Surrogate Models for Analyzing Performance Behavior of HPC Applications Using the RAJA Performance Suite

Befikir Bogale[1]*, Ian Lumsden[1]*, Dalal Sukkari[1], Dewi Yokelson[2], Stephanie Brink[2], Olga Pearce[2], and Michela Taufer[1]

[1] University of Tennessee, Knoxville, TN 37919, USA
[2] Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

Abstract. Optimizing supercomputer software requires identifying parameter configurations that maximize performance. However, the wide range of parameter values and their varying impact across systems make traditional identification methods insufficient, highlighting the need for new approaches to performance prediction and parameter tuning. We propose Surrogate-Based Modeling (SBM) as an efficient method for characterizing performance across the parameter landscape. Using data from the RAJA Performance Suite's computational kernels (RAJAPerf), we show that SBM outperforms the standard k-Nearest Neighbors (kNN) model, achieving predictions up to 54% more accurate while requiring 33% less data. Thus, SBM emerges as a powerful tool for enhancing performance predictions across diverse parameter combinations.

Keywords: Performance analysis · Surrogate-based modeling · Parameter tuning · Performance modeling · Magma Library.

1 Introduction

Parameters such as rank count, memory use, and data distribution strategies influence HPC application performance. These parameters vary widely in value, each affecting performance differently across diverse hardware platforms, from CPUs to GPUs and accelerators. Choosing the optimal parameter configuration is complex due to platform heterogeneity: a setting effective on one platform might fail on another. This requires extensive parameter sampling to map the performance landscape. Traditional methods of exploring configuration spaces involve either exhaustive sampling, which is accurate but computationally expensive, or local search algorithms (LSAs), which, like grid hill climbing or simulated annealing, are more efficient yet unpredictable, often converging to local optima or yielding inconsistent outcomes.

Surrogate-Based Modeling (SBM) [3, 4] offers a compelling alternative to exhaustive and local searches. By using limited samples, SBM builds predictive models to estimate performance landscapes, achieving near-optimal results by

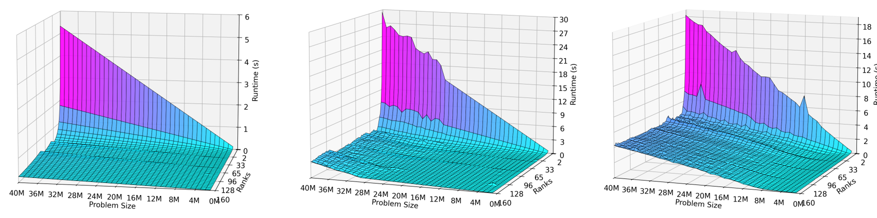
*Bogale and Lumsden contributed equally to this work.

carefully selecting the necessary sampling points. This method can predict optimal configurations not explored during learning, giving it an edge over traditional LSAs, while efficiently balancing exploration costs and prediction accuracy.

To assess SBM's role in tuning HPC configurations, we analyze the RAJA Performance Suite (RAJAPerf) [6], featuring microbenchmarks for optimizing kernels in the RAJA model. RAJAPerf includes varied kernels representing (i) compute-bound, with high arithmetic intensity, (ii) memory-bound, focusing on data movement, and (iii) hybrid-bound characteristics. We gather detailed performance data from three key kernels on heterogeneous platforms. By leveraging SBM, we show that a limited number of samples can effectively map the performance landscape, lowering prediction costs compared to full sampling.

2 Capturing Diverse Performance Patterns

The main resource constraints in HPC applications fall into three categories: compute-bound, memory-bound, and hybrid-bound workloads. RAJAPerf provides benchmark kernels that demonstrate these constraints in programming models such as OpenMP, CUDA, HIP, and SYCL, and classifies them accordingly [6]. Compute-bound kernels, such as Basic_TRAP_INT, are limited by



(a) Basic_TRAP_INT (b) Basic_INIT_VIEW1D (c) Stream_TRIAD

Fig. 1: Ground truth performance surfaces for compute-bound, memory-bound, and hybrid-bound kernels on CPU across problem sizes and ranks.

