A LOAD BALANCING METHOD FOR PARALLEL GENETIC CFD OPTIMIZATIONS

I. AIZPURUA*, O. LEHMKUHL*,†, R. BORRELL*,†, C.D. PEREZ-SEGARRA* AND A. OLIVA*

* Heat and Mass Transfer Technological Center (CTTC)
Universitat Politècnica de Catalunya - Barcelona Tech
ETSEIAT, Colom 11, 08222 Terrassa, Barcelona, Spain
e-mail: cttc@cttc.upc.edu - Web page: http://www.cttc.upc.edu

†Termo Fluids S.L.
Av. Jacquard 97 1-E, 08222 Terrassa, Barcelona, Spain
e-mail: termofluids@termofluids.com - Web page: http://termofluids.com

Key words: Genetic algorithm, Master-worker parallelization, Load balancing, Task scheduling, CFD optimization, Variable evaluation time

Abstract. CFD optimizations are characterized by long and highly variable objective function evaluation times. The aim of this study is to improve the parallel performance using a new load balancing master-worker parallelization strategy. First the new method is introduced and compared with the standard parallelization strategy. Finally two illustrative examples are provided.

1 INTRODUCTION

CFD optimizations are characterized by long objective function evaluation times. Additionally, these evaluation times have demonstrated to be highly variable due to changing geometries or meshes (e.g. adaptive meshes) or to the use of different models and solvers in the same optimization (step by step, CFD, etc.), for instance. These facts make CFD optimization a challenge in both reaching the global optimal solution and in the efficient usage of computational resources.

Genetic algorithms (see Figure 1) hold a prominent place among optimization algorithms and are commonly used for solving CFD problems. Parallelization becomes crucial as the population size grows up or fitness evaluation times become considerably larger than the time required by evolutionary operations. Research in the field is focusing on improving the island-model (coarse-grained parallelization) and the cellular genetic algorithm (fine-grained parallelization) [1]. But apparently no effort is being put into optimizing the master-worker model by means of dynamic task scheduling (global parallelization), even if it may operate in the lower layers of the island-model.

Asynchronous evaluation of individuals is currently the common practice in master-worker parallelization of genetic algorithms, being synchronization necessary only at the end of each generation. Research in this field is mainly focused on handling the heterogeneity and variability in run time of the computational environment, as it happens in

grid-computing. However, computational time waste may arise towards the end of each generation's evaluation, when the number of remaining individuals is not enough to occupy all CPUs and the work load becomes consequently unbalanced (see section 2). Most authors seem not to be concerned by this issue, as the cases they solve have short evaluation times. But this fact gains importance as evaluation times become longer and/or have great variability among individuals of the same generation [2], which is the case of CFD simulations.

The aim of this study is to propose a new load balancing master-worker parallelization strategy for designing an appropriate task schedule based on estimations of the individuals' evaluation times and dynamic work redistribution.

