Study of Time Performance and Energy Consumption in an HPC Environment

Internship report

Frédéric Becerril
Directed by: Dr. Emmanuel Jeannot, Dr. Mihail Popov, Dr. Laercio Lima Pilla

National Institute for Research in Digital Science and Technology

INRIA Bordeaux

August 2023

Contents

ABSTRACT

In high-performance computing (HPC) systems, the balance between time performance and energy consumption is a critical challenge. This study, conducted at INRIA Bordeaux, focuses on unraveling the key parameters that influence both performance and energy efficiency, with a special emphasis on the communication between processes and process placement strategies

The research explores the intricate relationships between various metrics, analyzing how they impact overall system efficiency. Through a comprehensive examination of existing theories, coupled with experimental protocols and programming implementations, the study seeks to answer questions concerning the correlations between performance-improving parameters and energy-reducing factors. Importantly, the results indicate that in some cases, optimizing for time performance can lead to poor energy consumption scenarios. In such situations, energy consumption could be optimized by more than five times by considering different strategies and trade-offs.

Keywords Energy consumption · Communication

1 Introduction

In today's high-performance computing (HPC) landscape, the optimization of time performance and energy consumption is of paramount importance. As computing environments become more complex and demanding, there is a growing need to understand how various parameters affect both performance and energy consumption.

The problem arises from the delicate balance that must be struck between achieving maximum computing performance while minimizing energy usage. This balance is critical in an era where energy efficiency is not only a financial concern but also has broad environmental implications.

1.1 High-Performance Computing: An Overview

High-Performance Computing (HPC) refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

HPC is usually associated with the process of coordinating many processors to carry out simultaneous computations. This can be achieved through the use of supercomputers, specialized computing hardware that contains thousands to millions of cores, or by clustering, which connects multiple machines to operate as a single system.

HPC systems are designed to handle and analyze enormous amounts of data at high speeds. These systems not only process data more rapidly, but also incorporate techniques for managing and coordinating the simultaneous processing of data by many CPUs.

HPC has wide-ranging applications across many fields. In scientific research, HPC is used to model complex biological processes, predict weather or climate trends, simulate nuclear reactions, and analyze genomic data, among other uses. In business, HPC is often used for data-intensive tasks like predicting market trends, optimizing supply chains, or protecting against cyber threats.

In recent years, the focus in HPC has begun to shift towards enhancing energy efficiency, as the energy demands of running large-scale, high-speed computations can be significant. Balancing computational performance with energy consumption is one of the key challenges in modern HPC, making it an important field of study in the context of sustainable computing.

1.2 Study Objectives

The motivation behind this study is to investigate the key parameters that influence time performance and energy consumption in an HPC environment. The research aims to answer vital questions:

- What parameters improve the performance of HPC systems?
- What parameters reduce energy consumption in these systems?
- Are the parameters that affect performance and energy consumption correlated?
- If these parameters are not correlated, what might be the underlying reasons?

A crucial aspect of this study is the examination of communication between processes in an HPC environment. Understanding the dynamics of inter-process communication is vital as it can have profound implications for both time performance and energy consumption. Specific metrics related to communication between processes will be carefully analyzed, including:

- The different process placement strategies and their impact.
- The impact of communication protocols and methods on performance and energy efficiency.

By answering these questions, this study seeks to provide insights that can help optimize both performance and energy efficiency in HPC systems, contributing to more sustainable computing practices.

2 Related Work

There has been significant research on the optimization of HPC applications for both performance and energy efficiency. The relevant works can be divided into different categories such as performance optimization, energy efficiency optimization, and process mapping or placement.

2.1 Performance and Energy Optimization

In the work by [1], the authors explored the impact of configuration options and different problem sizes on performance and energy efficiency. They found that well-adapted NUMA-related options and cache-prefetchers provide significantly more gains for energy than performance. The study showed that reusing optimization strategies across problem sizes limits the original gains. The authors addressed this gap using machine learning models to predict the best configuration for a target size using information collected on another size. The models achieved 88% and 85% of the native gains when cross-predicting across performance and energy and across problem sizes, respectively.

The authors of [2] followed a similar approach, investigating the impact of configuration options and problem sizes on performance and energy. They observed that energy efficiency can be optimized by using strategies different from those used for performance optimization. The study suggested that machine learning can effectively predict the optimal configuration for different scenarios.

Another study [3] focused on the energy proportionality of multicore processors. The authors discovered that energy proportionality does not hold true for multicore processors. They proposed a bi-objective optimization method for energy and performance, which uses two decision variables: the number of multithreaded kernels and the number of threads per kernel. The method was demonstrated to be efficient in optimizing both performance and dynamic energy consumption.

2.2 Process Mapping or Placement

Process mapping, or process placement, is an important technique for optimizing the execution of parallel applications. The authors of [4] presented TopoMatch, a library and algorithm for process placement that takes into account machine topology and process affinity. The library is versatile and offers features to address different factors such as sparsity of the input affinity matrix, speed and quality trade-off, and input noise impact.

3 Methods and Tools

To investigate time performance and energy consumption in HPC, a specific set of tools and methodologies was employed. This section outlines the key benchmarking applications, computing techniques, and measurement tools used during the research, setting the foundation for the findings discussed later.

3.1 Benchmark Application: MiniGhost

MiniGhost is a prevalent benchmarking tool in the HPC domain, designed to simulate the propagation of heat on a grid. Conceptually, the simulation begins with all grid values set to zero. The value of a specific grid cell is then altered, representing a heat source, and the application simulates the subsequent heat propagation across the grid. This simple yet powerful simulation offers a rich environment to investigate various HPC-related parameters and their impact on performance and energy efficiency.

