Providing Efficiency and Adaptivity on BSP Processes Rescheduling*

Rodrigo Righi, Laércio Pilla, Thiago dal Pai, Alexandre Carissimi, Philippe O. A. Navaux
Institute of Informatics - Federal University of Rio Grande do Sul - Porto Alegre, Brazil
{rodrigo.righi, llpilla, tbdpai, asc, navaux}@inf.ufrgs.br

Hans-Ulrich Heiss
Technical University Berlin - Berlin, Germany
heiss@cs.tu-berlin.de

Abstract

In this paper we describe a model for BSP (Bulk Synchronous Parallel) processes rescheduling called MigBSP. Considering the scope of BSP applications, its differential approach is the combination of three metrics - Memory, Computation and Communication - in order to measure the Potential of Migration of each BSP process. In this context, this paper addresses both the efficiency and the adaptivity perspectives of this model over our multi-cluster architecture. MigBSP presents significant gains of performance as well as a low overhead on application execution. Model's adaptation is expressed through the self-management control of rescheduling calls based on system state: if processes are balanced or not and if a pattern without migrations is found on the rescheduling calls. Finally, the results indicate that MigBSP is a viable solution in the management of BSP processes migration on multi-cluster architectures.

1. Introduction

The use of dynamic and heterogeneous resources is increasingly present in parallel and distributed computing areas. In these situations, the initial mapping of processes to resources may not remain efficient during application runtime, due to modifications on the state of the resources. Moreover, the amount of processing as well as the network interaction among the processes may vary during runtime. Concerning this, processes rescheduling (with preemptive migration) to new processors becomes relevant in order to minimize the waiting time for results. Especially, the rescheduling is useful on applications with synchronization points, since they always wait for the slowest process [3].

In this context, we have developed a processes rescheduling model called MigBSP that works over BSP (Bulk Synchronous Parallel) applications[4]. Besides the computation and communication phases of a BSP superstep, our model observes the migration operation costs and the process' memory to decide about processes transferring viability. The model may approximate processes that establish high communication recently and propose migrations of the slowest processes to faster clusters in CPU-bound applications. Aiming to reduce the model's intrusion on application runtime, we developed two adaptations that act over the frequency of rescheduling calls. Different from PUBWCL[1], MigBSP performs the tests for processes relocation according to the system state. The results showed that both adaptations are useful to: (i) improve application performance when migrations are done; (ii) provide model's viability, mainly when migrations do not occur due to cost penalties.

This paper aims to present some MigBSP's results when executing an application over a multi-cluster architecture. The results emphasize the importance of using multiple metrics to control migration and the impact of considering the environment feedback on the rescheduling frequency.

2. MigBSP: Rescheduling Model

MigBSP and its algorithms were described in details in [4]. We applied two adaptations in order to generate the least intrusiveness possible in BSP application. Both are explained in [4]. They manage the interval between two rescheduling calls. This interval is controlled by α ($\alpha \in \mathbb{N}^*$), which depends on the processes' balancing among the resources and on the periodicity of migrations.

The first adaptation considers the stability of each BSP process. α increases if the system tends to stability in conclusion time of each superstep and decreases otherwise. The last case means that the frequency of calls increases in order to turn the system stable quickly. The other adaptation considers the management of the variable *D* that controls the range to define the system as balanced or not. The idea is to increase this variable if processes rescheduling is activated for ω consecutive times but no migrations happened. This causes the growing of the interval in which the system is considered stable.

* This work was partially supported by CNPq and Microsoft

3. Evaluation Methodology

The main objectives of this evaluation are to observe the changes on performance when MigBSP controls the processes relocation and to analyze the impact of the developed adaptations during application runtime. We applied simulation in three scenarios: (i) Application execution simply; (ii) Application execution with scheduler without applying migration; (iii) Application execution with scheduler allowing migrations. The comparison between scenarios i and ii is useful to measure the model's overhead when we do not have processes migration viability. The comparison between scenarios i and iii is pertinent to observe the final performance when executing the rescheduling model.

We use the Simgrid Simulator[2] (MSG module), which makes possible application modeling and processes migration. Simgrid is deterministic, where a specific input always results in the same output. We assembled an infrastructure with four Sets, which is depicted in Figure 1. Each node has a single processor. Moreover, we modeled a BSP implementation of Lattice Boltzmann Method (LBM) to Simgrid using vertical domain decomposition. At each superstep, each process *i* computes a sub-lattice. After that, this process sends boundary data to its neighbor $i+1$. Besides Lattice Boltzmann, this scheme encompasses a broad spectrum of scientific computations, from mesh based solvers, signal processing to image processing algorithms.

