Energy-Optimal Configurations for Single-Node HPC Applications

1st Vitor R. G. Silva
Dept. of Electronics and Microeletronics
University of Mons
Mons, Belgium
vitor.ramosgomesdasilva@umons.ac.be

2nd Alex F. A. Furtunato
Directorate of Information Technology
Instituto Federal do Rio Grande do Norte
Natal, Brazil
alex.furtunato@ifrn.edu.br

3rd Kyriakos Georgiou
Dept. of Computer Science
University of Bristol
Bristol, United Kingdom
kyriakos.georgiou@bristol.ac.uk

4th Samuel Xavier-de-Souza
Dept. of Computer Engineering and Automation
Universidade Fedral do Rio Grande do Norte
Natal, Brazil
samuel@dca.ufrn.br

5th Carlos A. V. Sakuyama
Dept. of Electronics and Microeletronics
University of Mons
Mons, Belgium
carlos.valderrama@umons.ac.be

6th Kerstin Eder
Dept. of Computer Science
University of Bristol
Bristol, United Kingdom
kerstin.eder@bristol.ac.uk

***Abstract*—Energy efficiency is a growing concern for modern computing, especially for HPC due to operational costs and the environmental impact, considering that processors have an important role in this energy consumption. In this work, we propose a methodology to find energy-optimal frequency and number of active cores to run single-node HPC applications using an application-agnostic power model of the architecture and an architecture-aware performance model of the application. We characterize the application performance using machine learning, specifically the "Support Vector Regression" algorithm. Besides that, the power consumption is estimated by modeling CMOS dynamic and static power without knowledge of the application. So, The energy-optimal configuration is estimated by minimizing the product these two models outcomes, the power model and the performance model. Then, the final model can be used to find better frequency and number of cores to aim energy efficiency application execution. Results were obtained for four PARSEC applications and, with five different inputs shows that the proposed approach used substantially less energy when compared to the DVFS governor, in best cases and worst cases.**

***Index Terms*—Energy Efficient Software, Power Modeling, Performance Modeling.**

I. Introduction

Processors are the main contributor to the power consumption of High-Performance Computing (HPC) servers. They contribute between 20 and 40% to the total servers power draw [12]. Google's servers showed that during peak utilization processors consumed about 42% of the overall servers power consumption [2]. Reducing processor power consumption is an effective approach to reduce the whole system's power consumption. Therefore, modern processors incorporate several features for power management such as independent processing cores that can be disabled by the operating system [23], clock gating techniques for reducing the dynamic power dissipation of synchronous circuits [28] and Dynamic Voltage and Frequency Scaling (DVFS) [18].

DVFS has been demonstrated to be a very effective technique for reducing the power consumption of processors [1], [3], [11], [15], [16], [19], [21], [29]. The technique tries to optimize power consumption by adjusting the frequency according to the current load of the processor. Generally, the frequency scales with the intensity of the load and the voltage scales to the minimum value that enables the selected frequency. Among other aspects, DVFS helps reducing energy consumption because it allows memory-bounded programs to be executed more efficiently [26]. Nonetheless, aspects such as load variability may compromise the effectiveness of DVFS. Another important aspect that is typically not taken into account is the number of processing cores to be used by a parallel program. This choice is left to the user, which often is not trivial as shown in this paper.

We propose a methodology to find the operating frequency and number of active cores that minimize the total energy used to execute an HPC application on a single shared-memory HPC node.

The methodology uses an application-agnostic power model and an architecture-specific application characterization to model performance. The power model is based on the modeling of Complementary Metal-Oxide-Semiconductor (CMOS) logic in function of the operating frequency [24]. It models both the dynamic and static power. Besides operating frequency, the power model is also parametric to the number of active sockets and the number of active cores per socket.

Performance is modeled by characterizing the application on the target architecture. The idea is to predict the performance of the application at any given configuration. The model takes as inputs the operating frequency, the number of active cores and the input size. The modeling is done using a supervised learning method for regression called Support Vector Regression (SVR) [25].

To find the optimal-energy configurations, the algorithm minimizes the product of outcomes of the power and performance models. This approach was validated on four PARSEC applications [4] and compared to the *Ondemand* governor,

which is the default DVFS scheme for the Linux operating system. The results show that the proposed approach was able to find configurations that used about 14× less energy when compared to the worst case of the Ondemand governor. When compared to the best case of this DVFS scheme, i.e. when the user guesses the optimal number of cores to be used, the proposed approach was able to find configurations that used as much as 23% less energy to execute the target application. The overall average energy saving reached 6% for the proposed approach when compared to the best case and 790% when compared to the worst case.

The rest of this paper is organized as follows. Section II presents the proposed models for power, performance, and energy. The experimental setup and the fitting of the models are described in Section III. In Section IV, the results of applying the proposed approach to four PARSEC applications are presented. Related works are presented in Section V. Finally, conclusions are drawn and future work is proposed in SectionVI.

II. Models

In this Section, we present the proposed power and performance models that are used to estimate the minimum-energy consumption configuration.

A. Power Model

Some of the main factors that contribute to the CPU power consumption are the dynamic power consumption, the short-circuit power consumption, and the power loss due to the current leakage of transistors, [10], [13], [14], [22]. The complexity of the circuits of modern processors makes it very difficult to model their power consumption accurately. A viable approach for modeling the CPU's power draw is to model their building components, which are mainly made out of CMOS logic gates. Thus, modeling the power consumption for one logic gate and multiplying this by the total number of gates reduces the complexity of modeling the internal circuits but still provides the sufficient accuracy needed for making optimization decisions.

