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 Methodology

To evaluate our new sparse modeling technique and quantify it's accuracy and cost in comparison to our old method we conducted an extensive synthetic data analysis using different levels of noise, parameter-value selection strategies and modeler configurations. Therefore, we ran many different experiments considering 1, 2 and 3 model parameters. We chose to use upto 3 model parameters for the evaluation as this is the maximum number of parameters supported by the old modeling approach due to the high costs it requires. In order to find the optimal trade-off between cost reduction and model accuracy we varied the number of measurement points per parameter $P = \{5, 6, 7\}$ and the number of repetitions per measurement point $R = \{1, 2, 3, 4, 5, 6, 7\}$, while applying $\pm 5\%$ of noise to the generated measurements. For models with 3 configuration parameters we additionally experimented with different levels of noise $n = \{0\%, \pm 1\%, \pm 2\% \pm 5\%\}$ and parameter-value selection strategies, which determine the measurement points used for modeling. Apart from that, we investigated if there is a sweet spot for the number of repetitions per measurement point that counters the effects of noise and if there is an optimal number of measurement points per parameter that increases the model accuracy. Each experiment we conducted was evaluated over 100.000 synthethic test functions, were we analyzed and compared the results of the two model generators. However, for performance models with only one parameter this comparison is not necessary. In this special case the modeling procedure used by the old and the new sparse modeler are absolutely identical and provide the exact same results. Therefore, we simply searched for the optimal number of repetitions per measurement point and the optimal number of measurement points.

2 Environment

In order to conduct all of these experiments under the same conditions we used the following evaluation environment and general definitions.

Synthethic performance functions: For the evaluation of each experiment we generated a set of 100.000 unique synthethic test functions, which we created by instantiating our performance model normal form (PMNF) from Eq. 1 with random coefficients $c_l \in (0.01, 1000)$ and random exponents i_l and j_l, selected from the sets $I = \{0, 1, 2, 3\}$ and $J = \{0, 1, 2\}$. We used a seed for the generation of the coefficients and selection of the exponents, so that we could evaluate different experiment configurations with the same test functions and compare the results.

$$f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \cdot \prod_{l=1}^{m} x_l^{i_{kl}} \cdot log_2^{j_{kl}}(x_l) \tag{1}$$

When instantiating the PMNF we created synthetic performance functions such as the one represented by Eq. 2. Each function consists of m terms (m is the number of model parameters), which in turn can consist of a polynomial term, a logarithmic term or both. The behaviour of these different terms can either be additive or multiplicative, as shown by Eq. 2 and Eq. 3. The coefficients $c_0, c_1, \ldots , c_k$ are randomly generated from the interval $(0.01, 1000)$. The equations 4 and 5 show examples for such performance functions with 2 model parameters, while the equations 6 and 7 show examples for functions with 3 model parameters.

$$f(x) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \tag{2}$$

$$f(x) = c_0 + c_1(x^i)^{0|1} + (log_2^j(x))^{0|1} \tag{3}$$

$$\begin{aligned} f(x, y) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \cdot \\ (y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \end{aligned} \tag{4}$$

$$\begin{aligned} f(x, y) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} + \\ c_2(y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \end{aligned} \tag{5}$$

$$\begin{aligned} f(x, y, z) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \cdot \\ (y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \cdot \\ (z^i)^{0|1} \cdot (log_2^j(z))^{0|1} \end{aligned} \tag{6}$$

$$\begin{aligned} f(x, y, z) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} + \\ c_2(y^i)^{0|1} \cdot (log_2^j(y))^{0|1} + \\ c_3(z^i)^{0|1} \cdot (log_2^j(z))^{0|1} \end{aligned} \tag{7}$$

Theoretically it is possible to add more than one term per parameter, however while this can increase the model accuracy, it also requires a much larger number of measurements for modeling. Since our work focused on the cases where a cost reduction is actually possible, such functions were out of scope of the evaluation.