The MiniGhost application provides a plethora of adjustable parameters for users to tailor the simulation to their specific needs and to explore a wide range of scenarios. Some of the key parameters include:

- **Number of Stencils:** This determines how the heat value of a specific grid cell influences its neighboring cells, thereby affecting the heat propagation pattern.
- **Number of Variables:** By adjusting this parameter, users can simulate scenarios with varying degrees of complexity, possibly simulating different types of heat sources or multiple simultaneous heat introductions.
- **Grid Size:** The dimensions of the grid in X, Y, and Z axes can be adjusted, offering the flexibility to simulate both small-scale and large-scale heat propagation scenarios.
- **Grid Composition:** Users can define the structure of the grid, such as lines, squares, influencing how the heat propagates across the grid, and the number of communication.
- **Communication Protocol:** Given the significance of inter-process communication in HPC, MiniGhost allows users to select the communication protocol, which can significantly influence both time performance and energy consumption.

It is worth noting that MiniGhost is particularly targeted at MPI applications, which will be discussed in greater detail in the following section 3.2.

Given its versatility and relevance, MiniGhost was chosen for this study to simulate the HPC environment, enabling an in-depth exploration of the interplay between various parameters, their impact on HPC performance, and energy consumption.

3.2 Parallel Processing: MPI

MPI (Message Passing Interface) is a standardized and portable message-passing system designed to facilitate the development and execution of parallel applications across a multitude of parallel computing environments. For this study, MPI was crucial in orchestrating the execution of the MiniGhost application across multiple processes, potentially spanning one or several machines.

One of MPI's strengths lies in its flexibility in process placement. Users can meticulously control where specific processes are launched, enabling an exploration of various configurations and their subsequent impact on performance. This placement capability is particularly crucial for HPC applications like MiniGhost, where the positioning of processes can influence communication patterns and, by extension, overall execution time and energy consumption.

Beyond process execution, MPI provides a large set of metrics that are indispensable for a comprehensive analysis in HPC research. Among these, communication patterns stand out as especially significant. These patterns, which detail how data is exchanged between processes, offer insights into the efficiency and effectiveness of parallel execution. By analyzing these patterns, researchers can identify bottlenecks, inefficiencies, or opportunities for optimization.

In this study, MPI's capabilities were paramount in providing a realistic HPC environment, enabling an in-depth investigation of the relationship between process placement, communication patterns, and their combined influence on both performance and energy efficiency.

3.3 Platform: PlaFRIM

The research was conducted on PlaFRIM, a computing cluster located in Bordeaux, France, designed to support experiment-driven research in applied mathematics related to modeling and high-performance computing. PlaFRIM provided the necessary computational resources and environment to carry out the project's experiments and analyses.

3.4 Energy Consumption Measurement: LIKWID

To measure the energy consumption of the Minighost application and other relevant metrics, the LIKWID toolsuite was employed. LIKWID is a performance-oriented toolkit that supports various processors, including Intel, AMD, ARMv8, and POWER9, on the Linux operating system. It offers a range of command-line applications and a library that enabled for detailed monitoring and assessment of energy consumption and other performance aspects.

4 Observations and Results

The results section is a comprehensive compilation of empirical data obtained from numerous experiments designed to elucidate the intricate relationship between various software and hardware parameters in an HPC environment. Presented in the form of two distinct heatmaps, one for performance (Perf) and the other for energy consumption (Energy), the visualization offers a clear juxtaposition of software (SW) parameters on the y-axis and hardware (HW) parameters on the x-axis.

On the software side, the investigation delved into varying configurations, encompassing:

- The number of stencils used in the calculations is indicative of the number of neighboring data points used to compute the new temperature at a grid point. In this study, we considered different stencil configurations, represented as 2D5P, 2D9P, 3D7P, and 3D27P. (figure 1)
  - **2D5P**: In this configuration, the stencil includes the immediate neighbors in a 2D plane. The center point of the stencil is the grid point for which the new temperature is being computed. The neighboring points included are the four adjacent points in the North, South, East, and West directions, for a total of five points, including the center point.
  - **2D9P**: This configuration also operates in a 2D plane but includes the diagonal neighbors as well. Along with the four immediate neighbors, the stencil also takes into account the diagonal neighbors in the Northeast, Northwest, Southeast, and Southwest directions. This results in a total of nine points.
  - **3D7P**: This stencil configuration operates in a 3D space. It includes the immediate neighbors in the up, down, left, right, forward, and backward directions, similar to the 2D5P configuration, but adapted to 3D. The center point and the six neighboring points result in a total of seven points.
  - **3D27P**: In this configuration, the stencil operates in a 3D space and includes every adjacent neighbor. This encompasses all the points immediately adjacent to the center point in any direction, including diagonals, resulting in a total of 27 points.

- The number of variables, which ranged from 1 to 40.
- Communication protocols, specifically BSPMA and SVAF.
  - **BSPMA (Bulk Synchronous Parallel Message Aggregation)**: In simple terms, BSPMA waits until all processes have finished their calculations before allowing them to share information with each other. This way, the messages are all sent at the same time in a coordinated fashion. It's like waiting for everyone in a meeting to finish speaking before moving on to the next topic.
  - **SVAF (Single Value Aggregation Function)**: SVAF is different. As soon as a process finishes calculating one piece of information, it immediately sends it to the next process. It doesn't wait for others to finish their calculations. It's like passing a note to your neighbor in a meeting as soon as you write it down, without waiting for others.
- The scaling technique, categorized into Strong and Weak scaling. (4.2.2)

In juxtaposition, the hardware configurations explored were:

- The number of processes, varying from a singular process up to 32.
- Process placements, categorized as Scatter, Contiguous, or Compact. (4.1)

With this structured approach, the subsequent findings offer insights into how the interplay between software and hardware parameters impacts both time performance and energy consumption in an HPC setup.