floating-point operations (FLOPs) and benefit significantly from architectures with powerful compute units. Memory-bound kernels such as Stream_TRIAD are constrained by memory bandwidth and show strong performance gains on systems with high bandwidth memory (HBM). Hybrid-bound kernels, such as Basic_INIT_VIEW1D, require a balance of compute power and memory bandwidth, benefiting from both high FLOPs and fast memory. We use RAJAPerf to sample performance across hardware and programming models, generating performance patterns from one kernel per category. We run our tests using Benchpark [7], a reproducible benchmarking repository, with performance metrics collected using Caliper [1] and analyzed using Thicket [2], an open-source Python tool. We evaluated the performance on individual nodes of the Lassen system at LLNL, focusing on CPU-based configurations. Using IBM POWER9 CPUs, we vary the MPI ranks from 2 to 160 and the problem size from 1M to 40M in 1M steps, resulting in 3,200 unique data points. Figure 1 shows performance surfaces for the three selected kernels: Basic_TRAP_INT (compute-bound) maintains flat runtimes due to negligible memory bottlenecks; Stream_TRIAD (memory-bound) shows increasing runtime slopes with problem size; and Basic_INIT_VIEW1D (hybrid-bound) displays mixed behavior where runtime increases at large problem sizes due to memory pressure. Across all kernels, we observe performance cliffs caused by simultaneous multithreading (SMT) imbalance, especially when MPI ranks exceed multiples of the 40 physical cores per node. This effect is most pronounced in Basic_TRAP_INT, indicating greater sensitivity to SMT contention.

3 Generating Performance Approximations with SBM

Building a surrogate model involves approximating the performance landscape using a sampled subset of the parameter space. This data fits a functional model that captures the relationship between parameters and performance. The model is refined iteratively to improve prediction accuracy while balancing computational cost and efficiency.

Selecting the Surrogate Model. When building a surrogate model, choosing the functional form to approximate the performance landscape is critical. In this study, we choose **Polynomial Regression** due to its balance between accuracy and computational efficiency when modeling the performance landscape of RAJAPerf kernels.

Constructing the Polynomial Model. We build our SBM by fitting a **polynomial surface** to sampled data points using **Least Squares Regression**. Consider a scenario with n observed runtimes (data points) corresponding to different parameter settings. These parameters (e.g., number of ranks, number of threads per rank, or problem size) act as independent variables and are represented as vectors $\vec{x}_1, \ldots, \vec{x}_n$. The performance metric, serving as the dependent variable (e.g., application runtime or I/O bandwidth), is denoted as $z_1, \ldots, z_n$. Each data point is modeled by a polynomial function $z(\vec{x})$, expressed as:

$$z(\vec{x}) = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 x^2 + \beta_5 xy + \beta_6 y^2 + \cdots + \beta_D y^d,$$

where β_i are the polynomial coefficients, $\vec{x}$ contains the independent variables (e.g., x, y), and d is the polynomial degree. The total number of monomials is $d_m = \frac{(d+1)(d+2)}{2}$. In matrix form, the system is represented as $Z = X\beta$.

The **Least Squares Problem** is then solved as: $X^\top Z = X^\top X\beta$, where the solution is obtained using the **LU Decomposition** of the matrix $X^\top X$. We use the **MAGMA** library [5, 8], optimized for dense matrix computations on heterogeneous architectures. This enables efficient and scalable performance, effectively allowing the surrogate model to handle large datasets and higher polynomial degrees.

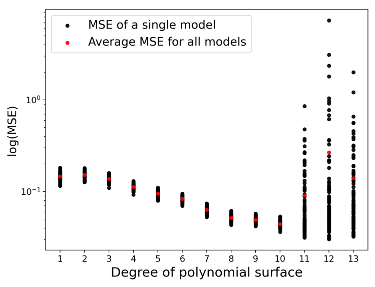
Model Validation and Selection. We validate our surrogate models using two key scoring metrics: **Mean Squared Error (MSE)** and **R-squared ($\mathbf{R^2}$)**. These metrics are the foundation for determining the near-optimal polynomial degree and number of sampled points required for the **k-fold cross-validation** procedure and the ensemble agreement approach upon which our assessment is based. They ensure the model maintains high accuracy and remains stable across application parameter settings. We use **k-fold cross-validation** with $k = 10$ to evaluate the performance of the surrogate model for each degree of polynomial d and a number of randomly sampled data points representing the performance measurements of a kernel. These data points are randomly shuffled and divided into k subsets, each subset serving as the validation set to prevent overfitting. For each degree d, given a certain number of sample points, we calculate the **MSE** and $\mathbf{R^2}$ values.

