Performance Prediction of Geophysics Numerical Kernels on Accelerator Architectures

Víctor Martínez, Matheus S. Serpa,
Philippe O. A. Navaux
Informatics Institute
UFRGS, Brazil
email: {victor.martinez, msserpa, navaux}@inf.ufrgs.br

Edson L. Padoin
Department of Exact Sciences
and Engineering
UNIJUI, Brazil
email: padoin@unijui.edu.br

Jairo Panetta
Computer Science Division
ITA, Brazil
email: jairo.panetta@gmail.com

Abstract—**In order to develop geophysics tools for exploration of energetic resources, numerical models are proposed to understand complex geological structures. They are solved from the discretization of Partial Differential Equations by the Finite Differences Method. This method creates a pattern that solves each point in a 3D domain, and it replicates the same calculation to compute all the data domain. Because of the quantity of calculations, solving the numerical kernels requires High Performance Computing. However, the complexity of current architectures may reduce the efficiency. In some cases, applications tuning have been used to improve the performance. In this context, predicting the performance from input parameters is a critical problem. 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 work focuses on the use of Machine Learning to predict the performance of geophysics numerical kernels on many-core architectures. Measures of hardware counters on a limited number of executions are used to build our predictive model. We have considered three different kernels (7-point Jacobi, seismic and acoustic wave propagation) to demonstrate the effectiveness of our approach. Results show that the performance can be predicted with high accuracy.**

Keywords*–*Machine Learning; Geophysics Applications; Many-core Systems; Performance Model

I. INTRODUCTION

Geophysics modeling remains fundamental to keep up with the demand for energetic resources. Thus, Oil and Gas industries rely on High Performance Computing (HPC) software as an economically viable way to reduce risks. Wave propagation algorithms are routinely used both in the oil and gas industry and in strong motion analysis in seismology. The finite-differences numerical method used for this problem also lies at the heart of a significant fraction of numerical solvers in other fields. In terms of computational efficiency, one of the main difficulties is to deal with the disadvantageous ratio between the limited point-wise computations and the intensive memory access required, leading to a memory-bound situation [1].

The objective of HPC applications is to optimize the performance. This comes from the increasing of complexity for many interdependent factors: vectorization, compiler optimizations, non-uniform memory access and several levels of memory, etc. Although a large body of literature on the optimization of this class of applications is available, predicting the performance on current architectures remains a challenge and it requires to search in a large set of input configurations [2]. 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 parameters. Recently, in [3] the authors present a methodology based on ML algorithms and simulation to obtain dynamic scheduling policies, whereas in [4] the authors proposed an ML-based scheme to tune the storage system and increase the I/O throughput.

This research was developed in the context of High Perfomance Computing for Enery Project (HPC4E). In this paper, we describe the procedure to predict the performance of stencil applications by a ML-based model, on accelerators architectures. The paper is organized as follows: Section II provides the fundamentals of geophysics models under study; Section III describes the methodology of our ML-based approach; Section IV presents the experiments, the performance prediction, and the model accuracy; Section V describes the related work. Finally, Section VI concludes this paper.

II. NUMERICAL KERNELS

In this section, we present the geophisycs numerical models. Due to its simplicity, the Finite-Differences Method (FDM) is widely used to design the geophysics models, when discretizing Partial Differential Equations (PDE). From the numerical analysis point of view, the FDM computational procedure consists in using the neighboring points in the north-south, east-west and forward-backward directions to evaluate the current grid point in the case of a three-dimensional Cartesian Grid. The algorithm then moves to the next point applying the same computation until the entire spatial grid has been traversed. The number of points used in each direction depends on the order of the approximation. This procedure is called stencil-based computation. The stencil sweep can be expressed as a triply nested loop presented in Figure 1.

```
 1: for each timestep do
 2:     compute in parallel
 3:     for each block in X-direction do
 4:         for each block in Y-direction do
 5:             for each block in Z-direction do
 6:                 compute stencil(3D tile)
 7:             end for
 8:         end for
 9:     end for
10: end for
```

Figure 1. Pseudocode for stencil algorithms.