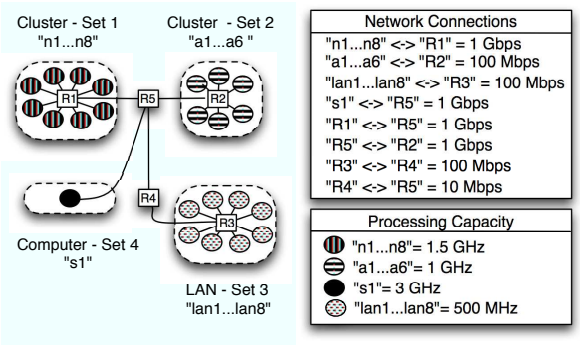

Figure 1: Infrastructure for simulations

Initial tests were executed using α = {4,8,16}. Furthermore, we observed the behavior of 10 BSP processes, employed ω equal to 3 and 0.5 for initial *D*. In addition, we choose only one process as candidate for migration at each rescheduling call. We applied two different initial processes-resources mappings, as follows: First mapping: $\{(p_1,n_1),\ (p_2,n_2),\ (p_3,n_3),\ (p_4,n_4),\ (p_5,n_5),\ (p_6,n_6),\ (p_7,n_7),\ (p_8,n_8),\ (p_9,a_1),\ (p_{10},a_2)\}$; Second mapping puts one process per Set circularly: $\{(p_1,n_1),\ (p_2,a_1),\ (p_3,lan_1),\ (p_4,s_1),\ (p_5,n_2),\ (p_6,a_2),\ (p_7,lan_2),\ (p_8,s_1),\ (p_9,n_3),\ (p_{10},a_3)\}$.

Besides such mappings, we modeled two types of tests: A and B. Superstep time is dominated by computation in test A, while in test B the communication among the processes is the most costly part. Each process executes 10^9 instructions and communicates 500 KBytes at each superstep in test A. The number of instructions changes to 10^6 in test B. Moreover, each process occupies 1 MB of memory. These values were adopted based on real executions of LBM in our clusters at UFRGS, Brazil.

4. Results and Discussions

Table 1 shows the results of test A and the first mapping. The system stays stable (α increases at each rescheduling call). This fact causes low intrusion of our model in application execution comparing scenarios i and ii. One migration occurred $\{(p_9,s_1)\}$ when testing 10 supersteps. A second migration happened when considering 50 or more supersteps: $\{(p_{10},s_1)\}$. Both happened in the first two rescheduling calls. The next calls indicate p_8 as the owner of the highest *PM* and Set 4 for destination. Nevertheless, the migration does not take place because it is inviable. Using α = 4, we obtained a profit of 33% after executing 2000 supersteps in comparison of scenarios i and iii. Analyzing scenario iii with α = 16, we detected that the first migration is postponed, resulting in a larger completion time (see Figure 2). α 16 just presented better performance than α 4 and α 8 when 10 supersteps were evaluated, because no call for rescheduling was done. However, the larger is α in scenario ii, the lower is the model's impact.

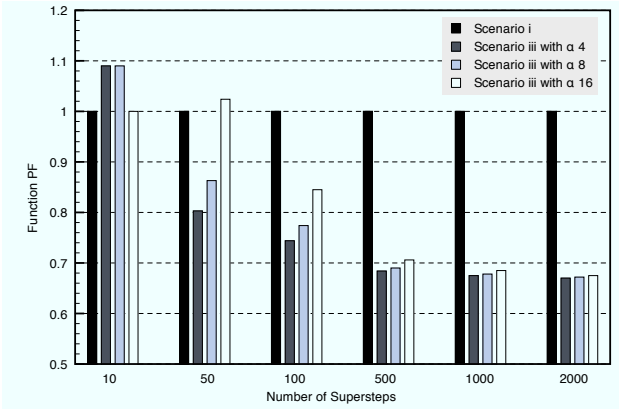

Figure 2: Comparison between scenarios i and iii when evaluating Test A and 1st Mapping. Function PF = $\frac{tested\ scenario}{scenario\ i}$

Table 2 shows the results where the amount of computation is reduced when compared to Test A. This turns the system unstable. The system will become stable either when two migrations are done (scenario iii) or when ω calls without migrations are reached (scenario ii). The model indicates two migrations $\{(p_9,n_7),(p_{10},n_8)\}$ in all executed supersteps, independent of the α used. We verified that the higher is the value of α, the better is the application performance. Concerning scenario iii and using α 4, five rescheduling calls are done when executing 50 su-

Table 1: Execution of all situations when dealing with Test A and the first initial mapping (time in seconds)

