2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Optimizing the Execution of Statistical Simulations for Human Evolution in Hyper-threaded Multicore Architectures

Raquel Dias, César A. F. De Rose
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
raquel.dias.001@acad.pucrs.br,
cesar.derose@pucrs.br

Antônio Tadeu Azevedo Gomes
National Laboratory for Scientific Computing (LNCC)
Rio de Janeiro, Brasil
atagomes@lncc.br

Nelson J. R. Fagundes
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
nelson.fagundes@ufrgs.br

***Abstract*—Simulations of statistical models have been used to validate theories of past events in evolution of species. Studies concerning human evolution are important for understanding about our history and biodiversity. However, these approaches use complex statistical models, leading to high computational cost. The present paper proposes optimization techniques for Hyper-threaded multicore architectures to improve the computational performance of these simulations. Combining granularity studies and Hyper-threading optimization, we improved the performance of simulations in more than 30%, if compared with common parallel execution (default parallelization applied by users). The performance was evaluated using a complex example of human evolution studies [1]. For this example, our techniques enable the user to decrease the simulation execution time from 50 days (sequential runtime) to less than 5 days. In addition, the evaluation has been extended for simulations running on multiple multicore cluster nodes. Our measurements show a high Speed-up, close to theoretical maximum, being 129 times faster for 160 computational cores. This represents an efficiency of 81%.**

Hyper-threading, ABCToolbox optimization; granularity; workloads; statistical simulations;

I. INTRODUCTION

Simulations of statistical models are important for evolutionary studies of many species. Studying the human evolution, for example, we can understand our own history. Many statistical models and theories concerning human evolution have been proposed [2], however the use of genetic data for estimating the relative likelihood among them has started recently [1][3]. These methods can be applied in order to evaluate the effects of demographic and historic events, which may have influenced the biodiversity of a given species. These analyses are performed by means of simulations that are based on statistical inference [4][5].

The evolutionary simulations are based on experimental data (genetic or morphologic) and historical or archeological data on a given species. Through this information the researcher may build a theory concerning the evolution of a species (demographic model or evolution scenario). The model, based on statistical parameters, is simulated and compared against other theories. This comparison is performed testing these models' probability of generating the current characteristics of the given species.

In simulations of high complexity, such as human evolution studies, it is necessary to develop extremely detailed statistical methods. Sampling simulations are performed with a high number of parameters that demand a higher computational cost. Furthermore, a high number of samples must be generated in advanced probabilistic models, in order to acquire results with high degrees of confidence. These features also lead to high computational cost, implying the need to use techniques to optimize performance.

The present work proposes and evaluates an optimization technique for improving the computational performance of evolutionary simulations that are based on statistical inference. Some parallelization criteria were evaluated, such as the use of parameter sweep and workload. Combined with the parallel execution, we applied the optimization techniques of granularity evaluation and use of Hyper-Threading. This work focused on examples of demographic models of human evolution, proposed by Fagundes *et al*.[1]. In addition, the tests have been extended to more nodes of a cluster, and for another example problem. The performance analysis was carried out in order to explore the adaptation of the present optimization in other cases, with diverse complexities and environments.

The present paper structure is defined as follows. In section 2 we detail the main features of the problem and the simulation tool that will be optimized. Section 3 describes related simulation tools, as well as the common method for parallelization applied for the tool which will be optimized. In section 4 we present the proposed optimization techniques. Section 5 presents the evaluation of the proposed techniques, comparing it with non-optimized execution. In section 6 is the discussion on the obtained results and the main advantages acquired through the presented optimization technique.

II. SIMULATIONS OF STATISTICAL MODELS FOR HUMAN EVOLUTION

Studies by Fagundes et al. [1] compare three alternative models of human evolution: (A) African origin, (B) assimilation models and (C) multiregional sources evolution. The model variants are distinguished by their patterns of population growth: exponential, instantaneous, and negative, with sources in multiple regions or a single one. The schematic representation of some of the compared models is demonstrated in Figure 1.

