Optimizing Water Cooling Applications on Shared Memory Systems

Edson L. Padoin[1,2], Andressa T. Diefenthaler[1], Matheus S. Serpa[2], Pablo J. Pavan[2], Emmanuell D. Carreño[3], Philippe O. A. Navaux[2], and Jean-François Mehaut[4]

[1] Department of Exact Sciences and Engineering
Regional University of the Northwest of the State of Rio Grande do Sul – UNIJUI
Ijuí, Brazil
{padoin,andressa.tais}@unijui.edu.br

[2] Informatics Institute
Federal University of Rio Grande do Sul – UFRGS
Porto Alegre, Brazil
{msserpa,pjpavan,navaux}@inf.ufrgs.br

[3] Department of Informatics
Federal University of Paraná – UFPR
Paraná, Brazil
edcarreno@inf.ufpr.br

[4] Laboratoire d'Informatique de Grenoble
University of Grenoble – UGA
Grenoble, France
jean-francois.mehaut@imag.fr

Abstract. The Network Search method is not yet widely used in computational simulations due to its high processing time in the solutions' calculation. In this sense, this paper seeks to analyze the gains achieved with the parallel implementation of the Network Search method algorithm for shared memory systems. The results achieved with the parallel implementation of the algorithm applied in a real water cooling system achieved a reduction of the total execution time by up to 160 times and reduction of energy consumption by up to 60 times. Given the significant reduction of the execution time achieved with the parallelization of the Network Search method, it can be applied in different scientific problems in substitution of other methods that have less accuracy in their results.

Keywords: Network Search Method · High Performance Computing · Water Cooling

1 Introduction

Computing has been responsible for significant changes in science. Through computers, problems that until now could not be solved, or that required a long time

to be solved, were within reach by the scientific community. The evolution of computer processors, which currently incorporate multiple processing units, allow the execution of tasks in parallel, allowing a reduction in execution time and an increase in the accuracy of the results. Thus, with the significant increases in computational power of computer architectures, the range of problems that can be treated computationally has been broadened.

These problems include the principles of thermodynamics, an area of physics dedicated to the study of heat movement, which helps explain different real physical phenomena related to temperature variation. In everyday life, it is possible to identify different situations and phenomena related to thermodynamics, such as the conservation of hot water in thermal bottles, which is directly related to the phenomenon of heat transfer, with heat, according to Halliday et al. [11], energy transferred from a system to the environment or vice versa by virtue of a temperature difference [7].

This water cooling process was modeled in an experiment involving four glass ampoules, differentiated by the presence or absence of vacuum and mirroring [6]. In order to obtain the mathematical model that describes the experimental data's behavior, the Inverse Problem's resolution was considered by the numerical method Network Search, which was validated by the Direct Problem's solution by Newton's Cooling Law.

However, the Network Search method is configured as an exhaustive method, which finds the best set of parameters by calculating the data distributions through all the possibilities of combinations of parameters, within the initially defined intervals, which demands a high time computational [25].

In this context, aiming to analyze the use of multicore processors in solving scientific problems, this paper presents a parallelization study of the Network Search method applied in a real application of water cooling in ampoules of thermal bottles. Our main contributions are:

- Parallel implementation of the Network Search method algorithm for shared memory systems;
- Analysis of the power demand, execution time and power consumption of the parallel version with mapping techniques;
- The tradeoff between execution time and energy consumption.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 4 describes the methodology used in the execution of the tests, the equipment used, the mathematical modeling of the real application through the inverse problem and direct problem. Results are discussed in Section 5, followed by conclusions and future work, in Section 6.

2 Related Work

Power consumption today is a central issue in the development of the next generations of supercomputers. Research efforts have focused on both performance and energy consumption, with priority being given to reducing power demand.

However, there are still gaps for parallelization of scientific applications. Some research has focused on the use of ARM processors and others the use of GPGPU accelerators to increase the energy efficiency of HPC systems.