There are three main components of power dissipation in digital CMOS circuits,

$$P_{total} = P_{static} + P_{leak} + P_{dynamic} \tag{1}$$

namely, static power P_{static}, dynamic power $P_{dynamic}$, and leakage power P_{leak}. According to [6], [24], the dynamic power and leakage power behavior can be approximated by:

$$P_{dynamic} = CV^2 f, \tag{2}$$

and

$$P_{leak} \propto V, \tag{3}$$

where C is the CMOS capacitance, V the voltage applied to the circuit and f the switching frequency.

Another common approximation is to expect a linear relationship between the voltage and the applied frequency [30]:

$$f \propto V \tag{4}$$

Thus, the proposed model for one processing core of a multi-core processor is derived by using (2), (3) and (4) to rewrite (1) as follows:

$$P_{total}(f) = c_1 f^3 + c_2 f + c_3, \tag{5}$$

where c_1, c_2, and c_3 are the model's parameters.

When we include the number of active cores p, the estimation of the power consumption of the whole processor becomes:

$$P_{total}(f, p) = p(c_1 f^3 + c_2 f) + c_3. \tag{6}$$

For systems that have more than one processor sockets, the power cost of enabling each socket can be considered. Adding the number of sockets s to the equation gives the final version of the power model used in this work:

$$P_{total}(f, p, s) = p(c_1 f^3 + c_2 f) + c_3 + c_4 s, \tag{7}$$

with c_4 being the model parameter for the number of sockets.

B. Performance Model

The performance model aims to estimate the application's execution time for a given target architecture based on a given operating frequency, number of active cores and input size.

The performance was modeled by sampling the execution time of the application for several combinations of discrete values of frequency, number of active cores and input size. The samples where used as a training set for a Support Vector Regression (SVR); a version of the Support Vector Machine (SVM) algorithm for regression proposed in [9].

Training the SVR means minimize the weights w subject to:

$$\begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases}$$

In our model x_i is a vector with the frequency, number of active cores and input size, y_i is the execution time measured. $\langle w, x_i \rangle + b - y_i$ is the predicted output time and ε is a free parameter that serves as a threshold.

C. Energy Model

By combining outcome of the power model described in section II-A and the SVR characterization of the application performance described in section II-B, we can estimate the total energy used by the application as follows:

$$E(f, p, s, N) = P(f, p, s) \times \mathrm{SVR}(f, p, N), \tag{8}$$

where $P(f, p, s)$ is the total power modeled by (eq. (1)), $\mathrm{SVM}(f, p, N)$ is the execution time estimated by the SVR characterization of the application, f is the frequency, p is the number of active cores, s is the number of sockets, and N is the input size.

With (eq. (8)), it is possible to calculate energy consumption estimations for every possible configuration. Then, the configuration that minimizes energy consumption for a given input can be selected. It is also possible to apply constraints

on the execution time, frequency, and the number of active cores although this is not considered in this work.

III. Experimental Setup

In the following subsections we present the software and hardware experimental setup used to validate the proposed approach.

A. Case-Study Applications

Four applications from the PARSEC parallel benchmark suite, version 3.0 [4], were used as case-studies. This suite focuses on emerging workloads and was designed to be representative of the next generation shared-memory programs for chip-multiprocessors. The four applications used in this work were chosen for being relatively straightforward to devise smaller input sizes from the standard native inputs. These are: Fuidanimate, Raytrace, Swaptions, and Blackscholes. A short description of each one follows.

1) Blackscholes: calculates the prices for a portfolio of European options analytically using the Black-Scholes partial differential equation. There is no closed-form expression for the Black-Scholes equation and as such it must be computed numerically. The program's inputs are the number of threads, the input file containing the options data, and the output file name.

2) Fuidanimate: uses an extension of the Smoothed Particle Hydrodynamics (SPH) method to simulate an incompressible fluid for interactive animation purposes. The inputs are the number of threads, the number of frames, and an input file with information of all fluid particles and his proprieties.

3) Raytrace: is a version of the raytracing method that is typically employed by real-time animations such as the ones used in computer games. It is optimized for speed rather than realism. The computational complexity of the algorithm depends on the resolution of the output image and the scene. The inputs used on this applications was the number of threads, the number of frames, a 3D object and the display resolution.

4) Swaptions: Uses the Heath-Jarrow-Morton (HJM) framework to price a portfolio of swaptions. Swaptions employs Monte Carlo (MC) simulation to compute the prices. The input to this program are the number of threads, number of swaptions and the number of trials.

B. Case-Study Architecture

In the experiments performed in this work, we used compute nodes that consists of two Intel Xeon E5-2698 v3 processors with sixteen cores each and two hardware threads for each core. The maximum non-turbo frequency is 2.3GHz, and the total physical memory of the node is 128GB (8×16GB). Turbo frequency and hardware multi-threading were disabled during all experiments. The operating system used is Linux CentOS 6.5, kernel 2.6.32.

The Linux kernel has many drivers available developed by the CPU manufacturers and the community [5]. The default driver is the "acpi-cpufreq" that uses policies implemented by so-called governors that dynamically decide the frequency values. Some of the governors available are Performance, Powersave, Ondemand, Conservative and Userspace. Performance and Powersave are static, and they set the frequency to the maximum and minimum allowed values, respectively. Ondemand and Conservative implement algorithms to estimate the CPU required capacity and adjust the processor frequency accordingly. Finally, Userspace allows the user to specify the frequency.