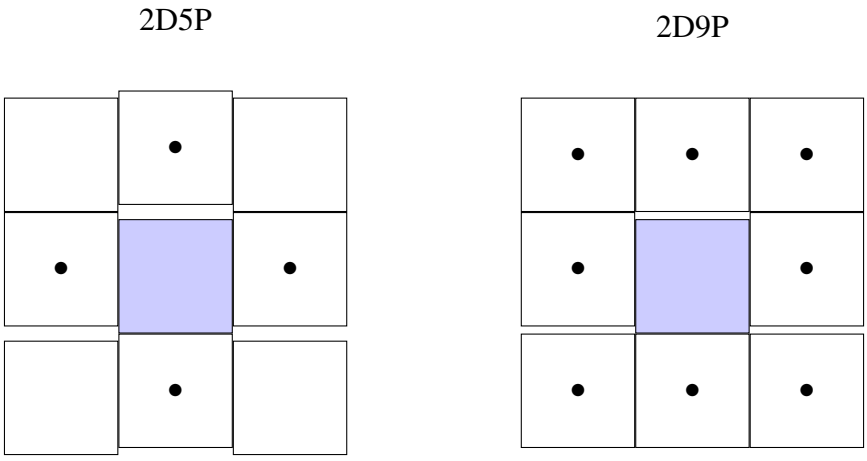

Figure 1: Visual representation of different stencil configurations.

4.1 Process Placement Configurations on a Dual-Socket System

Working within a dual-socket environment with 36 processors split evenly between the two sockets, the processes were strategically placed to explore the impact of distinct configurations:

1. **Scatter:** Processes are alternately distributed across the two sockets. Specifically, the first process is allocated to the 1st socket, followed by the second to the 2nd socket, and so on in an alternating pattern. This ensures that even-numbered processes are placed on the 1st socket, while the odd-numbered ones are on the 2nd socket.

2. **Contiguous:** The first half of the processes (n/2) are entirely dedicated to the 1st socket. Subsequently, the remaining processes, making up the other half (n/2), are designated for the 2nd socket.
3. **Compact:** All processes initially occupy the 1st socket to its fullest capacity. Once this socket is filled, any surplus processes overflow to the 2nd socket.

4.2 Performance and Energy Consumption

The heatmap illustrates the relative performance and energy consumption under various hardware parameters compared to the baseline configuration of 32 processes with scatter placement. The color spectrum ranges from blue to red, indicating a slowdown to speedup, respectively, with white representing no change.

4.2.1 Strong Scaling

From the heatmap for strong scaling (figure 2), it is evident that in terms of performance, the baseline configuration is optimal as no other configuration results in a speedup. Reducing the number of processes inevitably leads to a slowdown, implying that utilizing as many processes as possible is advantageous. In contrast, energy consumption behaves differently. For many cases, employing fewer processes (16 instead of 32) is more energy-efficient. This trend is particularly pronounced for smaller-scale applications characterized by fewer variables and neighbors.

These findings suggest that while maximizing the number of processes is beneficial for performance, a more nuanced approach is necessary for optimizing energy consumption. It highlights the trade-off between performance and energy efficiency and underscores the importance of considering the specific characteristics of the application in making decisions regarding process distribution.

4.2.2 Weak Scaling

In weak scaling, the problem size per processor is kept constant while the total problem size increases proportionally with the number of processors. For this study, throughput, computed as the total processed data divided by the number of processors, is the metric used for both performance and energy comparisons.

The heatmap for weak scaling (figure 3) shows that the optimal hardware configuration for both performance and energy is 32 processors, regardless of other variations in application parameters. This consistency is a direct result of the increased problem size that comes with more processors, which requires more computational resources.

The results suggest that when dealing with large datasets, using a greater number of processors is beneficial. The time saved from the increased parallelism outweighs the increase in power consumption. Consequently, the overall energy consumption decreases with the number of processors. This trend highlights the advantages of scaling up computational resources when faced with bigger data loads, as it can lead to both faster processing times and lower energy consumption.

4.2.3 Big Data Set

In order to validate the hypothesis that larger data sets may cause the behavior of performance and energy consumption to converge, an additional experiment was conducted with a manually increased grid size, ensuring that it remained constant for each process. A select few application parameters, which exhibited differing behavior in the previous heatmap, were chosen for this experiment.

The results of this experiment seem to confirm the hypothesis. As the size of the grid increases, the divergence between the behavior of energy consumption and performance diminishes. This convergence is particularly evident when compared to the results from smaller grid sizes. This indicates that the size of the data set can significantly impact the interplay between performance and energy consumption, and must be considered when optimizing for these metrics.