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.87

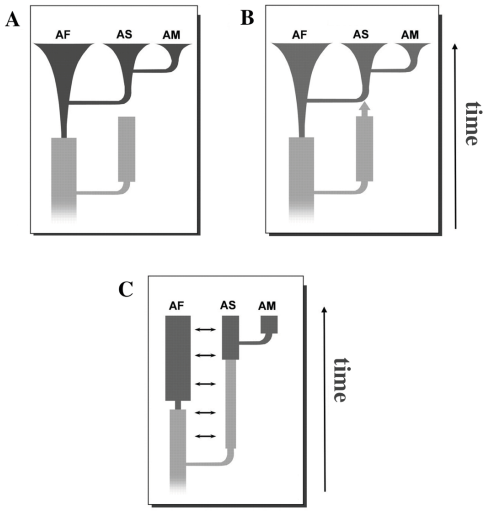

Figure 1. Schematic examples of simulation models for human evolution. Adapted from [1].

The demographic models represent modifications in the population size range over time. Narrowing represents decreasing of population size, gradual enlargements represent exponential growth (A and B), and immediate enlargements represent instantaneous growth (C).

In order to perform the simulations of the proposed evolutionary models, these were described through a set of discreet or continuous parameters. Some examples of such parameters are: minimum and maximum starting population size, migration rate, population growth rate, genetic variation rates, parameters deviations, and others. These approximations are performed based on historical and/or genetic data. Having a detailed description of the proposed theories, these models can be used in evolutionary simulation tools.

The work presented by Fagundes *et al*. is a representative example of current evolutionary studies, as for its computational cost and complexity levels. Considering this, we reproduce the same work by Fagundes et al., aiming to propose and evaluate optimization techniques against this current example workflow. We review the main features of this workflow and describe the proposed optimization techniques in the next sections.

A. Evolutionary Studies Workflow with ABCToolbox

The simulations of the described problem in this paper make use of a package for *Approximate Bayesian Computation*, known as ABCToolbox [6]. This tool consists in sampling algorithms, based on a likelihood distribution. These algorithms, named *Monte Carlo* (MC) and *Markov Chain Monte Carlo* (MCMC), enable the generation of sampling models. The samples are created by approximation methods, guided by delimiter parameters. [7]. The *workflow* of evolutionary studies consists in a set of steps for configuration, simulation and analysis.

Initially, the user gives as parameters a set of features that forms the evolutionary model or hypothesis that is being proposed (statistical model). It is done using an input file for the simulation program, which contains all the statistical parameters of the proposed evolutionary scenario. After that, using MC or MCMC methods combined with Bayesian inference, a random sampling procedure is performed. The sampling is delimited to likelihood intervals obtained from the proposed model. The underlying data generated in this step is a set of randomly created samples, whose statistical parameters and characteristics are assigned according to the boundaries of the proposed model. This method is well known and widely applied in several research lines. Among its main applications, are issues related to the sampling randomness, likelihood and inference [6].

After the sampling step, the evaluation of parameters distribution is performed for the evolutionary theory simulated. In order to do this, it is tested if the hypothesis of the proposed model may actually have leaded the current characteristics of human species. This test is performed comparing the experimental statistic data with simulated ones (summary statistics evaluation). In the end of this analysis, the output is a comparative score representing the likelihood of each compared model.

In cases that involve complex models, such as in the present work case, a huge number of samples is necessary. This occurs in order to ensure the reliability and randomness of the comparative procedures. In addition, they may be needed to rerun the simulations, adjusting parameters for calibrating the evolutionary scenarios. In this manner, initially the input parameters are evaluated for representativeness of the proposed model (parameters calibration). Figure 2 shows an overview of the workflow for evolutionary studies, where the simulation procedures of ABCToolbox are involved.

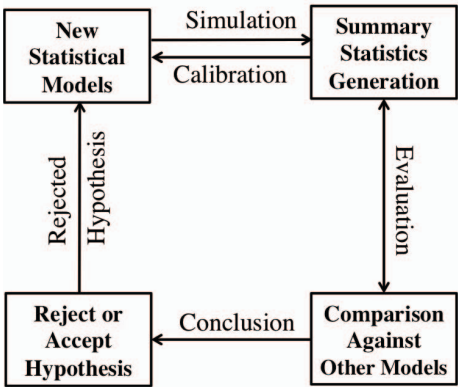

Figure 2. *Workflow* for evolutionary studies where ABCToolbox simulations are involved.