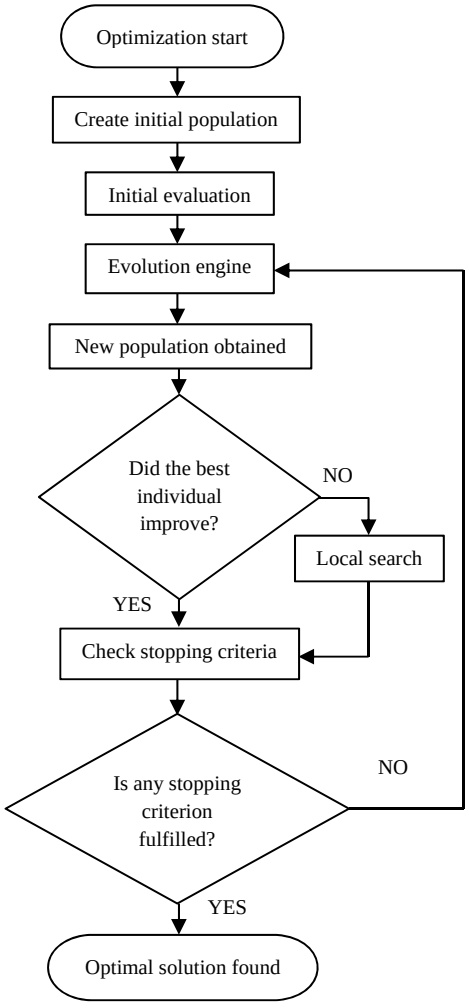

Figure 1: Standard genetic algorithm

2 STANDARD MASTER-WORKER PARALLELIZATION STRATEGY

In the standard master-worker parallelization strategy, the total number of available processors as well as the amount of processors assigned to the evaluation of an individual are fixed in the beginning of the optimization algorithm. This gives rise to groups of processors, being each group in charge of evaluating one individual at a time. In the example of Figure 2 (left), the number of processors assigned to the evaluation of each individual has been fixed to 4. This means that the number of individuals being evaluated simultaneously is 8, one individual in each group of processors.

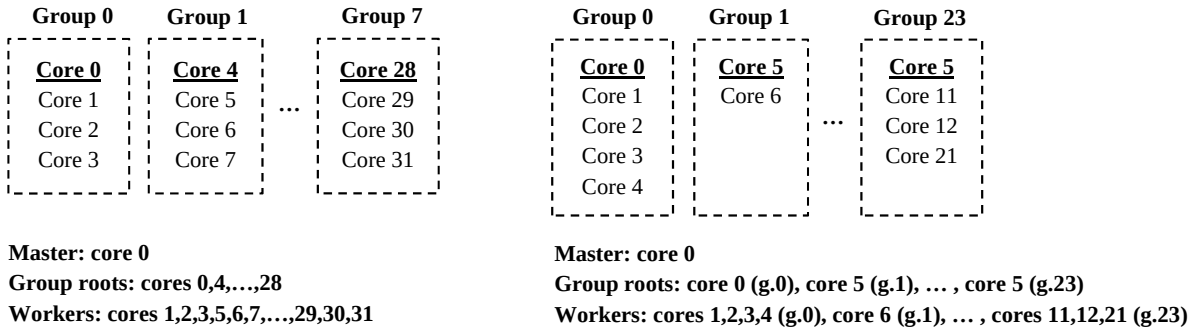

Figure 2: Processor groups in the standard master-worker parallelization strategy (left) and the load balancing strategy (right). There are 24 individuals to be evaluated in 32 processors in this example.

The standard master-worker parallelization algorithm is represented in Figure 3. Genetic operations take place in the master (core 0). When a batch of new individuals needs to be evaluated, the master sends these new individuals to the group roots and the parallel evaluation begins. Each group root checks which individuals are still pending, chooses one of them and carries out a parallel evaluation in coordination with the workers belonging to that group of processors. This process goes on until the individuals batch finishes. From then on, that group of processors remains inactive until the master needs to evaluate a new batch of individuals.

However, this strategy may lead to computational time waste in certain cases, as shown in Figure 4. The inefficiency lies on the fact that a group of processors remains inactive since the moment at which there are no more individuals to be selected in the batch until the last ongoing evaluation finishes. Hence, the worst situation takes place for long and variable evaluation times, which are common in CFD simulations.

3 A NEW LOAD BALANCING MASTER-WORKER PARALLELIZATION STRATEGY

In the new load balancing master-worker parallelization strategy proposed by the authors, only the total number of available processors is fixed at the beginning of the optimization algorithm. The strategy is composed by 2 major steps: estimation of fitness evaluation times on the one hand, and task scheduling on the other hand, solving at least

