Employing a study of the robustness metrics to assess the reliability of dynamic loop scheduling [*]

Srishti Srivastava[1,2,†], Florina M. Ciorba[3] and Ioana Banicescu[1,2]
Mississippi State University
[1]Department of Computer Science and Engineering
[2]Center for Computational Sciences - HPC^2
[3]Center for Advanced Vehicular Systems - HPC^2
{srishti@hpc, florina@cavs, ioana@cse}.msstate.edu

Abstract

To achieve best performance, scientific applications are executed on parallel and distributed heterogeneous computing systems. These applications often are computationally intensive, data parallel, irregular, and usually contain large loops that exhibit non-uniform characteristics depending upon their semantic structure during execution. These loops are the most data parallel and computationally intensive part of the applications, and therefore, the main focus of this work is on loop iterations scheduling. Improper scheduling of such loop iterations may lead to load imbalance, which is the dominant factor for performance degradation. A number of dynamic loop scheduling (DLS) techniques have been developed to address the issue of load imbalance for scientific applications on dynamically changing environments. The increasing demand for faster execution of iterations in larger simulations of more complex application models require that DLS provide robust, on-demand performance, on dynamically scalable, high performance computing systems. To evaluate the robustness of these DLS techniques two robustness metrics have previously been formulated to guarantee flexibility and resilience. In this work, we describe simulations of DLS techniques using Alea, a GridSim-based scheduling simulator. Based on the simulation results, we calculate the robustness metrics and show how to use them to determine the most robust DLS techniques.

1. Introduction

Scientific applications are often executed on parallel and distributed heterogeneous systems in order to reduce the time and cost to solution. A number of factors such as, computation time, overhead of communication and managing parallelism, and others, affect the performance of these applications when executed on parallel systems. One of the major performance degradation factors is the load imbalance among processors. Many scientific applications contain large loops which may have irregular iteration execution times due to variances in algorithmic and systemic characteristics. The irregular execution of the loop iterations often leads to load imbalance, which degrades the performance of scientific applications. A number of dynamic loop scheduling (DLS) techniques have been proposed to provide load balancing. However, scheduling scientific applications on large scale heterogeneous systems requires some mechanism to ensure the robustness of these DLS techniques. *Robustness* (reliability) of a DLS method w.r.t. system load is a measure of *flexibility* of its resource allocation w.r.t. the variation of the system load. Moreover, *robustness* (reliability) of a DLS method w.r.t. resource failures is a measure of *resilience* of its resource allocation w.r.t. the variation in the number of processor failures. The mechanism to as-

[*]This work was partially supported by the following grants: NSF #0934393, NSF DBI-0923469 and DOE #008860013.

[†]Srishti Srivastava is the student author.

sess the robustness of the DLS techniques becomes more important as the underlying systems are scaling rapidly from multi-core to peta-scale to exa-scale, and from homogeneous to heterogeneous. Recently, metrics have been formulated to assess the robustness of the DLS methods, with the main focus on the methods which are inherently more robust, such as, Factoring (FAC), Weighted Factoring (WF), Adaptive Weighted Factoring (AWF) and its variants AWF-batched (AWF-B), AWF-chunked (AWF-C) and, Adaptive Factoring (AF), which are discussed in [2, 5]. Previously, robustness of resource allocation or task scheduling has been addressed only for individual methods or for individual applications. The focus of this research work is on the application of the generalized robustness metrics for a number of non-adaptive DLS (NADLS) and adaptive DLS (ADLS) techniques, which can be used for any class of scientific applications. This paper presents the implementation of some of the DLS methods using the Alea simulator [7], which is specifically designed to study the advanced scheduling techniques in a Grid environment and is based on the latest GridSim 5.0 simulation toolkit [4]. The results obtained from the simulation of the DLS methods used to schedule several jobs/tasks on perturbation free environments and on environments with perturbations leading to load imbalance, are further employed to calculate the robustness metrics. The metrics are further used to determine the most robust DLS methods and ensure their robustness for unpredictable dynamically changing systems due to variable workloads and processor failures.

The rest of the paper is organized as follows. Section 2 outlines the robustness metrics and related work. Section 3 describes the implementation of the DLS methods and the calculation of the robustness metrics, while the results and their analysis are presented in Section 4. The conclusions, the significance of this work, and potential future work are discussed in Section 5.

