Performance Prediction of Acoustic Wave Numerical Kernel on Intel Xeon Phi Processor

Víctor Martínez[1], Matheus Serpa[1], Fabrice Dupros[2], Edson L. Padoin[3], and Philippe Navaux[1]

[1] Informatics Institute (INF), Federal University of Rio Grande do Sul (UFRGS), Av. Bento Gonçalves, 9500, Campus do Vale, 91501-970, Porto Alegre, Brazil
{victor.martinez,msserpa,navaux}@inf.ufrgs.br
[2] BRGM, Orléans, France
f.dupros@brgm.fr
[3] Regional University of Northwest of Rio Grande do Sul (UNIJUI), Ijuí, Brazil
padoin@unijui.edu.br

Abstract. Fast and accurate seismic processing workflow is a critical component for oil and gas exploration. In order to understand complex geological structures, the numerical kernels used mainly arise from the discretization of Partial Differential Equations (PDEs) and High Performance Computing methods play a major in seismic imaging. This leads to continuous efforts to adapt the softwares to support the new features of each architecture design and maintain performance level. In this context, predicting the performance on target processors is critical. This is particularly true regarding the high number of parameters to be tuned both at the hardware and the software levels (architectural features, compiler flags, memory policies, multithreading strategies). This paper focuses on the use of Machine Learning to predict the performance of acoustic wave numerical kernel on Intel Xeon Phi many-cores architecture. Low-level hardware counters (e.g. cache-misses and TLB misses) on a limited number of executions are used to build our predictive model. Our results show that performance can be predicted by simulations of hardware counters with high accuracy.

Keywords: Machine Learning, Geophysics Applications, Many-core Systems, Performance Model

1 Introduction

Geophysics exploration remains fundamental to the modern world to keep up with the demand for energetic resources. This endeavor results in expensive drilling costs (100M$-200M$), with less than 50% of accuracy per drill. Thus, Oil and Gas industries rely on software focused on High-Performance Computing (HPC) as an economically viable way to reduce risks.

Acoustic wave propagation approximation is the current backbone for seismic imaging tools. It has been extensively applied for imaging potential oil and gas

reservoirs beneath salt domes for the last five years. Such acoustic propagation engines should be continuously ported to the newest HPC hardware available to maintain competitiveness. At the same time, on the HPC hardware front, the days of faster single core CPUs are over, and the solutions adopted are being replaced by many-core technologies [5, 4].

On one hand, the trend for High Performance Computing (HPC) applications is to pay a higher cost in order to optimize the overall performance. This comes from the complexity of many interdependent factors (non-uniform memory access, vectorization, compiler optimizations, memory policies) at an architectural level that may severely influence the application's behavior. This is particularly true for stencil numerical kernels that are usually memory-bound.

On the other hand, Machine Learning (ML) is a comprehensive methodology for optimization that could be applied to find patterns on a large set of input parameters. Recently, ML algorithms have been used on HPC systems under different situations. In [16] the authors used ML algorithms to select the best job scheduling algorithm on heterogeneous platforms whereas in [2] the authors proposed an ML-based scheme to select the best I/O scheduling algorithm for different applications and input parameters. And recently, in [13] the authors used ML algorithms to predict the performance of stencil computations on multicore architectures.

In this paper, we extend the procedure to build a suitable ML-based performance model for a classical numerical model based on isotropic acoustic wave propagation for many-core architectures. The paper is organized as follows. Section 2 provides the fundamentals of the geophysical application under study. Section 3 describes the methodology of our ML-based approach. Section 4 presents configuration, simulation performance, and model accuracy. Section 5 describes related works. And finally, section 6 concludes this paper.

2 Acoustic wave equation

In this section, we describe the application used as benchmark that simulates the propagation of a single wavelet over time over a three-dimensional domain.

We consider the model formulated by the isotropic acoustic wave propagation (Equation 1) under Dirichlet boundary conditions over a finite 3D rectangular domain, prescribing $p = 0$ to all boundaries, and the isotropic acoustic wave propagation (Equation 2) with variable density where $p(x, y, z, t)$ is the acoustic pressure, $V(x, y, z)$ is the propagation speed and $\rho(x, y, z)$ is the media density.

$$\frac{1}{V^2}.\frac{\partial^2 p}{\partial t^2} = \nabla^2 p \tag{1}$$

$$\frac{1}{V^2}.\frac{\partial^2 p}{\partial t^2} = \nabla^2 p - \frac{\nabla \rho}{\rho}.\nabla p \tag{2}$$

The Laplace operator is discretized by a 12^{th} order finite differences approximation and the time derivatives are approximated by a 2^{nd} order finite

differences operator. *Petrobras*, the leading Brazilian oil company, provides a standalone mini-app of the previously described numerical method. This kernel represents the cornerstone of the classical Reverse Time Migration imaging procedure.