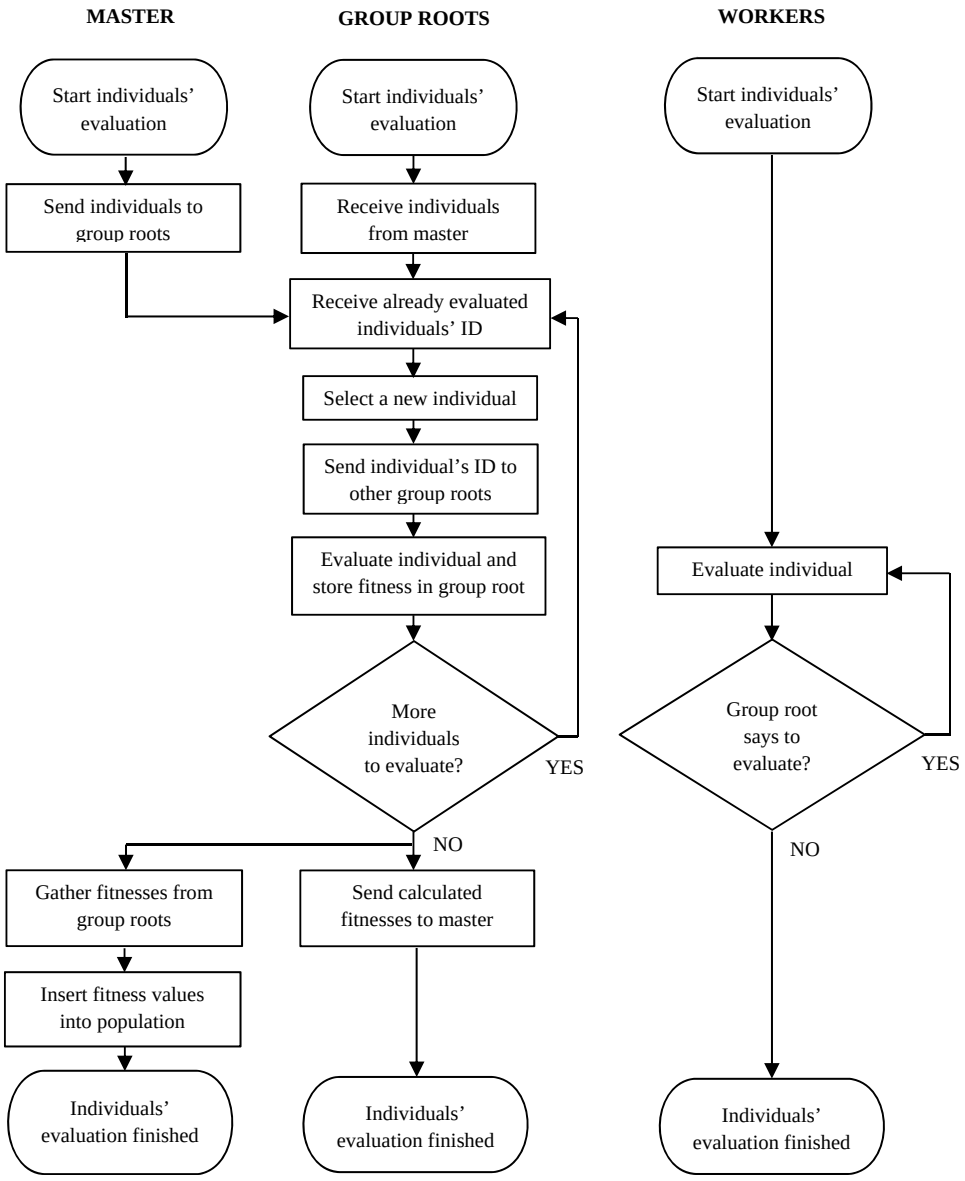

Figure 3: Standard master-worker parallelization strategy.

a load balancing combinatorial problem in each generation. Note that similar methods are being applied for scheduling parallel jobs on multicore clusters [3].

This strategy gives rise to a schedule similar to that shown in Figure 4 (right), where time savings in the evaluation process are noticeable. At the current development stage of the algorithm, the end user is in charge of providing a law for estimating evaluation times, but time estimation via response surfaces is being developed. With that information, the parallelization algorithm carries out a task schedule optimization by means