2. Related work

In this section, background work in addition to the one already presented in the introduction section, is being discussed. The formulation of the robustness metrics is based on the methodology proposed in [1], known as the Feature Perturbation Impact Analysis (FePIA) procedure. Although the methodology adopted and the overall idea behind calculating the robustness metric for resource allocations in [1] and the robustness metric for task scheduling in [3] and [9] is the same, they differ in the perspective taken to achieve the most robust application performance. While in [1] the authors investigate the robustness from the resources' perspective, herein we investigate achieving robustness from the applications point of view. Two robustness metrics, a *flexibility* metric to measure robustness against system load variations ($\mathbf{\Lambda}$), and a *resilience* metric to measure robustness against resource failures (**F**), were formulated in [3] and [9]. The performance feature for both metrics is identified to be the parallel execution time, T_{PAR}. One technique is considered most robust if it can handle the largest variation in perturbation and still have minimal variation from a fixed value of the performance feature, or if it has the lowest impact on the performance feature for a fixed variation interval of the perturbation parameter. Thus, the robustness (flexibility or resilience) metric is considered to be the robustness value of the least robust DLS technique.

The two robustness metrics for task allocation (or loop scheduling) using the DLS techniques, are presently in the process of being evaluated experimentally using simulations to be completed in the near future. It should be noted that this paper addresses the behavior of the DLS methods in the presence of resource failures, which is a novel contribution and has not been analyzed until now.

3. Implementation using a GridSim-based scheduler

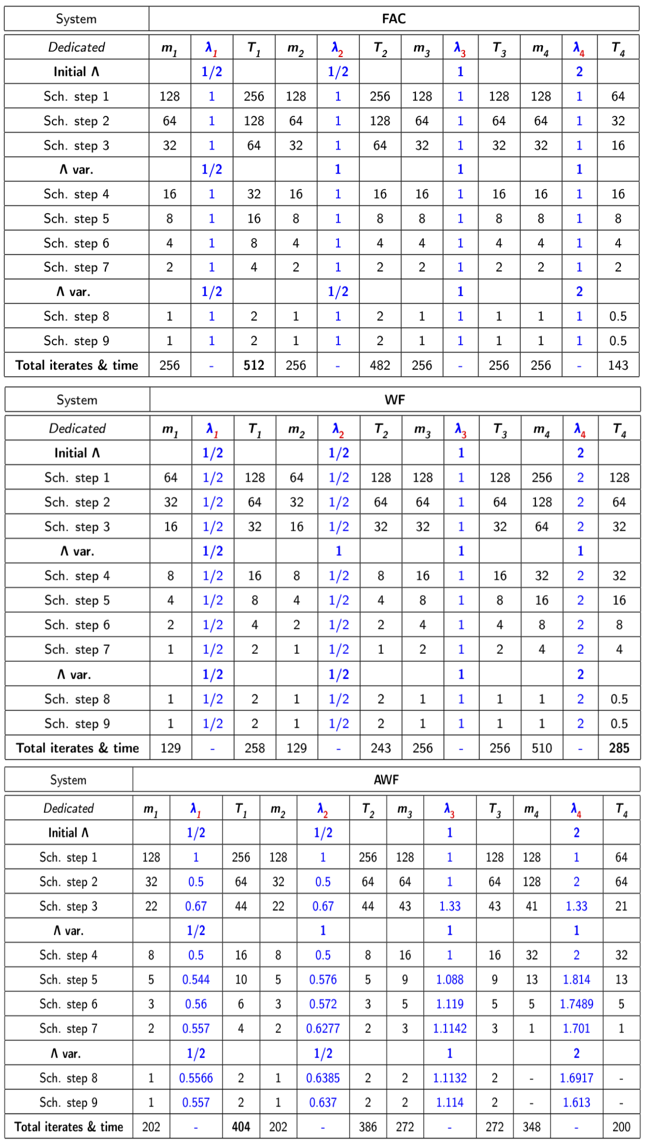
In this paper, we present a description of the ongoing work on implementing the DLS techniques using Alea, a GridSim-based scheduler. This section gives a brief overview of Alea and describes the way the DLS techniques are implemented into this open-source scheduler. Further in this section, we explain how the obtained simulation results can be used to determine the robustness value of each DLS method for a scientific application, and hence, to calculate the robustness metrics which can be used to guarantee their *flexibility* and *resilience*.