Step	Scen (i)	Scen (ii), α 4	Scen (iii), α 4	Scen (ii), α 8	Scen (iii), α 8	Scen (ii), α 16	Scen (iii), α 16
10	10.01	10.15	11.10	10.15	11.10	10.15	10.15
50	50.51	50.58	40.64	50.55	43.70	50.52	51.83
100	101.02	101.34	75.21	101.24	78.27	101.08	85.40
500	505.12	506.14	345.78	505.88	348.84	505.32	356.97
1000	1010.25	1011.87	682.49	1011.27	685.55	1010.74	692.68
2000	2020.50	2022.92	1354.90	2021.53	1357.97	2020.93	1365.10

Table 2: Execution of all situations when dealing with Test B and the first initial mapping (time in seconds)

Step	Scen (i)	Scen (ii), α 4	Scen (iii), α 4	Scen (ii), α 8	Scen (iii), α 8	Scen (ii), α 16	Scen (iii), α 16
10	0.11	1.45	1.89	1.10	1.11	0.11	0.11
50	0.56	3.67	4.30	2.88	3.34	1.70	2.42
100	1.12	5.89	5.57	4.88	4.61	3.26	3.58
500	5.63	10.86	9.60	9.57	8.73	8.10	7.80
1000	11.26	16.45	13.34	15.38	12.38	14.48	11.45
2000	22.52	27.89	19.64	26.81	18.68	25.32	17.75

persteps (at supersteps 4, 8, 12, 20 and 36). Analyzing the same scenario with α 16, only two calls are executed (at supersteps 16 and 32), what results in a lower impact. This explains the good performance achieved with higher values of α. Moreover, other point is that the amount of computation and communication in this configuration is not enough to overlap the migration costs on few supersteps. Figure 3 illustrates the performance for Test B and the first mapping. Besides the behaviors of the different values of α, we can conclude that we need a higher number of supersteps to gain with processes migration on IO-Bound applications.

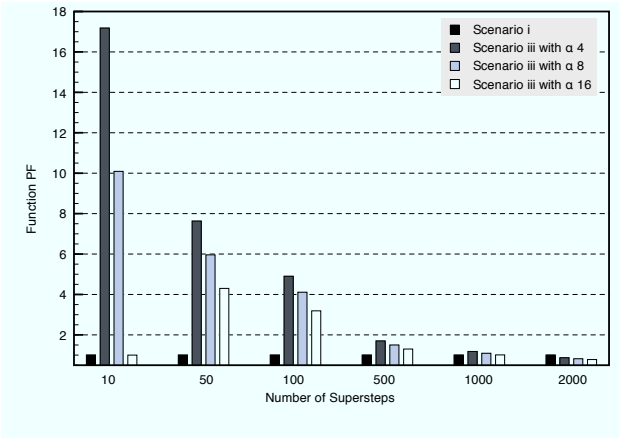

Figure 3: Comparison between scenarios i and iii when evaluating Test B and 1^{st} Mapping. Function PF = $\frac{tested\ scenario}{scenario\ i}$

Table 3 shows the results with second mapping and test A. The system stays stable with this configuration, allowing similar times for scenarios i and ii. We have one migration $\{(p_3, s_1)\}$ with 10 supersteps and a second $\{(p_7, s_1)\}$ when considering 50 or more supersteps. Both processes belonged to the slowest Set, which contributes to increase their *PM* values. Furthermore, these *PM*s refer to migrations to Set 4, which presents the highest clock rate. A reduction of $\approx$ 39% in time is achieved with our model when 2000 supersteps are evaluated. Furthermore, we verified that the second mapping provides longer execution times than the first one. This is associated with both the heterogeneity of the infrastructure and the application's behavior.

The results of test B with the second mapping are shown in Table 4. Firstly, we can observe that scenarios i and ii present similar times. Scenario iii achieves good performance, since it reorganizes the initial mapping, moving processes that present network interaction to a same Set. Two migrations occurred with 10 supersteps and α 4: $\{(p_3, a_4), (p_7, a_5)\}$. Both processes were executing on Set 3, which presents both the lowest processing capacity and the largest communication costs. Besides these migrations, we achieved the following when 50 supersteps are executed: $\{(p_2, n_4), (p_3, n_5), (p_6, n_6), (p_7, n_7), (p_{10}, n_8), (p_5, s_1)\}$. We can observe that processes p_3 and p_7 migrated twice, firstly from their initial location to Set 2 and after to Set 1. Maintaining this value of α, 2 more migrations are done with 100 supersteps: $\{(p_6, s_1), (p_7, s_1)\}$. In addition, the executions of 500 up to 2000 supersteps do not present more migrations. Analyzing scenarios iii, we can verify that the migrations occurred during the first supersteps (up to 100 or 500 depending on α). During such supersteps, we obtained better results with higher values of α, because there are less calls for processes rescheduling. However, the larger is the amount of supersteps, the higher is the gain with migrations using lower values of α. A high number of iterations amortizes the migration costs involved in the first supersteps.