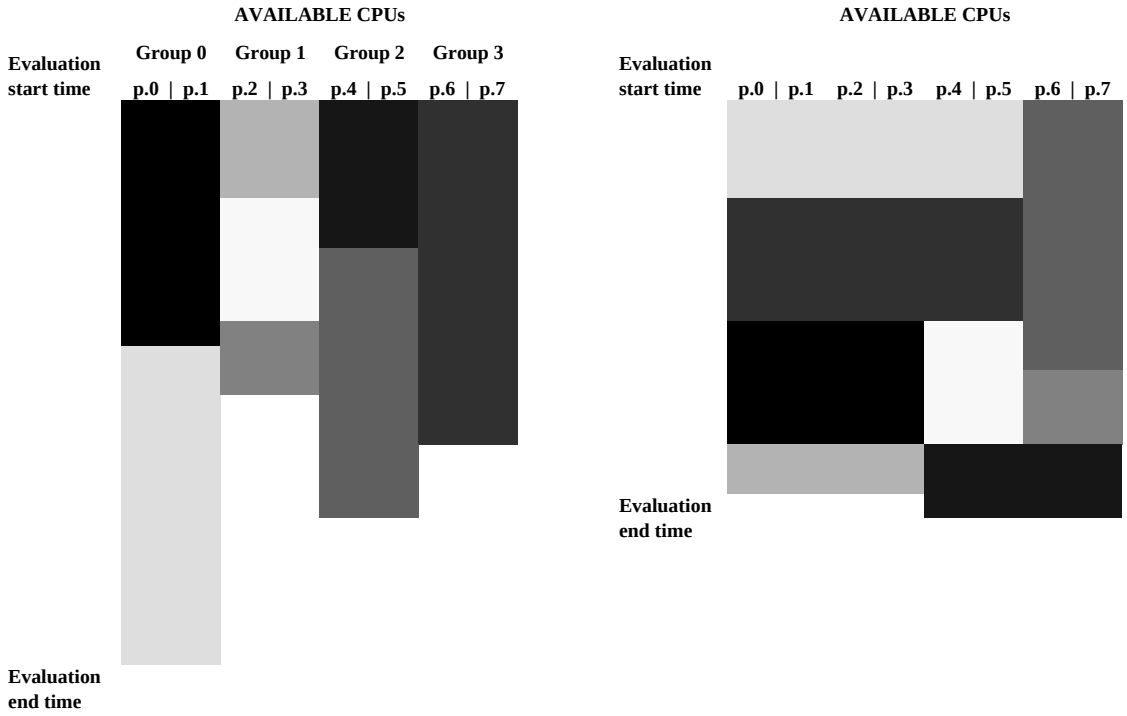

Figure 4: Work load distribution carried out by a standard master-worker parallelization strategy (left) and the load balancing strategy (right). In this example, 8 individuals are being evaluated in 8 CPUs. The vertical axes represent the total evaluation time. Available processors are shown in horizontal axes.

of a genetic algorithm, being the objective function the minimization of the generation's total evaluation time. As a result, a group of cores is assigned to each individual and a schedule defined. In the example of Figure 2 (right), a group of processors has been created per individual. The number of individuals being evaluated simultaneously is variable, opposed to the fixed number in the standard parallelization strategy. The new load balancing parallelization algorithm is represented in Figure 5.

4 ILLUSTRATIVE EXAMPLES

The new load balancing strategy has been tested in the following 2 examples.

4.1 Example 1: Refrigeration of a power electronic device

The first example consists on a simplified model of a power electronic device refrigerated by means of a fluid circulating through a certain amount of pipes (see Figure 6). The heat extraction from the power electronic device is to be maximized, being the only optimization variable the pipe diameter (and hence the number of pipes). The model has been constructed using 3D conduction for the device, a 1D fluid solver for the heat pipes and a 1D network solver for the inlet/outlet collectors.