The code was written in standard C and leverage from OpenMP directives for shared-memory parallelism. But Indeed, the parallelization strategy relies on the decomposition of the three-dimensional domain based on OpenMP loop features. There is another implementation for GPU architectures, but it is out of scope of this work and could be analyzed in future works.

Advanced optimizations (loop interchange, vectorization, thread and data mapping) strategies have been implemented to speedup the performance on Intel Xeon Phi processors. For sake of clarity of this paper focused on the impact of machine learning methodology, we will not go into too many details regarding the implementation and interesting readers can refer to [1].

3 Testbed and Machine Learning Methodology

In this section we describe our testbed configurations and the Machine Learning (ML) model. We used Intel Xeon Phi (*Knights Landing*) many-core platform to carry out the experiments. The detailed configurations are shown in Table 1.

Processor	Intel Xeon Phi 7520
Clock(GHz)	1.40
Cores	68
Sockets	1
Threads	272
L2 cache size (MB)	34

Table 1: Description of the Intel Xeon Phi architecture used for our experiments.

Based on this platform, Table 2 details all the configurations available for our optimization categories. As it can be noted, a brute force approach would be unfeasible due to the large number of simulations required (147,968), because some of these executions can take many hours (or days).

Optimization	Parameters	Total configurations
Number of threads	1	272
Scheduling policy	1	2
Chunk size	1	272
Total	**3**	**147,968**

Table 2: Configurations available for our optimization procedure.

3.1 Feature vectors

We emphasize that the selection of the relevant feature vectors is a key ingredient of our method. In our case, we considered a classical ML model with three layers of measurements (input, hidden, and output), which are described below:

1. **Input Layer** is defined by OpenMP implementation features such as the number of threads, the loop scheduling policy (static or dynamic), and the chunk size (which defines how many loop iterations will be assigned to each thread at a time).
2. **Hardware Counters Layer** is built on top of PAPI library to collect the most relevant metrics from the hardware counters total of last level total cache misses (measured by **PAPI_L2_TCM** event), and total of data translation lookaside buffer misses (measured by **PAPI_TLB_DM** event). We decided to use related cache values because stencils are memory bounded problems and the number of available counter is determined by the architecture.
3. **Performance Layer** represents the total elapsed time to solve the geophysical problem.

As we can see, the output depends on several parameters that create a n-dimensional problem and if we try to model it by a regression method it can not be solved by 2D or 3D classical models.

3.2 Machine Learning Model

Our ML model which is based on Support Vector Machines (SVM). This supervised ML approach has been introduced in [6] and then extended to regression problems for n-dimensional problems where support vectors are represented by kernel functions [8].

The main idea of SVMs is to expand hyperplanes through the output vector. It has been employed to classify non-linear problems with non-separable training data by a linear decision (i.e. hardware counters behavior in next section).

Our ML model was built on top of three consecutive layers, where output values of a layer are used as input values of the next layer (Figure 1). The input layer contains the configuration values from the input vector. The hidden

layer contains two SVMs that take values from the input vector to simulate the behavior of hardware counters presented in the previous section. Because hardware counters have very large values it was necessary to perform a dynamic range compression (log transformation) between the hidden layer and the output layer, this is a very common technique used in digital image processing to avoid raw data [11]. Finally, the output layer contains one SVM that takes each simulated value from the hidden layer to obtain the corresponding execution time value.

