Performance Modeling at a Discount
An extended evaluation for the IPDPS 2020 paper submission

This document provides an extended version of the evaluation of the IPDPS 2020 paper submission ‘Performance Modeling at a Discount’. In order to ensure the general reproducibility of our results this document provides a detailed explanation of the complete evaluation process.

1 Evaluation methodology

As presented in our submission, we evaluated our new sparse modeling technique with a synthetic data analysis and compared its accuracy and cost to our old modeling approach. Additionally, we analyzed if there is a sweet spot for the number of repetitions per measurement point to counter the effects of noise, and if there is an optimal number of measurement points per parameter that increases the model accuracy. From our previous research we had only empirical values for the number of repetitions and measurement points per parameter. Therefore, it was important to answer these questions before optimizing model accuracy and cost. We did this analysis for 1, 2, and 3 model parameters. However, modeling only 1 parameter is a special case, as both modelers use an identical approach when modeling only a single parameter. Since, there is no option in selecting the measurement points, we only analyzed the optimal number of repetitions and measurement points per parameter. For 2 and 3 model parameters we compared the results of the old modeler, that uses all available measurements for modeling, and the sparse modeler that uses different combinations of points for modeling. For 3 parameters we also exposed the two modeling approaches to different amounts of noise (0%,±1%,±2%,±5%) and tested other parameter-value selection strategies apart from our parameter-value selection heuristic.

2 Evaluation environment

TODO...

3 Evaluation metrics (accuracy and cost)

To evaluate our sparse modeling technique and quantify it’s accuracy and cost in comparison to the current state of the art we conduct an extensive synthetic data analysis using different levels of noise, environment and modeler configurations. In order to simulate the noise one would experience taking performance measurements on a large computing cluster, we run several identical experiments with different levels of noise $n = \{0, 1, 2, 5\}$ %. For each of these experiments we generate a fixed size set of 100.000 test functions by instantiating our performance model normal form from Eq. **??** with random coefficients $c_l \in (0.01, 1000)$ and random exponents i_l and j_l selected from the sets $I = \{1, 2, 3\}$ and $J = \{1, 2\}$. Since we do not want inactive function terms for the analysis, $\{0\}$ is not allowed as an exponent. Furtheremore, we choose to use 3 parameter functions for the evaluation as this is the maximum number of parameters supported by the current state of the art modeler. The Equations 1 and 2 show an example for the two distinct types of functions (additive, multiplicative) we obtain using the described methodology.

$$\begin{aligned} f(x,y,z) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \cdot \\ (y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \cdot \\ (z^i)^{0|1} \cdot (log_2^j(z))^{0|1} \end{aligned} \tag{1}$$

$$\begin{aligned} f(x,y,z) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} + \\ c_2(y^i)^{0|1} \cdot (log_2^j(y))^{0|1} + \\ c_3(z^i)^{0|1} \cdot (log_2^j(z))^{0|1} \end{aligned} \tag{2}$$

It is important to mention that both, the current state of the art as well as the new sparse modeler, are capable of predicting more complex functions than shown in Eq. 1 or 2. Though, in order to predict such functions correctly a much larger amount of performance measurements is required. Therefore, they are not suitable to quantify the accuracy and cost reduction of the sparse modeling approach.

Before a function is used for the evaluation the contribution of each term tc in regard to the overall function value is checked. When modeling with noisy data it is important that the contribution of a term is higher than the amount of noise in the system. Otherwise the modeler can not distinguish between system noise and actual application behaviour. Therefore, we only allow terms contributing at least 3 times the amount of noise, resulting in the following configuration for $tc = \{1, 3, 6, 15\}$ %.

We evaluate each function in our test set with 5^m points, where $m = 3$ is the number of parameters, using fixed parameter values for $x = \{4, 8, 16, 32, 64\}$, $y = \{10, 20, 30, 40, 50\}$ and $z = \{2, 4, 6, 8, 10\}$. This provides us with a set of 125 measurements that we use as input for our two model generators. While the state of the art modeler requires all of these measurements for modeling, the sparse modeler only uses a small subset of points depending on its configuration. In principle there are 4 major configuration parameters that influence the point selection of the sparse modeler:

- *The single parameter point selection strategy:* determines which points are used to model the single parameter functions for x, y and z (e.g. cheapest, most expensive points).
- *The number of additional measurement points:* specifies the number of additionally used measurement points for modeling the multiparameter function besides the baseline points that were used for creating the single parameter functions (e.g. 0, 1, 2).

- *The multi parameter point selection strategy:* defines which additional points are used to model the multi parameter function (e.g. cheapest, most expensive points).

Figure **??** shows the results of our accuracy analysis for different modeler configurations over the generated test sets of 100.000 synthetic performance functions. Each subfigure represents a separate experiment with a distinct level of noise. The y-axis shows the percentage of correctly predicted models categorized by the modeler configuration on the x-axis. A model is either identical, when it exactly matches the generated test function, or correct, when the lead-order term is predicted correctly, or incorrect, when the lead-order term is also different. We call a model with a successfully identified lead-order term correct because the modeler was able to predict the core behaviour of the function correctly. Furthermore, it is not the ambition of our approach to predict every possible performance function correctly, but rather to identify scalability bugs in all kind of HPC applications. For every experiment we analysed 7 different modeler configurations, with the state of the art modeler functioning as the baseline. Each of these configurations is represented by a label consisting of a letter representing either the baseline (B) or sparse modeler (S) and a number representing the total number of measurements points used for modeling. *S13* for example stands for a version of the sparse modeler that only uses 13 points for modeling.