In the first group, Andreolli et al. [1], the authors focused on acoustic wave propagation equations, choosing the optimization techniques from systematically tuning the algorithm. The usage of collaborative thread blocking, cache blocking, register re-use, vectorization, and loop redistribution resulted in significant performance improvements. Our proposal chooses a largely used seismic imaging simulation based on the acoustic wave propagation and provides a deeper evaluation of the hardware impact of the optimizations applied to the Xeon and Xeon Phi processors. Blake et al. [2] developed a comparison between multicore processors. In this study, aspects such as caching and microarchitecture are analyzed from ARM Cortex-A9, Intel Atom, XMOS XS1-G4, Intel Core i7, and Sun Niagara T2 processors. Dongarra et al. [8] analyze the energy efficiency of equipment with ARM, Intel, AMD, and NVIDIA processors. The results point to the better energy efficiency of ARM processors. However, their energy consumption measurements considered the entire system and not just the processing unit. Similar, in the work of Valero et al. [28] are presented results of Cortex-A9 architecture. In this work, an efficiency of up to 8 GFLOPS is estimated for the Cortex-A15 ARM processors. The use of ARM and Intel processors has been an important topic of research in scientific applications of Padoin et al. [20], highlighting the metrics runtime, power demand, and power consumption. Liu et al. [17] propose an approach based on profiling to determine thread-to-core mapping on the Knights Corner architecture that depends on the location of the distributed tag directory, achieving significant reductions on communication latency. Caballero et al. [4] studied the effect of different optimizations on elastic wave propagation equations, achieving more than an order of magnitude of improvement compared with the basic OpenMP parallel version.

In the second group, several studies have evaluated the performance and power consumption of GPUs. Huang et al. [13], compare performance between CPUs and GPUs using matrix multiplication algorithms. The authors conclude that heterogeneous systems with GPUs achieve performances up to 46 times higher with energy consumption up to 17 times lower. Buck et al. [3] propose four applications to evaluate the performance of heterogeneous architectures. Jiao et al. [16] utilized a subset of these applications to analyze the performance and energy efficiency of CPUs and GPUs when their dynamically changed voltage and clock frequency. Both authors indicate that using GPUs is the right course to achieve green computing. In a similar work, Padoin et al. [21] investigate the energy efficiency of a heterogeneous system (CPU + GPU) using a scientific application. In Luk et al. [18], the authors propose a methodology that automatically performs workload mapping and allocation in heterogeneous systems. Its approach reduces execution time by up to 25% and power consumption by up to 20% when compared to workload allocation statically.

The third group focused on process mapping as an effective way to improve the performance of parallel applications and propose new methods to perform

the mapping more efficiently. Tousimojarad and Vanderbauwhede [27] show that the default thread mapping of Linux is inefficient when the number of threads is as large as on a many-core processor and presents a new thread mapping policy that uses the amount of time that each core does useful work to find the best target core for each thread. Liu et al. [17] propose an approach based on profiling to determine thread-to-core mapping on the Knights Corner architecture that depends on the location of the distributed tag directory, achieving significant reductions on communication latency. He, Chen, and Tang [12] introduces NestedMP, an extension to OpenMP that allows the programmer to give information about the structure of the tasks tree to the runtime, which then performs a locality-aware thread mapping. Cruz et al. [5] improve state of the art by performing a very detailed analysis of the impact of thread mapping on communication and load balancing in two many-core systems from Intel, namely Knights Corner and Knights Landing. They observed that the widely used metric of CPU time provides very inaccurate information for load balancing. They also evaluated the usage of thread mapping based on the communication and load information of the applications to improve the performance of many-core systems. Serpa et al. [23] focus on Intels multi-core Xeon and many-core accelerator Xeon Phi Knights Landing, which can host several hundreds of threads on the same CPU. Execution time was reduced by up to 25.2% and 18.5% on Intel Xeon and Xeon Phi Knights Landing, respectively.

Other works explore the use of ARM processors and GPGPU accelerators to improve runtime and power consumption. This work seeks to explore the use of the various processing units present in the current multicore processors, making it possible to use the network searching method. Thus, this paper discusses the gains of the parallelization of this method based on aspects of a real thermodynamics application.

3 Thermodynamics and Computational Models

Thermodynamics is related to the different phenomena of everyday life that involve temperature variations, which instigate as to its cause or effect. Among these situations, it is worth noting the cooling of the hot water in thermal bottles. The scientific application that refers to the processes of transfer of heat in ampoules of thermal bottles is not recent. The invention of these containers is related to the creation of the Dewar flask in the 19th century by the Scottish physicist-chemist James Dewar ($1842 - 1923$). When evidencing that the best thermal insulator is the vacuum, Dewar began to manufacture bottles with double walls, leaving a space between them (vacuum) and coating its interior with a silver film, in order to reflect the radiation [14].

The thermos bottle is a container composed of an external body (jar or bottle made of different materials such as plastic, stainless steel) and an internal part constituted by an ampoule (usually glass). These bottles are widely used in the daily life of individuals to conserve the temperature of drinks, hot or cold, and are also manufactured in different capacities and with different storage systems.