Implementing DLS techniques using Alea

A detailed description of the Alea simulator is given in [7]. To implement the DLS techniques in Alea, we only require knowledge of the job submission system, the scheduler and, the resources. The job submission system is responsible for submitting the job descriptions, which in our case are the individual loop iterates, to the scheduler. The input to the job submission system is a workload file, which is often a text file containing the different jobs and their respective descriptions. The format adopted in this work for job descriptions is the Standard Workload Format (SWF) [6]. We use a workload generator called Lublin [8], written in C, to generate the SWF file. The scheduler receives all the jobs and schedules them to the available resources using the incorporated DLS techniques. The job submission system is notified upon the completion of a job and calculates the job-related statistics (execution time, scheduling overhead, and others).

Calculating the robustness metrics

The *robustness value* of a DLS method is given by the *robustness radius*, r_{DLS}, which is the maximum increase in the perturbation factor (system load or resource failure) in any direction from its original value, such that there is no tolerance interval violation for the parallel execution time, T_{PAR}. Similarly, r_{DLS} could also be the increase in the value of T_{PAR} from its original value for a fixed variation in the value of the perturbation parameter. In this work we focus on the latter to calculate r_{DLS}. To emulate system load variations, we inject interactive jobs with unknown arrival and execution times in the simulation; these jobs execute on the same machines as our application, in a space-sharing fashion. We maintain the system load and its variation in a vector, $\mathbf{\Lambda} = [\lambda_1, \ldots, \lambda_P]$, where each λ_j, $1 \leq j \leq P$, maintains the load on machine m_j. Similarly, to emulate resource failures, we consider a vector $\mathbf{F} = [f_1, f_2, ..., f_P]$, where each f_j, $1 \leq j \leq P$, is a binary value such that 0 indicates a machine being alive and 1 indicates a machine failure. In contrast to [3] and [9], where failures were assumed to occur only once during the simulation, herein we consider that they can occur at multiple times during the simulation. The *robustness metric*, $\rho_{DLS}(\Phi, \Pi)$, is the minimum of all robustness radii values, where Φ is the performance feature and Π is the perturbation parameter.



System	FAC											
Dedicated	m_1	λ_1	T_1	m_2	λ_2	T_2	m_3	λ_3	T_3	m_4	λ_4	T_4
Initial Λ		**1/2**			**1/2**			**1**			**2**	
Sch. step 1	128	1	256	128	1	256	128	1	128	128	1	64
Sch. step 2	64	1	128	64	1	128	64	1	64	64	1	32
Sch. step 3	32	1	64	32	1	64	32	1	32	32	1	16
Λ var.		**1/2**			**1**			**1**			**1**	
Sch. step 4	16	1	32	16	1	16	16	1	16	16	1	16
Sch. step 5	8	1	16	8	1	8	8	1	8	8	1	8
Sch. step 6	4	1	8	4	1	4	4	1	4	4	1	4
Sch. step 7	2	1	4	2	1	2	2	1	2	2	1	2
Λ var.		**1/2**			**1/2**			**1**			**2**	
Sch. step 8	1	1	2	1	1	2	1	1	1	1	1	0.5
Sch. step 9	1	1	2	1	1	2	1	1	1	1	1	0.5
Total iterates & time	256	-	**512**	256	-	482	256	-	256	256	-	143

System	WF											
Dedicated	m_1	λ_1	T_1	m_2	λ_2	T_2	m_3	λ_3	T_3	m_4	λ_4	T_4
Initial Λ		**1/2**			**1/2**			**1**			**2**	
Sch. step 1	64	1/2	128	64	1/2	128	128	1	128	256	2	128
Sch. step 2	32	1/2	64	32	1/2	64	64	1	64	128	2	64
Sch. step 3	16	1/2	32	16	1/2	32	32	1	32	64	2	32
Λ var.		**1/2**			**1**			**1**			**1**	
Sch. step 4	8	1/2	16	8	1/2	8	16	1	16	32	2	32
Sch. step 5	4	1/2	8	4	1/2	4	8	1	8	16	2	16
Sch. step 6	2	1/2	4	2	1/2	2	4	1	4	8	2	8
Sch. step 7	1	1/2	2	1	1/2	1	2	1	2	4	2	4
Λ var.		**1/2**			**1/2**			**1**			**2**	
Sch. step 8	1	1/2	2	1	1/2	2	1	1	1	1	2	0.5
Sch. step 9	1	1/2	2	1	1/2	2	1	1	1	1	2	0.5
Total iterates & time	129	-	258	129	-	243	256	-	256	510	-	**285**