A. 7-point Jacobi

The stencil model of 7-point Jacobi is given by the explicit 3D heat equation described in [5] and presented in (1):

$$\begin{aligned}B_{i,j,k} =&\alpha A_{i,j,k}\\ &+\beta(A_{i-1,j,k}+A_{i,j-1,k}+A_{i,j,k-1}\\ &\quad+A_{i+1,j,k}+A_{i,j+1,k}+A_{i,j,k+1})\end{aligned} \tag{1}$$

This stencil performs a single Jacobi (out-of-place) iteration. Thus, reads and writes occur in two distinct arrays (A, B), where each subscript represent the 3D index into array A or B. For each grid point, this stencil will execute 8 floating point operations [6]. Figure 2 illustrates the size of Jacobi stencil.

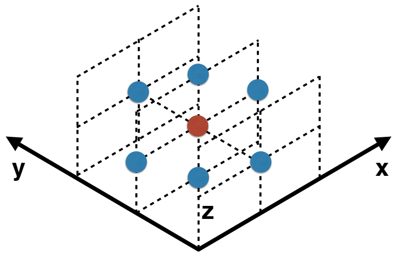


Figure 2. Seven-point Jacobi stencil [7].

B. Seismic Wave Propagation

The seismic waves radiated from an earthquake are often simulated under the assumption of an elastic medium although the waves attenuate due to some anelasticity. This numerical kernel corresponds to the discretization of the elastodynamics equation and it is of great importance both for seismic hazard assessment, as well as for the oil and gas industry. In our case, we consider a standard fourth order in space and second order in time approximation. This algorithm corresponds to the evaluation of six stress components (three in the diagonal direction and three off-diagonal) and three velocity components. A detailed description of the numerical modeling of seismic waves on multi-core platforms is presented in [8].

$$\begin{aligned}\sigma^{n+1}_{xx}(i,j,k) =&\sigma^{n}_{xx}(i,j,k)\\ &+A_1[a_1(V^n_x(i+\frac{1}{2},j,k)-V^n_x(i-\frac{1}{2},j,k))\\ &\quad+a_2(V^n_y(i,j+\frac{1}{2},k)-V^n_y(i,j-\frac{1}{2},k))\\ &\quad+a_3(V^n_z(i,j,k+\frac{1}{2})-V^n_z(i,j,k-\frac{1}{2}))]\\ &+B_1[a_1(V^n_x(i+\frac{3}{2},j,k)-V^n_x(i-\frac{3}{2},j,k))\\ &\quad+a_2(V^n_y(i,j+\frac{3}{2},k)-V^n_y(i,j-\frac{3}{2},k))\\ &\quad+a_3(V^n_z(i,j,k+\frac{3}{2})-V^n_z(i,j,k-\frac{3}{2}))]\end{aligned} \tag{2}$$

Equation (2) provides a synthetic view of the computation of one of the diagonal components, where i, j, k represent a tensor field component in Cartesian coordinates (x, y, z), and V and σ represent the velocity and stress fields, respectively. Figure 3 illustrates the size of the seismic wave propagation stencil applied to calculate velocity and stress components.

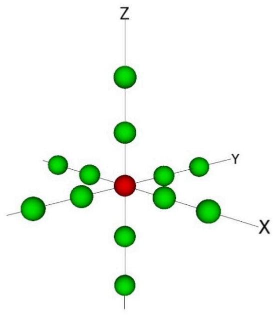


Figure 3. Seismic wave propagation stencil [9].

C. Acoustic Wave Propagation

The 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. We consider the model formulated by the isotropic acoustic wave propagation under Dirichlet boundary conditions over a finite 3D rectangular domain, prescribing to all boundaries, and the isotropic acoustic wave propagation. Numerical method solves (3) and is detailed in [10].

$$\begin{aligned}C_{i,jk} =&a_0C_{i,j,k}\\ &+a_1(C_{i-1,j,k}+C_{i+1,j,k}+C_{i,j-1,k}\\ &\quad+C_{i,j+1,k}+C_{i,j,k-1}+C_{i,j,k+1})\\ &+a_2(C_{i-2,j,k}+C_{i+2,j,k}+C_{i,j-2,k}\\ &\quad+C_{i,j+2,k},+C_{i,j,k-2}+C_{i,j,k+2})\\ &+a_3(C_{i-3,j,k}+C_{i+3,j,k}+C_{i,j-3,k}\\ &\quad+C_{i,j+3,k}+C_{i,j,k-3}+C_{i,j,k+3})\end{aligned} \tag{3}$$

Propagation speed depends on variable density, the acoustic pressure, and the media density. Figure 4 illustrates the size of the acoustic wave propagation stencil.