It is to be understood that the ampoules currently manufactured are composed of two glass walls, with or without a vacuum between them, and may or may not be mirrored. According to [19] these constituent materials are determinants for the operation of the containers, which is based on the principle of avoiding the exchange of heat between the contents of its interior and the environment. Seeking to prevent the occurrence of the three forms of heat transfer: i) Conduction: the heat exchange occurs by the shock of the particles that make up the system [15]. It is avoided due to the material in which the ampoule is manufactured, the glass, which is considered good thermal insulation; ii) Convection: occurs in the fluids (gases and liquids), caused by the difference in the density of the system components [11]. In the ampoules, it is avoided due to the vacuum between the double walls of glass and the insulating cover, which prevents the contact between hot water and air; and iii) Radiation: it occurs from electromagnetic waves, and the mirrored walls cause the infrared rays emitted by the hot water to be reflected, attenuating the heat exchanges. Despite these characteristics, there are no perfect insulating materials, and therefore, there is still heat exchange, which causes the hot water inside the containers to cool down over time.

In conducting heat transfer, the exchange occurs by the shock of the particles. In this way, conduction is avoided due to the material from which it is produced, the glass, which is considered excellent thermal insulation, because it has high thermal resistance (R). A property obtained by the ratio between the thickness L of a plate and the thermal conductivity k of the material in which it is manufactured: $R = \frac{L}{k}$ (the glass is an insulation material, has a low thermal conductivity (k) $1,0 W/mK$, which guarantees its higher capacity to conserve heat) [11].

The vacuum region between the ampoules' double walls also serves to prevent heat transfer by conduction and convection. The convection exchange occurs in the fluids (gases and liquids), caused by the system components' density variation [11]. In this way, the vacuum, as well as the presence of an insulating cover (to keep the bottle closed), prevent contact between hot water and air, thereby attenuating this form of heat transfer.

When the ampoules are mirrored, the infrared rays emitted by the hot water are reflected, attenuating the heat exchanges by radiation and aiding in the water temperature's conservation. In this way, due to the ampoules' characteristics, the heat transfer can be reduced - by conduction, convection, and radiation - and conserve the hot water for longer inside the thermal bottles. However, there are no perfect insulating materials, and after a while, the water cools [19].

3.1 Mathematical Modeling

For the data's mathematical modeling, we considered the equation of the natural exponential function decreasing (since the temperature of the water decreases and tends to stabilize over time), that is, $y = A * e^{-Bx} + C$, with three non-linearizable parameters A, B and C. In [6], the characteristic equations of the data sets (nonlinear adjustment of curves) were obtained, from the resolution of

the Inverse Problem through the Network Search method, being these validated by means of a comparison with the equations and determination coefficients found by the resolution of the Direct Problem, considering, for this, the Law of Cooling of Newton's bodies.

Diefenthaler et. al [6] model the problem of heat transfer in bottles. For this, a practical experiment related to the principles of thermodynamics was carried out and the four types of glass ampoules efficiency, differentiated by the presence or absence of vacuum and mirroring, from the cooling process the hot water over time was analyzed. The experimental data obtained were modeled, is possible to find the cooling curves' equations in each container from the Inverse Problem's resolution through the Network Search method, which was validated through the comparison with the results provided by the Direct Problem resolution, by Newton's Cooling Law. From the developed one, the method's efficiency was evidenced, because the results were coherent, and high determination coefficients were obtained.

In each ampoule, a volume of 1ℓ of hot water was added at an initial temperature of $70C$. Temperature measurements were performed using four Mercury thermometers, ranging from $-10C$ to $110C$. Temperatures were recorded every $20min$ ($1200s$ by SI) in degrees Celsius, for a time of 7 hours ($25200s$), totaling 22 measurements. To obtain the data, in [6] the temperature of 1ℓ of water at an initial temperature of $70C$ over 7 hours ($25200s$) was measured, which were measured every $20min$ ($1200s$), totalizing a set of 22 data. This article considers only the bulb manufactured in a vacuum and mirrored glass, since it presents better performance in heat conservation, as can be observed in the results shown in Figure 1.

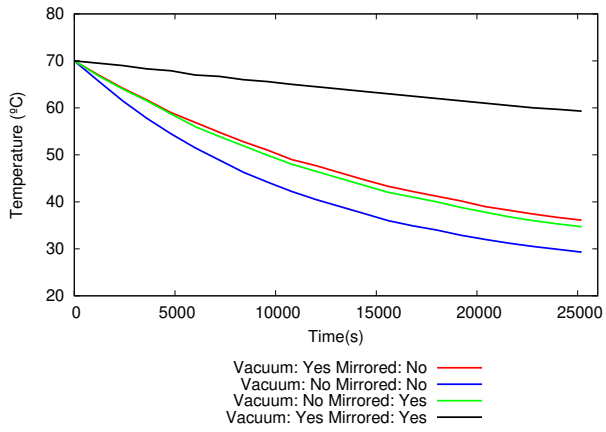


Fig. 1. Data measured in the laboratory in the process of water cooling in ampoules.