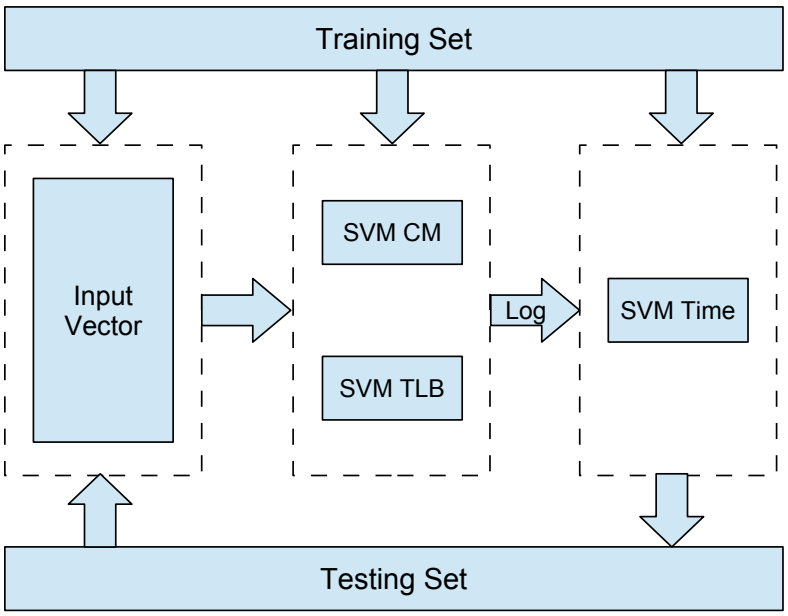

Fig. 1: Flowchart of our machine learning model.

4 Experiments

In this section we present the results of our prediction model. The use-case used as benchmark for our experiments corresponds to a three-dimensional stencil of size 1024 x 256 x 256.

4.1 Preliminary results

Hardware Counters Figures 2 and 3 illustrate how the performance of our kernel is affected by the input variables and their relations with hardware counters. Each point represents one experiment.

For instance, Figure 2 shows that the scheduling policy creates two separated sets when Time values are related with L2 cache misses. The same behavior is

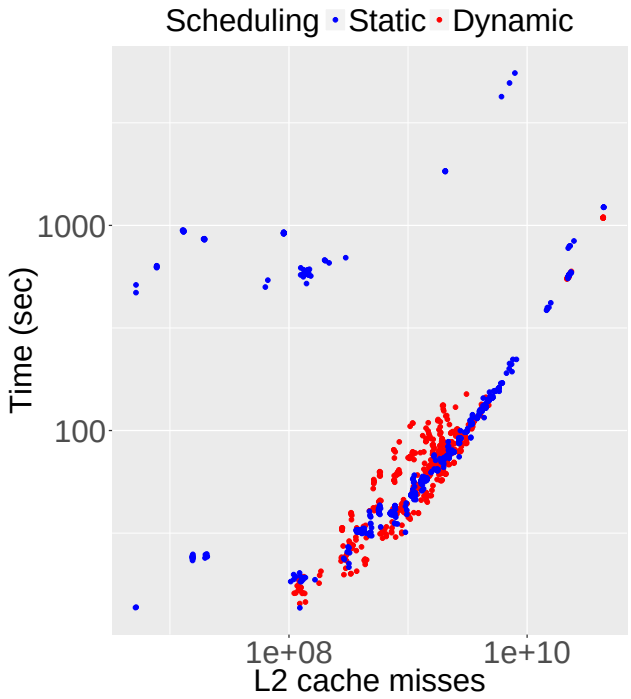

Fig. 2: L2 cache misses when varying the scheduling policy.

observed in Figure 3 when chunk size create several sets if the elapsed time is observed with respect to the amount of TLB data misses.

We can resume this behavior as follows, changing input values affects the performance creating several separated areas in the graphic representation, each color represents a different value for the input value, then these areas could be separated by hyperplanes from SVM. And we note that minor cache misses are related with better performance, as expected. The sparse values are related with number of threads, chunk and scheduler changes in the input vector.

Elapsed time This work is focused on performance prediction, and we use the time as measurement to be analyzed. Firstly, we found that timing is highly affected by the input values, the standard deviation of time measure shows this variability. Some configurations take more time than the walltime available on cluster (3 hours), and we can not use these input values for our experiments.

In Table 3 we present the common statistical values: mean, minimum and maximum values, and standard deviation. We can see that difference between best and worst performance is more than 400x, and standard deviation is more than 2x the mean.