Lastly, after developing and comparing representative statistical models for each theory, it is possible to choose the higher scored model as the most feasible or reliable between the compared ones. If one of the hypotheses presents higher feasibility among others, so this is accepted as the most

consistent. If the opposite occurs, so the hypothesis is less probable than others compared.

The simulation performed in the present work takes a sequential run time of around 50 days, in an average computer. It implies the need for dividing the workload between multiple processors, in order to reduce the execution time.

III. Related Work

Many parallelization strategies for MCMC methods have been developed [8][9][10][11]. Besides the ABCToolbox simulation tools, other evolutionary simulation frameworks have been investigated for improving performance. An example is the PBPI program, which combines MCMC methods with Bayesian inference for performing phylogenetic studies [12]. The parallelism of PBPI was developed by means of distributing MCMC sampling methods. PBPI tests have demonstrated a moderate performance in a Beowulf cluster of 32 nodes [13][14]. The program presented a weak scaling, with low efficiency for higher number of cores, with a *Speed-up* of 46 for 64 cores (71.9% efficiency), 53 for 128 (41.4%) and 75 for 256 cores (29.3%).

Another program, known as MrBayes, is an example of parallel implementation for evolutionary simulations with the use of MCMC methods [15]. For its parallelization two programing models were used: message passing interface and shared memory. For the MrBayes program, such kind of implementation was necessary because its simulation procedures make use of a sampling algorithm variant purely based on MCMC. Unlike ABCToolbox, this variant, known as *Metropolis Coupled MCMC Methods,* there is interdependence between processes. Tests have demonstrated a moderate scaling, with *Speed-up* of 21 for 32 cores, with 65.6% efficiency, for the parallel implementation of MrBayes [15].

In the present work, techniques for improving the performance of evolutionary simulations of ABCToolbox are evaluated. The performance evaluation is focused on demographic model examples evaluated by Fagundes et al. [1]. ABCToolbox has a workload for which little effort is required to separate the problem in parallel tasks. For the parallel execution the simulation parameters were distributed and the workload granularity was analyzed. For the optimization of parallel execution *Hyper-Threading* was employed [16] combined with granularity optimization. In this work we introduce a new optimization technique in order to improve the performance of ABCToolbox.

A. Selecting a Template (Heading 2)

The ABCToolbox can be considered as an embarrassingly parallel program, being parallelized through the distribution of input parameters between processes, with no major efforts. Its parallel execution is well known and applied by users [17]. Unlike MrBayes program, ABCToolbox does not present dependence between processes, which enables its parallelization, without the need for message passing interface. For the parallel run, the total number of simulation iterations is divided among the total number of processes.

In the parallel ABCToolbox, the total number of iterations is settled by an input parameter named as *nbsims*. For its parallel run, the value of *nbsims* is divided by the total number of processes, represented by *np*. An example of a ABCToolbox run, with the division of this parameter for 4 processes, is described in Table 1.

In this example, the parameter *nbsims* (1000 iterations) is divided by *np* (4 processors), resulting in 250 iterations by processor. *Input* represents the statistical parameters for the model. The parallel run of ABCToolbox will be used in order to compare its gain of performance against the present optimization techniques proposed.

TABLE I. Example of ABCToolbox Parallel Execution

np	*nbsims*	Run commands generated
1	1000	ABCSampler input 1000
4	*nbsims/np*	ABCSampler input 250
		ABCSampler input 250
		ABCSampler input 250
		ABCSampler input 250

B. Workload Granularity

The granularity, which is the ratio between processing and communication, is an important issue for the performance of parallel programs [18][19]. Although not having interdependence among processes, at the end of execution, each ABCToolbox process writes its results to hard disk. This kind of communication may interfere in performance if the workload grain is too small. This issue was considered before applying the presented herein optimization.

Another feature of ABCToolbox is that, for each iteration of simulation, it takes approximately the same runtime, without remarkable variations. There is no big difference between processes workloads. All these issues were considered for the performance analysis and employment of HT optimization. The granularity evaluation was performed in order to explore the performance of parallel ABCToolbox, combined with the use of HT.

IV. Optimizations Through the Use of *Hyper-Threading*

The technique explored for improving the performance of evolutionary simulations, using parallel ABCToolbox, was the use of HT combined with granularity optimizations. HT is an Intel proprietary technology that works replicating modules of a given processor. With HT, a physical processor starts to be considered by the operating system as two logical processors. In this manner, processor modules that are idle can be available for better use of performance [16].