In this work, changing the frequency of the cores was done using the Linux "acpi-cpufreq" driver. The number of active cores was changed by modifying the appropriate Linux virtual files. Both changes require root privileges. In practice, this approach can be brought into production by allowing the resource manager to perform this changes for the user using pre- and post-scripts for job submissions with energy consumption requirements.

C. Fitting the Power Model

To fit the power-model equation, the CPU was stressed up to 100% and power information was acquired from the Intelligent Platform Management Interface (IPMI) sensors with a sampling rate of about one sample per second. IPMI provides information about variables and resources such as the system's temperature, voltage, fans and power supplies; using independent sensors attach to the hardware.

The power was collected for all combinations of frequency — starting from 1.2 GHz and increasing by 100 MHz each time until 2.2 GHz is reached, and possible numbers of active cores — from 1 to 32. Between each test the CPU was left idle until it cooled down to avoid interference on the next test.

The coefficients of (7), c_1, c_2, c_3 and c_4, were found by performing multi-linear regression on the data collected. The retrieved fitting can be seen on Fig. 1.

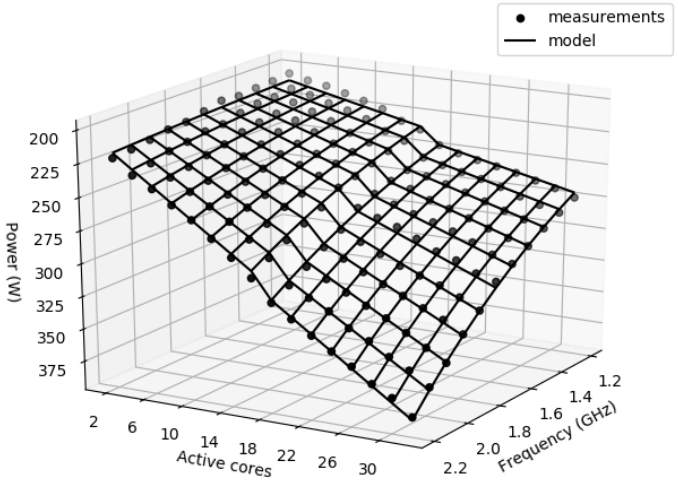


Fig. 1. Power model fitting. The dots represent real power measurements and the solid lines represents the modeled power.

The equation for estimating the power in the target architecture turned to be:

$$P_{total}(f, p, s) = p(0.29f^3 + 0.97f) + 198.59 + 9.18s, \quad (9)$$

where the unit for frequency is GHz.

To validate this model was calculated the absolute percentage error, i.e. the mean of the perceptual error on each point. This metric was chosen because of the significant difference between the smallest and the biggest values and it is calculated as follows:

$$\sum_{i}^{\#\text{samples}} \frac{|y_i - y_{\text{model}}|}{y_i}. \tag{10}$$

The resulting absolute percentage error was 0.75% and the root-mean squared error was 2.38W.

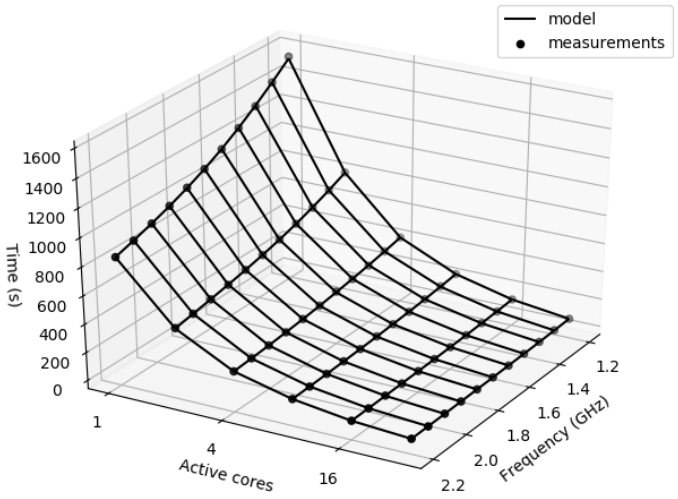


Fig. 2. Fluidanimate's performance model. The dots represent real performance measurements and the solid lines represent the modeled performance for various numbers of active cores and frequencies when running for input size 3.

D. Performance Characterization

To characterize an application, we ran it for all different numbers of active cores in the range of $1 <= p <= 32$, for all the frequencies in the range of $1.2 <= f <= 2.2$ using 100MHz steps, and for 5 different input sizes.

The input sizes were chosen in such a way that the average execution time was in the order of minutes. The sampled power information, on every second, was used to calculate the real energy usage. The total time to complete the characterization varied between one and two days, depending on the application.

The SVR model was built using the collected data. A grid search was used to tune the model parameters. In this case, a Radial Base Function (RBF) kernel and the penalty for the wrong term of 10×10^3 and gamma 0.5 [20]. To train the SVR, the data collected was divided into two parts, 90% for training and 10% to test the accuracy.

The model was validated also using a cross-validation k-fold with k equal to 10, using the Mean Absolute Error (MAE) and Percentage Absolute Error (PAE) as metrics. The average results of the cross validation can be seen in Table I.

TABLE I
PERFORMANCE-MODEL'S CROSS VALIDATION ERRORS

Application	MAE	PAE
Blackscholes	2.01	4.6%
Fluidanimate	6.65	1.89%
Raytrace	3.77	0.87%
Swaptions	2.29	2.56%

The results of the characterization can be seen in Figs. 2, 3, 4, and 5.