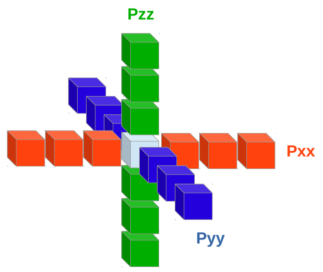


Figure 4. Acoustic wave propagation stencil [10].

Petrobras, the leading Brazilian oil company, provides a standalone mini-app of the previously described numerical method. 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.

III. Testbed and Prediction Model Methodology

In this section, we describe our testbed, the runtime configurations and introduce the ML model.

A. Testbed

We used the Intel Xeon Phi many-core platform to carry out the experiments (code-name *Knights Landing, or KNL*). The detailed configurations are shown in Table I.

TABLE I. DESCRIPTION OF THE INTEL XEON PHI (KNL) ARCHITECTURE.

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

The KNL brings in new memory technology, a high bandwidth on package memory called Multi-Channel DRAM (MCDRAM). MCDRAM is a high bandwidth, low capacity (up to 16GB) memory. MCDRAM has three memory modes and can be configured as *cache* mode to work as a third level cache; in *flat* mode, both the MCDRAM memory and the DDR memory act as regular memory and are mapped into the same system address space as a distinct NUMA node (allocatable memory); and the *hybrid* mode, the MCDRAM is partitioned such that either a half or a quarter of the MCDRAM is used as cache, and the rest is used as flat memory [11] [12].

The scientific applications implemented on the KNL architectures are developed in openMP, and as we can explained in Section II a stencil is built by three nested for loops. Then, scheduling influences the application performance. OpenMP scheduling can be defined in runtime by the `OMP_SCHEDULE` environment variable. This variable is a string formatted by two parameters: scheduling policy and chunk size. Four different loop scheduling policies can be provided to OpenMP: *Static* divide the loop into equal-sized chunks; *Dynamic* uses the internal work queue to give a chunk-sized block of loop iterations to each thread; *Guided* is similar to dynamic, but the chunk size starts off large and decreases to better handle load imbalance between iterations; and *auto*, when the decision regarding scheduling is delegated to the compiler. The optional parameter (chunk), when specified, must be a positive integer and defines how many loop iterations will be assigned to each thread at a time [13].

Based on this platform, Table II details all the available configurations for the optimization categories. As it can be noted, a brute force approach would be unfeasible due to the large number of simulations required (443,904), because some of these executions can take many hours (or days).

TABLE II. AVAILABLE CONFIGURATIONS FOR THE OPTIMIZATION PROCEDURE.

Optimization	Parameters	Total configurations
Number of threads	1	272
Chunk size	1	272
Scheduling policy	1	3
Memory mode	1	2
Total	**3**	**443,904**

B. Feature vectors

We emphasize that selection of the relevant feature vectors is a key ingredient of our method, which are described below:

1) The **Runtime vector** is defined by OpenMP implementation features such as the number of threads (`OMP_NUM_THREADS` environment varible), the loop scheduling policy (static, dynamic or guided), the chunk size, and the memory mode (cache or flat).
2) The **Hardware Counters vector** is built on top of PAPI library, to collect the most relevant metrics from the hardware counters: total of L2 cache misses (measured by `PAPI_L2_TCM` event), and total of cycles (measured by `PAPI_TOT_CYC` event) [14]. We decided to use related cache values because stencils are memory bounded problems, and the number of available counters is determined by the architecture.
3) The **Performance vector** represents the total elapsed time to solve the geophysical problem.

As we can see, the performance 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.

C. Prediction Model

The proposed ML model is based on Support Vector Machines (SVM), which is a supervised ML approach introduced in [15] and extended to regression problems where support vectors are represented by kernel functions [16]. 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 surface (i.e., hardware counters behavior in previous Section III-B).

