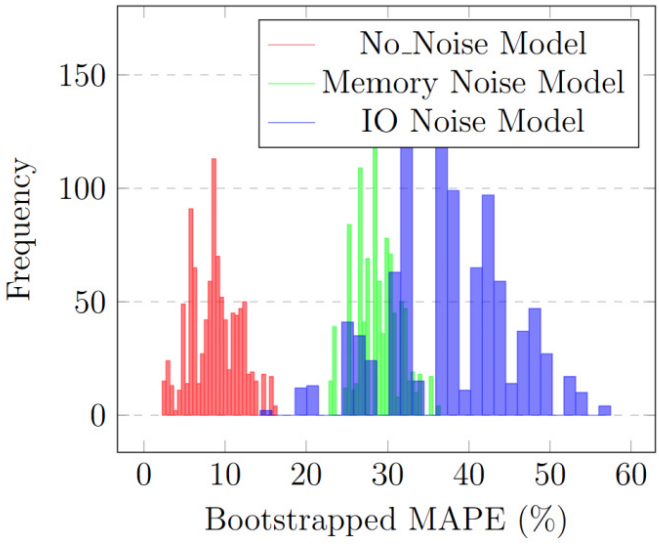
Hessisches Kompetenzzentrum
für Hochleistungsrechnen

Noise-Sensitivity of Algorithms and Applications



Project Manager
Ali Cebeci

Principal Investigator
Prof. Dr. Felix Wolf

Project Term
2024 - 2025

Clusters
Lichtenberg II Cluster Darmstadt

Institute
Parallel Programming

University
Technische Universität Darmstadt

Introduction

In modern scientific research, HPC systems are irreplaceable for advancing knowledge across various fields. However, as these systems scale to accommodate increasingly more computational resources, the necessity of creating a robust environment becomes more critical. Noise in large-scale systems can greatly affect program behavior, leading to unpredictability. Therefore, mitigating the impact of noise in applications and algorithms is crucial for ensuring predictable program behavior in high-performance computing. The complexity of modern scientific applications often demands large-scale computations that cannot be effectively handled by desktop computers. These applications typically involve massive datasets, intensive calculations, and complex algorithms that require parallel processing, distributed memory, and specialized hardware available only in HPC environments. Desktop systems lack the computational power and resources to accurately simulate the large-scale computations. Only through HPC systems can we model and assess the noise impact on a much larger, more representative scale, providing a more accurate understanding of how noise affects both the performance and behavior of algorithms in high-performance environments.

Methods

Before analyzing the noise sensitivity of LULESH 2.0, we first assessed run-to-run variation to distinguish inherent system variability from artificially induced noise. Moving forward with the analysis, we evaluated the predictive accuracy of performance models. Specifically, we trained three models:

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
TECHNISCHE UNIVERSITÄT DARMSTADT
UNIKASSEL VERSITÄT
Philipps Universität Marburg

noiseless, I/O-noise affected, and memory-noisy model for five repetitions for each input size. We then compared these models based on the execution time of the most significant functions. To assess accuracy, we calculated the Mean Absolute Percentage Error and resampled the values to establish confidence intervals, thereby mitigating potential biases from a small dataset. By comparing the overall trend and the error of the runs, the impact of each noise type could be identified and evaluated.

Results

The results indicate that the system exhibits inherent run-to-run variation, which must be accounted for in performance analysis. Despite this variability, the proposed method successfully identified the impact of noise. Notably, some functions were more susceptible to noise than others, with variations in execution time depending on the type of noise introduced. By quantifying these effects, the method enabled a detailed assessment of noise sensitivity.

Discussion

The results highlight that the extent of noise impact depends on program characteristics. Some programs exhibited greater sensitivity to I/O noise, while others were more affected by memory noise, or in some cases, both. This variation can be explained by the functionality of individual functions. Communication-intensive functions suffered primarily from I/O noise, whereas memory-bound computational functions experienced significant slowdowns due to memory noise. Understanding these patterns is crucial for optimizing system performance and designing noise-resilient applications. Future work should focus on refining noise detection methods to distinguish between different noise sources more precisely. Additionally, utilizing compiler-level analysis or hardware counters could provide deeper insights into how noise propagates through different system components, leading to more targeted noise-mitigation strategies. While the proposed method effectively identifies noise and its impact on program execution, it does not precisely pinpoint the exact source of performance degradation. Achieving this level of precision would require a more granular analysis. Tools such as LLVM could provide the necessary instrumentation to strengthen noise-sensitivity analysis by offering deeper insights into function-level execution behavior.

Figures

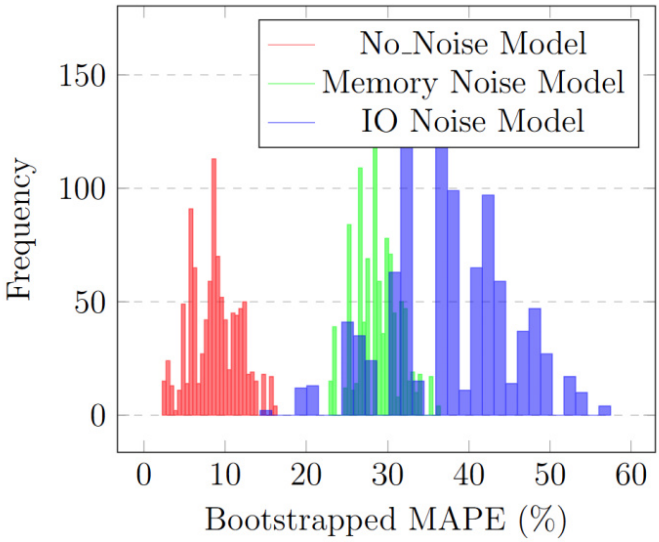


Figure 1: Bootstrapped MAPE values of function MPI Waitall.

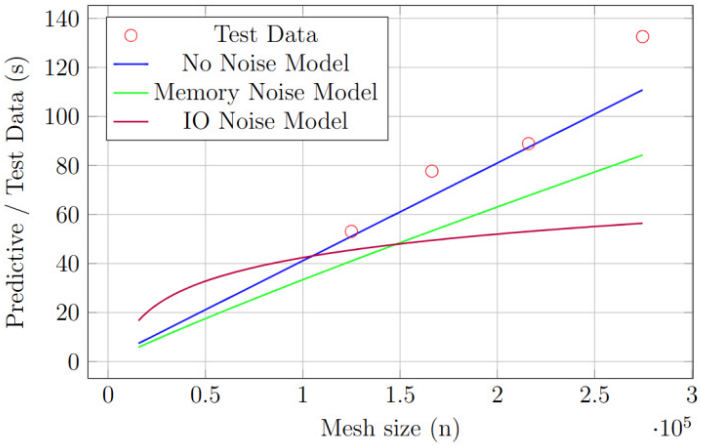


Figure 2: Performance Models of MPI Waitall.

Last Update: 2025-05-27 15:34