Benchmark tests have demonstrated that the performance of Intel Xeon processors can be improved 30% with HT [16]. Without the use of HT, in order to obtain the same improve of performance, it would be necessary to obtain new physical processors. Considering this, HT is a strategy regarded as an advantageous feature at costs and benefits levels, for improving the performance of multiprocessors without any additional cost.

Many program features may influence the performance of the HT technology, such as: workload balance, interdependence between processes, use of memory, access to memory, etc. Among them, an important feature for obtaining performance by the use of HT is the memory usage of applications. With HT enabled, the number of logical processors is duplicated, what consequently increases the number of memory accesses. This may harm this technique efficiency, when applied to high memory usage programs.

The characteristics of ABCToolbox program support the use of HT optimization technique: there is no dependence among processes (no need of message passing interface), low memory usage (a maximum of usage around 35%, for a node with 16GB available) and a balanced workload between processes. The tests for this optimization technique were focused in the simulations of statistical models described in the present work, analyzed by Fagundes et al. [1]. However, the present optimization was also evaluated for another example, in order to verify its adaptation to other kinds of problems.

V. PERFORMANCE EVALUATION

The present work was performed through the following computational environments:

- Atlântica cluster (10 nodes): 2 Intel Xeon Quad-Core E5520 2.27 GHZ processors, *Hyper-Threading*, 16 cores per node, 16 GB of memory. *Cluster* located in Laboratório de Alto Desempenho (LAD-PUCRS) [1]. 10 nodes were used for performance testing. These machines are connected by Gigabit Ethernet network.
- Pantanal cluster (6 nodes): 2 Intel Xeon 3.6 GHZ processors, *Hyper-Threading*, 4 cores per node, 2 GB of memory. *Cluster* located in Laboratório de Alto Desempenho (LAD-PUCRS)[1]. These machines are connected by Gigabit Ethernet network. One node used for performance testing.
- Altix-Xe cluster: 2 Intel Xeon Quad-Core E5520 2.27 GHZ processors, *Hyper-Threading*, 16 cores per node, 24 GB of memory. *Cluster* located in Laboratório Nacional de Computação Científica (LNCC) [2]. These machines are connected by Gigabit Ethernet network. One node used for performance testing.
- Rachserver: 2 Intel Xeon 8 cores, 2.00 GHZ processors, Hyper-Threading (32 cores per node), 64 GB of memory. Server located located in Laboratório de Alto Desempenho (LAD-PUCRS)[1].

All nodes are multiprocessors from clusters. The environments rely on the use of resource scheduling TORQUE e SGE [20][21]. There is the same processor model in both clusters Atlântica (TORQUE) and Altix-Xe (SGE). However, since each one has different resource scheduling managers, both were kept for the development of *scripts* for parameter sweep. In addition, variations in resources administration may influence performance.

A. Parallel ABCToolbox

The present optimizations were compared against the default ABCToolbox. For this, tests were performed with parallel ABCToolbox (without optimization), 5 times, for a simulation of 10 iterations. The statistical parameters were applied according to the simulation scenario described in this work. The runtime was measured for this test set, obtaining an average runtime. For measuring the performance, the acceleration factor, or Speed-up, was calculated [22][23]. For measuring the gain of performance regarding the computational power available, efficiency factor was calculated. This value indicates the effective use of processors [23]. The results demonstrate a high potential for parallelization through the division of nbsims parameter, since among all the tests the acceleration factor presented a speed-up close to the maximum theoretical value expected (Figure 3).

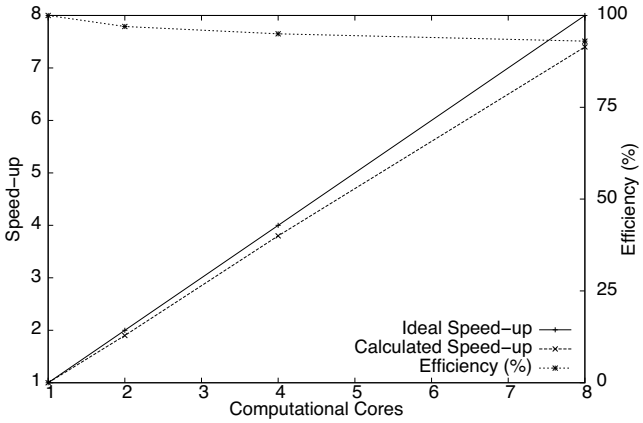

Figure 3. Performance acquired using division of nbsims parameter, using a node o Atlântica cluster.