System	AWF											
Dedicated	m_1	λ_1	T_1	m_2	λ_2	T_2	m_3	λ_3	T_3	m_4	λ_4	T_4
Initial Λ		**1/2**			**1/2**			**1**			**2**	
Sch. step 1	128	1	256	128	1	256	128	1	128	128	1	64
Sch. step 2	32	0.5	64	32	0.5	64	64	1	64	128	2	64
Sch. step 3	22	0.67	44	22	0.67	44	43	1.33	43	41	1.33	21
Λ var.		**1/2**			**1**			**1**			**1**	
Sch. step 4	8	0.5	16	8	0.5	8	16	1	16	32	2	32
Sch. step 5	5	0.544	10	5	0.576	5	9	1.088	9	13	1.814	13
Sch. step 6	3	0.56	6	3	0.572	3	5	1.119	5	5	1.7489	5
Sch. step 7	2	0.557	4	2	0.6277	2	3	1.1142	3	1	1.701	1
Λ var.		**1/2**			**1/2**			**1**			**2**	
Sch. step 8	1	0.5566	2	1	0.6385	2	2	1.1132	2	-	1.6917	-
Sch. step 9	1	0.557	2	1	0.637	2	2	1.114	2	-	1.613	-
Total iterates & time	202	-	**404**	202	-	386	272	-	272	348	-	200

Figure 1. Execution of FAC, WF and AWF, respectively, in a loaded environment without resource failures for 1024 loop iterates on 4 heterogeneous processors; m_j represents the number of iterates allocated to processor j, T_j is the execution time of processor j, and λ_j is the load on processor j.

4. Simulation results and analysis

This section illustrates a small scale example of the results obtained from the simulations. Due to space limitations, we include results for a small number of loop iterates (1024) scheduled on 4 het-

System	FAC											
Proc. weights	1/2			1/2			1			2		
Dedicated	m_1	f_1	T_1	m_2	f_2	T_2	m_3	f_3	T_3	m_4	f_4	T_4
Initial F		**0**			**0**			**0**			**0**	
Sch. step 1	128	0	256	128	0	256	128	0	128	128	0	64
Sch. step 2	64	0	128	64	0	128	64	0	64	64	0	32
Sch. step 3	32	0	64	32	0	64	32	0	32	32	0	16
F var.		**0**			**1**			**0**			**0**	
Sch. step 4	22	0	44	-	1	-	22	0	22	20	0	10
Sch. step 5	11	0	22	-	1	-	11	0	11	10	0	5
Sch. step 6	6	0	12	-	1	-	6	0	6	4	0	2
Sch. step 7	3	0	6	-	1	-	3	0	3	2	0	1
F var.		**0**			**1**			**1**			**0**	
Sch. step 8	2	0	4	-	1	-	-	1	-	2	0	1
Sch. step 9	1	0	2	-	1	-	-	1	-	1	0	0.5
Sch. step 10	1	0	2	-	1	-	-	1	-	1	0	0.5
Total iterates & time	270	-	**540**	-	-	-	-	-	-	264	-	132

System	WF											
Proc. weights	1/2			1/2			1			2		
Dedicated	m_1	f_1	T_1	m_2	f_2	T_2	m_3	f_3	T_3	m_4	f_4	T_4
Initial F		**0**			**0**			**0**			**0**	
Sch. step 1	64	0	128	64	0	128	128	0	128	256	0	128
Sch. step 2	32	0	64	32	0	64	64	0	64	128	0	64
Sch. step 3	16	0	32	16	0	32	32	0	32	64	0	32
F var.		**0**			**1**			**0**			**0**	
Sch. step 4	11	0	22	-	1	-	22	0	22	31	0	15.5
Sch. step 5	6	0	12	-	1	-	11	0	11	15	0	7.5
Sch. step 6	3	0	6	-	1	-	6	0	6	7	0	3.5
Sch. step 7	2	0	4	-	1	-	3	0	3	3	0	1.5
F var.		**0**			**1**			**1**			**0**	
Sch. step 8	1	0	2	-	1	-	-	1	-	3	0	1.5
Sch. step 9	1	0	2	-	1	-	-	1	-	1	0	0.5
Sch. step 10	1	0	2	-	1	-	-	1	-	1	0	0.5
Total iterates & time	137	-	**274**	-	-	-	-	-	-	509	-	254