3.2 Inverse Problem via Network Search Method

The problem is configured as an Inverse Problem since from the realized experiment we have the phenomenon's effects (the experimental data), but its causes are unknown, as well as the equation (mathematical model) that represents the phenomenon and allows obtaining these data. In this way, the Inverse Problem consists of determining unknown causes from desired or observed effects [9] and [25]. In a resolution by the Direct Problem, the cause and the equation are known - Newton's Cooling Law - from which the effects are determined - the temperature that the water reaches at a given time -. Already in the resolution by the Inverse Problem, we have only the experimental data and, observing its behavior and using numerical methods; it is possible to model the data and obtain an equation that "causes these effects."

Thus, in the case of an inverse problem, the problem presented here is considered an ill-posed problem. According to [10], a well-put problem is one that presents three characteristics: i) Existence of solution; ii) Uniqueness (existence of a single solution), and iii) Stability of the solution (the solution has a continuous dependency *smooth* with the input data).

In order to determine the parameters of the equation $y = A * e^{-Bx} + C$, in [6] it was proposed the resolution through the Network Search method, which consists in the definition of valid intervals for each parameter to be estimated and of several partitions for each interval. This method finds the best set of parameters by exhaustion, calculating the distributions of the data through all possibilities of combinations of parameters, within the defined initial intervals [24]. Thus, according to [26], this method can be considered a method of suboptimal solutions, since there is no convergence criterion and the guarantee that the optimal solution belongs to the predefined intervals.

This method was implemented based on information: number of points (22), Vector X (time vector in seconds), Vector Y (temperature vector in Celsius), previous intervals in which the parameter values (A - 0 to 50, B - -0.01 to 0 and C - 0 to 50) and the number in which each of these intervals (1000 divisions) will be divided. Thus, the following algorithm was used to determine the Z parameters:

Step 1 - It is estimated the intervals (minimum and maximum) of values of each parameter A, B, C, that contain the optimal value of A, B, C;

Step 2 - Construct a partition of n points in which each interval is divided (in this case, we have 1000 possible values for A, 1000 for$B$and 1000 possibilities for the value of C);

Step 3 - For each set of values (*A1*,*A2*, ..., *A1000*; *B1*, *B2* ..., *B1000*; *C1*, *C2*, ..., *C1000*), solve the function that represents the Direct Problem;

Step 4 - Calculate the differences between the estimated solutions and the experimental data; and

Step 5 - Identify the smallest difference, which corresponds to the set of parameters A,B and C optimal for the estimated interval.

This method was chosen because of its efficiency and practicality because it is possible to define the previous intervals for each parameter to be estimated in

this situation and also because it does not involve the calculation of derivatives; however, is a method that demands long execution time.

There are different methods for solving an Inverse Problem since the choice of the Network Search algorithm is due to its efficiency and practicality since it is possible to predefine intervals for the parameters to be estimated (A, B, and C) and not wrap or derivative calculation. Although the method is exhaustive (combining all possibilities of solution), this problem seeks to determine only three parameters, a number that does not require a large execution time, which allows a solution via the Network Search method in an easy and efficient.

The values' choice of the intervals of A,B, and C and the number of subdivisions were made from the scatter diagram of the collected data. Through it, it was evidenced that the temperature decreases over time, but remains positive until reaching the thermal equilibrium with the medium, which was maintained at a constant temperature of $23C$, which justifies the choice of intervals from 0 to 50 for parameters A and C. As for parameter B, it was verified that the temperature variation is minimal for a time long interval (in seconds) and will be negative, due to the cooling process (decreasing exponential), so opted by the range of -0.01 to 0, adopting 1000 divisions for all intervals.

4 Methodology

This section describes the methodology used in this study. First, it presents the execution environment for parallel execution. In the sequence is presented the measurement methodology used in the experiments and the real thermodynamics application that was modeled through the inverse problem and validated from the direct problem.

4.1 Execution Environment

The platform used for the experiments is an Altix UV 2000 designed by SGI. The platform is composed of 24 NUMA nodes. Each node has an Intel Xeon processor E5-4640 Sandy Bridge-EP x86-64 processor with eight physical cores of 2.40 GHz. Each core of the Intel Xeon E5-4640 has L1 cache memories of 32 KB for instruction and 32 KB for data and 256 KB of L2 cache. All eight cores share a cache of 20 MB L3. Each node has 32 GB of DDR3 memory, which is shared with other nodes in a cc-NUMA form through SGI's NUMAlink6. In general, this platform has 192 physical cores and 768 GB of DDR3 memory. The platform runs an unmodified SUSE Linux Enterprise Server operating system with kernel 3.0.101-0.29 installed. All applications were compiled with GCC 4.8.2. Table 1 displays the main execution environment's characteristics.