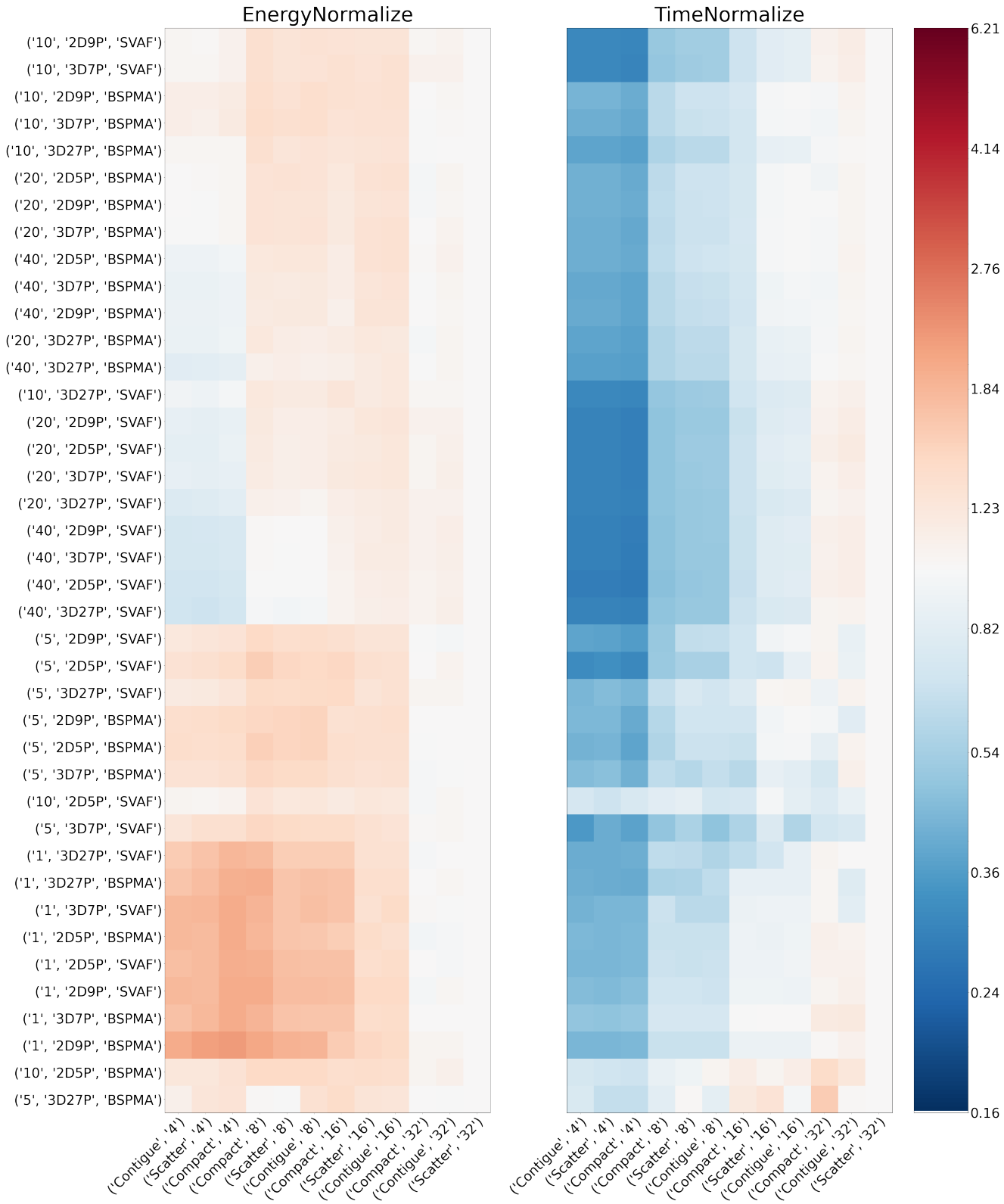

Figure 2: Heatmap for strong scaling

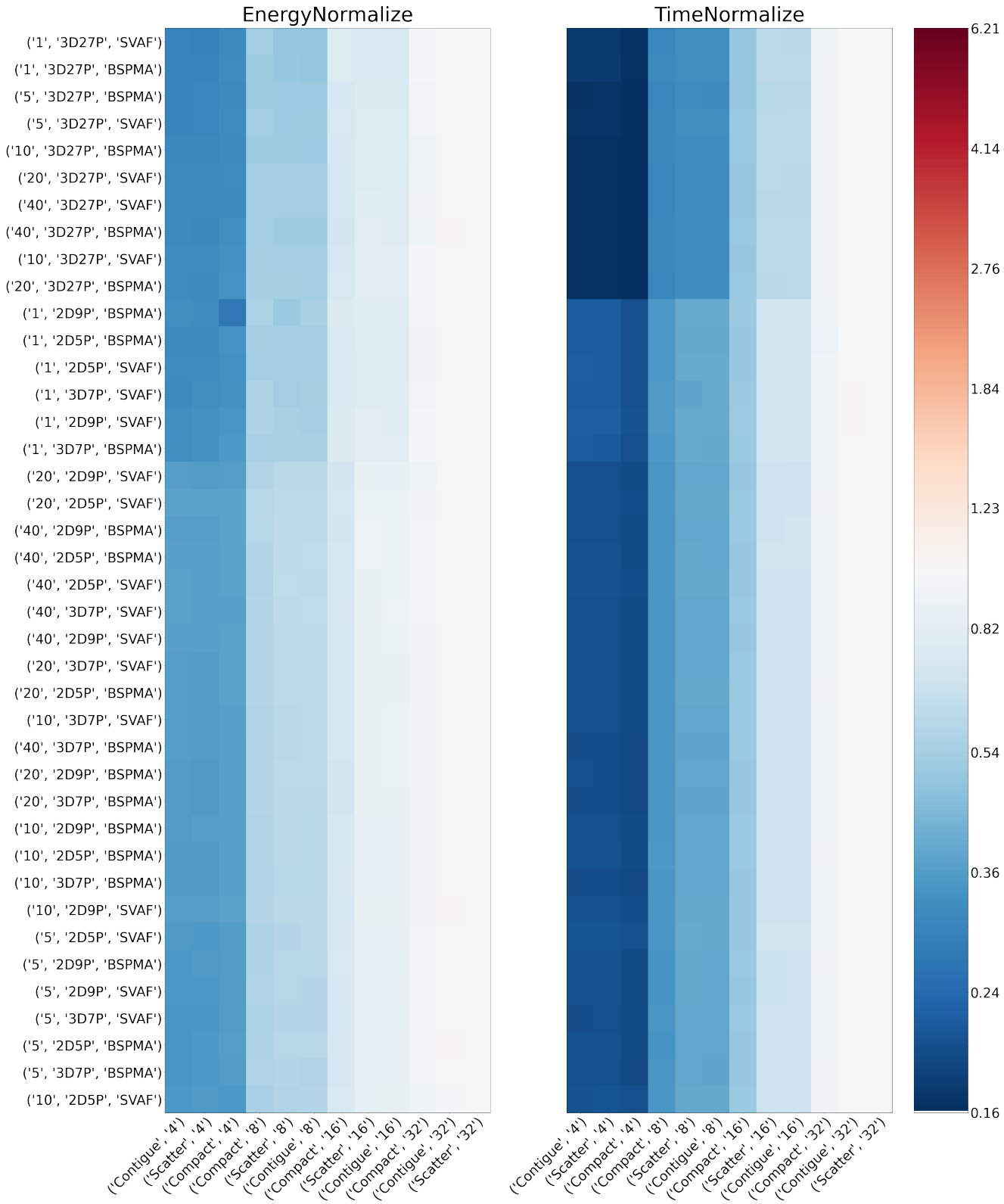

Figure 3: Heatmap for weak scaling

The results obtained from the experiments provided valuable insights into the performance and energy consumption characteristics of the miniGhost application under various hardware and software parameters. It was observed that there exists a trade-off between performance and energy efficiency, and the optimal configuration for one does not necessarily translate to the best configuration for the other. This trade-off was especially pronounced under different process distributions and application parameters. Additionally, it was determined that the size of the data set plays a critical role in influencing the behavior of energy consumption and performance. This was confirmed through a separate experiment with a manually increased grid size, where the divergence between the two metrics diminished as the grid size increased.

With these insights in mind, the next step is to develop a predictive model that can estimate the performance and energy consumption of the miniGhost application for various configurations without the need for running the application. By doing so, we aim to understand the underlying factors contributing to these metrics and provide recommendations for optimal configurations that balance performance and energy efficiency.

4.3 Correlations and deviations between performance and energy consumption