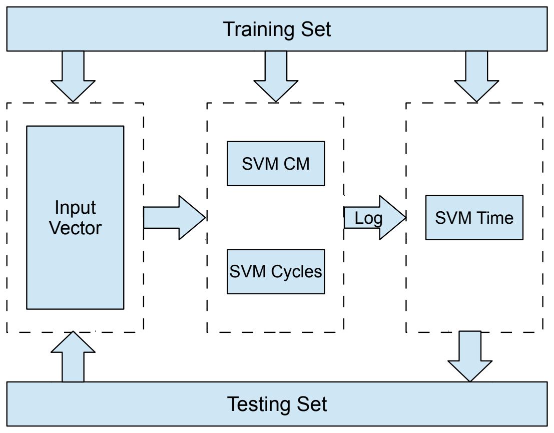


Figure 5. Flowchart of the proposed ML-based model.

In our case, we considered a classical ML model with three layers of measurements (input, hidden, and output) related with vectors explained in Section III-B, and extends the work done by [17], (Figure 5). The input layer contains the values from the runtime vector. The hidden layer contains two SVMs that take values from the input vector to simulate the behavior of hardware counters. Finally, the output layer contains one SVM that takes each simulated value from the hidden layer to obtain the predicted execution time. Our model was implemented by *e1071* R package.

IV. Experimental results

In this section, we present the results of our prediction model. We used a three-dimensional grid of size 512x512x512, and 190 time iterations, as benchmark for our experiments.

A. Training and validation sets

We created a training set by randomly selecting a subset from the configuration parameters presented in Table II. Then, for each experiment we measured the hardware counters (L2 cache misses, and total cycles) and the performance (execution time). Because hardware counters have very large values, it was necessary to perform a dynamic range compression based on a logarithmic transformation, between the hidden layer and the output layer, as shown in Figure 5.

A random testing set was used since all SVMs in both the hidden and the output layers are trained to predict the execution time values. After that, we measured the accuracy of the model using statistical estimators. Table III presents the total number of experiments that were performed to obtain the training and validation sets. It is remarkable that total number of experiments used for testing and validation, for each stencil, is lower than 1% of total configurations described in Table II.

TABLE III. NUMBER OF EXPERIMENTS

	Training	Testing	**Total**
Jacobi	334	3007	**3341**
Seismic	346	3122	**3468**
Acoustic	335	3021	**3356**

B. Hardware Counters Behavior

Figure 6 illustrates how the hardware counters measurements are affected by the input variables. Each point represents one experiment when varying the input parameters described in Section III-B; the X domain represents the L2 cache misses, the Y axis represents the total cycles, and the color represents the different values for the input parameter. For instance, Figure 6(a) represents the scheduling policy (green is dynamic, red is guided, and blue is static) for the Jacobi stencil. We can see how the static scheduling tends to be separated from other values. Figure 6(b) shows the number of threads used to solve the seismic stencil. We also can see how each value create one easily separated area. Figure 6(c) also presents how chunk size trends to creates separated areas for the acoustic stencil. We can resume this behavior as follows: the input values changing affects the application performance and creates several separated areas in the graphic representation. As each color represents a different value for the input value these areas could be separated by hyperplanes from a SVM.

C. Prediction Results

We evaluate the model with two statistical estimators: the Root Mean Square Error (RMSE) and the Coefficient of Determination (R-squared); they represent the standard deviation of prediction errors and how close the regression approximates the predicted and actual data, respectively. As it can be noted in Table IV, the regression model is highly accurate. For the 7-point Jacobi implementation, the model presented an accuracy of up to 97.39%. For the seismic wave implementation, on the other hand, the model presented an accuracy of up to 85.62% and the acoustic wave propagation model has 93.2% accuracy. These results are similar to prediction of multi-core architectures presented in [18] and [17].

