Adaptive Sampling for Performance Characterization of Application Kernels

Pablo de Oliveira Castro *[1,2], Eric Petit[1], Asma Farjallah[1], and William Jalby[1,2]

[1]PRISM, Université de Versailles Saint-Quentin-en-Yvelines, France
[2]Exascale Computing Research, France

This is the pre-print version of the following article: Adaptive Sampling for Performance Characterization of Application Kernels, P. de Oliveira Castro, E. Petit, A. Farjallah, W. Jalby, Concurrency and Computation: Practice and Experience. (2013) doi: 10.1002/cpe.3097 which has been published in final form at `http://onlinelibrary.wiley.com/doi/10.1002/cpe.3097`.

Abstract

Characterizing performance is essential to optimize programs and architectures. The open source Adaptive Sampling Kit (ASK) measures the performance trade-off in large design spaces. Exhaustively sampling all sets of parameters is computationally intractable. Therefore, ASK concentrates exploration in the most irregular regions of the design space through multiple adaptive sampling strategies. The paper presents the ASK architecture and a set of adaptive sampling strategies, including a new approach: Hierarchical Variance Sampling. ASK's usage is demonstrated on three performance characterization problems: memory stride accesses, jacobian stencil code and an industrial seismic application using 3D stencils. ASK builds accurate models of performance with a small number of measures. It considerably reduces the cost of performance exploration. For instance, the jacobian stencil code design space, which has more than 31×10^8 combinations of parameters, is accurately predicted using only 1 500 combinations.

1 Introduction

Understanding architecture behavior is crucial to tune applications and develop more efficient hardware. An accurate performance model captures all interactions among the system's elements such as: multiple cores with an out-of-order

*pablo.oliveira@prism.uvsq.fr

dispatch or complex memory hierarchies. Building analytical models is increasingly difficult with the complexity growth of current architectures.

An alternative approach considers the architecture as a black box and empirically measures its performance response. The downside of the approach is the exploration time needed to sample the design space. As the number of considered factors grows – cache levels, problem size, number of threads, thread mappings, and access patterns – the size of the design space explodes and exhaustively measuring each combination of factors becomes infeasible.

To mitigate the problem, we must sample only a limited number of combinations, which we will call *points* in the following. From the sampled points, we build a surrogate performance model to study, predict, and improve architecture and application performance within the design space. Samples should be chosen with care to faithfully represent the whole design space. A good sampling strategy should capture the performance accurately with the minimal number of samples.

The two fundamental elements of a sampling pipeline are the sampling strategy and the surrogate model.

1. The sampling strategy decides what combinations of the design space should be explored.

2. The surrogate model extrapolates from the sampled combinations a prediction on the full design space.

The Adaptive Sampling Kit (ASK) gathers many state-of-the-art sampling strategies and surrogate models in a common framework simplifying the process. The user provides ASK with a description of the design space parameters. Then, ASK automatically selects the points that should be sampled, measure their response, and returns a model that predicts the performance of any given set of parameters. Therefore, with a small set of measurements ASK can produce an accurate performance map of the full design space.

Choosing an adequate sampling strategy is not simple: for best results one must carefully consider the interaction between the sampling strategy and the surrogate model [1]. Many implementations of sampling strategies are available, but they all use different configurations and interfaces. Therefore, building and refining sampling strategies is difficult. ASK addresses this problem by providing a common interface to these different strategies and models. Designed around a modular architecture, ASK facilitates building complex sampling pipelines. ASK also provides reporting and model validation modules to assess the quality of the sampling and ease the experimental setup exploration for performance characterization. The paper's main contributions are:

- a common toolbox, ASK, gathering state-of-the-art sampling strategies, and simple to integrate with existing measurement tools,

- a new sampling strategy, Hierarchical Variance Sampling (HVS), which mitigates sampling bias by using confidence bounds,

- an evaluation of the framework, and of HVS, on three representative performance characterization experiments.

Section 2 discusses related work. Section 3 succinctly presents ASK's architecture and usage. Section 4 explains the HVS strategy. Section 5 details the interaction between HVS sampling strategy and the Generalized Boosted Model (GBM). Section 6 evaluates ASK on two performance studies: memory stride accesses and 2D stencils. Finally, section 7 presents a performance case study based on ASK of a seismic simulator kernel extracted from an industrial code.

This paper is an extended version of the work presented at the 18^{th} Euro-Par international conference [2].

2 Related Work

There are two kinds of sampling strategies: space filling designs and adaptive sampling. Space filling designs select a fixed number of samples with sensible statistical properties such as uniformly covering the space or avoiding clusters. For instance, Latin Hyper Cube designs [3] are built by dividing each dimension into equal sized intervals. Points are selected so the projection of the design on any dimension contains exactly one sample per interval. Maximin designs [4] maximize the minimum distance between any pair of samples; therefore spreading the samples over the entire experimental space. Finally, low discrepancy sequences [5] choose samples with low discrepancy: given an arbitrary region of the design space, the number of samples inside this region is almost proportional to the region's size. By construction, the sequences uniformly distribute points in space. These designs are often better than Random Sampling, which may clump samples together [6].

Space filling designs choose all points in one single draw before starting the experiment. Instead, adaptive sampling strategies iteratively adjust the sampling grid to the complexity of the design space. By observing already measured samples, they identify the most irregular regions of the design space. Further samples are drawn preferentially from the irregular regions, which are harder to explore.

The definition of irregular regions varies depending on the sampling strategy. Variance-reduction strategies focus the sampling in regions with high variance. The rationale is: irregular regions require more measurements to be accurately modeled. Query-by-Committee strategies build a committee of models trained with different parameters and compare the committee's predictions on all the candidate samples. Selected samples are the ones where the committee's models disagree the most. Adaptive Multiple Additive Regression Trees (AMART) [7] is a recent Query-by-Committee approach based on Generalized Boosted Models (GBM) [8], it selects non-clustered samples with maximal disagreement. Another recent approach by Gramacy et al. [9] combines the Tree Gaussian Process (TGP) [10] model with adaptive sampling strategies [11]. For an extensive review of adaptive sampling strategies please refer to Settles [12].

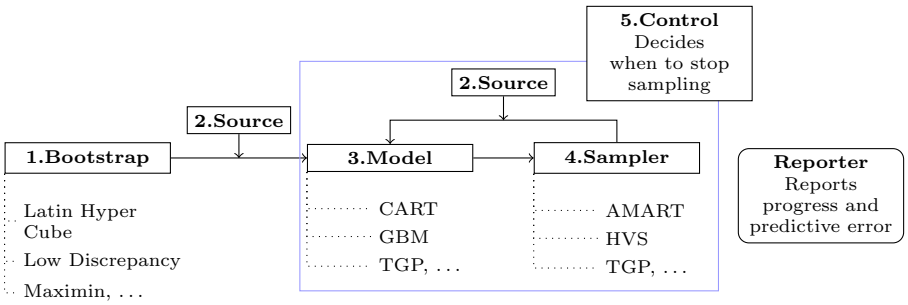


Figure 1: ASK pipeline

The Surrogate Modeling Toolbox (SUMO) [13] offers a Matlab toolbox building surrogate models for computer experiments. SUMO's execution flow is similar to ASK's: both allow configuring the model and sampling strategy to fully automate an experiment plan. SUMO focuses mainly on building and controlling surrogate models, offering a large set of models. It contains algorithms for optimizing model parameters, validating the models, and helping users choose a model. A recent approach, LOLA-Voronoi, is included, which finds trade-offs between uniformly exploring the space and concentrating on nonlinear regions of the space [14]. SUMO is open source but restricted to academic use and depends on the proprietary Matlab toolbox.