In the realm of high-performance computing, there's a common assumption that optimizing for performance will also optimize for energy consumption. While in some contexts, especially larger-scale applications, this holds true with an increase in the number of processes leading to benefits in both performance and energy conservation. However, our findings suggest that this is not a universal truth. For smaller applications, an over-reliance on increasing process counts can lead to disproportionate energy expenditures without a corresponding gain in performance.

A closer examination of configurations further highlights this divergence. When considering the top 5 configurations optimized solely for time, they underperform in energy efficiency. Surprisingly, the reverse doesn't hold true: the top 5 configurations for energy conservation demonstrate competitive time performance. This suggests that a configuration honed for energy efficiency doesn't significantly compromise on performance.

Thus, it raises a compelling argument: while optimizing for time is a natural instinct in HPC settings, there might be a greater overall advantage, both in performance and sustainability, in shifting our focus towards energy-centric configurations.

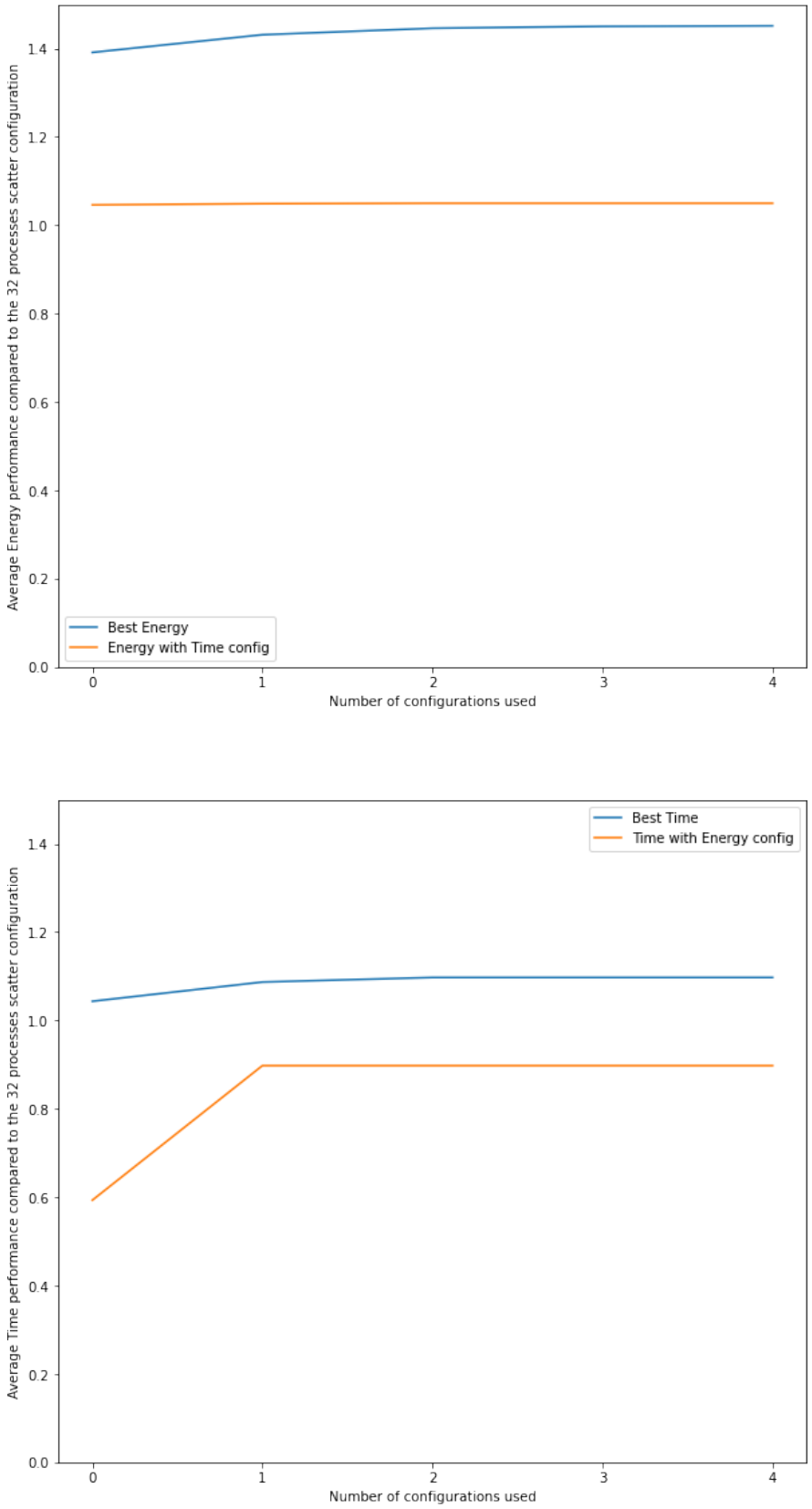

Figure 4: Performance when using different metrics as an optimization

5 Modeling and Analysis

In this section, we will outline the process of creating a predictive model for the performance and energy consumption of the miniGhost application. We will utilize the data obtained from the previous experiments as well as various modeling techniques to build and validate the model. The goal of this model is to provide accurate predictions for the performance and energy consumption of the miniGhost application under different hardware and software parameters, enabling better decision-making and optimization without the need for running the application.