TABLE IV. STATISTICAL ESTIMATORS OF OUR PREDICTION MODEL

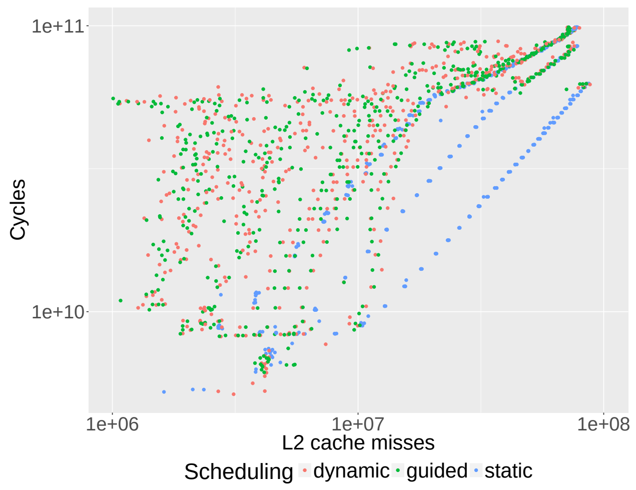
	RMSE	R-squared
Jacobi	2.9894	0.9739
Seismic	21.0183	0.8562
Acoustic	40.0748	0.9322

V. Related Work

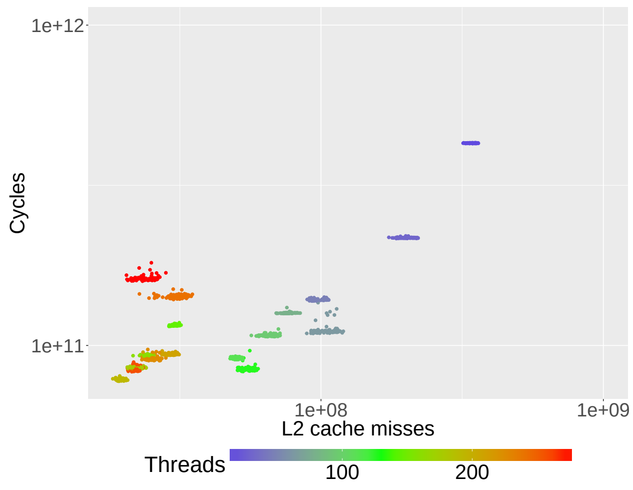
Recent HPC architectures, including accelerators and co-processors, proved to be well suited for geophysics simulations, outperforming the general purpose processors in efficiency. And some works are developed to optimize and to predict the performance. Because numerical kernels are a memory-bound problem, common optimizations are cache-based. In [19], the authors worked on target cache reuse methodologies, and demonstrated that memory system organization reduce the efficacy of traditional cache-blocking optimizations for stencil computation on multiple architectures (multi-core and accelerators). In the same way, in [8], 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.

At high level, adjusting the runtime parameters can optimize the performance of stencil applications. In [20], 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 [2], the authors analyzed the performance of task scheduling algorithms. They concluded that different scheduling policies combined with different task sizes may affect the efficiency and performance of seismic wave kernels. In [21], 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.

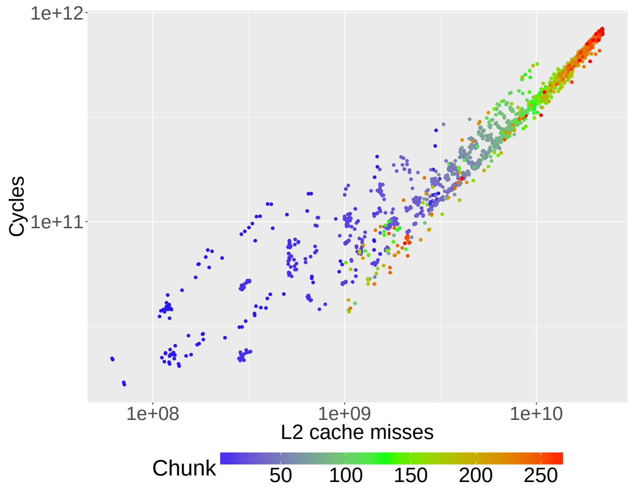
Methods for automatic code generation are used at runtime level. In [22], the authors presented a stencil auto-tuning framework for multi-core architectures that converts a sequential stencil expression into tuned parallel implementations. In [23], the authors present an automatic source-to-source transformations framework. Thus, in [24], the authors suggest using runtime reconfiguration, and a performance model, to reduce resource consumption. Analogously, in [25], the authors automatically generate a highly optimized stencil code for



(a) Jacobi stencil



(b) Seismic stencil



(c) Acoustic stencil

Figure 6. Hardware counters behavior.

multiple target architectures. Overall, the main problem of these works is that the search domain can be very large and searching for the best configuration would take too much time.