4.2 Measurement Methodology

To analyze the performance, power demand, energy consumption, and define a Tradeoff of performance $\times$ energy. The EMonDaemon [22] tool was used to collect and analyze runtime and processor power demand during execution. This

Table 1. Equipment's Configuration

Platform	Intel Sandy Bridge-EP
Manufacturer	Intel
Processor Model	E5-4640
Clock Frequency	2,4 GHz
Number of cores	8
Memory	32 GB
Cache L1	64 KB
Cache L2	256 KB
Cache L3	20 MB
Manufacturing Technology	32 nm
Instruction Set Architecture	AVX
Floating-Point Unit (FPU)	VFPv3
Out-of-order Execution	Yes

tool facilitates the relation of this information since the current systems have different interfaces to collect information about its components. Also, some of the existing tools provide data to be parsed only after execution. Thus, for performing the EMonDaemon tool tests, it has been configured to perform processor power demand measurements every 1 s.

In order to increase the performance of this application in multicore systems, a parallel version was implemented in language *C* with *OpenMP* [5], popular programming for shared memory systems. The parallel version of the application distributes the iteration intervals between the available cores. For the mapping of threads, the tool *hw ıock* [6] was used.

Each of the tests performed in this work was repeated 10 times, to achieve a relative error of less than 5% and 95% of statistical confidence for a Student's t-distribution. Between each of the tests, the system was left in *idle* for at least 20 seconds, so that the power demand of the system stabilized.

5 Results

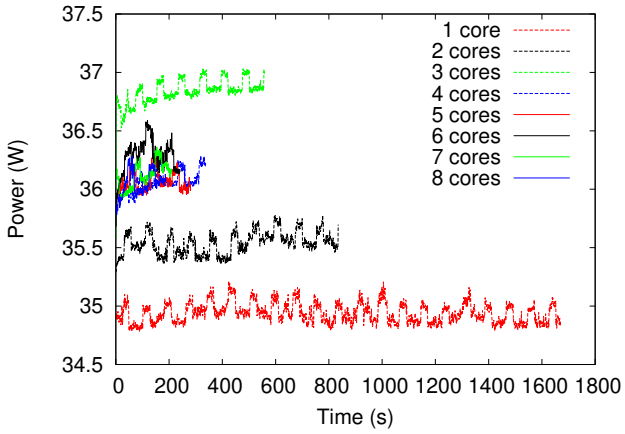
This section describes the results measured with the Network Search method's parallel runtime applied in a real water cooling system in ampoules. For the measurements, the platform described in Section 4.1 and the EMonDaemon tool. In order to facilitate the analysis, the results were organized in the following order:

- Power demand of processors with or without thread mapping;
- Execution Time versus energy consumption with or without thread mapping; and
- Tradeoff between Execution Time and Energy Consumption of the parallel version.

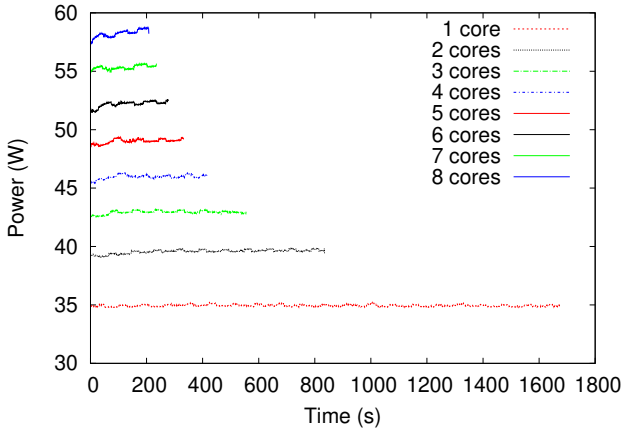
[5] http://www.openmp.org/
[6] https://www.open-mpi.org/projects/hwloc/

5.1 Power Demand

Our testbed equipment has 24 processors, each one with 8 cores, totaling 192 cores. Thus, in the first test, each processor power demand was measured when the equipment was idle. This measurement is important to define the power demand in idle and its an increasing during the executions of each test. In the idle state, the average power demand measured of the processors was (20.54 W).



(a) The power demand of the executions *without* threads mapping



(b) The power demand of the executions *with* threads mapping

Fig. 2. Instantaneous power measured for each processor during executions.

The first test was performed without the thread mapping. 8 threads were created leaving to the operating system the decision to choose which of the 192 cores to run each one of the threads. Thus, the threads were randomly mapped in cores of different processors. For the power analysis, it then measured of each one of the processors used in the execution. In Figure 2(a) is showed the instantaneous power of the processors used in the tests. Each row represents an execution of the application for several threads. From the analysis of the results, we can see a small difference between the processors' power demands, which had a variation of 8%. In this test, the power varies between 34.77 W and 37.03 W.