Measurements: After generating the syntethic performance functions we then generated the corresponding measurements. Therefore, we evaluated each function in our test set for 5^m parameter value combinations, where $m \in \{1, 2, 3\}$ is the number of parameters, drawn from predefined parameter values for $x = \{4, 8, 16, 32, 64\}$, $y = \{10, 20, 30, 40, 50\}$, and $z = \{2, 4, 6, 8, 10\}$. For functions with 3 parameters, consequently we created 125 measurements points and 625 measurements, when considering 5 repetitions per point. In order to generate the measurements for the selected metric we simply inserted the defined parameter values into the generated functions.

Noise: To simulate the noise one would experience taking performance measurements on a large computing cluster, we ran several identical experiments with different levels of noise $n = \{0\%, \pm 1\%, \pm 2\%, \pm 5\%\}$. Therefore, for each measurement we generated an additional random value $v \in [-1, 1]$. This value represents the relative divergence from the actual measured value. A value of $v = 1$ represents a divergence of the maximum allowed noise level e.g. $+5\%$. A $v = -1$ represents a divergence of the minimum allowed noise level e.g. -5%. By performing this calculation for all measurements we were able immitate realistic system noise that is not uniformly distributed.

Term contribution: Before we used one of the generated functions for the evaluation, the contribution of each term tc in regard to the overall function value was 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 (signal). Therefore, we only allowed terms contributing at least 3 times the amount of noise, resulting in the following configuration for $tc = \{1\%, 3\%, 6\%, 15\%\}$.

Sparse modeler configuration: The generated measurements are the inputs for the 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 3 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).

Based on these configuration parameters the sparse modeler produces different results. For each experiment and all of its functions we saved the respective result of the two modelers, so that we can analyze them using different evaluation metrics.

3 Metrics

To evaluate the results of the synthetic analysis, we considered two aspects, accuracy and cost.

Cost: To define cost, we try to replicate the scenario of the tool being used to determine the scalability of an application, which is the one most often encountered in practice. We imagine that one parameter, x, represents the number of processes and that the metric we measure is the total runtime per process. The cost of the measurement (e.g., the total core hours to run the experiment) is therefore not the same for all samples. Given that the exhaustive modeler requires all measurements, we consider their accumulated cost to be 100%, and determine which percentage of it the sparse modeler requires.

Accuracy: To define accuracy, we consider the next parameter values in every series (e.g., $x = 128$, $y = 60$, and $z = 12$) and evaluate the resulting model for this combination. Then, we verify whether the predicted value is within $\{\pm5, \pm10, \pm15, \pm20\}$ % of the actual value.

Additionally, we took a look at a different accuracy measure, the term identification of the modeler. Depending on the number of functions terms one can evaluate if the modeler is able to correctly predict the entire function (we call this an identical model), only the lead orderm term (the major term that influences the function behaviour), or neither of them (we call this an incorrect model).

4 Results

The following sections discusses the results of the synthetic evaluation of the sparse modeling approach. It is structured as the folders in this Git repository. Each section will discuss the findings for the different numbers of model parameters. Then the evaluation is further devided in experiments with different amounts of measurement points per parameter and the repetitions per measurement point. For the evaluation with 3 parameters we additionally provide experiments with different noise levels and parameter-value selection strategies. Each of the sub-folders in the Git repository contains a README file that further describes the organization of the files and how to interpret them.

4.1 One model parameter

As described previously, both modelers provide identical results when modeling functions with only one parameter. Since their implementation is the same we focused on finding the optimal number of repetitions and measurement points to increase the accuracy and noise robustness of models with one parameter. Furthermore, it was not possible to reduce the modeling costs in this case. For one parameter we require a minimum of 5 points in order to be able to model different types of logarithmic and polynomial functions. Using less points for modeling would decrease the costs, however it would also drastically reduce model accuracy and we are not willing to make this trade.

From our previous research we had aquired empirical values for the number of repetitions ($R = 5$) and measurement points per parameter ($P = 5$). The results of our synthethic evaluation for one parameter show that these values were already very good. We found that...

... adding more measurements for each parameter, using six or seven samples rather than five. We concluded that the quality improvement does not warrant the increase in effort. We further found that repeating each measurement four times provides the optimal cost/benefit ratio in most cases, with the exception of single parameter models where even two repetitions can be sufficient...

4.2 Two model parameters

Text.

4.3 Three model parameters

Text.

Noise levels: Text.

Parameter-value selection strategies: Text.

5 Conclusion