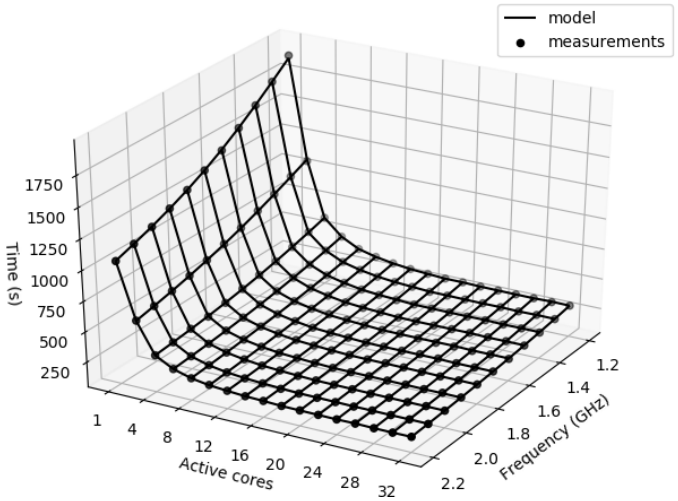


Fig. 3. Raytrace's performance model. The dots represent real performance measurements and the solid lines represent the modeled performance for various numbers of active cores and frequencies when running for input size 3.

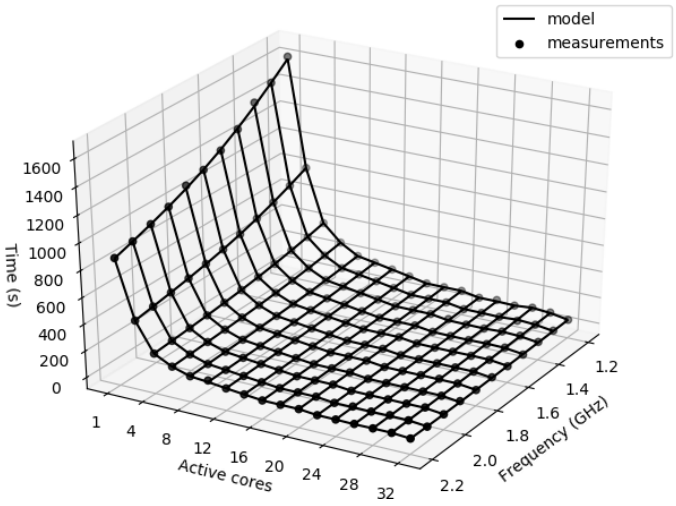


Fig. 4. Swaptions's performance model. The dots represent real performance measurements and the solid lines represent the modeled performance for various numbers of active cores and frequencies when running for input size 3.

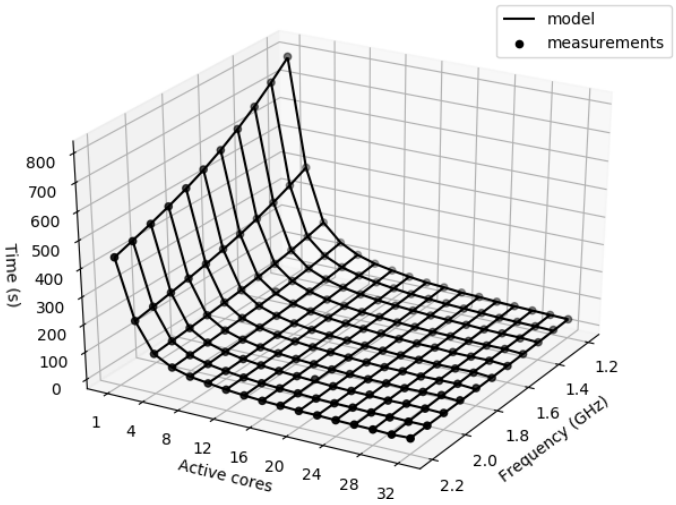


Fig. 5. Blackscholes's performance model. The dots represent real performance measurements and the solid lines represent the modeled performance for various numbers of active cores and frequencies when running for input size 3.

IV. Experimental Results

In this Section, we present results for the energy model that we introduced in Section II based on the parameter fitting described in Sections III-C and III-D. First, we compare and comment the model in contrast with the actual energy measurements. Finally, we evaluate the effectiveness of the proposed approach by comparing it to the Linux default Ondemand DVFS governor.

A. Measured versus Modeled Energy

The energy measurements were obtained by integrating the power measurements over the total execution time of the application. The power measurements were made using the IPMI sensors with a sampling rate of about one sample per second.

Figs. 6, 7, 9, and 8 plot the measured and modeled energy consumption for Blackscholes, Fuidanimate, Raytrace, and Swaptions, respectfully, for varying the number of active cores and operating frequency, running with the mid-size input.

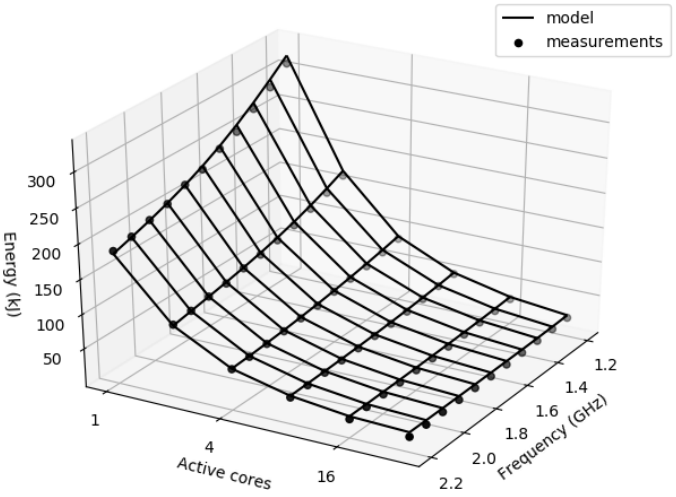


Fig. 6. Fluidanimate's energy measurements versus modeled energy consumption varying the number of active cores and operating frequency, running with the input size 3.