We incorporate **Ensemble Agreement** in our approach to assess the stability of our models. This technique involves training multiple surrogate models with slightly different configurations or data subsets and comparing their predictions. We calculate average **MSE** and $\mathbf{R^2}$ to measure the consistency across different models. Stable metrics (i.e., low standard deviation) are a proxy for high agreement between models and suggest that the surrogate model can generalize well across different computing platforms in HPC environments. We assess prediction stability using k-fold cross-validation and ensemble agreement, ensuring consistent validation and model generalization.

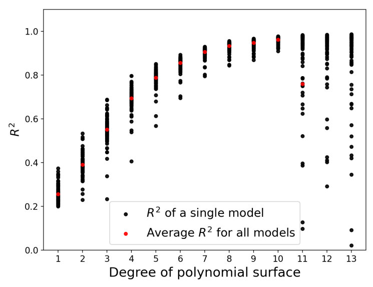
After performing **Ensemble Agreement**, we use the calculated **MSE** and $\mathbf{R^2}$ to determine the near-optimal degree and number of sampled points. We select the degree d_{best} that minimizes the average **MSE** and maximizes the average $\mathbf{R^2}$ as the near-optimal degree for the surrogate model. This process ensures the model generalizes well across different data subsets and avoids overfitting, improving its performance across the application parameter settings (i.e., the problem size and ranks). To determine the optimal number of sample points to train the surrogate model, we use $\mathbf{MSE'(x)}$, which is **the first derivative** of **MSE** with respect to the number of sampled points x. This metric measures the rate of change in **MSE** with respect to the number of points sampled and can find the optimal point where the model accuracy improves without overfitting. This derivative tracks how the **MSE** changes as the number of sampled points increases. We look for the point where the slope of the **MSE** curve transitions from negative to positive. This point represents the **optimal number of sampled points** where adding more data points no longer leads to significant improvements in the model. With the optimal polynomial degree and sample size, the validated model provides a computationally efficient approach to predicting performance metrics, significantly reducing the sampling cost.

4 Performance Predictions

Using the data from Section 2 and the methodology in Section 3, we predict the performance of the RAJAPerf kernels. We determine the near-optimal polynomial degree, identify the near-optimal number of sampled points required, and compare the performance of SBM with kNN under the same conditions.



(a) MSE for Basic_TRAP_INT



(b) R^2 for Basic_TRAP_INT

Fig. 2: MSE and R^2 for SBM on CPU using full dataset for polynomial degrees from 1 to 13 for Basic_TRAP_INT and Stream_TRIAD.

Optimal Polynomial Degree Selection Opting for the right polynomial degree is crucial for accurate SBM performance prediction, aiming to minimize MSE while maximizing R^2. Figure 2 shows the MSE and R^2 for SBM using the Basic_TRAP_INT kernel on a CPU, examining degrees 1 to 13. Increasing the degree reduces MSE, indicating better data fit until it levels off around degree 10. Beyond this, further increases offer little error decrease. Higher degrees also raise R^2 values, indicating better variance explanation. For the kernel, R^2 approaches 1 by degree 10, highlighting improved predictive capacity. The findings underscore that degree 10 offers a balance of complexity and accuracy, while higher degrees risk overfitting and inconsistent prediction.

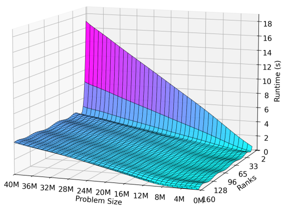
Similar MSE and R^2 patterns were seen for the Basic_INIT_VIEW1D and Stream_TRIAD kernels. Table 1 presents average metrics for 100 models with degrees ranging from 2 to 12. Bold cells indicate the optimal degree's MSE and R^2 for each kernel. Detailed figures showing the surfaces created by models with these optimal degrees are not shown due to space constraints, but the main observations from these figures are as follows. For Basic_TRAP_INT (CPU) at degree 10, runtime consistency across problem sizes in the predicted surface reflects the kernel's compute-bound nature, with predictions closely matching the actual data. For Basic_INIT_VIEW1D (CPU) at degree 10, the model identifies the hybrid-bound trend but fails to model sharp transitions due to SMT imbalance. For Stream_TRIAD (CPU) at degree 10, as expected, predicted runtime increases with problem size, accurately reflecting its memory-bound nature in the degree 10 SBM model.