A second test was performed using the thread mapping with the tool (*hw_lock*). The main goal this test is to execute all threads on the same processor; each one is mapped to each of the cores. In this test, given the different amount of cores used, there are different power demands for each execution. When only 1 core is used, the average power demand was 34.9 W. On the other hand, the power demand achieves 57.8 W when all 8 cores were used.

However, as the execution time and the power demand determine the total energy spent, the next section will analyze the impact of these variations.

5.2 Execution Time and Energy Consumption with Thread Mapping

The execution time of the sequential version of the application is 1679.2 s for a matrix order equal to 1.000 (Table 2).

With the parallel version's development that uses shared memory, the execution time has been reduced from $1679,2$ s to $210,1$ s, representing a gain or *speedup* of 7.99 to 8 cores, that is, a speedup practically linear.

Statistical tests indicate that the runtimes' results presented in this section are significantly similar since small variations were observed in the measurements performed between the two sets of tests performed (with or without mapping). Thus, in this section, for the energy consumption analysis and comparison, the average execution time measured in the two tests was taken as the basis.

In the first test (without thread mapping) different processors were used. Thus, to compute the total energy spent was considered the processors' average power that performs threads added to the processors' average power demand that remained in idle state.

In this way, it is noticed that with the parallel implementation is achieved a reduction of 79.98% in total energy consumption when running the application with 8 threads in 8 different processors. The total energy spent was reduced from 300.2 KJ to 60.1 KJ. The runtimes, the power demands, and total energy consumption for different threads numbers are presented in Table 2.

In the second test was used the threads mapping aiming at the execution of the 8 threads in the 8 cores of the same processor. As shown in the previous section, for each threads number, there is a processor power demand, which varies from 34.9 to 57.8 W. On the other hand, processors that have remained idle have an average power demand of $20,546$ W. Thus, running 8 threads on 1 single processor was achieved a reduction of up to 85.88% in total energy

Table 2. Execution time, power demand and power consumption *without* thread mapping

Processors	1	2	3	4	5	6	7	8
Time (s)	**1679,2**	839,7	561,1	419,5	337,2	281,7	241,3	**210,1**
Power of used processors (W)	34,9	71,0	110,2	143,6	179,3	216,2	250,9	286,0
Power of processors in idle (W)	143,8	123,3	102,7	82,2	61,6	41,1	20,5	0,0
Energy (KJ)	300,2	163,1	119,5	94,7	81,2	72,5	65,5	60,1
Energy Save (%)		45,66	60,19	68,44	72,93	75,85	78,18	**79,98**

consumption. It is observed, however, that with the use of thread mapping, it was possible to reduce energy consumption even further, which was 5.9% higher than the unmapped tests. The tests' results with mapping for different threads number are presented in Table 3.

5.3 Execution Time x Energy Consumption Tradeoff

As the mapping presented a more significant reduction in energy consumption, a third test was carried out aiming to relate the runtime to the total amount of energy consumed when the number of threads is increased until all the system's cores are used. In Figure 3 shows the execution times and power consumption when varied the number of threads from 1 to 192.

We realize that, with the parallel implementation, was obtained a total execution time's reduction of 160 times for 192 cores, that is, the time was reduced from 1679.2 s to 10.3 s. The total power consumption was also reduced with the parallel version. In the sequential version, considering the 24 processors octa cores of the system, the energy consumption was $866,5$ KJ. With the parallel implementation that uses the 192 available cores, the consumption has been reduced to 14.4 KJ, which represents a reduction of 60.3 times.

Two factors justify the difference between time and consumption gains, 160 times, and 60.3 times, both for 192 cores. The first is linear velocity observed up to approximately 70 threads. From these cores used, the execution time's

Table 3. Execution time, power demand and power consumption *with* thread mapping

Processors	1	2	3	4	5	6	7	8
Time (s)	**1679,2**	839,7	561,1	419,5	337,2	281,7	241,3	**210,1**
Power of used processors (W)	34,9	39,5	42,7	45,9	48,8	51,8	54,9	57,8
Energy of used processors (KJ)	58,6	33,1	24,0	19,2	16,4	14,6	13,2	12,2
Total Power Demand (W)	178,8	183,3	186,6	189,7	192,6	195,6	198,8	201,6
Total Energy Spent (KJ)	300,2	153,9	104,7	79,6	64,9	55,1	48,0	42,4
Energy Save (%)		48,72	65,12	73,49	78,36	81,64	84,02	**85,88**

