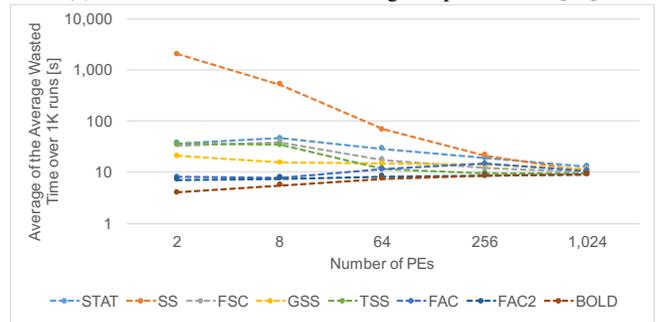


(d) 1,024 tasks - Relative discrepancy percentage between the simulation values and those from the original publication [14]

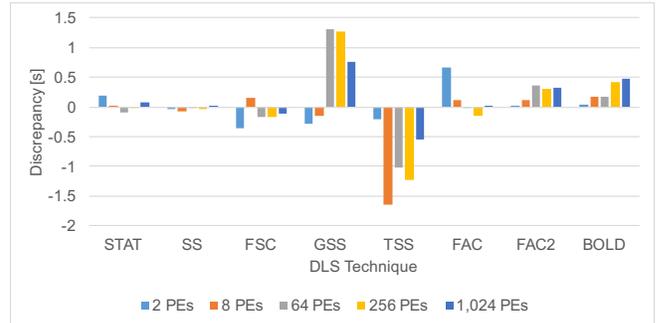
Figure 5: Results 1,024 tasks



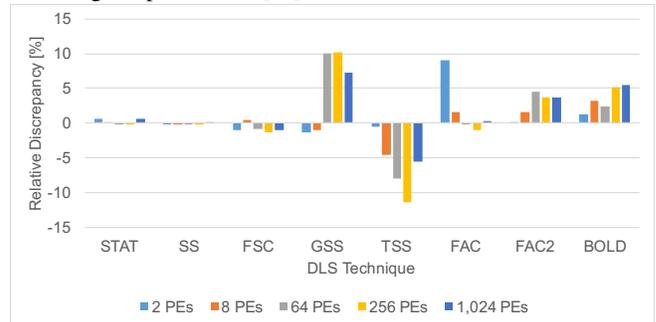
(a) 8,192 tasks - Values from original publication [14]



(b) 8,192 tasks - Values from SimGrid-MSG simulation

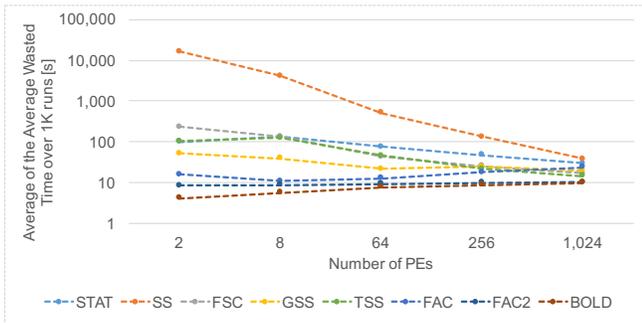


(c) 8,192 tasks - Discrepancy between simulation values and those from original publication [14]

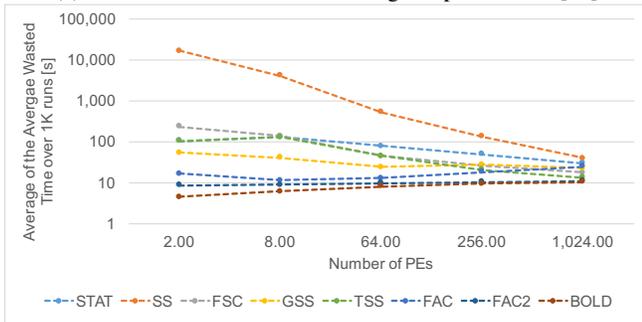


(d) 8,192 tasks - Relative discrepancy percentage between the simulation values and those from the original publication [14]