ASK specifically targets adaptive sampling for performance characterization, unlike SUMO. It includes recent state-of-the-art approaches that were successfully applied to computer experiments [9] and performance characterization [7]. Simpson et al. [1] show one must consider different trade-offs when choosing a sampling strategy: affinity with the surrogate model or studied response, accuracy, or cost of predicting new samples. Therefore, ASK comes with a large set of approaches to cover different sampling scenarios including Latin Hyper Cube designs, Maximin designs, Low discrepancy designs, AMART, and TGP. Additionally, ASK includes a new approach, Hierarchical Variance Sampling (HVS).

3 ASK Architecture

ASK's flexibility and extensibility come from its modular architecture. When running an experiment, ASK follows the pipeline presented in Figure 1:

1. A *bootstrap* module selects an initial batch of points. ASK provides standard bootstrap modules for the space filling designs described in Section 2: Latin Hyper Cube, Low Discrepancy, Maximin, and Random.

2. A *source* module, usually provided by the user, receives a list of requested points. The source module computes the actual measurements for the requested factors and returns the response.

3. A *model* module builds a surrogate model for the experiment on the sampled points. Currently ASK provides CART [15], GBM [8, 16], and TGP [9] models.

4. A *sampler* module iteratively selects a new set of points to measure. Some sampler modules are simple and do not depend on the surrogate model. For instance, the `random` sampler selects a random combination of factors and the `latin` sampler iteratively augments an initial Latin Hyper Cube design. Other sampler modules are more complex and base their decisions on the surrogate model.

5. A *control* module decides when the sampling process ends. ASK includes two basic strategies: stopping when it has sampled a predefined amount of points or stopping when the accuracy improvement stays under a given threshold for a number of iterations.

From the user perspective, setting up an ASK experiment is a three-step process. First, the range and type of each factor is described by writing an experiment configuration file in the JavaScript Object Notation (JSON) format. ASK accepts real, integer, or categorical factors. Then, users write a *source* wrapper around their measuring setup. The interface is straightforward: the wrapper receives a combination of factors to measure and returns their response. Finally, users choose which bootstrap, model, sampler, control, and reporter modules to execute. Module configuration is also done through the configuration file. ASK provides fallback default values if parameters are omitted from the configuration. An excerpt of a configuration with two factors and the hierarchical sampler module follows:

```
"factors": [{"name": "image-size",
             "type": "integer",
             "range": {"min": 0, "max": 600}},
            {"name": "stencil-size",
             "type": "categorical",
             "values": ["small", "medium", "large"]}],
"modules": {"sampler": {"executable": "sampler/HVS",
                        "params": {"nsamples":50}}}
```

Editing the configuration file quickly replaces any part of the ASK experiment pipeline with a different module. For example, by replacing `sampler/HVS` with `sampler/latin` the user replays the same experiment with the same parameters but using a Latin Hyper Cube sampler instead of Hierarchical Variance Sampling. All the modules have clearly defined interfaces and are organized to follow the *separation of concerns* principle [17]. This organization allows the user to quickly integrate custom made modules to the ASK pipeline.

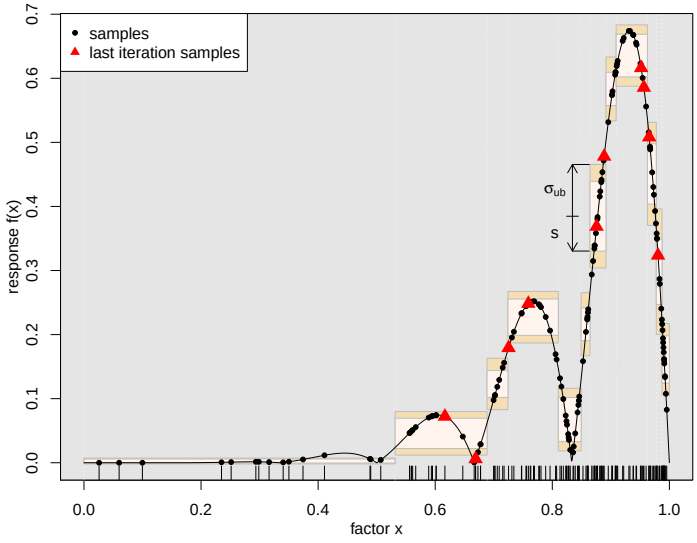


Figure 2: HVS on a synthetic 1D benchmark after fifteen drawings of ten samples each. The true response, $f(x) = x^5|sin(6.\pi.x)|$, is the solid line. CART partitions the factor dimension into intervals, represented by the boxes horizontal extension. For each interval, the estimated standard deviation, s, is in a light color and the upper bound of the standard deviation, σ_{ub}, is dark. HVS selects more samples in the irregular regions.

4 Hierarchical Variance Sampling

Many adaptive learning strategies are susceptible to bias because the sampler makes incorrect decisions based on an incomplete view of the design space. For instance, the sampler may ignore a region although it contains big variations because previous samplings missed the variations.

To mitigate the problem, ASK includes the new Hierarchical Variance Sampling, HVS. The key idea of HVS is to reduce the bias using confidence intervals that correct the variance estimation. HVS partitions the exploration space into regions and measures the variance of each region. A statistical correction depending on the number of samples is applied to obtain an upper bound of the variance. Further samples are then selected proportionally to the upper bound and size of each region. By using a confidence upper bound on the variance, the sampler is less greedy in its exploration and is less likely to overlook interesting regions. In others words, the sampler is less likely to ignore a region until the number of sampled points is enough to confidently decide the region has low variance.

HVS is similar to Dasgupta et al. [18] proposing a hierarchical approach for classification tasks using confidence bounds to reduce the sampling bias. The Dasgupta et al. approach is only applicable to classification tasks with a binary or discrete response. Because ASK models performance, which is continuous response, we could not directly use the Dasgupta et al. approach, and instead developed HVS.

To divide the design space into regions, HVS uses the Classification and Regression Trees (CART) partition algorithm [15] with the Analysis of Variance (ANOVA) splitting criteria [19] and prunes the tree to optimize cross validation error. At each step, the space is divided into two regions so the sum of the regions variance is smaller than the variance of the whole space. The result of a CART partitioning is shown in Figure 2 where each box depicts a region.

After partitioning, HVS samples the most problematic regions and ignores the ones with low variance. The sampler only knows the empiric variance s^2 that depends on previous sampling decisions; to reduce bias HVS derives an upper bound of the true variance σ^2. Assuming a close to normal region's distribution, HVS computes an upper bound of the true variance σ^2 satisfying $\sigma^2 < \frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}} = \sigma^2_{ub}$ with a $1-\alpha$ confidence[1]. To reduce the bias HVS uses the corrected upper bound accounting for the number of samples drawn. The variance upper bound computation assumes an underlying normal distribution per region, which may not be the case. We believe HVS could be improved by using a more accurate variance upper bound.