5.1 Modeling without Communication Metrics

In the first approach, we will develop a predictive model for the performance and energy consumption of the miniGhost application without considering communication metrics. This model will solely rely on the application an hardware parameters such as the number of stencils, number of variables, size of the map, and process placement. The aim of this approach is to assess the accuracy of a model that ignores the communication aspect. We will evaluate the model's predictions against the actual performance and energy consumption data and analyze the error rate to determine the significance of communication metrics in predicting these metrics.

5.1.1 Predicting Time Using Amdahl's Law

In this subsection, we will explore a predictive model for the execution time of the miniGhost application using Amdahl's Law, a formula used to find the potential improvement in the execution time of a task when only a portion of it can be parallelized. This is a fundamental concept in parallel computing and is crucial for understanding the limits of parallel performance.

Amdahl's Law is mathematically expressed as follows:

$$T(p) = \alpha \cdot T_1 + (1 - \alpha) \cdot \frac{T_1}{p}$$

where $T(p)$ is the total execution time with p processes, T_1 is the execution time with one process, α is the fraction of the execution time that is sequential (non-parallelizable), and p is the number of processes.

The assumptions underlying this version of Amdahl's Law are:

1. The workload consists of two distinct parts: the sequential part and the parallelizable part.
2. The parallelizable part is perfectly parallelizable, meaning that the execution time of this part is divided by the number of processes, and no overhead is introduced due to parallelization.

In the context of predicting execution time without considering communication metrics, we will use Amdahl's Law to model the impact of changes in the number of processors on the execution time of the miniGhost application. By identifying the sequential fraction α of the execution time and the execution time with one process T_1, we can predict the execution time for various numbers of processes p.

The predictive model based on Amdahl's Law, although generally accurate for many applications, faces significant limitations in the context of the miniGhost application, particularly in estimating the sequential fraction α. Through experimental results, it was

observed that the estimated α varied significantly across different application configurations. This variability is a critical issue as it indicates that the proportion of the code that is sequential and non-parallelizable is not consistent across different software parameters.

A possible explanation for this inconsistency is the complex interplay between various software parameters and the underlying hardware architecture. Factors such as memory access patterns, and data dependencies can all influence the proportion of the code that is sequential. Additionally, the time taken for communication between processes could be another contributing factor to this inconsistency. As the number of processes increases, the communication overhead may become more pronounced, affecting the overall execution time of the application.

To assess the accuracy of the model, the mean error was computed by comparing the predicted execution time with the actual execution time across different configurations. While the mean error was found to be relatively low, the model's accuracy dropped significantly when using the mean value of α estimated across different application configurations. This resulted in a mean error of up to 10% and, in some cases, exceeded 30%.

These limitations indicate that predicting the sequential fraction α in the context of the miniGhost application is a complex and challenging task. The variability in the estimated α values and the high error rates when using a mean value suggest that the model, although valuable in some scenarios, may not be sufficiently robust for predicting the performance of the miniGhost application across different software parameter configurations. In the following sections, we will explore the role of communication metrics in enhancing the accuracy of the model.

5.1.2 Power Model without Communication Metrics

In the following, when we will talk about power, we will talk about the mean power measure during the execution of the application.

The power model used for this study is based on a simple linear relationship between power and the number of processors:

$$P(p) = \text{base\_power} + \beta \cdot p$$

In this model, base_power represents the power used by the machine just to keep it running, while β is a constant representing the additional power required to supply each additional processor. The model assumes that the power used by the machine is a combination of the static power required to maintain the machine's operation and the dynamic power associated with the additional processors.

To estimate the parameters base_power and β, we utilized linear regression analysis on the collected data. The results showed that the model could effectively predict the power of the miniGhost application under different hardware configurations.

This simple power model has shown promising results in predicting power. However, as with the performance model, there is room for improvement. In the following sections, we will explore how communication metrics can be incorporated into the power model to further enhance its predictive accuracy.

5.2 Modeling with Communication Metrics

In the previous sections, we explored models to predict the performance and energy consumption of the miniGhost application without considering communication metrics. While these models yielded reasonably accurate predictions for many scenarios, they were less accurate for small-scale applications. Small applications, which have fewer variables

and neighbors, have a higher proportion of communication compared to computation. Thus, the time spent in communication can become a significant factor affecting the overall performance and energy consumption of these applications.

Communication between processes can introduce additional overheads, such as data transfer times and synchronization delays. In this section, we aim to improve the accuracy of our performance and energy consumption models by incorporating communication metrics. By accounting for the effects of communication, we hope to achieve better predictive accuracy, especially for small-scale applications where communication overheads have a more pronounced impact.

5.2.1 Predicting Time using Amdahl's Law and Communication overhead

In this section, we will refine our time prediction model by splitting the time spent by an application into two components: calculation time and communication time. The calculation time corresponds to the actual computation performed by the processes, while the communication time accounts for the overhead introduced by inter-process communication. As we have seen in the previous section, Amdahl's Law is a useful model for predicting calculation time. However, it does not consider the impact of communication overhead.