32 CPUs were used for the tests. Evaluation times were predicted according to a

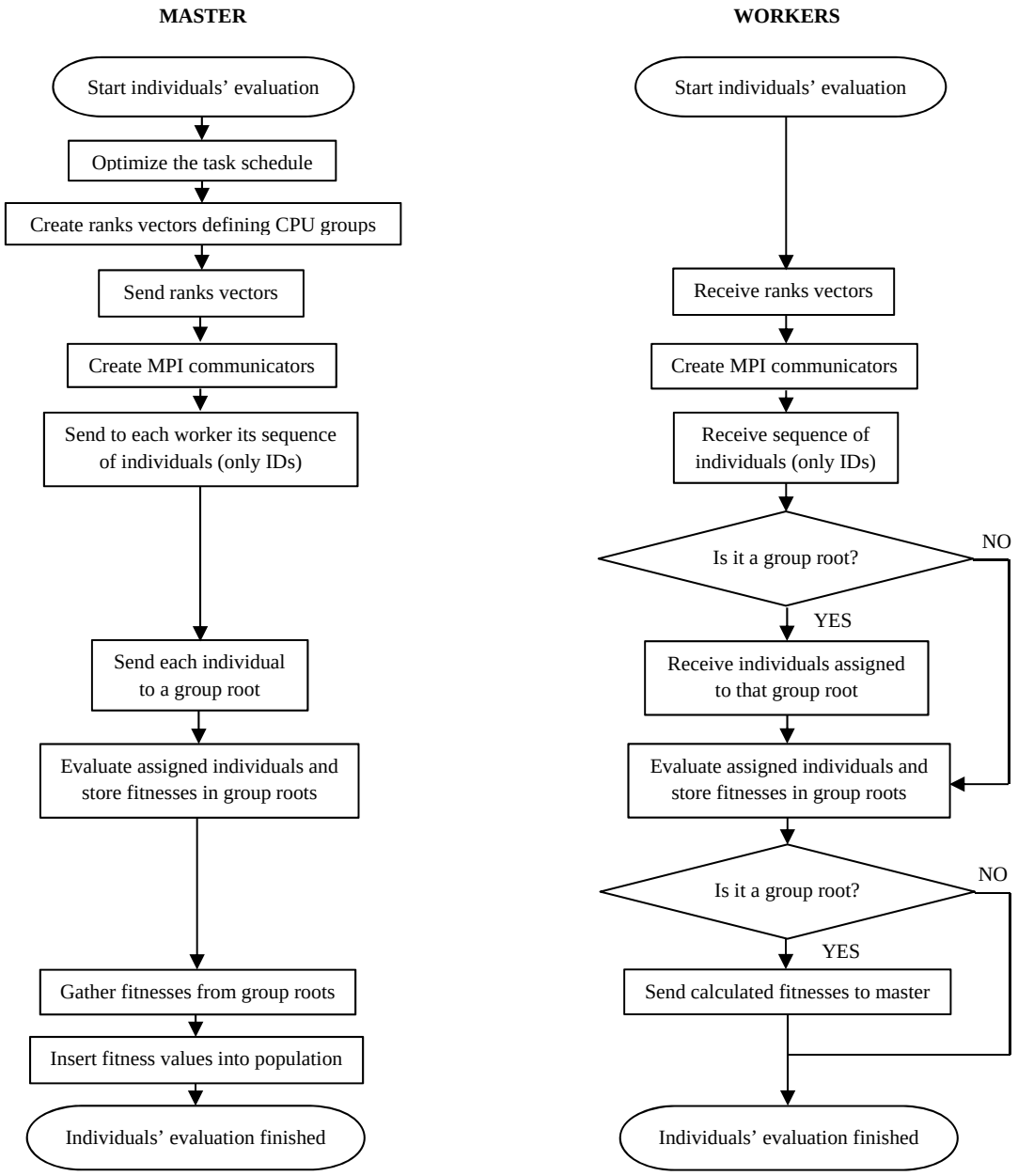

Figure 5: Load balancing master-worker parallelization strategy.

precomputed table (Table 1). In the standard parallelization strategy, 4 processors were assigned to each individual. On the other hand, the load balancing strategy had freedom to assign from 1 to 8 cores per individual. The obtained results are shown in Table 2.

The load balancing strategy outperformed the standard strategy. It can be observed that the scalability was not linear and using 4 cores per individual was not efficient in the

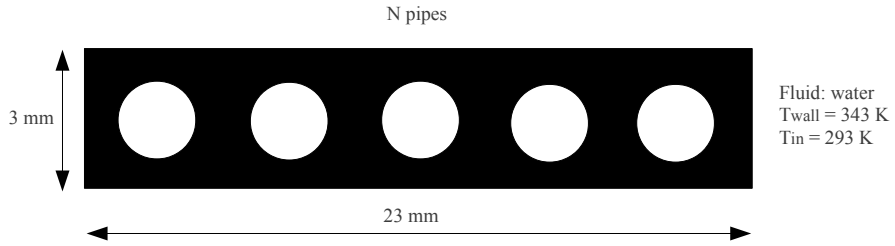

Figure 6: Power electronic device refrigerated by a fluid circulating through pipes.

Table 1: Precomputed evaluation times for Example 1.

	Evaluation times (seconds)		
n Pipes	**1 core**	**4 cores**	**8 cores**
8	70.73	38.70	37.66
76	199.49	65.00	38.85

Table 2: Comparison of parallel strategies for Example 1.