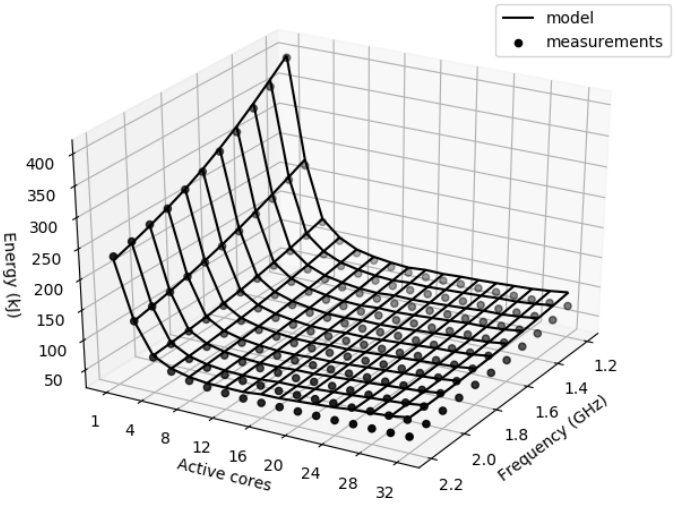


Fig. 7. Raytrace's energy measurements versus modeled energy consumption varying the number of active cores and operating frequency, running with the input size 3.

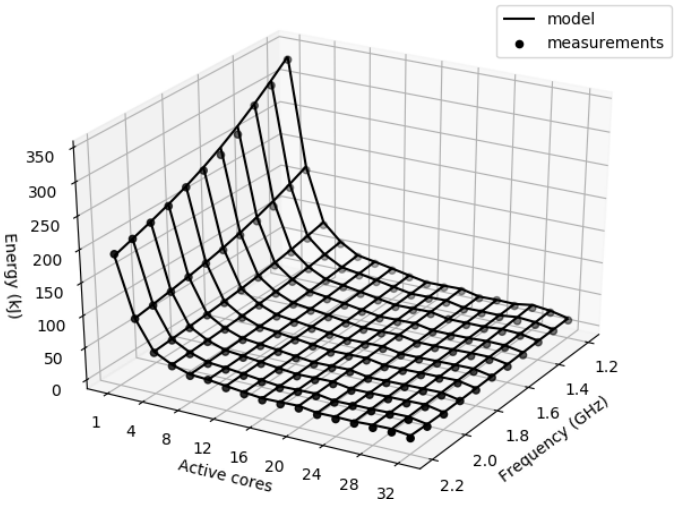


Fig. 8. Swaptions's energy measurements versus modeled energy consumption varying the number of active cores and operating frequency, running with the input size 3.

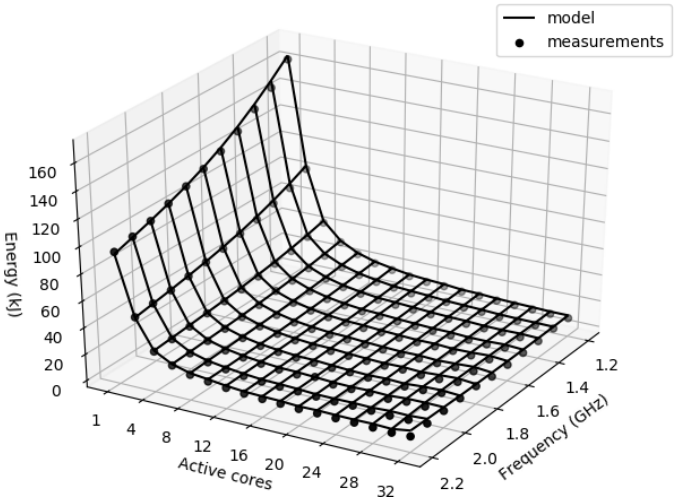


Fig. 9. Blackscholes's energy measurements versus modeled energy consumption varying the number of active cores and operating frequency, running with the input size 3.

In general, for the case-study applications and case-study architecture, the optimal-energy configurations tend to be the ones using the highest frequency, which characterizes a race-to-idle rather than a pace-to-idle optimal behavior [17]. This can be explained by the large static power observed in the considered architecture, evidenced by the large c_3 parameter in (7) that was fitted in (9). With a large static power, using a pace-to-idle strategy, i.e. the use of frequencies lower than the maximum, is expected to be effective only if the sum of the leakage and the dynamic power parcels is larger than the static power parcel. Based on the fitted power model, this would never happen, i.e. the sum of leakage and dynamic power is always less than the static power,

$$p(0.29f^3 + 0.97f) + 9.18s < 198.59,$$

even if we use the maximum number of cores, $p = 32$ and $s = 2$, and the maximum frequency, $f = 2.2$. Nevertheless, race-to-idle was not always the best strategy because energy scales with the execution time, which in turn scales inversely with the number of active cores and the operating frequency, and because power scales linearly with the number of cores, but exponentially with the frequency.

The optimal number of active cores depends on the parallel scalability of the application. The more scalable the application, the more cores it requires to minimize energy. A scalable application can increasingly exchange the speedup of more cores with lower frequencies in order to spend less energy. This is because of the linear relationship between power and number of cores and the exponential relationship between power and frequency.