Our refined model will utilize Amdahl's Law to predict the calculation time, incorporating an adjusted value for the proportion of sequential work α that accounts for the more parallelizable nature of the calculation phase. Fortunately, the miniGhost application provides valuable insight into the communication time, enabling us to accurately measure the time each process takes to communicate with others. By leveraging these communication metrics, we can predict the communication time, thereby enhancing the accuracy of our overall time prediction model.

By separating the calculation and communication components of the application, we aim to achieve a more accurate representation of the total execution time, especially for small-scale applications where communication overhead is more significant.

The results of the refined model incorporating both calculation and communication times have proven to be more accurate in predicting execution time, particularly for small-scale applications. By distinguishing between calculation and communication times, we have successfully reduced the average value of α by a factor of 2, indicating a more parallelizable nature of the calculation phase. While the overall error rate remains relatively stable, the error on small applications has seen a significant reduction. This improvement is crucial, as it allows us to more accurately evaluate the optimal number of processes for different application scales, leading to better resource allocation and more efficient execution. The integration of communication metrics from miniGhost has been instrumental in achieving these improvements, highlighting the importance of considering communication overhead when modeling parallel applications.

The following figures illustrate the results on some application whith high error rate.

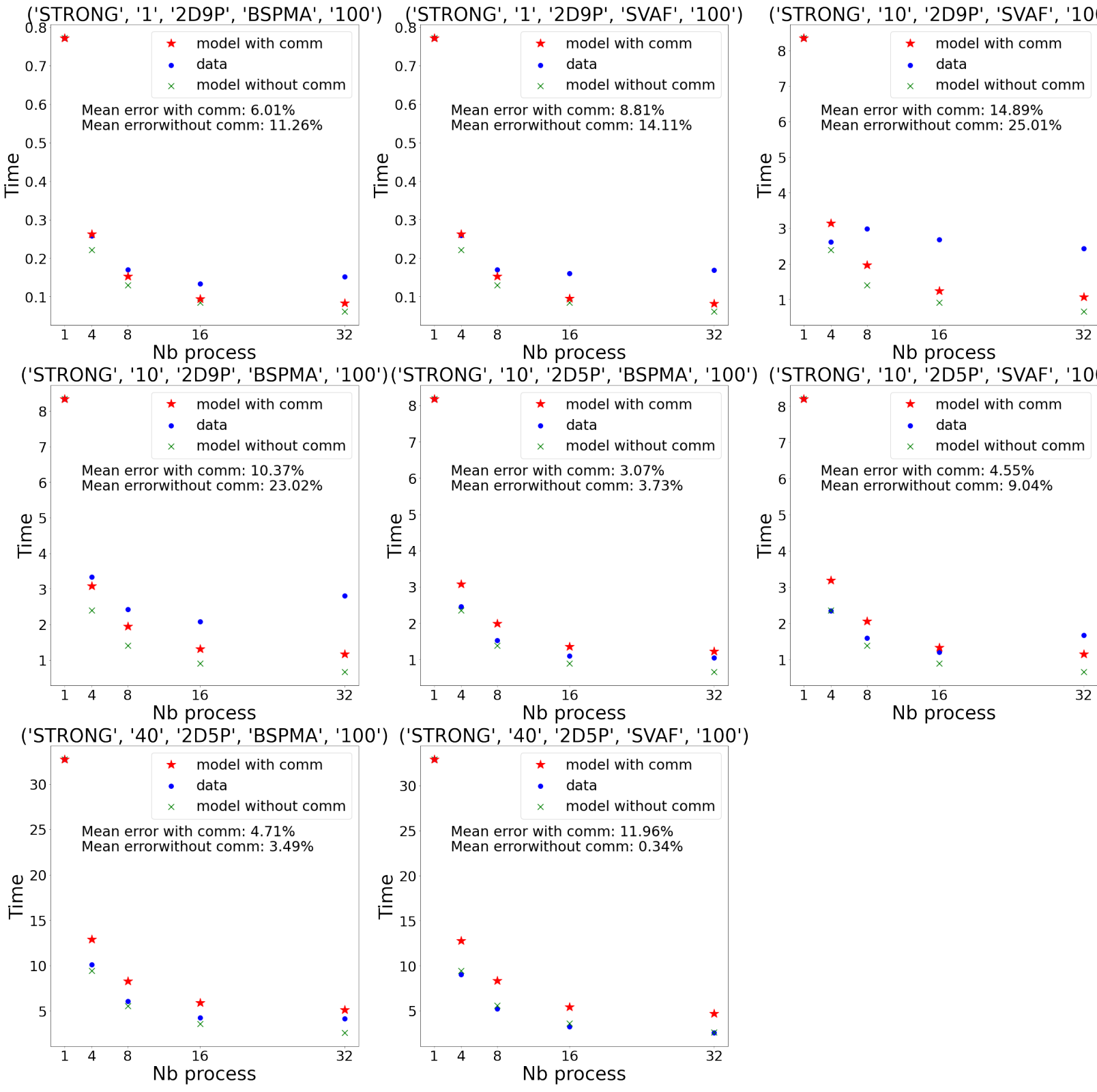

Figure 5: time model

5.2.2 Power Consumption Model with Communication Metrics

In this section, we extend the power model to take into account the effects of communication overhead on power. Communication between processes in parallel applications can significantly impact the power of an application, as the power requirements of communication can be different from computation. In the previous model, we assumed that all processes have a uniform energy consumption, regardless of their activity. However, this assumption does not hold when considering communication overhead, as the energy requirements of a process that is communicating may differ from a process that is performing computations.