For each region, Figure 2 plots the estimated standard deviation s, light colored, and upper-bound σ_{ub}, dark colored. As shown in Figure 2, samples are selected proportionally to the variance upper bound multiplied by the size of the region. New samples, marked as triangles, are chosen inside the largest boxes. HVS selects few samples in the $[0, 0.5]$ region, which has a flat profile.

If the goal of the sampling is to reduce the absolute error of the model, then the HVS strategy is adequate because it concentrates on high-variance regions. On the other hand, if the goal is to reduce the relative error[2] of the model, it is better to concentrate on regions with high relative variance, $\frac{s^2}{\bar{x}^2}$. HVSrelative is an alternate version of HVS using relative variance with an appropriate confidence interval [20]. Section 6 evaluates the HVS and HVSrelative sampling in two performance studies. Section 7 uses HVS to explore the performance of a seismic imaging application.

5 GBM Model and HVS Sampler Interactions

Fitting a model is a trade-off between the model complexity and the accuracy. When tuning the model, it is important to take into account the design space and the sampling strategies. In this section we study the interactions between the GBM model, used extensively in our experiments in sections 6 and 7, and

[1] $1-\alpha = 0.9$ confidence bound is default. χ is the Chi distribution.

[2] sometimes called percentage error

the HVS sampler.

Traditional regression trees approaches, such as CART [15], partition the design space into regions and fit a constant or linear model to each region. The partitioning is represented by a tree where each leaf corresponds to one of the regions in the partitioning. GBM improves over CART by combining the predictive power of many individual trees. GBM models the response as a function $f(\vec{x})$ where $\vec{x}$ is the input vector of factors. The function f is defined as a linear combination of regression trees. Each regression tree models the interactions among a subset of factors.

GBM improves the accuracy of f by minimizing a loss function, such as $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i))^2$, which computes for every point i, the mean squared error between the measured response y_i and the prediction $f(\vec{x}_i)$. This formulation implies that every $\vec{x}_i$ has the same effect on the final model. Therefore, if the space is not uniformly explored, regions with few samples will be underrepresented in the loss function.

Because adaptive sampling strategies sample unequally the space, they affect GBM's loss function computation. Consider, for instance, the synthetic 1D response defined by the following piece-wise function:

$$\forall x \in [1:100] \quad s(x) = \begin{cases} 1 & \text{if } 0 < x \leq 20 \quad (\text{region } R_1) \\ x - 20 & \text{if } 20 < x \leq 40 \quad (\text{region } R_2) \\ 20 & \text{if } 40 < x \leq 100 \quad (\text{region } R_3) \end{cases}$$

An adaptive strategy such as HVS will sample few points from the R_1 and R_3 regions because they are flat. Figure 3 presents a sampling experiment where the region R_2 is sampled ten times more than the other two regions. Since GBM's loss function is dominated by R_2, the model is trained specifically for the middle region at the expense of accuracy in R_1 and R_3. In figure 3 the model, labelled *unweighted*, clearly shows a lack of accuracy in the first and third regions.

To avoid this effect, ASK takes advantage of the GBM weights feature [8], which replaces the above loss function by $\sum_{i=1}^{n} w_i.(y_i - f(\vec{x}_i))^2$ where w_i is the weight of point i. To remove the sampling bias introduced by the adaptive strategy we set all weights in region r to $w_r = \frac{s_r}{n_r}$, where s_r is the proportion of the total design space occupied by region r and n_r is the proportion of samples in region r. In the unweighted formula, the importance of a region is proportional to the number of samples it received, n_r. With the weighted formula, this value is multiplied by the region weight, $n_r.w_r = n_r.\frac{s_r}{n_r} = s_r$. This ensures that the importance of a region is proportional only to its size but is independent of the number of samples it received.

Figure 3 compares on the synthetic example the true response to the weighted and unweighted GBM models using 5000 trees. As expected, the weighted GBM model fits better the first and third regions.

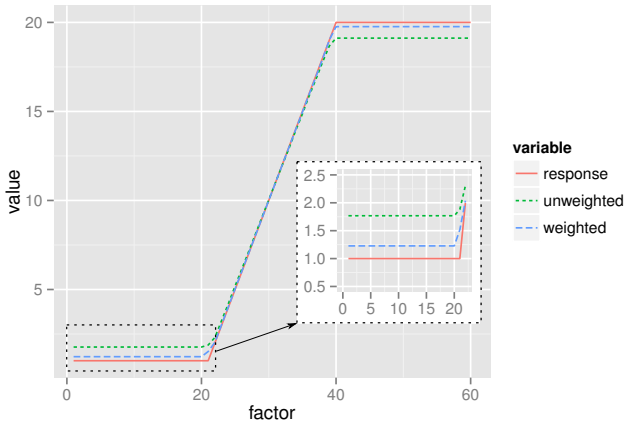


Figure 3: Prediction of the same synthetic response by weighted and unweighted 5000 trees GBM models

6 Experimental Study

In this section, we present two performance characterization experiments conducted using ASK.

The first experiment examines the performance of a synthetic microbenchmark called ai_aik. Ai_aik explores the impact of strided accesses to a same array inside a loop. The design space comprises 400 000 different combinations of two factors: loop trip N and k-stride. The design space is large and variable enough to challenge sampling strategies. Nonetheless, it is small enough to be measured exhaustively providing an exact baseline to rate the effectiveness of the sampling strategies.

The second experiment validates the strategies on a large experimental space: performance of 2D cross-shaped stencils of varying size on a parallel SMP. A wide range of scientific applications use stencils: for instance, Jacobi computation [21, 22] uses a 2×2 stencil and high-order finite-difference calculations [23] use a 6×6 stencil.

The design space is composed of five parameters: the $N \times M$ size of the matrix, the $X \times Y$ size of the stencil, and T the number of concurrent threads used. The design space size has more than 7×10^8 points in a 8-core system and more than 31×10^8 points in a 32-core system.

Since an exhaustive measurement is computationally infeasible, the prediction accuracy is evaluated by computing the error of each strategy on a test set of 25 600 points independently measured. The test set contains 12 800 points chosen randomly and 12 800 points distributed in a regular grid configuration over the design space. Measuring the test set takes more than twelve hours of computation on a 32-core Nehalem machine.

All studied sampling strategies use random seeds, which can slightly change

the predictive error achieved by different ASK runs. Therefore, the median error, among nine different runs, is reported when comparing strategies.

Experiments ran with six of the sampling strategies included in ASK: AMART, HVS, HVSrelative, Latin Hyper Cube, TGP, and Random. All the benchmarks were compiled with ICC 12.0.0 version. The strategies were called with the following set of default parameters on both experiments:

Samples All the strategies sampled in batches of fifty points per iteration.

Bootstrapping All the strategies were bootstrapped with samples from the same Latin Hyper Cube design, except Random, which was bootstrapped with a batch of random points.