Another feature of its execution, to be considered before applying optimizations, is the workload granularity. The disk writing time keeps the same in ABCToolbox. This occurs due to the characteristics of this algorithm, where regardless the simulation size, the summary statistics calculated from the sampling return a fixed number of values. These values are calculated as an overall descriptive statistics summary from all samples (average, standard deviation, variance, etc.).

Therefore, the size of simulated samples may vary while the statistic summary results will present the same size. This issue may be related with what is observed in Figure 4. 10 repetitions were performed for each test described, with each

[1] http://www.pucrs.br/ideia/lad

[2] http://www.lncc.br

workload size. The average value of runtime and its standard deviations were calculated. Through the results, we observe that a minimum 200 iterations by process is an recommended grain size, in order to obtain a better performance, in diverse computational environments.

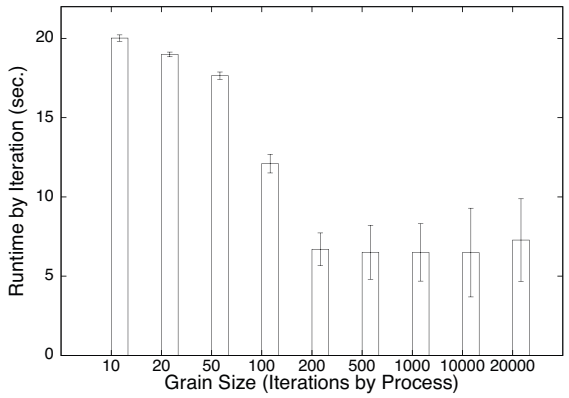

Figure 4. Workload granularity using HT (16 cores) in a node of Atlântica cluster.

Many factors may affect results of granularity, such as: memory access, cache size, processor clock, and disk access. This workload granularity optimization was combined with HT in order to improve performance.

B. *Use of Hyper-Threading combined with grain size optimization*

Through our tests, we observe that using HT combined with inappropriate granularity may dramatically decrease HT efficiency, as shown in figure 5. In addition, using at least a grain size of 200 iterations by computational core, the efficiency of HT improves substantially (29%), compared with smaller grain sizes tests.

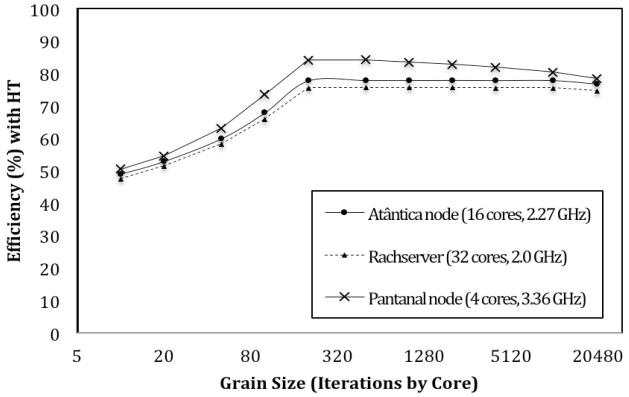

Figure 5. Efficiency using HT combined with grain sizes optimizations.

The same performance evaluation described in previous section was carried out for evaluating the use of HT technology. The evaluation was performed in order to compare the gain of performance by the use of HT against the parallel ABCToolbox commonly used (without HT). The improvement in the runtime was evaluated for each one of the computational environments described. The performance results with HT, combined with the parameters division, are demonstrated in Figure 6. The performance for all the computational environments can be observed.

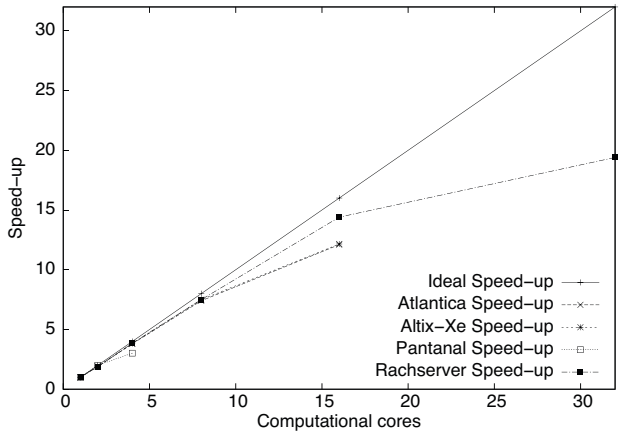

Figure 6. Performance acquired with Hyper-Threading.