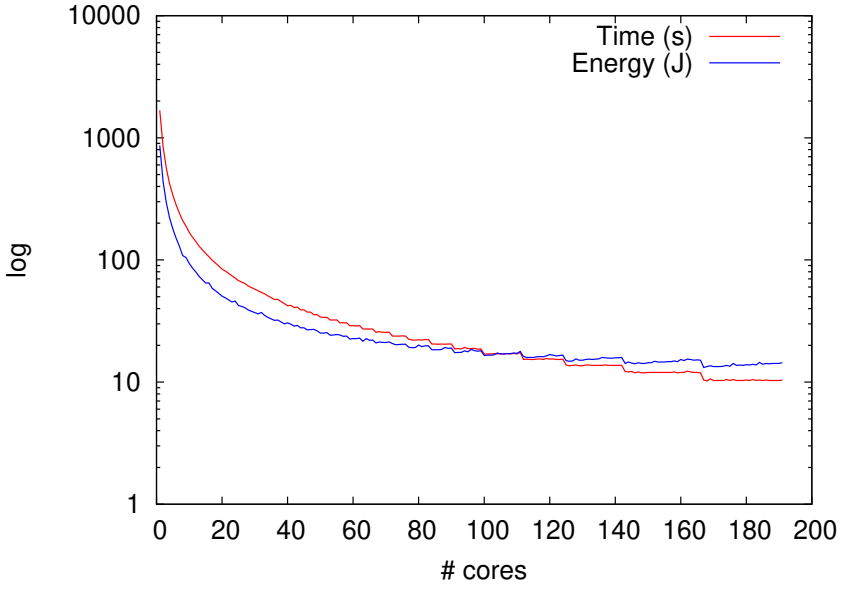


Fig. 3. Execution time and energy consumption for different number of threads.

reduction became smaller tending to stabilize, as shown in Figure 3. The second is due to the increase in static power demand of approximately 14.9 W each inclusion of a new processor in the parallel application execution.

However, according to the results obtained, it can be concluded that the Network Search method is shown as a practical and accurate and efficient method when used in its parallel version. Thus, the problem related to the high sequential version's execution time could be solved with parallel implementation and execution in multicore environments.

6 Conclusions

In this paper, we discuss our gains of up to 160× in time and up to 60× in energy consumption achieved with parallelization and process mapping in multicore processors of Network Search method applied to an actual water cooling application.

We showed that parallel Network Search Method has good efficiency and accuracy, as well as the influence the computer's configurations and architecture. We also show that Network Search Method can be applied in different scientific problems in substitution of other methods that have less accuracy in their results.

Future work includes working with other methods of parameter determination (Inverse Problem) to see if the simulation time is better than this one.

Acknowledgments

This work has been partially supported by *PETROBRAS* oil and gas company under Ref. 2016/00133-9 and the project GREEN-CLOUD: Computação em Cloud com Computação Sustentavel (Ref. 16/2551-0000 488-9), from FAPERGS and CNPq Brazil, program PRONEX 12/2014 and CNPq-Universal 436339/2018-8. We also thank *RICAP*, partially funded by the Ibero-American Program of Science and Technology for Development (*CYTED*), Ref. 517RT0529.

References

1. Andreolli, C., Thierry, P., Borges, L., Skinner, G., Yount, C.: Characterization and Optimization Methodology Applied to Stencil Computations. In: Reinders, J., Jeffers, J. (eds.) High Performance Parallelism Pearls, pp. 377–396. Morgan Kaufmann, Boston (2015)
2. Blake, G., Dreslinski, R., Mudge, T.: A Survey of Multicore Processors. IEEE Signal Proc. Magazine **26**(6), 26–37 (2009)
3. Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., Hanrahan, P.: Brook for gpus: stream computing on graphics hardware. ACM Transactions on Graphics (TOG) **23**(3), 777–786 (2004)
4. Caballero, D., Farrés, A., Duran, A., Hanzich, M., Fernández, S., Martorell, X.: Optimizing Fully Anisotropic Elastic Propagation on Intel Xeon Phi Coprocessors. In: 2nd EAGE Workshop on HPC for Upstream. pp. 1–6 (2015)
5. Cruz, E.H., Diener, M., Serpa, M.S., Navaux, P.O.A., Pilla, L., Koren, I.: Improving communication and load balancing with thread mapping in manycore systems. In: Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference on. pp. 93–100. IEEE (2018)
6. Diefenthler, A.T., Avi, P.C.: Determinaćão da curva de resfriamento da água em ampolas de garrafas térmicas. In: Anais do VII MCSUL - Conferncia Sul em Modelagem Computacional. Rio Grande/RS (2016)
7. Diefenthler, A.T., Avi, P.C., Padoin, E.L.: Processamento paralelo na determinaćão da curva de resfriamento da gua pelo método de procura em rede. In: Congresso Nacional de Matemtica Aplicada e Computacional (CNMAC). So Jos dos Campos/SP (2017)
8. Dongarra, J., Luszczek, P.: Anatomy of a Globally Recursive Embedded Linpack Benchmark. In: 16th IEEE High Performance Extreme Computing Conference (HPEC) (2012)
9. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems:Mathematics and its Applications. Springer (1996), 322 p
10. Hadarmard, J.: Lectures on the Cauchy problem in linear partial differential equation. New Haven Yale University Press (1923), 338 p
11. Halliday, D., Resnick, R., Walker, J.: Fundamentos de fsica, volume 2: gravitação, ondas e termodinâmica. 8.ed., LTC, Rio de Janeiro (2009), 295 p
12. He, J., Chen, W., Tang, Z.: Nestedmp: Enabling cache-aware thread mapping for nested parallel shared memory applications. Parallel Computing **51**, 56–66 (2016)
13. Huang, S., Xiao, S., Feng, W.c.: On the energy efficiency of graphics processing units for scientific computing. In: Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. pp. 1–8. IEEE (2009)