Surrogate Model Tuning accurately the model parameters is important to get accurate performance predictions. The TGP strategy uses the tgpllm model with its default parameters and the adaptive sampling setup described in section 3.6 of [10]. The other strategies used GBM [8] with the following parameters:

- `distribution`: the loss function, described in section 5, was selected to minimize squared error.
- `shrinkage`: this parameters sets the gradient descent step, which controls the contribution of each new tree added to the model. It was set to 0.01 as recommended in [8].
- `ntrees`: the maximum number of trees used in the model was set to 3 000.
- `depth`: the maximum depth of each tree was set to 8. This parameters controls the maximum number of interactions between factors. For example, trees of depth one with only one cut, allow no interactions between factors.
- `minobsinnode`: the minimum number of samples required to sprout a tree leaf was set to 10, the default value.

Manually tuning model parameters is tedious and requires precise knowledge of the model's internals. Strategies to automatically explore and optimize model parameters have been proposed [24, 13]. We plan to extend ASK with such strategies, to facilitate usage by non expert users.

AMART ran with a committee size of twenty as recommended by Li et al. [7].

TGP used the Active Learning-Cohn [11] sampling strategy.

HVS, HVSrelative used a confidence bound of $1 - \alpha = 0.9$.

Section 6.1 validates ASK on an exhaustive stride access experiment. Section 6.2 validates ASK on a large design space that cannot be explored exhaustively: multi-core performance of cross-shaped stencils.

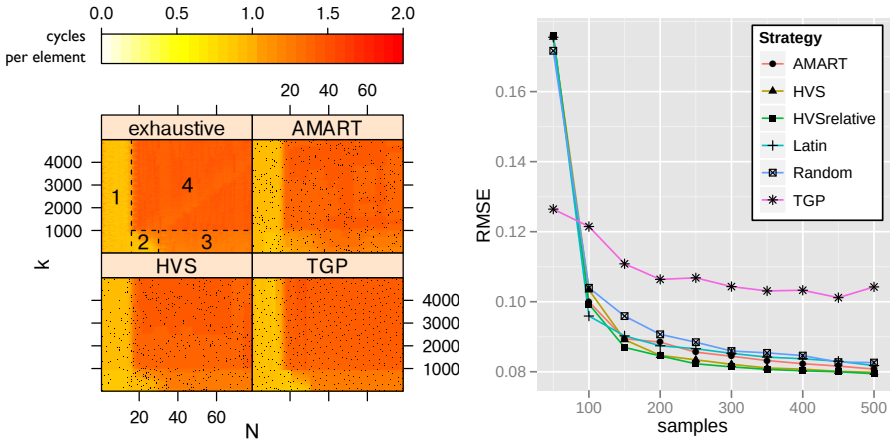


Figure 4: Stride experiments: (*Left*) exhaustive level plot shows the true response of the studied kernel in cycles per element. AMART, HVS, and TGP respectively show the predicted response of each strategy. Black dots indicate the position of the sampled points. (*Right*) RMSE is plotted for each strategy and number of samples. The median among nine runs of each strategy was taken to remove random seed effects.

6.1 Stride Microbenchmark: ai_aik

This section studies the stride memory accesses of the following ai_aik kernel:

```
for(i=0;i<N*256;i++) {
  res[i]=a[i]+a[i+2*k];
}
```

The baseline is an exhaustive measurement of cycle per element performance on a Xeon L5609 quad-core 1.87GHz with 8GB of RAM. The measurements revealed the four zones on the left side of Figure 4:

1. `zone 1`: the kernel is fastest because both i and $i+2k$ accesses fit in L1.
2. `zone 2`: is a transition zone between `zone 1` and `zone 3`, under the diagonal, the accessed array elements still fit in L1.
3. `zone 3`: accesses do not fit fully in L1 anymore but the performance is still acceptable because the accesses $i+2k$ prefetch the data needed for the i accesses in future iterations.
4. `zone 4`: performance is the worst because accesses do not fully fit in L1 and the $2k$ distance is too wide to reuse the data prefetched by the $i+2k$ accesses of the previous iterations.

The preceding exhaustive analysis required 400 000 measurements of the ai_aik kernel. ASK ran the same experiment with different sampling strategies

stopping at only 500 samples. Each strategy's accuracy was determined by comparing its predictions to the exhaustive baseline.

Comparing the predicted and exhaustive responses in Figure 4 shows that TGP and HVS capture the four zones in detail while AMART is less accurate in Zone 2. The diagonal effect in Zone 2 introduces high variance. Therefore, HVS concentrates sampling on Zone 2, capturing it accurately.

The Root Mean Square Error (RMSE) is the standard metric in the literature to evaluate a model's accuracy. Figure 4, right side, shows the RMSE of each strategy. The strategies, except TGP, are comparable in terms of convergence speed and reached accuracy. TGP scores a poor RMSE performance compared to the other strategies. GBM surrogate model seems to be a better fit than TGP for this experiment.

In the experiment, ASK's 500 point sampling successfully captures the performance features of the design space. ASK uses eight hundred times less samples than the exhaustive analysis (400 000 samples), while preserving accuracy.

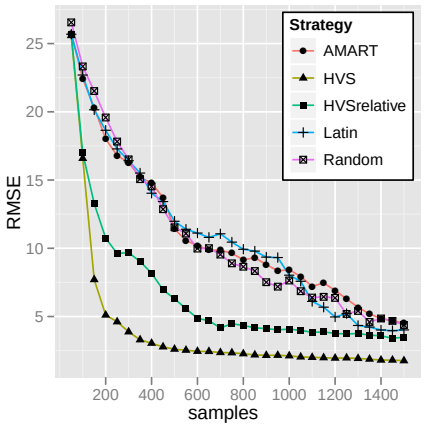
6.2 2D Stencil Characterization

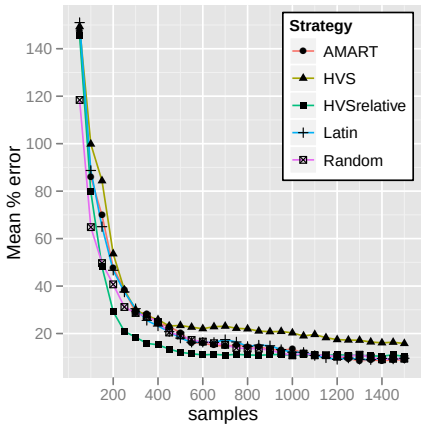
The stencil code characterization is infeasible with exhaustive exploration, but is possible using ASK's adaptive sampling strategies. In the studied stencil code, Figure 5a, five factors can be tuned – X and $Y \in \{1, 2, 4, 8, 16\}$ the horizontal and vertical sizes of the stencil, $N \in [64, 2048]$ the number of rows of the matrix, $M \in [64, 2048]$ the number of columns of the matrix, and $T \in [1, 32]$ the number of threads. The stencil was studied on two Nehalem architectures: an 8-core dual-socket Xeon E5620 at 2.40GHz with 24GB of RAM and a 32-core four-socket Xeon X7550 at 2.00GHz with 128GB of RAM. The OpenMP mapping policy was set to `Scatter`. The error was evaluated on an independent test set of 25 600 points.

TGP was not used during the second experiment because it does not handle categorical variables [10]. The computation time needed to select samples with HVS, HVSrelative, and Latin is negligible compared to the time required to measure a batch of samples. AMART is a Query-by-Committee strategy, which generates a prediction on all the candidate points, for twenty different models. In the stencil experiment the number of candidate points is in the order of billions, computing a prediction on all of them is not possible. Therefore, as suggested by Gramacy [10], ASK's AMART implementation reduces the number of candidate points to one thousand with a Latin Hyper Cube presampling.