B. Proposed Approach versus Ondemand Linux Governor

We have compared the energy consumption of the four case-study applications using the energy-optimal configurations provided by the proposed approach to the energy consumption resulted by use of the Linux default DVFS governor, Ondemand. Since the governor does not choose the number of active cores, we executed each application using 1, 2, 4, 8,$\cdots$, 28, 30, and 32 cores, accounting for the best and the worst cases of energy consumption. Tables II, III, IV and V present these results for Fuidanimate, Raytrace, Swaptions, and Blackscholes, respectively.

TABLE II
FLUIDANIMATE MINIMAL ENERGY

Input	Mean Freq. in GHz (#Cores)	Energy in KJ	Mean Freq. in GHz (#Cores)	Energy in KJ	Freq. in GHz (#Cores)	Energy in KJ	Min. Save(%)	Max. Save(%)
1	1.85 (32)	4.85	2.29 (1)	32.38	2.0 (32)	4.15	16.90	680.31
2	1.88 (32)	9.35	2.29 (1)	66.77	2.0 (32)	7.89	18.60	746.54
3	1.89 (32)	18.82	2.30 (1)	135.00	2.0 (32)	16.98	10.86	695.04
4	2.08 (32)	37.80	2.30 (1)	272.55	2.1 (32)	33.20	13.84	720.82
5	2.00 (32)	76.28	2.30 (1)	546.84	2.2 (32)	66.83	14.14	718.24
	Ondemand Min.		Ondemand Max.		Proposed			

TABLE III
RAYTRACE MINIMAL ENERGY

Input	Mean Freq. in GHz (#Cores)	Energy in KJ	Mean Freq. in GHz (#Cores)	Energy in KJ	Freq. in GHz (#Cores)	Energy in KJ	Save Min.(%)	Save Max.(%)
1	1.30 (4)	38.56	2.29 (1)	60.29	2.2 (6)	37.92	1,70	59.01
2	1.32 (8)	43.59	2.30 (1)	98.11	2.2 (10)	39.93	9.16	145.68
3	1.65 (16)	49.40	2.30 (1)	168.82	2.2 (14)	45.77	7.94	268.84
4	1.62 (32)	55.61	2.30 (1)	299.83	2.2 (22)	52.99	4.94	465.83
5	1.77 (32)	69.33	2.30 (1)	520.34	2.2 (26)	67.28	3.05	673.39
	Ondemand Min.		Ondemand Max.		Proposed			

TABLE IV
SWAPTIONS MINIMAL ENERGY

Input	Mean Freq. in GHz (#Cores)	Energy in KJ	Mean Freq. in GHz (#Cores)	Energy in KJ	Freq. in GHz (#Cores)	Energy in KJ	Min. Save(%)	Max. Save(%)
1	2.15 (32)	5.88	2.29 (1)	80.08	2.2 (32)	5.73	2.57	1297.82
2	2.00 (32)	9,21	2.30 (1)	106.84	2.2 (32)	7,81	17.90	1267.59
3	2.22 (32)	10.37	2.30 (1)	133.41	2.0 (32)	9.90	4.70	1247.58
4	2.02 (32)	14.29	2.30 (1)	160.34	2.0 (32)	12.33	15.95	1200.85
5	2.08 (32)	15.82	2.30 (1)	186.39	1.9 (32)	14.45	9.50	1190.15
	Ondemand Min.		Ondemand Max.		Proposed			