Figure 3 illustrates the impact of selecting different polynomial degrees than the optimal degree 10 on the accuracy of the SBM for the Stream_TRIAD kernel on CPU. Figure 3a shows the model with the optimal degree of 10, striking the right balance between flexibility and generalization. In Figure 3b, the model is underfitted with a polynomial degree of 5, resulting in an overly simplistic surface that fails to capture the complex memory-bound behavior of Stream_TRIAD. On the other hand, Figure 3c shows overfitting with a polynomial degree of

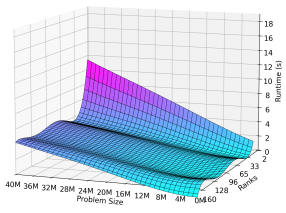
Table 1: Average MSE and R^2 for SBM degrees ranging from 1 to 13 using all data points for k-fold. Bold cells indicate the MSE and R^2 for the best degree.

	Basic_TRAP_INT CPU		Basic_INIT_VIEW1D CPU		Stream_TRIAD CPU	
Degree	MSE	R^2	MSE	R^2	MSE	R^2
2	0.1521	0.3907	0.7247	0.4616	0.4045	0.7407
4	0.1128	0.6939	0.5873	0.6932	0.3721	0.8363
6	0.0833	0.8599	0.4304	0.8592	0.2723	0.9253
8	0.0520	0.9335	0.2891	0.9316	0.1837	0.9656
10	**0.0443**	**0.9623**	**0.2299**	**0.9613**	**0.1374**	**0.9819**
12	0.2678	-5.1395	1.3101	-4.988	0.8093	-2.1725

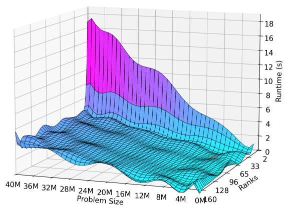
13, where the model captures noise and irregularities in the data, leading to unrealistic fluctuations in the predicted surface. Similar trends were observed in the other kernels and are not reported because of space constraints.



(a) Fit (d=10)



(b) Underfit (d=5)

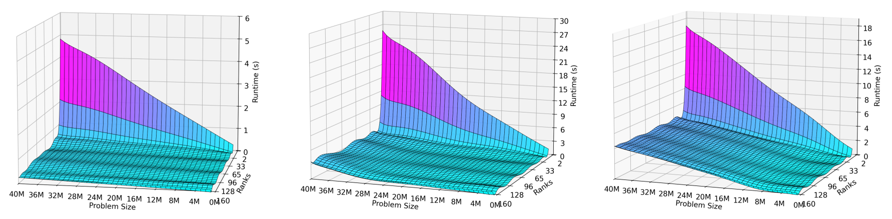


(c) Overfit (d=13)

Fig. 3: Overfitting and underfitting on SBM predictions for Stream_TRIAD showing the impact of polynomial degree d on model accuracy.

Near-Optimal Number of Sampled Points. Accurate performance predictions in HPC applications require efficient sampling strategies to minimize computational costs while maintaining model accuracy. We measure the derivatives of MSE with respect to the number of sampled points for each kernel's optimal SBM degree. We identify the point where the derivative changes sign, indicating the near-optimal number of sampled points beyond which adding more points yields a low accuracy gain while increasing the cost of the model generation. This point represents the number of sampled points where the SBM accuracy plateaus, signifying no significant improvement in model performance with additional data.

Across the considered CPU kernels, this optimal number of sampled points is approximately 2200, which is lower than the original 3200 points used. Figure 4 shows the surfaces generated by SBM using the best degree (10) and the best number of data points for each kernel (2200). Using the selected degree and the reduced number of sampled points identified through the MSE derivative analysis, the predicted surfaces in the figure closely match the patterns observed in Figure 1, effectively capturing performance trends while significantly reducing sampling cost.



(a) Basic_TRAP_INT (b) Basic_INIT_VIEW1D (c) Stream_TRIAD

Fig. 4: Predicted surfaces generated by SBM models using the best degree and the best number of sampled data points for each kernel

Table 2: Performance of the best SBM and kNN models (best value in bold).