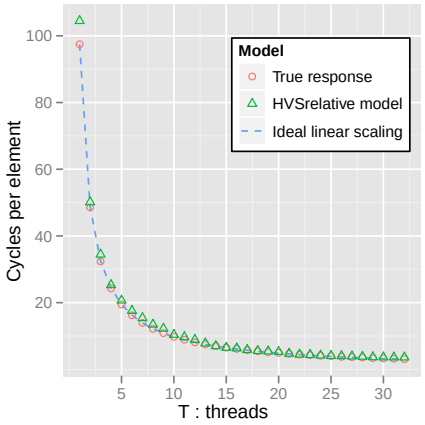
The sampling strategies' accuracy is measured in terms of RMSE and mean relative error, in Figure 5b. Here only the 32-core results are examined because the sampling strategies' accuracy was similar for both the 8 and 32-core architectures. For RMSE, HVS outperforms all other strategies both in quality of the final model, 1.76 RMSE, and speed of convergence. For mean percentage error, AMART achieves the best final result, 8.89%, followed closely by Latin, Random, and HVSrelative. HVSrelative converges faster than the others.

Overall, HVSrelative is the best compromise between RMSE and percentage error because it achieves low final errors and converges quickly to an accurate





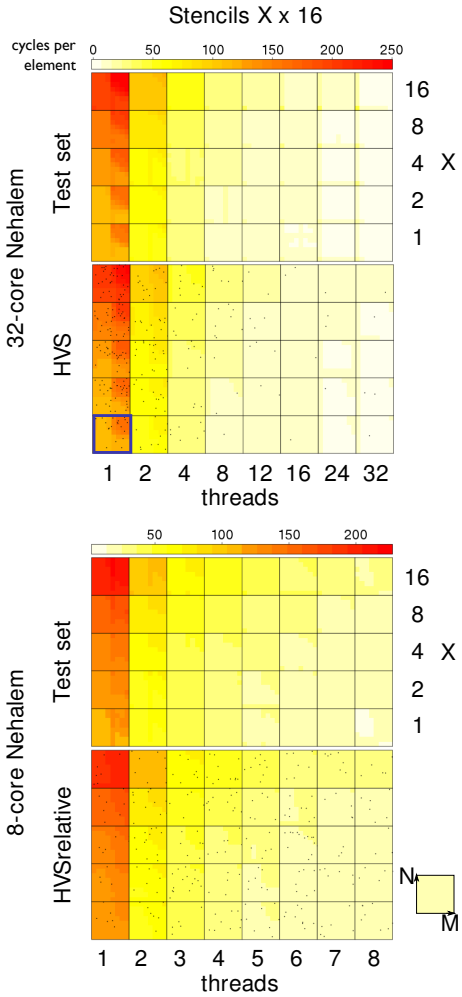
(b) Error curves for the exploration on 32 cores. The median among nine runs of each strategy was taken to remove random seed effects.



(d) Scalability for the 8×8 stencil on a $1\,000 \times 1\,000$ matrix.

```
#pragma omp parallel for
for(i=Y; i<N-Y; i++)
  for(j=X; j<M-X; j++) {
    for(k=j-X; k<=j+X; k++)
      out[i][j] += in[i][k];
    for(l=i-Y; l<=i+Y; l++)
      out[i][j] += in[l][j];
  }
```

(a) Stencil code evaluated



(c) Stencil $X \times 16$ predicted performance vs. test set performance in cycles per element.

Figure 5: Stencil experiments

model. Using only eight hundred samples, HVSrelative predicts the test set with a mean absolute error of 1.51 cycles and a mean relative error of 10.97%.

Figure 5c shows the performance prediction for HVS and HVSrelative on the $X \times 16$ stencils. Each square represents a unique (X, Y, T) configuration. Inside each square the performance is plotted depending on the matrix size $N \times M$. For example the outlined left-bottom square plots the performance predicted by HVS for a 1×16 stencil with one thread.

The kernel is slowest for high Y stencils whose column order accesses stress the cache. In comparison, X stencil's size impact on performance is negligible. Performance degrades for large matrices, as shown from the darker top-right corners of each square, probably because the matrices exceed the L2 cache capacity. Therefore, choosing an adequate blocking factor should improve the performance. On ASK HVS' 32-core model the mean speed-up obtained by varying the matrix size is 1.65. Using a small matrix with a high number of threads is detrimental because the cost of synchronization predominates.

On both 8 and 32-core targets the stencil code scales linearly up to, respectively, 8 and 32 threads. As an example, the scalability of the application was studied on the 8×8 stencil on a $1\,000 \times 1\,000$ matrix. Figure 5d shows the performance per number of threads predicted by the HVSrelative strategy. The prediction follows the measured true response. The matrix sizes explored fit into the socket's 18Mb L3 cache, additionnaly the 2D stencil benefits from data reuse, which explains the strong scaling.

Measuring the whole design space would take centuries whereas ASK adaptive sampling took less than an hour of experiment time with a 1.76 RMSE. ASK is both efficient when dealing with small design spaces, as in ai_aik, and large design space, as in the stencil codes experiment.

7 Seismic Modeling Performance Study

In Sections 6, synthetic application kernels were used to validate ASK sampling strategies. This section presents a performance study conducted using ASK on a finite-difference time-domain (FDTD) kernel extracted from an industrial seismic imaging code.

Section 7.1 presents the seismic application and the design parameters considered for FDTD. Section 7.2 details the steps to set up ASK and evaluates the sampling accuracy. Finally, section 7.3 presents the performance characteristics of the FDTD kernel captured by the performance model generated by ASK.

7.1 FDTD kernel presentation and design parameters

Industry relies on depth imaging applications to model the subsurface and thus locate precisely oil and gas deposits. Reverse Time Migration (RTM) [25] is a commonly used application as it represents a good compromise between the quality of the image rendered and the time to solution. The first step in RTM consists of building synthetic seismograms based on a model of the crust. This

process is called the seismic modeling. The second step is a retro-propagation of the waves recorded during exploration campaigns. These waves are the reflection of the vibrations sent in the earth. We adjust the seismic model using the collected data in order to get the actual image of the subsurface. The accuracy of the image depends on the numerical methods implemented in the application. For geophysical imaging, one can solve numerically the wave equation using three distinct approaches: spectral method, strong formulation, and weak formulation [26].

The spectral method is based on a dual representation of the space and is generally used on layered and simple geophysical structures. On the other hand, finite-element methods based on the weak formulation are used on complex structures of the earth but demand more computational resources. Here we consider the strong formulation of the partial differential equation which offers a good trade-off between the image accuracy and the computational cost.

Our study focuses on finite-difference time-domain (FDTD) implementations in isotropic medium. The equation governing the wave propagation in such medium is,

$$\frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} = \Delta U,$$

where U designates the wave field and c the velocity, constant in our case. In a finite-difference approach, each value of the field is updated using a combination of its neighboring values. Figure 7 shows the discretization of the wave equation. The number of neighbors in each direction defines the order in space of the stencil. Our implementation uses the centered explicit stencil shown in figure 6. In the following, p denotes the half order of the stencil. For example, in figure 6, $2p$ is equal to 8 which corresponds to the order of the Taylor expansion in each dimension of the space.