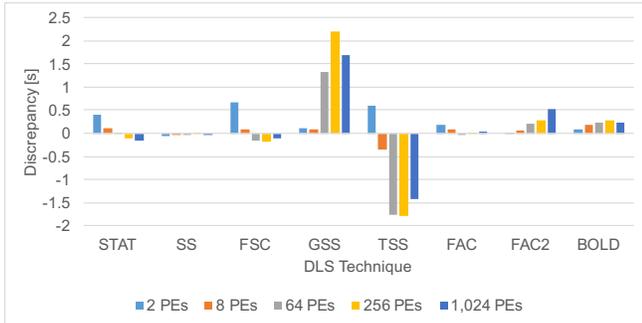
Figure 6: Results 8,192 tasks



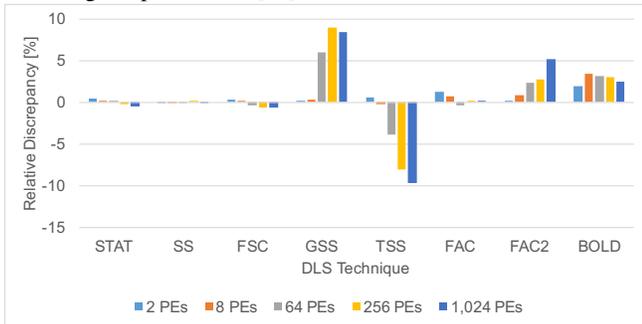
(a) 65,536 tasks - Values from original publication [14]



(b) 65,536 tasks - Values from SimGrid-MSG simulation

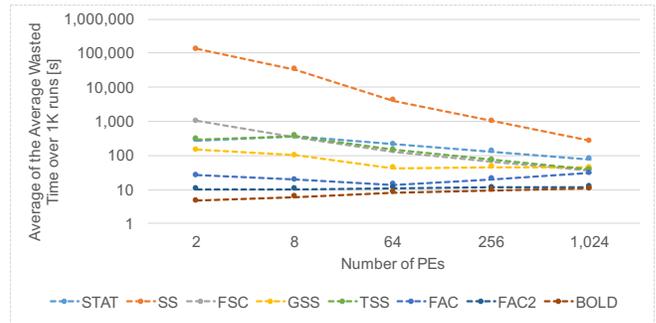


(c) 65,536 tasks - Discrepancy between simulation values and those from original publication [14]

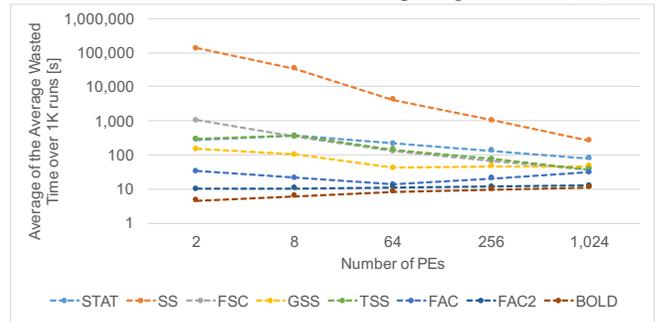


(d) 65,536 tasks - Relative discrepancy percentage between the simulation values and those from the original publication [14]

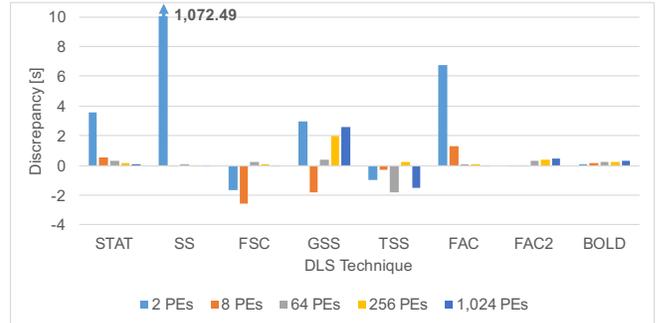
Figure 7: Results 65,536 tasks



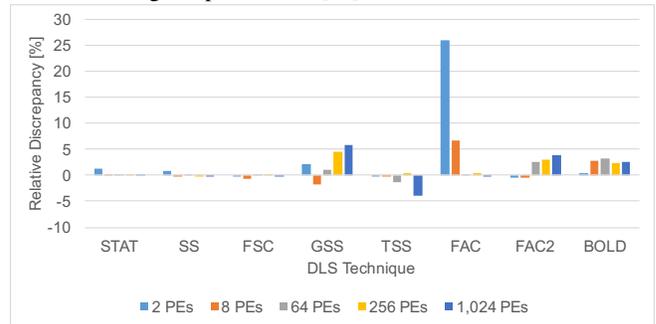
(a) 524,288 tasks - Values from original publication [14]