Other works investigated the accuracy of regression models and ML algorithms in different contexts. In [26] 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 [27] 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 [28] 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 [18], the authors improved performance of stencil computations by using a model based on cache misses. In [17], the authors proposed a ML model to predict and to optimize the performance of stencil computations on multi-core architectures.

VI. Conclusion

We presented a ML-based model to predict the performance of stencil computations on many-core architectures. We showed that the performance of three well-known stencil kernels (7-point Jacobi, seismic and acoustic wave propagation) can be predicted with a high accuracy by using the hardware counters.

Results from this work extend the ML-based strategy described in [17] for performance optimization of the elastodynamics equation on multi-core architectures. Our model is not restricted to Xeon Phi platforms and can also be implemented into architectures with the available hardware counters to obtain the cache-related measures. We used the PAPI library but we think that it could be implemented with another library (i.e., perf, pin, etc.). The future work can be summarized in two statements: First, we believe that our model can be integrated into an auto-tuning framework to find the best performance configuration for a given stencil kernel; second, we intend to design a model based on unsupervised ML algorithms to further improve our results.

Acknowledgments

This work has been granted by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)*, the *Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)*, the *Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS)*. Research has received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the *HPC4E Project*, grant agreement n.º 689772. It was also supported by Intel under the Modern Code project, and *PETROBRAS* oil company provided the acoustic wave numerical kernel code, under Ref. 2016/00133-9. The authors also thank to Bureau de Recherches Géologiques et Minières *(BRGM)* for providing the 7-point Jacobi and the seismic wave propagation codes.

References

[1] F. Dupros, H. Do, and H. Aochi, “On scalability issues of the elastodynamics equations on multicore platforms,” in Proceedings of the International Conference on Computational Science, ICCS 2013, Barcelona, Spain, 5-7 June, 2013, 2013, pp. 1226–1234.

[2] V. Martínez et al., “Towards seismic wave modeling on heterogeneous many-core architectures using task-based runtime system,” in Computer Architecture and High Performance Computing (SBAC-PAD), 2015 27th International Symposium on, Oct 2015, pp. 1–8.

[3] D. Carastan-Santos and R. Y. de Camargo, “Obtaining dynamic scheduling policies with simulation and machine learning,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC ’17. New York, NY, USA: ACM, 2017, pp. 32:1–32:13. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126955

[4] Y. Li et al., “Capes: Unsupervised storage performance tuning using neural network-based deep reinforcement learning,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC ’17. New York, NY, USA: ACM, 2017, pp. 42:1–42:14. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126951

[5] K. Datta et al., “Optimization and performance modeling of stencil computations on modern microprocessors,” SIAM Rev., vol. 51, no. 1, Feb. 2009, pp. 129–159. [Online]. Available: http://dx.doi.org/10.1137/070693199

[6] ——, “Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures,” in Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, ser. SC ’08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 4:1–4:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1413370.1413375

[7] A. Nguyen et al., “3.5-d blocking optimization for stencil computations on modern cpus and gpus,” in 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, Nov 2010, pp. 1–13.

[8] F. Dupros, F. Boulahya, H. Aochi, and P. Thierry, “Communication-avoiding seismic numerical kernels on multicore processors,” in International Conference on High Performance Computing and Communications (HPCC), Aug 2015, pp. 330–335.

[9] D. Michéa and D. Komatitsch, “Accelerating a three-dimensional finite-difference wave propagation code using gpu graphics cards,” Geophysical Journal International, vol. 182, no. 1, 2010, pp. 389–402. [Online]. Available: http://dx.doi.org/10.1111/j.1365-246X.2010.04616.x

[10] R. F. Vilela, “Profiling of the finite differences wave equation resolution problem in xeon phi coprocessor,” Master’s thesis, Federal University of Rio de Janeiro, COPPE, http://www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2017/2016033170–53/file, 2017.