System	AWF											
Proc. weights	1/2			1/2			1			2		
Dedicated	m_1	f_1	T_1	m_2	f_2	T_2	m_3	f_3	T_3	m_4	f_4	T_4
Initial F		**0**			**0**			**0**			**0**	
Sch. step 1	128	0	256	128	0	256	128	0	128	128	0	64
Sch. step 2	32	0	64	32	0	64	64	0	64	128	0	64
Sch. step 3	22	0	44	22	0	44	43	0	43	41	0	21
F var.		**0**			**1**			**0**			**0**	
Sch. step 4	24	0	48	-	1	-	15	0	15	25	0	12.5
Sch. step 5	7	0	14	-	1	-	11	0	11	14	0	7
Sch. step 6	4	0	8	-	1	-	6	0	6	6	0	3
Sch. step 7	2	0	4	-	1	-	3	0	3	3	0	1.5
F var.		**0**			**1**			**1**			**0**	
Sch. step 8	1	0	2	-	1	-	-	1	-	3	0	6
Sch. step 9	1	0	2	-	1	-	-	1	-	1	0	0.5
Sch. step 10	1	0	2	-	1	-	-	1	-	1	0	0.5
Total iterates & time	222	-	**444**	-	-	-	-	-	-	350	-	180

Figure 2. Execution of FAC, WF and AWF in a dedicated system with resource failures for 1024 loop iterates on 4 heterogeneous processors; m_j represents the number of iterates allocated to processor j, T_j is the execution time of processor j, and f_j denotes the failure status of processor j.

erogeneous processors. Figure 1 shows the results obtained for FAC, WF and AWF, respectively, in the heterogeneous system in the presence of system load variations. To emulate an uncertain environment the system load $\mathbf{\Lambda}$ is varied from its initial value after the scheduling step 3 and it resumes to its initial state after scheduling step 7. Figure 2 shows the performance of the FAC, WF and AWF techniques for the dedicated heterogeneous system in the presence of resource failures. The resource failures occur after the completion of scheduling step 3 and 7. We assume that a resource failure is permanent. Both Figure 1 and 2, display three columns corresponding to every processor. The first column of a processor m_j shows the number of tasks allocated to it, the second column shows the estimated load on m_j in Figure 1, and the processor failure status in Figure 2, and and the third column shows its finishing time. Finally, Figure 3 shows the derivation of the robustness values for the three DLS methods and the calculation of the two robustness metrics. Figure 3 also shows the use of the tolerance factors τ_1, τ_2, τ_3, as well as the best, average and worst case analysis of the robustness of the DLS methods based on the different values of the three tolerance factors. τ_1 is used to calculate the *flexibility* metric as the minimum of the robustness values of all DLS methods as follows:

$$T_{PAR}(\mathbf{\Lambda}) \leq \tau_1 \cdot T_{PAR}(\mathbf{\Lambda}^{orig}) \text{ such that}$$
$$\tau_1^{best} = 1 \qquad \text{(condition 1.a)}$$
$$\tau_1^{avg} = 1.25 \qquad \text{(condition 1.b)}$$
$$\tau_1^{worst} = 1.50 \qquad \text{(condition 1.c)}$$

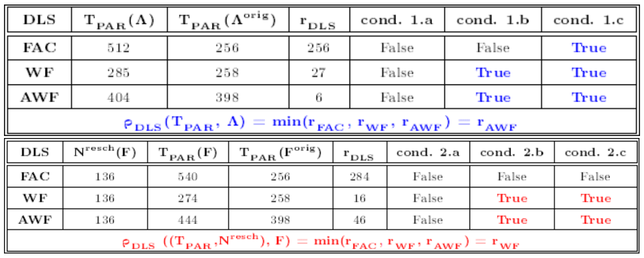
where $T_{PAR}(\mathbf{\Lambda})$ is the parallel time computed in the presence of system load variations and $T_{PAR}(\mathbf{\Lambda}^{orig})$ is the parallel execution time in the dedicated heterogeneous system. Similarly, for resource failures the *resilience* metric is obtained as the minimum of the robustness values of all DLS methods as follows:

$$(N^{resch}(\mathbf{F}) \leq \tau_2 \cdot N) \ \wedge \ (T_{PAR}(\mathbf{F}) \leq \tau_3 \cdot T_{PAR}(F^{orig}))$$

such that

$$(\tau_2^{best} = 0) \ \wedge \ (\tau_3^{best} = 1) \qquad \text{(condition 2.a)}$$
$$(\tau_2^{avg} = 0.25) \ \wedge \ (\tau_3^{avg} = 1.25) \qquad \text{(condition 2.b)}$$
$$(\tau_2^{worst} = 0.50) \ \wedge \ (\tau_3^{worst} = 1.50) \qquad \text{(condition 2.c)}$$

where $N^{resch}(\mathbf{F})$ is the number of iterates to be rescheduled when a resource fails. In Figure 3, for every DLS method we give a TRUE or FALSE value, when the method does or does not satisfy a particular condition. In the top part of the table, we calculate the flexibility metric based on the conditions satisfied by each technique. Similarly, we calculate the resilience metric in the lower part of the table.