For the Atlântica and Altix-Xe clusters nodes (2 x Intel Xeon Quad-Core E5520 2.27GHZ), it was observed a gain of performance of ~38% if compared with parallel ABCToolbox without using HT, nor granularity optimization. For Pantanal cluster (Intel Xeon 3.6GHZ), the performance improvement was around 51% of efficiency. The results from 1 to 8 cores represent the optimization without HT (Figure 3). The acceleration obtained for 16 cores represents the use of HT for the same 8 cores used in previous test (Figure 5). For the Rachserver (2 x Intel Xeon 8 cores X6550 2.00 GHZ), it was observed an improvement of performance of ~29% using HT (32 cores), if compared with parallel ABCToolbox not using HT (16 cores). In Figure 5, 32 cores represent the acceleration achieved with HT for Rachserver.

The present simulation for human evolution needs 5 million iterations to achieve a desired degree of confidence or representativeness. This total value of iterations takes a sequential time of around 50 days. In present work, this time has been reduced in approximately 90%. In other words, a simulation that takes around 50 days of sequential time, has become 12 times faster, running at less than 5 days. The results of runtime for this simulation can be observed in Figure 7.

The performance improvement obtained (~38%) using HT combined with appropriate grain size is better than the value expected through the use of HT (30%) [16]. This good adaptation to this problem may occur due to the better use of the processors idle time. It may happen during the sampling steps, writing in disk, or through the transition between such activities.

The memory usage is another important issue that may influence performance. Enabling HT, the number of logical processors is duplicated, what consequently increases the memory access. This could harm the efficiency of this

technique, when applied to programs that consume large amount of memory. However, this problem was not observed in ABCToolbox, since its maximum peaks of memory usage have not surpassed 35% from the total available memory.

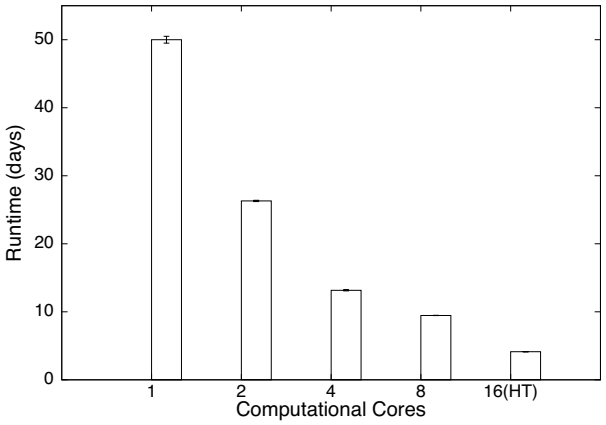

Figure 7. Results of runtime for sampling simulation with 5 million iterations.

The first tests were performed inside a single machine, only exploring the features of the computational environment and the simulation tool. Further, the same evaluation was extended to the remaining nodes of Atlântica cluster. The same optimizations described were applied, through 10 nodes (160 cores with HT). The performance evaluation results, comparing the optimized use of HT enabled and disabled, for all cluster nodes, are demonstrated in Figure 8.

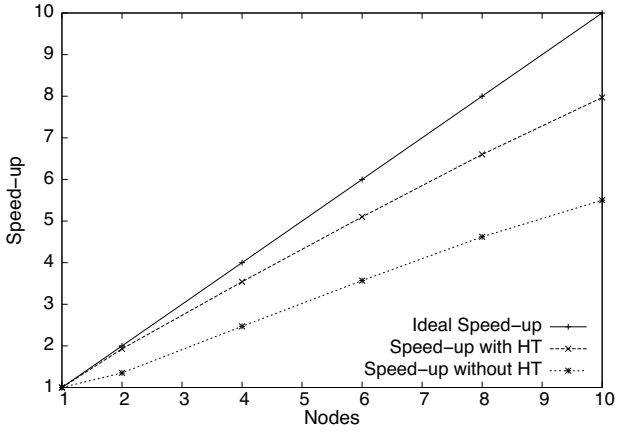

Figure 8. Performance obtained with Hyper-threading optimization among all Atlântica cluster nodes.