[11] S. H., “Intel® Xeon Phi^{TM} Processor 7200 Family Memory Management Optimizations,” https://software.intel.com/en-us/articles/intel-xeon-phi-processor-7200-family-memory-management-optimizations, 2016, [retrieved: april, 2018].

[12] M. P., “An Intro to MCDRAM High Bandwidth Memory) on Knights Landing,” https://software.intel.com/en-us/blogs/2016/01/20/an-intro-to-mcdram-high-bandwidth-memory-on-knights-landing, 2016, [retrieved: april, 2018].

[13] Intel, “OpenMP* Loop Scheduling,” https://software.intel.com/en-us/articles/openmp-loop-scheduling, 2014, [retrieved: april, 2018].

[14] P. Mucci and ICL, “PAPI Library,” http://icl.cs.utk.edu/papi/, 2014, [retrieved: april, 2018].

[15] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, 1995, pp. 273–297. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[16] H. Drucker et al., “Support vector regression machines,” in Advances in Neural Information Processing Systems 9. MIT Press, 1997, pp. 155–161.

[17] V. Martínez, F. Dupros, M. Castro, and P. Navaux, “Performance improvement of stencil computations for multi-core architectures based on machine learning,” Procedia Computer Science, vol. 108, 2017, pp. 305 – 314, international Conference on Computational Science, {ICCS} 2017, 12-14 June 2017, Zurich, Switzerland.

[18] R. de la Cruz and M. Araya-Polo, Modeling Stencil Computations on Modern HPC Architectures. Cham: Springer International Publishing, 2015, pp. 149–171.

[19] K. Datta et al., Scientific Computing with Multicore and Accelerators. Taylor & Francis Group: CRC Press, 2010, ch. Auto-Tuning Stencil Computations on Multicore and Accelerators.

[20] C. Andreolli et al., “Chapter 23 - Characterization and Optimization Methodology Applied to Stencil Computations,” in High Performance Parallelism Pearls, J. Reinders and J. Jeffers, Eds. Boston: Morgan Kaufmann, 2015, pp. 377 – 396.

[21] D. Caballero et al., “Optimizing Fully Anisotropic Elastic Propagation on Intel Xeon Phi Coprocessors,” in 2nd EAGE Workshop on HPC for Upstream, 2015, p. 4.

[22] S. Kamil et al., “An auto-tuning framework for parallel multicore stencil computations,” in IEEE International Symposium on Parallel Distributed Processing (IPDPS), April 2010, pp. 1–12.

[23] J. Cronsioe, B. Videau, and V. Marangozova-Martin, “BOAST: Bringing Optimization through Automatic Source-to-Source Transformations,” in International Symposium on Embedded Multicore/Manycore System-on-Chip (MCSoC). Tokyo, Japan: IEEE Computer Society, 2013, pp. 129 – 134.

[24] X. Niu, Q. Jin, W. Luk, and S. Weston, “A Self-Aware Tuning and Self-Aware Evaluation Method for Finite-Difference Applications in Reconfigurable Systems,” ACM Trans. on Reconf. Technology and Systems, vol. 7, no. 2, 2014, pp. 15:1–15:19.

[25] N. Kukreja et al., “Devito: Automated fast finite difference computation,” in Procs. of the 6th Intl. Workshop on Domain-Spec. Lang. and High-Level Frameworks for HPC, ser. WOLFHPC ’16. IEEE Press, 2016, pp. 11–19.

[26] J. K. Rai, A. Negi, R. Wankar, and K. D. Nayak, “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), July 2009, pp. 213–219.

[27] L. Weng, C. Liu, and J.-L. Gaudiot, “Scheduling optimization in multicore multithreaded microprocessors through dynamic modeling,” in Proceedings of the ACM International Conference on Computing Frontiers, ser. CF ’13. New York, NY, USA: ACM, 2013, pp. 5:1–5:10.

[28] A. Ganapathi, K. Datta, A. Fox, and D. Patterson, “A case for machine learning to optimize multicore performance,” in Proceedings of the First USENIX Conference on Hot Topics in Parallelism, ser. HotPar’09. Berkeley, CA, USA: USENIX Association, 2009, pp. 1–1.