(b) 524,288 tasks - Values from SimGrid-MSG simulation



(c) 524,288 tasks - Discrepancy between simulation values and those from original publication [14]



(d) 524,288 tasks - Relative discrepancy percentage between the simulation values and those from the original publication [14]

Figure 8: Results 524,288 tasks

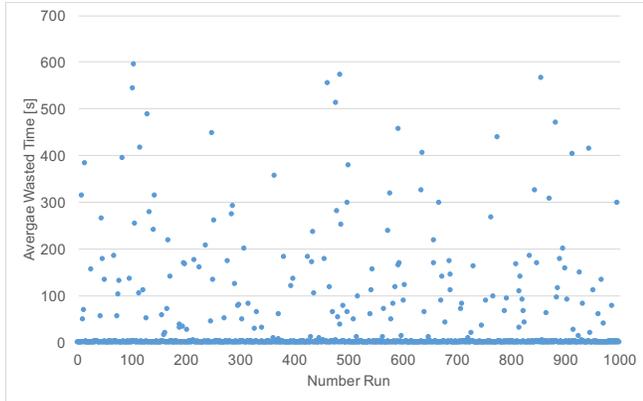


Figure 9: Average wasted time for each of the 1,000 runs of FAC using SimGrid-MSG with 2 workers and 524,288 tasks

The maximum absolute discrepancy for this experiments is 1,072.5s (Figure 8c). This high value is caused by the very high, and therefore, not exactly given, average wasted time (described in Section III-B) of the experiment itself of  $1.3 \cdot 10^5$ s. The relative discrepancy for these experiments is lower than 0.9%.

Taken as whole, the relative discrepancy is lower than 10%, excluding the outlier FAC in the experiments with 2 PEs. The average wasted time for each run for the outlier FAC with 2 PEs is shown in Figure 9. Only 15 values are higher than 400s, which corresponds to 1.5% of all values. The exclusion of these values from the computation of the average wasted over all runs leads to a value of 25.82s. With this value, the relative discrepancy is lower than 1%. As the task execution times are generated with a random number generator using the exponential distribution, the authors of [14] obtained a lower amount of such high values of the average wasted time for each run, and therefore, a better result for the average wasted time for all runs.

## V. REPRODUCIBILITY OF THIS WORK

Any interested reader should be able to reproduce 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 is freely available online<sup>3</sup>.

## VI. CONCLUSION AND FUTURE WORK

This work presents reproducibility efforts of experiments using DLS techniques given in earlier publications, through

<sup>3</sup><https://cloudstore.zih.tu-dresden.de/index.php/s/UiLqNpohuS4OQUF>

their implementation in SimGrid-MSG, for the purpose to verify their performance results. The reproducibility of the results from [12] was unsuccessful. Potential reasons could be a strong influence of the chunk calculation in GSS during execution, inaccurate network parameters in the SimGrid-MSG simulation, and the different parallelization strategies (implicit vs. explicit). The measurements in the publication are obtained on a shared memory system using implicit parallelism. The SimGrid-MSG interface implements a master-worker execution model, which requires explicit management of parallelism.

In [14], the information given regarding the application, the system (in this case a simulator) and the execution are very detailed. The reproducibility was successful for all 20 experiments for each of the eight DLS techniques, STAT, SS, FSC, GSS, TSS, FAC, FAC2, and BOLD. With increasing number of tasks, the relative difference between the values given in the publication and the SimGrid-MSG values is decreasing.

This 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 (AF, AWF, and AWF-B/C). This verified implementation allows well-founded research concerning the 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] 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 (IPDPS 2012) - Parallel and Distributed Scientific and Engineering Computing Workshop*. IEEE Computer Society Press, May 2012, pp. 1343–1351.
- [2] 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 (IPDPS 2013) - Parallel and Distributed Scientific and Engineering Computing Workshop*. IEEE Computer Society Press, May 2013, pp. 1429–1438.

- [3] 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 (ISPD 2015)*. IEEE Computer Society Press, June 2015, pp. 194–203.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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 (IPDPS 2002) - Heterogeneous Computing Workshop*. IEEE Computer Society Press, April 2002, 12 pp. (on CDROM).
- [9] 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 (IPDPS 2003) - Parallel and Distributed Scientific and Engineering Computing with Applications Workshop*. IEEE Computer Society Press, April 2003, 7 pp. (on CDROM).
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.