DLS	$T_{PAR}(\Lambda)$	$T_{PAR}(\Lambda^{orig})$	r_{DLS}	cond. 1.a	cond. 1.b	cond. 1.c
FAC	512	256	256	False	False	True
WF	285	258	27	False	True	True
AWF	404	398	6	False	True	True
$\rho_{DLS}(T_{PAR}, \Lambda) = \min(r_{FAC}, r_{WF}, r_{AWF}) = r_{AWF}$						

DLS	$N^{resch}(F)$	$T_{PAR}(F)$	$T_{PAR}(F^{orig})$	r_{DLS}	cond. 2.a	cond. 2.b	cond. 2.c
FAC	136	540	256	284	False	False	False
WF	136	274	258	16	False	True	True
AWF	136	444	398	46	False	True	True
$\rho_{DLS}((T_{PAR}, N^{resch}), F) = \min(r_{FAC}, r_{WF}, r_{AWF}) = r_{WF}$							

Figure 3. Calculating the robustness radii and determining the robustness (flexibility and resilience) metrics

5. Conclusions: usefulness and future work

Calculating the robustness metrics depends on the finishing time of each processor, the total parallel time T_{PAR}, the number of tasks assigned to every processor and certain other application, system or algorithm specific parameters which can be obtained *a priori*. Further, the metrics can be integrated into the scheduler to adapt and steer the scheduling decisions autonomously. The robustness metrics are also useful when used in conjunction with other performance metrics, such as makespan, to select the most robust DLS technique among the techniques which yield equal performance from the parallel execution time and cost points of view. The constraints of different types of applications, such as time critical or non-time critical applications, can be imposed by varying the bounds on the values of the tolerance factors, τ_1, τ_2, and τ_3. This enables the use of these metrics in conjunction with other performance metrics, such as, makespan, resource utilization, and others, to ensure the robust execution of scientific applications on heterogeneous computing systems. Using Alea, we can simulate the execution of different types of applications, on different types of computing systems and run them over a large number of application time-steps, with the required level of detail to assess the robustness of the DLS techniques.

The immediate future work plan includes: (1) accounting for other types of perturbations and calculating individual robustness metrics for each type, (2) formulating the robustness metric for multiple perturbations, (3) integrating the robustness metrics within the scheduler for selecting the most robust technique during the execution of the application.

References

[1] S. Ali, A. Maciejewski, H. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. IEEE Transactions on Parallel and Distributed Systems, 15(7):630 – 641, Jul. 2004.

[2] I. Banicescu and R. L. Cariño. Addressing the stochastic nature of scientific computations via dynamic loop scheduling. Trans. on Num. Anal.-Sp. Iss. on Combinatorial Sci. Comp., 21:66–80, 2005.

[3] I. Banicescu, F. M. Ciorba, and R. L. Carino. Towards the robustness of dynamic loop scheduling on large-scale heterogeneous distributed systems. Proc. of IEEE Int'l Symp. on Par. and Dist. Comp., 0:129–132, 2009.

[4] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and Computation: Practice and Experience (CCPE), 14(13):1175–1220, 2002.

[5] R. L. Cariño and I. Banicescu. Dynamic scheduling parallel loops with variable iterate execution times. In Proc. of the IEEE/ACM Int'l Par. and Dist. Proc. Symp. (IPDPS-PDSECA 2002) (CDROM). IEEE Computer Society Press, 2002.

[6] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In IPPS/SPDP '99/JSSPP '99: Proc. of the Job Sch. Strat. for Par. Proc., pages 67–90, London, UK, 1999. Springer-Verlag.

[7] D. Klusáček and H. Rudová. Alea 2 – job scheduling simulator. In Proc. of the 3rd Int'l ICST Conf. on Sim. Tools and Techn. (SIMUTools 2010). ICST, 2010.

[8] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Par. and Dist. Comp., 63(11):1105–1122, 2003.

[9] S. Srivastava, I. Banicescu, and F. Ciorba. Investigating the robustness of adaptive dynamic loop scheduling on heterogeneous computing systems. In Proc. of the IEEE/ACM Int'l Par. and Dist. Proc. Symp. (IPDPS2010-PDSEC), CDROM, IEEE Comp. Soc. Press, pages 1 –8, Apr. 2010.