Table 5 presents the execution of 1000 supersteps on scenarios ii and iii without model's adaptations. We can observe that our adaptations have more impact over unstable

Table 3: Execution of all situations when dealing with Test A and the second initial mapping (time in seconds)

Step	Scen (i)	Scen (ii), α 4	Scen (iii), α 4	Scen (ii), α 8	Scen (iii), α 8	Scen (ii), α 16	Scen (iii), α 16
10	22.00	22.58	23.59	22.45	23.99	22.00	22.00
50	110.01	110.65	79.65	110.48	88.14	110.12	107.10
100	220.03	220.89	147.82	220.58	156.31	220.21	174.10
500	1100.48	1101.05	687.18	1100.80	695.66	1100.43	714.63
1000	2200.39	2201.02	1359.87	2200.72	1368.35	2200.38	1386.32
2000	4400.73	4401.88	2703.25	4401.36	2712.73	4400.96	2730.70

Table 4: Execution of all situations when dealing with Test B and the second initial mapping (time in seconds)

Step	Scen (i)	Scen (ii), α 4	Scen (iii), α 4	Scen (ii), α 8	Scen (iii), α 8	Scen (ii), α 16	Scen (iii), α 16
10	2.02	3.6	3.2	2.82	2.82	2.02	2.02
50	10.11	14.11	9.78	13.31	9.30	12.51	8.52
100	20.23	25.03	12.00	24.23	11.56	23.43	12.59
500	101.18	107.58	15.50	106.78	16.11	105.98	18.33
1000	202.36	209.56	18.39	208.77	19.00	207.97	21.22
2000	404.73	412.74	23.16	411.94	23.77	411.14	25.98

Table 5: Evaluating 1000 supersteps on both scenarios ii and iii without model's adaptations (time in seconds)

Scen	Test A + First Mapping		Test B + First Mapping		Test A + Second Mapping		Test B + Second Mapping	
	α = 4	α = 16	α = 4	α = 16	α = 4	α = 16	α = 4	α = 16
(ii)	1030.32	1018.58	258.79	72.65	2235.87	2207.96	402.30	251.91
(iii)	924.41	744.33	255.64	67.52	1600.12	1431.27	255.63	72.14

and communication-bound environments. For example, using α 16, 1000 supersteps, Test B and the second mapping, we achieved 21.22s and 72.14s on scenario iii. The first time uses adaptations, while the other does not. Finally, we conclude that **both adaptations are essential to achieve performance and to turn our model viable**.

5. Conclusion and Future Works

This paper shows some results of MigBSP. The main contributions of MigBSP are twofold: efficiency and adaptivity on BSP processes rescheduling.

- Efficiency - MigBSP combines three metrics - Computation, Communication and Memory - in a scalar one called PM (Potential of Migration). PM considers processes and Sets, not performing all possible processes-resources tests. MigBSP takes different decisions depending on the application behavior (CPU or IO bound). Our model obtained good results on both cases (with gains greater than 30%). MigBSP migrates processes to faster Sets in the first case and suggests the approximation of processes in the other one.
- Adaptivity - MigBSP performs the rescheduling according to the system state. We use an adaptable index called α, which informs the interval between calls for rescheduling. The general adaptations' ideas are: (i) to postpone this call if the system is balanced or, otherwise, to turn it more frequent; (ii) to delay the rescheduling call if the system presents a pattern without migrations on several calls for rescheduling. Comparing the situations in which developed adaptations are tested with other in which they are disabled , we can conclude that the adaptations turn the model viable to reschedule BSP processes.

We observed that our experiments prioritized the heterogeneity issue. Thus, future works include experiments over dynamic environments. Finally, we are testing LU decomposition application with MigBSP on our multi-cluster architecture.

References

[1] O. Bonorden, J. Gehweiler, and F. M. auf der Heide. Load balancing strategies in a web computing environment. In *Conference on Parallel Processing and Applied Mathematics (PPAM)*, pages 839–846, Poznan, Poland, 2005.

[2] H. Casanova, A. Legrand, and M. Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Conference on Computer Modeling and Simulation (uksim)*, pages 126–131, 2008. IEEE.

[3] L. Chen, C.-L. Wang, and F. Lau. Process reassignment with reduced migration cost in grid load rebalancing. *Parallel and Distributed Processing, 2008. IPDPS 2008*, pages 1–13, 2008.

[4] R. da Rosa Righi, L. L. Pilla, A. S. Carissimi, P. O. Navaux, and H.-U. Heiss. Applying processes rescheduling over irregular bsp application. *Computational Science – ICCS 2009*, volume 5544 of *LNCS*, pages 213–223. Springer, 2009.