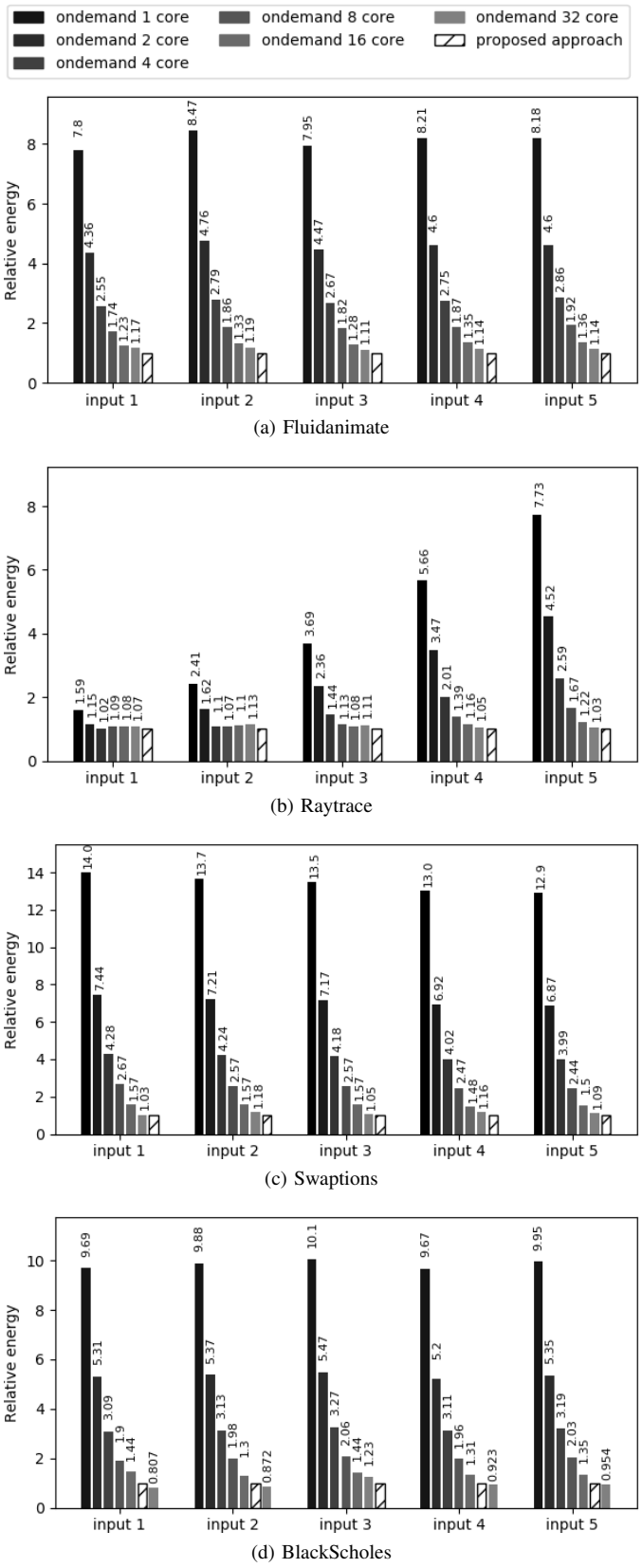
TABLE V
BALCKSCHOELS MINIMAL ENERGY

Input	Mean Freq. in GHz (#Cores)	Energy in KJ	Mean Freq. in GHz (#Cores)	Energy in KJ	Freq. in GHz (#Cores)	Energy in KJ	Min. Save (%)	Max. Save(%)
1	1.57 (32)	1.36	2.27 (1)	16.35	2.2 (30)	1.69	-19.32	869.00
2	2.09 (32)	2.93	2.24 (1)	33.16	1.8 (32)	3.36	-12.78	887.93
3	1.82 (32)	8.08	2.23 (1)	65.97	2.2 (30)	6.55	23.31	907.02
4	2.01 (32)	12.59	2.14 (1)	131.85	2.2 (26)	13.64	-7.66	866.97
5	1.97 (32)	25.29	1.57 (1)	263.89	2.2 (28)	26.52	-4.61	895.23
	Ondemand Min.		Ondemand Max.		Proposed			

In most cases, the proposed approach obtained better results than the best cases of the Ondemand governor. For Blackscholes, the proposed approach was only better than the Ondemand best case for input number 3. On average, the proposed method was 6% better than the best case of the Ondemand governor.

In all cases, the method proposed here outperformed the worst case of the Ondemand governor. On average, the difference in energy consumption was about 790%, being 1298% the maximum difference and 59% the minimum. In general, the energy consumption of the DFVS scheme was larger for smaller numbers of cores. Nonetheless, it was not always the case that the best number of cores for this scheme was the maximum, i.e. 32 cores. Possibly, for architectures with larger number of cores, choosing the exact number the minimizes energy consumption would be less evident.

Fig. 10 shows the behavior of the energy consumption for all tested cases of the Ondemand governor and the proposed approach with values normalized to the energy consumption of the proposed approach.



(a) Fluidanimate

(b) Raytrace

(c) Swaptions

(d) BlackScholes

Fig. 10. Energy consumption of the Ondemand governor for power-of-2 numbers of cores and the proposed approach. The values are relative to the energy of the proposed approach.

V. Related Work

DVFS is the most common technique employed to obtain energy savings on multi-core systems. Thus, the technique has been extensively researched with the aim of providing strategies for selecting the optimal voltage and frequency for a specific application and architecture. In [1] the authors utilized two algorithms for scaling the frequency of the processors: a human-immune system inspired algorithm to monitor the server's power and performance states; and a fuzzy logic based algorithm for changing the server's performance state. [7] introduced a scaling method for determining the system's optimal operation points for the number of threads and DVFS settings.

In [8], an approach that considers instantaneous system activity states was proposed. In this case, the memory and network activity were used to generate a DVFS management setting.

Performance counters have also been used to perform effective DVFS. In [27], the authors used a Continuous Adaptive DVFS based on a performance model of the processor. The model was based on sampling the hardware's performance counters at regular intervals to predict performance/energy workloads. Base on these predictions appropriate voltage, and frequency settings were selected.

In this work, we introduce a power and a performance model to find energy-optimal operating frequency and number of active cores for applications running on specific multi-core platforms. Our approach does not use the DVFS manager to control the processor voltage and frequency settings. This new approach can obtain better results than DVFS strategies as was shown in section IV.

The success obtained from this approach is possibly due to the fact that the use previous knowledge of the application's performance on the target architecture can expose sufficiently relevant information, such as parallel speedups, that is harder to guess in runtime techniques based on DVFS.

The use of an application-agnostic power modeling for the target architecture helps to make the technique portable to other applications. That is, to estimate the energy-optimal frequency and number of active cores for a new application, only a performance characterization is needed.

VI. Conclusion and future work

In this paper, we propose a new approach to optimize the energy efficiency of single-node batch HPC applications. In contrast to existing scheduling algorithms, our technique utilizes the application's runtime profile, and a power model of the compute node to predict the optimal frequency and number of cores to be used. This proven effective in reducing the energy consumption of applications.

Results from four parallel PARSEC applications running on an HPC node with two sixteen-core processors show that the novel approach outperforms the default Linux DVFS scheme on its best case with an average of 6% energy savings. In its worst case, the savings were about 790%, on average.

A weakness of the proposed technique is the need for information about the input size of the application before execution. A possible solution would be to use performance counters, present in all modern HPC processors, to guess the input size based on previously trained data.