Figure **??** shows that for perfect data the baseline modeler *B125* predicts all of the test functions correctly. The sparse modeler *S13* on the other hand predicts only 92.9% of the test functions correctly. Though, it only uses the minimum number of base points required for modeling. Since we have a perfect cubic matrix of measurement points the sparse modeler is able to reuse some of these points, similar to the example shown in Figure **??**. Therefore, a minimum of 13 base points is sufficient for modeling. The next configuration *S14* only uses one additional point and already achieves a modeling accuracy of 99.9%. When using all available points the sparse modeler *S125* achieves the same result as *B125*.

For the next experiment, which is shown in Figure **??**, we introduced 1% of noise to the measurements. This results in an overall small decrease of correctly identified models among all configurations. Now 93.3% of the models predicted by *B125* are identical, 3.4% are correct and 3.2% are incorrect. Similar to the experiment without noise the results of *S13* are much worse than the other configurations with about 24.9% of incorrectly predicted models. This indicates that the base points itself are not sufficient to create accuracte performance models. Again using only one additional measurement point increases the accuracy of the correctly predicted functions to 93.2%. Though, when we keep adding additional measurement points the accuracy increase is neglectable in comparison to the increasing costs. Since the cost reduction of the modeling process is the main target of our new approach, we choose to work with the cheapest measurement points available. Hence, we adjusted the configuration parameters of the sparse modeler accordingly and set the single and multi parameter point selection strategies to use only the cheapest points for modeling. With this strategy *S13* in average uses only 1.53% of the cost used by the state of the art modeler *B125*. *S14* achieves an average cost reduction of 98.45% in comparison to *B125*, while retaining 99% of its accuracy.

Figure **??** shows the achieved model accuracy for 2% of noise. In comparison to the previous experiments the behaviour of the different modeler configurations is very similar. However, due to the increased amount of noise the number of identical models is slightly smaller, about 85.4%, while the number of correctly identified lead-order terms has increased to 6.4% for all modelers. These results show that our approach is able to predict over 90% of the models correctly even when exposed to high amounts of noise, exceeding those one would expect when taking measurements on a real computing cluster.

For our last experiment, which is shown in Figure **??**, we choose an even larger amount of noise to demonstrate that the previously observed behaviour is not changing. As expected the increased amount of noise leads to an increased number of incorrectly predicted models. For 5% of noise *S14* predicts 62.2% identical models, 12.3% correct models and 25.4% incorrect models. Furtheremore, these results indicate that we can not create reliable performance models from data with 5 or more percent of noise, as the error rate exceeds 25%.

Besides analyzing the accuracy of the created models we also evaluated their scalability. Figure **??** shows the results of the synthetic scalability analysis we conducted using the same set of test functions, modeler configurations and parameter values. Instead of analyzing the function terms we take a look at the divergence between the predicted and actual value at a specific measurement point $P(x, y, z)$. Therefore, we take the maximum parameter values of $x = 64$, $y = 50$ and $z = 10$ and multiply them with a scaling factor $s = \{2, 4, 8, 16, 32\}$. A model is correct when the difference at the resulting measurement point P_s is within the percentage of noise, in this case 2%.

In general we observed the same behaviour as in our previous experiments. The modeler configurations *B125* and *S125* performed identically, still predicting over 88% of the models correctly, even though we applied a scaling factor of 32× to the original parameter values. Similar as before *S13* achieves the lowest accuracy of all configurations. However, *S14* again indicates that one additional measurement point is enough to retain 99% of *B125s* accuracy. With *S14* we obtain a percentage of 91.1% of correct models at 2×, 88.2% at 4×, 87.5% at 8×, 87.4% at 16× and 87.3% at 32× scaling. Other configurations do not show a significant increase in accuracy, although they have a much higher cost. These results prove that the sparse modeler is capable of successfully predicting function behaviour at larger scales, using only the cheapest points for modeling.

In conclusion the 3 parameter synthetic evaluation showed that the sparse modeler is able to achieve the same accuracy as the state of the art modeler, using only the cheapest base points and at least one additional point for modeling. One additional point seems to be necessary in order to prevent overfitting on the base points, and the behaviour expressed by the single parameter models. Even though we were able to reduce the modeling costs drastically by applying our simple point selection strategy, that solely focuses on the reduction of the modeling costs, it is still not clear if this strategy also leads to the highest possible accuracy.

As described in Section **??** we want to identify the selection strategy that leads to the best models and minimal cost at the same time. Therefore, further investigation is necessary which will be done with our parameter value selection AI.

- we can use cheap points for base points - we can use cheap points for add points - we need at least on additional point to improve the accuracy - we can easily afford one or two additional points as they are very cheap in comparison to the previous cost of the full matrix - adding more points does not increase the accuracy of the method - this stays the same with increasing amounts of noise - therefore, our method is prone to noise - we can reduce the number of points to 14 (one additional, 13 base points) - and save upto x% cost - though, it is not clear which combination of points leads to the best and cheapest result at the same time. - maybe the cheapest points or the baseline does not necessary represent the best result?

3.1 Parameter Value Selection

Mini batch gradient descent = 1 < batch size < training set size.

3.2 Application Measurements

From the literature the expected base behaviour for Relearn is $\mathcal{O}(n \log_2^2 n + p)$. Before we were not able to predict the part with p, with our new modeler we are able to predict it! We used these measurement points for the prediction of the model: $p = \{32, 64, 128, 256, 512\}$ and $n = \{5000, 6000, 7000, 8000, 9000\}$ where p are the number of processes and n is the problem size.

The result of the old modeler is like the expected behaviour in the literature. The sparse modeler with the cheapest base points and all additional points comes to the same result as the old modeler with all points. The sparse modeler produces the same result only using the cheap base points without any additional points. More points do not necessarily increase accuracy but do increase the cost. So in conclusion we get the same model for only 14.67% of the cost. We get this result without any repetitions on the measurement points.