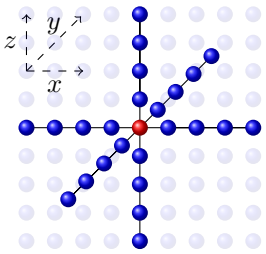


Figure 6: 3D centered stencil of order 8 ($p = 4$).

$$U^{t+1}_{i,j,k} = 2U^{t}_{i,j,k} - U^{t-1}_{i,j,k} + c^2\Delta t^2 \left\{ \frac{1}{\Delta x^2}\sum_{m=-p}^{p} b_m U^{t}_{i+m,j,k} + \frac{1}{\Delta y^2}\sum_{m=-p}^{p} b_m U^{t}_{i,j+m,k} + \frac{1}{\Delta z^2}\sum_{m=-p}^{p} b_m U^{t}_{i,j,k+m} \right\}$$

Figure 7: Discretization of the pressure field U using a stencil of order $2p$ in space and 2 in time. Δt and $(\Delta x, \Delta y, \Delta z)$ designate discretization steps in time and space respectively. b_m are constant coefficients weighting neighboring values that are the result of the Taylor expansion.

Performance optimization of stencil computations is widely studied by the community [21]. Common optimizations target data locality through spatial and temporal cache blocking, loop tiling and padding. Most of these optimiza-

tions are machine dependent and need to be manually tuned, for instance by selecting the best blocking size and padding width. Performance modeling helps the tuning process by exploring the performance trade-offs in a large parameter space.

We explore the following parameters in the FDTD implementation:

- the grid size (X, Y, Z) where each dimension is independently selected in the range $[768 : 1536]$ by steps of 128,
- the half stencil order p in $[1 : 8]$,
- the number of threads in $\{4, 8, 16, 32\}$,
- the number of blocks (NX, NY, NZ) respectively on X direction in $\{1, 2, 4, 8, 16\}$, Y and Z directions in $\{4, 16, 32, 64, 128\}$,
- the variant of the algorithm. The first variant, `isotropic`, corresponds to the code in algorithm 1. The second variant, `isotropic-split`, is the same code after loop splitting.

The possible factor combinations amount to more than 2.7 million.

7.2 ASK experimental setup

To produce the results presented in section 7.3, we build a performance model of the FDTD kernel using ASK. The target machine is the same Nehalem 32-core four-socket Xeon X7550 at 2.00GHz with 128GB of RAM that was used in the 2D stencil experiment in section 6.2. Threads were distributed evenly among sockets with `KMP_AFFINITY=scatter`.

Points are sampled in 16 batches of 50 points, using three different sampling strategies: HVS, HVSrelative and Random. The surrogate model, GBM, was manually tuned on a small set of points, before starting the full ASK experiment. The selected GBM parameters, described in section 6, are: `shrinkage`=0.05, `ntrees`=10 000, `depth`=9, `minobsinnode`=3.

In order to evaluate the error, we measure the real response of 3225 randomly selected points, which represent more than 60 hours of experiments on the 32-core machine. Since the test set is small compared to the design space, confidence intervals of the prediction error are computed using 1000 ordinary bootstrap iterations [27].

Figure 8 shows the RMSE and mean percentage error for the three sampling strategies. The experiment is stopped after 16 sampling steps to show ASK capability of building a performance model with a limited number of samples. HVS RMSE error curve seems to be still progressing. The most accurate model is built using HVS and GBM with a final mean error of 7.71% and an RMSE error of 4.14. The difference between HVS and random sampling in this experiment is small. Our intuition is that the design space has a high variability and gives few optimization opportunities to HVS which benefits mostly from flat regions. HVSrelative RMSE error is higher than the other methods. Since

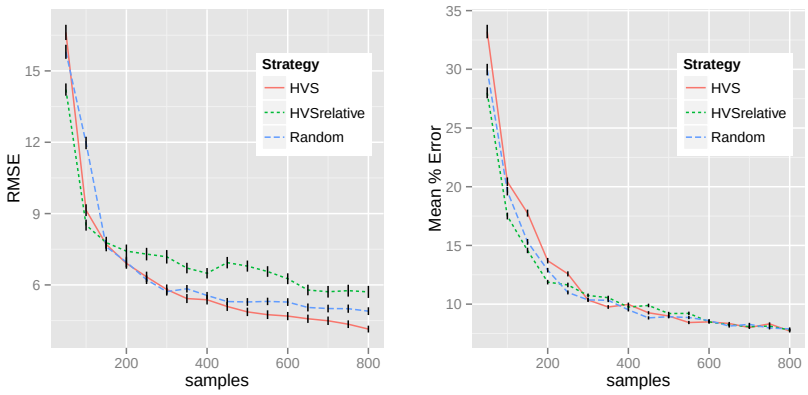


Figure 8: Root mean square error and Mean percentage error for the three tested strategies in the FDTD case study. The error is evaluated against a randomly selected test set of 3225 points. The vertical black lines show the bootstrap confidence intervals.

HVSrelative optimizes for relative error, this result is not surprising. However, HVSrelative does not show relative error improvement compared to the other methods in this experiment.

A GBM performance model of the FDTD kernel is built using the HVS 800 points sampling. Next section, uses this model to study the interactions between the different parameters of the FDTD kernel.

7.3 Performance characterization of stencil computation

The GBM model offers a useful feature that allows sorting the model factors by their relative influence. Figure 9 shows the relative influence of the different factors considered in our experiment. The relative influence is computed using the method proposed by Friedman in [16] and determines how much a given variable affects the response in the GBM model. Dominant factors of performance are the order of the stencil, the code variant, number of blocks on X and Y and the number of threads. Figure 9 gives an insight on the parameters to consider in priority to enhance performance. The following paragraphs give more details on the way these parameters affect performance. The metric used is the number of cycles required per lattice update.

The Variant Influence We consider two variants of the FDTD implementation. The first variant, `isotropic` in algorithm 1 computes the Laplacian in a triple nested loop. In the second variant, `isotropic-split`, the inner loop is split into p smaller loops. Each split loop corresponds to one of the b_p factored blocks. Figure 10 shows that the number of cycles per update increases for both variants as the stencil order increases. Yet, for high stencil orders with $2.p > 10$,

Algorithm 1 3D loop to update the wave equation U for the time step $t+1$.