	Averaged evaluation times (seconds)		
	Standard	**Balanced**	**Time saving**
Initial eval.	177.67	103.73	41.6%
Generation 1	181.59	95.74	47.3%
Generation 2	199.53	96.70	51.5%
Generation 3	195.73	128.95	34.1%
Generation 4	185.30	129.17	30.3%
Generation 5	161.10	108.46	32.7%

standard strategy (most individuals had between 8 and 40 pipes). The load balancing strategy noticed this and used no more than 2 cores per individual.

4.2 Example 2: Theoretical case based on Example 1

In the second example, the behavior of Example 1 is simulated by means of an analytical function. In this case, the objective function is a parabolic function with the optimal point in 60 pipes and linear parallel scalability has been imposed to the evaluation of the objective function. Evaluation times are fixed according to Table 3.

Table 3: Evaluation times fixed for Example 2.

	Evaluation times (seconds)	
n Pipes	**1 core**	**32 cores**
8	1273.14	39.79
76	3590.82	112.21

128 CPUs have been used for the tests. In the standard parallelization strategy, 16 processors were assigned to each individual. On the other hand, the load balancing strategy had freedom to assign from 1 to 32 cores per individual. The obtained results are shown in Table 4.

In this case the load balancing strategy outperformed the standard strategy again, but needed longer time for solving the combinatorial scheduling problem. This is due to the size of the problem, since more CPUs per individual can be used and this fact slows down the overall resolution time.

Table 4: Comparison of parallel strategies for Example 2.

	Averaged evaluation times (seconds)			
	Standard	**Balanced**	**Time saving**	**Scheduling time (seconds)**
Initial eval.	420.94	356.64	15.3%	28.39
Generation 1	392.13	358.03	8.7%	46.78
Generation 2	426.23	386.76	9.3%	44.62
Generation 3	488.78	429.28	12.2%	37.27
Generation 4	460.35	420.94	8.6%	29.55
Generation 5	494.45	460.92	6.8%	30.62
Generation 6	454.74	429.56	5.5%	40.06
Generation 7	625.18	489.38	21.7%	28.67
Generation 8	494.50	489.59	1.0%	50.13
Generation 9	494.52	432.84	12.5%	44.96
Generation 10	557.06	495.66	11.0%	33.53

5 CONCLUSIONS

The state of the art of parallelization strategies for genetic CFD optimizations has been analyzed. This has allowed to find a problem of parallel computational efficiency in optimizations with long and highly variable evaluation times. In order to tackle this issue, a new load balancing master-worker parallelization strategy has been proposed, composed by 2 major steps: estimation of fitness evaluation times and task scheduling. Then, the new strategy has been tested in 2 small scale cases, showing considerable improvement potential for the computational efficiency of genetic algorithms.

The future lines of work include the development of an algorithm for estimating evaluation times, the addition of dynamic time scheduling strategies, the improvement of MPI communications using one-sided techniques, the improvement of the resolution of the task scheduling combinatorial problem (both the resolution time and the solution's quality) and the application of the new load balancing strategy to massively parallel simulations.

6 ACKNOWLEDGMENTS

This work has been financially supported by the Ministerio de Ciencia e Innovación, Spain (ENE-2014-60577-R) and an FPU doctoral grant (Formación de Profesorado Universitario) awarded by the Ministerio de Educación, Cultura y Deporte (Spain) to I. Aizpurua (FPU12/06265). The authors thankfully acknowledge these institutions.

REFERENCES

[1] Knysh, D.S. and Kureichik, V.M. *Parallel genetic algorithms: a survey and problem state of the art*. Journal of Comp. and Syst. Sciences Int., 49 (4), 579-589 (2010).

[2] M.S. Eldred, W.E. Hart, B.D. Schimel and B.G. van Bloemen Waanders *Multilevel parallelism for optimization on MP computers: theory and experiment*. Proceedings of the 8th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, Long Beach, CA, Sep 2000, AIAA-2000-4818.

[3] G. Utrera, J. Corbalan and J. Labarta *Scheduling parallel jobs on multicore clusters using CPU oversubscription*. The J. of Supercomputing, 68 (3), 1113-1140 (2014).