Kernel	Model	Degree	k	# Pts	μ(MSE)	σ(MSE)	$\mu(R^2)$	$\sigma(R^2)$
Stream_TRIAD (CPU)	SBM	10	N/A	2200	**0.1453**	**0.0121**	**0.9814**	**0.0046**
	kNN	N/A	3	2700	0.1982	0.1391	0.9415	0.0312
Basic_INIT_VIEW1D (CPU)	SBM	10	N/A	2200	**0.2403**	**0.0210**	**0.9585**	**0.0167**
	kNN	N/A	3	2700	0.5189	0.4148	0.8900	0.0560
Basic_TRAP_INT (CPU)	SBM	10	N/A	2200	0.0465	**0.0037**	**0.9586**	**0.0186**
	kNN	N/A	3	2700	**0.0188**	0.0143	0.8874	0.0547

Comparison with kNN. To evaluate the effectiveness of SBM, it is crucial to compare it with existing commonly used methods, with k-Nearest Neighbors (kNN) being a popular choice for performance modeling. We compare SBM's capability to capture kernel patterns with kNN under the same circumstances. We use the best degree and number of sampled points identified earlier in the paper for SBM, and, for kNN, we use an optimal k and number of sampled points determined using the validation and selection methodology described in Section 3. Table 2 shows each kernel's average and standard deviation of MSE and R^2 for SBM and kNN. These results show that SBM consistently demonstrates better average accuracy and stability (i.e., lower standard deviation) for more complex surfaces, such as those observed with CPU kernels, while requiring fewer sampled points. SBM predictions are up to 54% more accurate and require up to 33% fewer sampled points than kNN.

5 Related Work

Diverse strategies for performance modeling are found in the literature, utilizing methods from surrogate-based models to data-driven insights. Travis et al. [3] optimized MapReduce job settings with polynomial surrogate models. Machine learning frameworks assist in performance modeling. Scikit-learn provides multiple regression models. An observation-based black-box method [9] predicts performance metrics using short partial executions that capitalize on predictable behavior after startup. Our research extends previous work by employing SBM on various platforms like CPUs and GPUs, and on different loop-based computational kernels from the RAJA Performance Suite. Our approach improves scalability over large parameter spaces by integrating polynomial regression with the MAGMA library via SBM [8, 5].

6 Conclusions and Future Work

Our SBM approach, which strategically integrates MSE, R^2, k-fold cross-validation, Ensemble Agreement, and MSE$'$(x), optimizes surrogate models with computational efficiency and high accuracy. It boosts accuracy by up to 54% and reduces sampling needs by 33% compared to traditional methods, significantly cutting sampling costs while ensuring stability and generalization. These advancements enhance the ability to predict performance across various application parameters, establishing SBM as an essential tool for optimizing HPC configurations.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under grant numbers 2331152, 2334945, and 2103845. The work was also performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 24-SI-005 (LLNL-CONF-2003169).

References

1. Boehme, D., Gamblin, T., Beckingsale, D., Bremer, P.T., Gimenez, A., LeGendre, M., Pearce, O., Schulz, M.: Caliper: Performance Introspection for HPC Software Stacks. In: Proc. of SC16 (2016)
2. Brink, S., McKinsey, M., Boehme, D., Scully-Allison, C., Lumsden, I., Hawkins, D., Lüttgau, J., Isaacs, K., Taufer, M., Pearce, O.: Thicket: Seeing the Performance Experiment Forest for the Individual Run Trees. In: Proc. of HPDC (2023)
3. Johnston, T., Mohammad, A., Cicotti, P., Taufer, M.: Performance Tuning of MapReduce Jobs using Surrogate-based Modeling. In: Proc. of ICCS (2015)
4. Johnston, T., Zanin, C., Taufer, M.: HYPPO: A Hybrid, Piecewise Polynomial Modeling Technique for Non-Smooth Surfaces. In: Proc. of SBAC-PAD (2016)
5. Nath, R., Tomov, S., Dongarra, J.: Accelerating GPU Kernels for Dense Linear Algebra. In: Proc. of VECPAR (2010)

6. Pearce, O., Burmark, J., Hornung, R., Bogale, B., Lumsden, I., McKinsey, M., Yokelson, D., Boehme, D., Brink, S., Taufer, M., Scogland, T.: RAJA Performance Suite: Performance Portability Analysis with Caliper and Thicket (2024)
7. Pearce, O., Scott, A., Becker, G., Haque, R., Hanford, N., Brink, S., Jacobsen, D., Poxon, H., Domke, J., Gamblin, T.: Towards Collaborative Continuous Benchmarking for HPC. In: Proc. of the SC23 Workshops (2023)
8. Tomov, S., Dongarra, J., Baboulin, M.: Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Computing **36**(5-6), 232–240 (2010)
9. Yang, L., Ma, X., Mueller, F.: Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution. In: Proc. of ACM/IEEE SC (2005)