We verified a good adaptation of the optimization techniques along the cluster nodes, presenting a performance close to the theoretical Speed-up. We have obtained an acceleration factor of 8.1 for 10 multicore nodes, which corresponds 129,1 for 160 cores (81% efficiency), representing an improvement of ~38% when compared with the disabled HT performance. With 160 cores, the runtime was decreased from 50 days to less than 9 hours using HT, and 13,8 hours without HT. A more detailed description of results obtained for each node is shown in Table 2, where the performance and efficiency are demonstrated for each optimization technique applied in this work. These tests where carried out in Atlântica cluster (10 multicore nodes, 16 computational cores with HT).

TABLE II. PERFORMANCE OF OPTIMIZATION TECHNIQUES EVALUATED IN A MULTICORE CLUSTER

	Nodes	1	2	4	6	8	10
	Cores	16	32	64	80	128	160
Speed-up	Granularity	8	15.8	31.3	38	56	69.2
	HT	14.6	21.9	40.2	48.3	75	91.7
	HT + Granularity	15.3	30.9	56.6	68	106	129.1
Efficiency (%)	Granularity	50	49	49	48	44	43
	HT	91	69	63	60	59	57
	HT + Granularity	96	97	88	85	83	81

The performance achieved surpasses the one obtained with the parallel version of PBPI program (~75 for 256 cores)[14]. The performance also surpasses the one obtained through the parallel MrBayes program. MrBayes achieved a Speed-up of ~21 in 32 cores [15]. With the present optimization for ABCToolbox, we observe an acceleration of ~30, for 32 cores.

VI. CONCLUSION

Sampling simulations of statistical models for inference are important for understanding several issues related to the evolution of many species. These simulations present high complexity, which implies a high computational cost. For each evolutionary model used for the problem described in present work, 5 million of simulation steps or iterations are necessary. Such repetitions take an average runtime of 50 days, for each model, if running sequentially in a common computer. This high computational cost leads to the investigation of optimization techniques that can decrease the runtime in parallel architectures.

Considering this issue, we proposed and evaluated techniques for optimizing the performance of simulations with ABCToolbox. The performance evaluation focused on a representative problem of high computational cost. The use of Hyper-threading and granularity evaluation were proposed as optimization techniques. Currently, Hyper-threading technology is present in most of new processor models of Intel. The expected gain of performance (30%), plus the suitable features presented in ABCToolbox motivated the use of HT. Besides, workload granularity optimization was evaluated in order to improve performance combined with HT technology.

Through the proposed optimizations, the runtime of ABCToolbox program was reduced by 12 times, for 16 computational cores in one cluster node. Furthermore, the tests were extended for other cluster nodes. Our best result was verified when optimizing for 10 nodes (160 cores), with a *Speed-up* close to the theoretical value expected (129) and 81% efficiency. We can verify that the optimization not only is adapted for one kind of problem, but also for a simpler example.

Further, the proposed optimization techniques also can be applied to other programs for evolutionary simulations. Like ABCToolbox, other programs are used to perform the same kind of evolutionary studies described in this work. Although varying some features, they share runtime and complexity characteristics among their simulation strategies. For instance, PBPI and MrBayes programs use MC and MCMC methods, which are very similar to the ones used in ABCToolbox [14][15]. Considering this, it is expected that our optimizations may be successfully applied to such programs for evolutionary simulations. On the other hand, other optimization techniques, such as cache usage optimization, could be adapted and evaluated for ABCToolbox in future work.

REFERENCES

[1] Fagundes N.J.R, et al., "Statistical Evaluation of Alternative Models of Human Evolution", PNAS, 6/11/2007, pp. 17614-17619.

[2] Stringer C., "Modern Human Origins: Progress and Prospects", Philosophical Transactions of Royal Society B., 2001, pp. 563–579.

[3] Ray N., et al., "A Statistical Evaluation of Models for the Initial Settlement of the American Continent Emphasizes the Importance of Gene Flow with Asia", Molecular Biology Evolution, Oxford University Press, Oxford, UK, 2010, pp. 337-345.