for $k \leftarrow Zmin, Zmax$ **do**
  for $j \leftarrow Ymin, Ymax$ **do**
    for $i \leftarrow Xmin, Xmax$ **do**

$$
\begin{aligned}
Laplacian\_U = {} & b_0 * U_t(i,j,k) \\
& + b_1 * \{U_t(i-1,j\ \ ,k\ \ ) + U_t(i+1,j\ \ ,k\ \ ) \\
& \quad + U_t(i\ \ ,j-1,k\ \ ) + U_t(i\ \ ,j+1,k\ \ ) \\
& \quad + U_t(i\ \ ,j\ \ ,k-1) + U_t(i\ \ ,j\ \ ,k+1)\} \\
& \ldots \\
& + b_p * \{U_t(i-p,j\ \ ,k\ \ ) + U_t(i+p,j\ \ ,k\ \ ) \\
& \quad + U_t(i\ \ ,j-p,k\ \ ) + U_t(i\ \ ,j+p,k\ \ ) \\
& \quad + U_t(i\ \ ,j\ \ ,k-p) + U_t(i\ \ ,j\ \ ,k+p)\}
\end{aligned}
$$

$$U_{t+1} = 2\ U_t - U_{t-1} + c^2\ \Delta t^2\ Laplacian\_U$$

    end for
  end for
end for

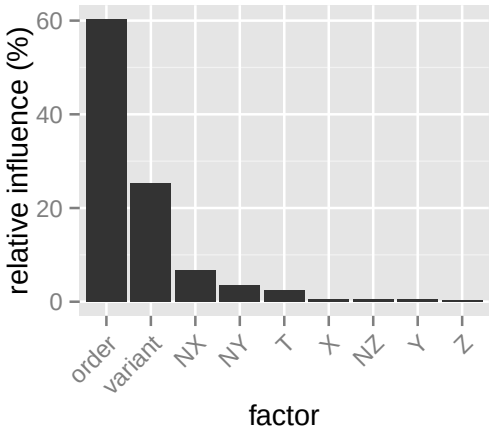


Figure 9: Relative influence of input factors in the FDTD kernel. The influence determines how much a factor affects the response. For the GBM model, it is computed as described in [16].

the `isotropic` variant is significantly more costly than the `isotropic-split` variant. The goal of loop splitting is to lower the register pressure and the number of concurrent memory streams. It is particularly effective on large loop bodies, such as the ones needed by high-order stencils.

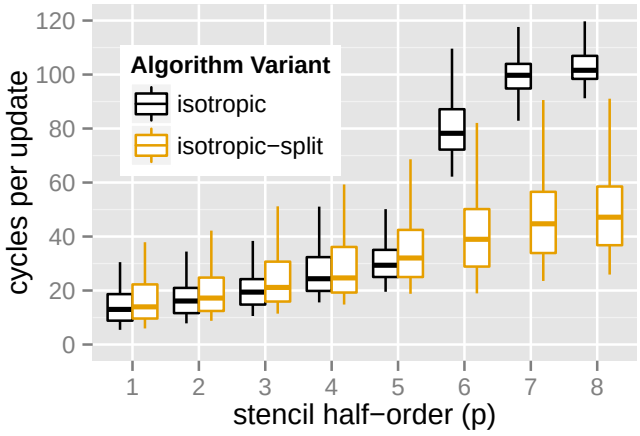


Figure 10: Performance of the `isotropic` and `isotropic-split` code variants. For p larger than five, the isotropic-split version is significantly faster.

The Blocking Influence This section studies the impact of the spatial cache blocking on performance in the `isotropic-split` variant with $p = 4$. Results are similar for other configurations. Cache blocking aims to increase data locality by reusing data before eviction from the cache. The loop illustrated in algorithm 1 can be blocked across the three dimensions.

Figure 11 illustrates the impact of blocking on the innermost dimension X. The grid is tiled with blocks: increasing the number of blocks reduces each block size. Performance deteriorates as the number of blocks on dimension X increases. Indeed, for small block sizes on X, the hardware prefetcher streams additional data, which are evicted from the cache before being used. Therefore, using small X block sizes result in an increase of memory traffic. The number of blocks in the X dimension should be kept low.

On the other hand, the outer Y and Z loops should be blocked to make the data working set of all the threads fit the cache. Figure 12 shows the correlation between the number of Y blocks and the number of threads. For high number of threads, configurations with a high number of Y blocks are the best. All the threads in the same socket share the same Last Level Cache (LLC). As the number of threads increases, the cache budget per thread decreases requiring smaller block sizes. Blocking across Z also helps, but the pay-off is smaller since it exposes less data reuse. Similar conclusions can be found in other studies on performance optimization of stencil computations such as [28, 29].

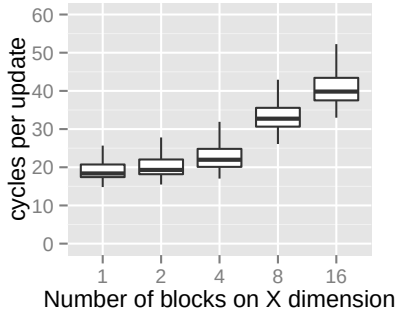


Figure 11: Performance of different X blocking configurations. The size of the blocks is inversely proportional to the number of blocks. Large blocks sizes across X exhibit the better performance.

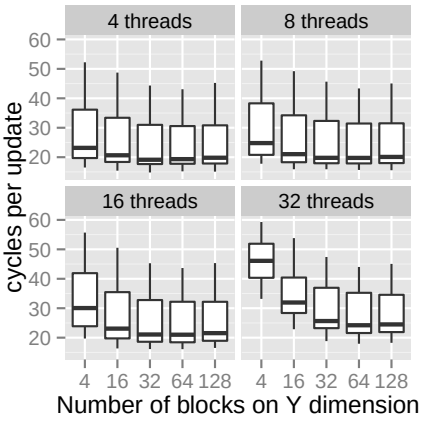


Figure 12: Performance of different Y blocking configurations. For a high number of threads, reducing the Y block size improves performance.

Scalability This section studies the strong scalability of a stencil of order 8 ($p = 4$) with the best variant and blocking parameters determined in the previous analysis. The selected parameters are `isotropic-split`, $p = 4$, $NX = 1$, $NY = 128$, and $NZ = 32$.

Figure 13 shows the scalability for two different grid sizes.

The kernel scales well up to 16 threads. ASK pinpoints a potential scalability problem, at 32 threads the speedup is around 26. Since the results are extrapolated from the model, we cross validate this result by direct measurement. Despite some small local discrepancies, the model predicts the general trend.

The model generated by ASK allowed us to tune the variant, the blocking factor, and to detect a scalability problem. The analysis presented here reaches similar conclusions to previous studies on stencil performance. Our results with ASK show that adaptive sampling is a valid approach for real application performance exploration.

8 Conclusion

Adaptive sampling techniques drastically reduce exploration time of large design spaces. Nevertheless, choosing the right technique can be difficult for a performance architect. ASK provides an homogeneous interface to multiple state-of-the-art sampling strategies, making the process easier. Adding new strategies to the framework is straightforward due to ASK's modular architecture.

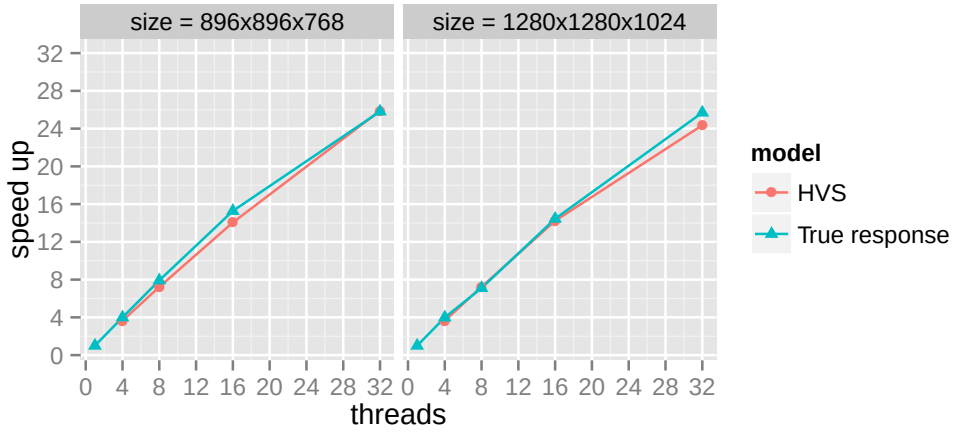


Figure 13: Scalability for the `isotropic-split` implementation with half order $p = 4$, $NX = 1$, $NY = 128$, and $NZ = 32$. The speedup is computed using the single thread performance with the same parameters.