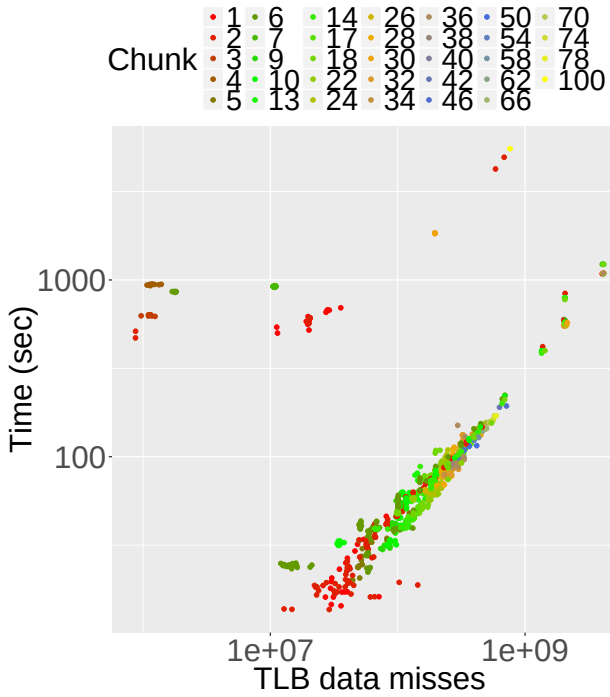

Fig. 3: TLB data misses when varying the chunk size.

	Mean	Min	Max	Standard Deviation (SD)
Time (s)	164.88	13.66	5,530.95	367.72

Table 3: Impact of the input parameters (OpenMP scheduling policies, chunks or number of threads) on the elapsed time.

4.2 Performance prediction

Training and validation We created a training set by randomly selecting a subset from the configuration set presented in Table 2. Then, for each experiment we measured the hardware counters (L2 cache misses, and data translation lookaside buffer misses) and performance values (execution time).

A random testing set was used since all SVMs in both the hidden and the output layers are trained to calculate new execution time values. After that, we measured the accuracy of the model using statistical estimators. Table 4 presents the total number of experiments that were performed to obtain the training and validation sets.

Set	Number of Experiments
Training	808
Testing	203
Total	**1,011**

Table 4: Number of experiments in the training and the testing sets.

Results and discussions We use the performance values from validation set and predicted values from our model to evaluate the model with two statistical estimators: root mean square error (RMSE) and the coefficient of determination (R-square). The former represents the standard deviation of the differences between predicted values and real values whereas the latter represents how close the regression approximates the real data (R-square ranges from zero to one, equal to zero indicates regression with no one prediction and equal to one indicates a perfect fit of data prediction).

As it can be noted in Table 5, the RMSE value confirms that deviation of time value is high, and the approximation of R-square is close to 94%, then we get a highly accurate regression.

Estimators	**Value**
RMSE	154.04
R-square	0.94

Table 5: Accuracy of our predictive modelling based on ML. We provide two statistical estimators.

5 Related Works

Recent architectures, including accelerators and coprocessors, proved to be well suited for geophysics, magneto-hydrodynamics and flow simulations, outperforming the general purpose processors in efficiency. And some works are developed to optimize and predict the performance of these kind of applications.

Thus, In [12], the authors automatically generate a highly optimized stencil code for multiple target architectures. In [14], the authors suggest using runtime reconfiguration, and a performance model, to reduce resource consumption. In [3], the authors 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 [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. In the same way, in [9], the authors worked on target cache reuse methodologies across single and multiple stencil sweeps, examining cache-aware algorithms as well as cache-oblivious techniques in order to build robust implementations

Other works investigated the accuracy of regression models and ML algorithms in different contexts. In [15] the authors compared ML algorithms for characterizing the shared L2 cache behavior of programs on multi-core processors. The results showed that regression models trained on a given L2 cache architecture are reasonably transferable to other L2 cache architectures. In [17] the authors proposed a dynamic scheduling policy based on a regression model that is capable of responding to the changing behaviors of threads during execution.

Finally, in [10] the authors applied ML techniques to explore stencil configurations (code transformations, compiler flags, architectural features and optimization parameters). Their approach is able to select a suitable configuration that gives the best execution time and energy consumption. In [7], the authors improved performance of stencil computations by using a model based on cache misses. In [13], the authors proposed a ML model to predict performance of stencil computations on multicore architectures.

6 Conclusion

In this paper, we introduced a predictive performance modeling strategy for geophysical numerical kernel on many-core architectures. We showed that performance of the simplified acoustic wave equation can be predicted with a high accuracy (95%) based on hardware counters.