[4] Knowles L.L., "Statistical Phylogeography", Annual Review of Ecology, Evolution and Systematics, Annual Reviews, Palo Alto, CA, USA, 1/12/2009, pp. 593-612.

[5] Nielsen R., and Beaumont M.A., "Statistical Inferences in Phylogeography", Molecular Biology, Blackwell Publishing Ltd, Malden, MA, USA, 4/2/2009, pp. 1034-1047.

[6] Wegmann D., Leuenberger C., Neeuenschwander and Excoffier L, "ABCtoolbox: a Versatile Toolkit for Approximate Bayesian Computatons", BMC Bioinformatics, BioMed Central, 2010, vol. 11, pp. 1-7.

[7] Gilks W.R., "Markov Chain Monte Carlo", Encyclopedia of Biostatistics, Blackwell Publishing Ltd, Malden, MA, USA, 15/7/2005.

[8] R. Salakhutdinov, and A. Mnih, "Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo", Proceedings of the 25th International Conference of Machine Learning (ICML '08), ACM, New York, USA, 2008, pp. 880-887.

[9] Whiley M., and Wilson S.P., "Parallel algorithms for Markov Chain Monte Carlo Methods in Latent Spatial Gaussian Models", Statistics and Computing, Kluwer Academic Publishers, Hingham, MA, USA, 8/2004, pp. 171-179.

[10] Campillo F., Rakotozafy R., and Rosi V., "Parallel and Interacting Markov Chain Monte Carlo Algorithm", Mathematics and Computers in Simulation, Elsevier Science Publishers, Amsterdam, Netherlands, 8/2009, pp. 3424-3433.

[11] Corander J., Ekdahl M., and Koski T., "Parallel Interacting MCMC for Learning of Topologies of Graphical Models", Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Hingham, MA, USA, 12/2008, 431-456.

[12] Feng X., "High Performance, Bayesian-Based Phylogenetic Inference Framework", University of South California, Columbia, SC, USA, 2006.

[13] Feng X., Buell D.A., Rose J.R., and Waddell P.J., "Parallel Algorithms for Bayesian Phylogenetic Inference", Journal of Parallel and Distributed Computing – High-performance Computational Biology, Academic Press, Orlando, FL, USA, 7/2003, pp. 707-718.

[14] Feng X., Cameron K.W., Buell D.A., "PBPI: A High Performance Implementation of Bayesian Phylogenetic Inference", SC '06 Proceedings of the 2006 ACM/IEEE conference on Supercomputing, ACM, New York, NY, USA, 2006, article 75.

[15] Altekar G., Dwarkadas S., Huelsenbeck J.P., and Ronquist F., "Parallel Metropolis Coupled Markov Chain Monte Carlo for Bayesian Phylogenetic Inference", Bioinformatics, Oxford University Press, Oxford, UK, 2/2004, pp. 407-415.

[16] D. Marr, et al., "Hyper-Threading Technology Architecture and Microarchitecture", Intel Technology Journal Q1, 2002, pp. 4-15.

[17] Wegmann D., Leuenberger C., e Excoffier L., "Using ABCToolbox", University of Bern, Bern, Switzerland, 30/9/2009, pp. 51.

[18] Oliver R.L., and Teller P.J., "Are All Scientific Workloads Equal?", Performance, Computing and Communications Conference (IPCCC '99), IEEE International, 6/8/2002, pp. 284-290.

[19] Connelly C., and Ellis C.S. "A Workload Characterization for Coarse-Grain Multiprocessors", Proceedings, 9th International Parallel Processing Symposium, IEEE, 6/8/2002, pp. 393-397.

[20] G. Staples, "TORQUE Resource Manager", Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, ACM, New York, USA, Artigo 8.

[21] Gentzsch W., "Sun Grid Engine: towards creating a compute power grid", First IEEE/ACM International Symposium on Cluster Computing and the Grid, Brisbane, Australia, 7/8/2001, pp. 35-36.

[22] Karp A.H., and Flatt H.P., "Measuring Parallel Processor Performance", Communications of the ACM, ACM, New York, USA, 3/1990, pp. 539-543.

[23] Sabetta A., and Koziolek H., "Measuring Performance Metrics: Techniques and Tools", Lecture Notes in Computer Science, Springer, 2008, pp. 226-232.