Future work will improve the proposed energy model by taking into account more relevant information, such as the percentage of CPU utilization. This can enable the identification of different phases of the target program and thus, it will enable more fine-grained changes of the frequency and, perhaps, the number of active cores, to further improve the results presented here.

References

[1] Ionut Anghel, Tudor Cioara, Ioan Salomie, Georgiana Copil, Daniel Moldovan, and Cristina Pop. Dynamic Frequency Scaling Algorithms for. In *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 485–491, Cluj-Napoca, Romania, 2011. IEEE.

[2] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[3] Robert Basmadjian and Hermann de Meer. Evaluating and modeling power consumption of multi-core processors. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 12:1–12:10, New York, NY, USA, 2012. ACM.

[4] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques - PACT '08*, page 72, 2008.

[5] Len Brown, Robert Moore, David Shaohua Li, Luming Yu, Anil Keshavamurthy, and Venkatesh Pallipadi. ACPI in Linux. *Symposium A Quarterly Journal In Modern Foreign Literatures*, 51:51, 2005.

[6] Pf Butzen and Rp Ribas. Leakage Current in Sub-Micrometer CMOS Gates. *Universidade Federal do Rio Grande do Sul*, pages 1–30, 2007.

[7] Ryan Cochran, Can Hankendi, Ayse Coskun, and Sherief Reda. Identifying the optimal energy-efficient operating points of parallel workloads. In *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, pages 608–615. IEEE, 2011.

[8] Georges Da Costa and Jean Marc Pierson. DVFS governor for HPC: Higher, faster, greener. *Proceedings - 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2015*, pages 533–540, 2015.

[9] Harris Drucker, Chris J C Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in Neural Information Processing Dystems*, 1:155–161, 1997.

[10] Zhihui Du, Rong Ge, Victor W Lee, Richard Vuduc, David A Bader, and Ligang He. Modeling the Power Variability of Core Speed Scaling on Homogeneous Multicore Systems. *Hindawi Scientific Programming*, page 13, 2017.

[11] Armen Dzhagaryan and Aleksandar Milenković. Impact of thread and frequency scaling on performance and energy in modern multicores. *Proceedings of the 2014 ACM Southeast Regional Conference on - ACM SE '14*, pages 1–6, 2014.

[12] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Computer Architecture News*, 35(2):13, 2007.

[13] Bhavishya Goel and Sally A. McKee. A Methodology for Modeling Dynamic and Static Power Consumption for Multicore Processors. *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 273–282, 2016.

[14] R. Gonzalez, B.M. Gordon, and M.A. Horowitz. Supply and threshold voltage scaling for low power CMOS. *IEEE Journal of Solid-State Circuits*, 32(8):1210–1216, 1997.

[15] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. An Energy Efficiency Feature Survey of the Intel Haswell Processor. *Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2015*, pages 896–904, 2015.

[16] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):13, 2012.

[17] David H. K. Kim, Connor Imes, and Henry Hoffmann. Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics. In *Proceedings of the 2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*, CPSNA '15, pages 78–85, Washington, DC, USA, 2015. IEEE Computer Society.

[18] Sparsh Mittal. A survey of techniques for improving energy efficiency in embedded computing systems. *International Journal of Computer Aided Engineering and Technology*, 6(4):440, 2014.

[19] Akihiko Miyoshi, Charles Lefurgy, Eric Van Hensbergen, Ram Rajamony, and Raj Rajkumar. Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling. *Proceedings of the 16th international conference on Supercomputing - ICS '02*, page 35, 2002.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Ilia Pietri and Rizos Sakellariou. Energy-aware workflow scheduling using frequency scaling. *Proceedings of the 43rd ICPPW*, 2014.

[22] Thomas Rauber, Gudula Rünger, Michael Schwind, Haibin Xu, and Simon Melzner. (P1) Energy measurement, modeling, and prediction for processors with frequency scaling. *The Journal of Supercomputing*, 70(3):1451–1476, 2014.

[23] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, and Doron Rajwan. Power-management architecture of the intel microarchitecture code-named Sandy Bridge. *IEEE Micro*, 32(2):20–27, 2012.

[24] Abul Sarwar. Cmos power consumption and cpd calculation. *Proceeding: Design Considerations for Logic Products*, 1997.

[25] Alexander J Smola and Bernhard Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 14(3):199–222, 2004.

[26] Marco Spiga, Mattia Spiga, Andrea Alimonda, Salvatore Carta, Francesco Aymerich, and Andrea Acquaviva. Exploiting memory-boundedness in energy-efficient hard real-time scheduling. *Industrial Embedded Systems - IES'2006*, 2006.

[27] Vasileios Spiliopoulos, Stefanos Kaxiras, and Georgios Keramidas. Green governors: A framework for continuously adaptive DVFS. *2011 International Green Computing Conference and Workshops, IGCC 2011*, 2011.

[28] Nandita Srinivasan, Navamitha S. Prakash, Shalakha D., Sivaranjani D., Swetha Sri Lakshmi G., and B. Bala Tripura Sundari. Power Reduction by Clock Gating Technique. *Procedia Technology*, 21:631–635, 2015.

[29] Matthew Travers. CPU Power Consumption Experiments and Results Analysis of Intel i7-4820K. *μSystems Research Group, School of Electrical and Electronic Engineering, Newcastle University.*, 2015.

[30] S. Usman, S. U. Khan, and S. Khan. A comparative study of voltage/frequency scaling in noc. In *IEEE International Conference on Electro-Information Technology , EIT 2013*, pages 1–5, May 2013.