Moreover, the results from this work extend the based-ML strategy described in [13] for performance optimization of the elastodynamics equation on multicore architectures. Our model is not restricted to Xeon Phi platforms and can also be implemented into architectures with the available hardware counters to measure the cache-related behavior, we use the PAPI library but we believe that it don't limit our model and could be implemented with another library. Our future works can be summarized in the following lines.

Firstly, we expect to extend our methodology in order to capture complex behaviors (vectorization capabilities, data mapping). Secondly, we intend to design a model based on unsupervised ML algorithms to further improve our results. Finally, we believe that a general model can be integrated into an auto-tuning framework to find the best performance configuration for a given stencil kernel.

7 Acknowledgments

For computer time, this research partly used the resources of Colfax Research. This work has been granted by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (*CAPES*), Conselho Nacional de Desenvolvimento Científico e Tecnológico (*CNPq*), Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (*FAPERGS*). The authors thank Jairo Panetta from Aeronautics Institute of Technology (*ITA*) and *PETROBRAS* oil company for providing the acoustic wave numerical kernel code. It was also supported by *Intel Corporation* under the Modern Code Project. Research has received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the *HPC4E* Project, grant agreement n° 689772. We also thank to *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.: Chapter 23 - 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. Boito, F.Z., Kassick, R.V., Navaux, P.O.A., Denneulin, Y.: Automatic I/O scheduling algorithm selection for parallel file systems. Concurrency and Computation: Practice and Experience 28(8), 2457–2472 (2016), cpe.3606
3. 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 (2015)
4. Clapp, R.G.: Seismic Processing and the Computer Revolution(s). In: SEG Technical Program Expanded Abstracts 2015. pp. 4832–4837 (2015)
5. Clapp, R.G., Fu, H., Lindtjorn, O.: Selecting the right hardware for reverse time migration. The Leading Edge 29(1), 48–58 (2010)
6. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
7. de la Cruz, R., Araya-Polo, M.: Modeling Stencil Computations on Modern HPC Architectures, pp. 149–171. Springer International Publishing, Cham (2015)
8. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. In: Advances in Neural Information Processing Systems 9. pp. 155–161. MIT Press (1997)
9. Dupros, F., Boulahya, F., Aochi, H., Thierry, P.: Communication-avoiding seismic numerical kernels on multicore processors. In: International Conference on High Performance Computing and Communications (HPCC). pp. 330–335 (Aug 2015)
10. Ganapathi, A., Datta, K., Fox, A., Patterson, D.: A case for machine learning to optimize multicore performance. In: Proceedings of the First USENIX Conference on Hot Topics in Parallelism. pp. 1–1. HotPar'09, USENIX Association, Berkeley, CA, USA (2009)
11. Gonzalez, R.C., Woods, R.E.: Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2006)

12. Kukreja, N., Louboutin, M., Vieira, F., Luporini, F., Lange, M., Gorman, G.: Devito: Automated fast finite difference computation. In: Procs. of the 6th Intl. Workshop on Domain-Spec. Lang. and High-Level Frameworks for HPC. pp. 11–19. WOLFHPC '16, IEEE Press (2016)
13. Martínez, V., Dupros, F., Castro, M., Navaux, P.: Performance improvement of stencil computations for multi-core architectures based on machine learning. Procedia Computer Science 108, 305 – 314 (2017), international Conference on Computational Science, {ICCS} 2017, 12-14 June 2017, Zurich, Switzerland
14. Niu, X., Jin, Q., Luk, W., Weston, S.: A Self-Aware Tuning and Self-Aware Evaluation Method for Finite-Difference Applications in Reconfigurable Systems. ACM Trans. on Reconf. Technology and Systems 7(2) (2014)
15. Rai, J.K., Negi, A., Wankar, R., Nayak, K.D.: On prediction accuracy of machine learning algorithms for characterizing shared l2 cache behavior of programs on multicore processors. In: International Conference on Computational Intelligence, Communication Systems and Networks (CICSYN). pp. 213–219 (July 2009)
16. Vladuic, D., Cernivec, A., Slivnik, B.: Improving job scheduling in grid environments with use of simple machine learning methods. In: International Conference on Information Technology: New Generations. pp. 177–182 (April 2009)
17. Weng, L., Liu, C., Gaudiot, J.L.: Scheduling optimization in multicore multithreaded microprocessors through dynamic modeling. In: Proceedings of the ACM International Conference on Computing Frontiers. pp. 5:1–5:10. CF '13, ACM, New York, NY, USA (2013)