The new HVS strategy reduces experimental bias and is comparable with other state-of-the-art approaches. It even outperforms them in two case studies. The 2D stencil code design space, which has more than 31×10^8 points, was accurately predicted using only 1 500 points.

ASK was used to characterize the performance of a finite-difference time-domain kernel used in industrial seismic imaging applications. The produced performance model identifies the parameters affecting performance and captures the interaction between different factors. Such a model can be used to identify performance problems, select among different code variants or blocking factors, or tune an application for a given architecture.

Currently, users must try different models manually to find what is best suited to their experiment. Tuning the model parameters can also be difficult and time consuming for a non expert user. Automatic model and sampling selection techniques [30] and automatic model tuning techniques [24, 13] applied to performance experiments will be investigated in future work.

Our preliminary set of experiments shows that the performance characterization field could benefit from adaptive sampling techniques. ASK is open source, the last version is available at `http://code.google.com/p/adaptive-sampling-kit`. The experimental data and benchmarks used to produce the paper results are available at the same URL.

Acknowledgments

The authors thank T. Guillet and J.C. Beyler from Intel for their fruitful comments. We also thank H. Calandra from Total for his help with the FDTD kernels. This work has been conducted conjointly by the Exascale Computing Research laboratory, thanks to the support of CEA, GENCI,

Intel, UVSQ, and by the LRC ITACA laboratory, thanks to the support of the French Ministry for Economy, Industry, and Employment throught the ITEA2 project H4H. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the CEA, GENCI, Intel, or UVSQ.

References

[1] Simpson T, Lin D, Chen W. Sampling strategies for computer experiments: design and analysis. *International Journal of Reliability and Applications* 2001; **2**(3):209–240.

[2] de Oliveira Castro P, Petit E, Beyler JC, Jalby W. ASK: Adaptive Sampling Kit for Performance Characterization. *Euro-Par 2012 Parallel Processing - 18th International Conference*, *Lecture Notes in Computer Science*, vol. 7484, Kaklamanis C, Papatheodorou TS, Spirakis PG (eds.), Springer, 2012; 89–101.

[3] Stein M. Large sample properties of simulations using Latin hypercube sampling. *Technometrics* 1987; :143–151.

[4] Johnson M, Moore L, Ylvisaker D. Minimax and maximin distance designs. *Journal of statistical planning and inference* 1990; **26**(2):131–148.

[5] Diwekar U, Kalagnanam J. Efficient sampling technique for optimization under uncertainty. *AIChE Journal* 1997; **43**(2):440–447.

[6] Shirley P, *et al.*. Discrepancy as a quality measure for sample distributions. *Proceedings of Eurographics 91*, Eurographics Society, 1991; 183–94.

[7] Li B, Peng L, Ramadass B. Accurate and efficient processor performance prediction via regression tree based modeling. *Journal of Systems Architecture* 2009; **55**(10-12):457–467.

[8] Ridgeway G. Generalized Boosted Models: A guide to the gbm package. *Update* 2007; **1**:1.

[9] Gramacy R, Lee H. Adaptive design and analysis of supercomputer experiments. *Technometrics* 2009; **51**(2):130–145.

[10] Gramacy R. tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. *Journal of Statistical Software* 2007; **19**(9):6.

[11] Cohn D. Neural network exploration using optimal experiment design. *Neural Networks* 1996; **9**(6):1071–1083.

[12] Settles B. Active Learning Literature Survey. *Science* 1995; **10**(3):237–304.

[13] Gorissen D, Couckuyt I, Demeester P, Dhaene T, Crombecq K. A surrogate modeling and adaptive sampling toolbox for computer based design. *The Journal of Machine Learning Research* 2010; **11**:2051–2055.

[14] Crombecq K, Gorissen D, Deschrijver D, Dhaene T. A Novel Hybrid Sequential Design Strategy for Global Surrogate Modeling of Computer Experiments. *SIAM Journal on Scientific Computing* 2011; **33**:1948.

[15] Breiman L, Friedman J, Olshen R, Stone C, Steinberg D, Colla P. CART: Classification and regression trees. *Wadsworth: Belmont, CA* 1983; .

[16] Friedman J. Greedy function approximation: a gradient boosting machine.(english summary). *Ann. Statist* 2001; **29**(5):1189–1232.

[17] Hürsch WL, Lopes CV. Separation of concerns. *Technical Report* 1995.

[18] Dasgupta S, Hsu D. Hierarchical sampling for active learning. *Proceedings of the 25th international conference on Machine learning*, ACM, 2008; 208–215.

[19] Atkinson E, Therneau T. An introduction to recursive partitioning using the RPART routines. *Rochester: Mayo Foundation* 2000; .

[20] McKay A. Distribution of the Coefficient of Variation and the Extended t Distribution. *Journal of the Royal Statistical Society* 1932; **95**(4):695–698.

[21] Datta K, Murphy M, Volkov V, Williams S, Carter J, Oliker L, Patterson D, Shalf J, Yelick K. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, IEEE Press, 2008; 1–12.

[22] Treibig J, Wellein G, Hager G. Efficient multicore-aware parallelization strategies for iterative stencil computations. *J. Comput. Science* 2011; **2**(2):130–137.

[23] Dursun H, Nomura K, Peng L, Seymour R, Wang W, Kalia R, Nakano A, Vashishta P. A multilevel parallelization framework for high-order stencil computations. *Euro-Par 2009 Parallel Processing* 2009; :642–653.

[24] Kuhn M. Building predictive models in r using the caret package. *Journal of Statistical Software* 2008; **28**(5):1–26.

[25] Baysal E. Reverse time migration. *Geophysics* Nov 1983; **48**(11):1514, doi:10.1190/1.1441434.

[26] Virieux J, Calandra H, Plessix R. A review of the spectral, pseudo-spectral, finite-difference and finite-element modelling techniques for geophysical imaging. *Geophysical Prospecting* 2011; **59**(5):794–813, doi:10.1111/j.1365-2478.2011.00967.x.

[27] Efron B, Tibshirani R. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science* 1986; **1**(1):54–75.

[28] Nguyen A, Satish N, Chhugani J, Kim C, Dubey P. 3.5-D Blocking Optimization for Stencil Computations on Modern CPUs and GPUs. *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, IEEE Computer Society: Washington, DC, USA, 2010; 1–13, doi:http://dx.doi.org/10.1109/SC.2010.2.

[29] Rivera G, Tseng CW. Tiling optimizations for 3D scientific computations. *Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '00, IEEE Computer Society: Washington, DC, USA, 2000.

[30] Maron O, Moore A. Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute* 1993; :263.