14. Ibarra, J.R.M.: La materia a muy bajas temperaturas. Revista Ingenieras, n. 38 **11**, 7–16 (Enero-Marzo 2008)
15. Incropera, F.P., Dewitt, D.P.: Fundamentos de transferncia de calor e de massa. 6.ed., LTC, Rio de Janeiro (2011), 698 p
16. Jiao, Y., Lin, H., Balaji, P., Feng, W.: Power and performance characterization of computational kernels on the gpu. In: Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom). pp. 221–228. IEEE (2010)
17. Liu, G., Schmidt, T., Dömer, R., Dingankar, A., Kirkpatrick, D.: Optimizing thread-to-core mapping on manycore platforms with distributed tag directories. In: Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific. pp. 429–434. IEEE (2015)
18. Luk, C., Hong, S., Kim, H.: Qilin: exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. pp. 45–55. ACM (2009)
19. Marques, N.R.L., Araujo, I.S.: Física térmica. In: Textos de apoio ao professor de Física, n.5. vol. 20. UFRGS, Instituto de Física, Porto Alegre/RS (2009), 73 p
20. Padoin, E.L., de Oliveira, D.A.G., Velho, P., Navaux, P.O.A.: Time-to-solution and energy-to-solution: A comparison between ARM and xeon. In: Third Workshop on Applications for Multi-Core Architectures (WAMCA SBAC-PAD). pp. 48–53. New York, USA (2012). https://doi.org/10.1109/WAMCA.2012.10
21. Padoin, E.L., Pilla, L.L., Boito, F.Z., Kassick, R.V., Velho, P., Navaux, P.O.A.: Evaluating application performance and energy consumption on hybrid CPU+GPU architecture. Cluster Computing **16**(3), 511–525 (2013), 10.1007/s10586-012-0219-6
22. Padoin, E.L., Pilla, L.L., Castro, M., Boito, F.Z., Navaux, P.O.A., Mehaut, J.F.: Performance/energy trade-off in scientific computing: The case of ARM big.LITTLE and Intel Sandy Bridge. IET Computers & Digital Techniques **2**(3), 1–14 (2014)
23. Serpa, M.S., Krause, A.M., Cruz, E.H., Navaux, P.O.A., Pasin, M., Felber, P.: Optimizing machine learning algorithms on multi-core and many-core architectures using thread and data mapping. In: Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference on. pp. 329–333. IEEE (2018)
24. Silva, B.F.: Mtodo da procura em rede melhorado: uma proposta para a estimao dos parmetros do modelo de rakhmatov e vrudhula. Dissertao de Mestrado do Programa de Ps-Graduao em Modelagem Matemtica da Uniju (Maio/2013), 64 p
25. Silva Neto, A.J.: Tcnicas de inteligncia computacional inspiradas na natureza: Aplicaes em problemas inversos em transferncia radiativa. In: Notas em Matemtica Aplicada, 2. ed. vol. 41. SBMAC, So Carlos/SP (2012), 148 p
26. Silva Neto, A.J., Moura Neto, F.D.: Problemas Inversos: Conceitos Fundamentais e Aplicaes. UERJ, Rio de Janeiro (2005), 168 p
27. Tousimojarad, A., Vanderbauwhede, W.: An efficient thread mapping strategy for multiprogramming on manycore processors. Parallel Computing: Accelerating Computational Science and Engineering (CSE), Advances in Parallel Computing **25** (2014)
28. Valero, M.: Towards Exaflop Supercomputers. In: High Perfomance Computing Academic Research Network (HPC-net). pp. 1–117. Rio Patras, Greece (2011)