To account for this, we introduce a new model that distinguishes between the power used by process during communication and computation. The model is based on the idea that a process consumes energy at different rates depending on whether it is communicating or computing. Specifically, we model the power consumption of an application as the sum of two components: the power used by processes during communication and the power used by processes during computation. This model is described by the formula:

$$P(p) = \alpha_1 \cdot p \cdot \text{comm\_proportion} + \alpha_2 \cdot p \cdot (1 - \text{comm\_proportion}) + \beta$$

where $P(p)$ is the power of the application with p processes, comm_proportion is the proportion of the total time that the processes spend in communication, α_1 and α_2 are coefficients representing the energy consumption rates of processes during communication and computation, respectively, and β is a constant term representing the base power of the system. We use this model to fit the energy consumption data from our experiments and analyze the impact of communication overhead on the energy efficiency of parallel applications.

The updated power consumption model incorporating communication metrics has been evaluated using experimental data. Unfortunately, the results indicate that the modifications do not lead to a substantial improvement in the accuracy of power consumption predictions compared to the previous model. The mean error rate remains largely unchanged, suggesting that the inclusion of communication metrics does not offer a meaningful improvement in predictive power.

Moreover, the new model introduces a greater risk of overfitting due to the increased number of variables in the formula. Overfitting occurs when a model becomes too complex and begins to fit the noise in the data rather than the underlying trend. As a result, overfitted models may perform well on the training data but are less likely to generalize well to new, unseen data. In this case, the addition of extra variables for the power consumed by processes during communication and computation increases the complexity of the model, making it more susceptible to overfitting.

Overall, the results suggest that while communication overhead may have an impact on the mean power used by the application, its contribution is not significant enough to warrant the additional complexity of the new model. It may be more beneficial to focus on other factors that have a greater influence on power consumption or to explore alternative modeling techniques that are less prone to overfitting.

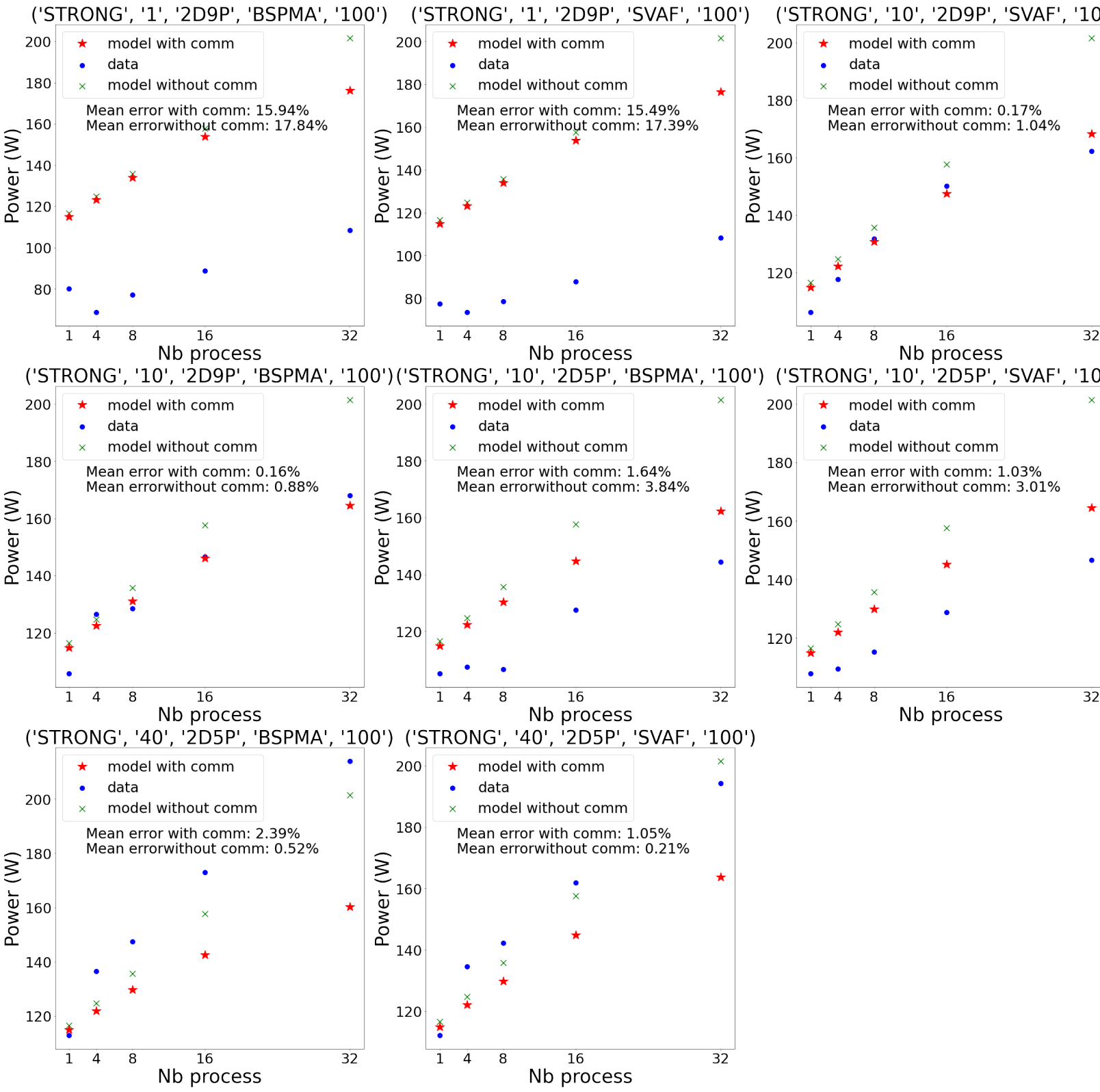

Figure 6: power model

5.3 Combined Energy Model Results

The energy model, which combines the time and power models, has been evaluated both with and without the inclusion of communication metrics. When the communication metrics were not included, the model produced decent results, often correctly predicting the optimal number of processes needed to minimize energy consumption. However, the model without communication metrics may not generalize well to other types of applications or scenarios.

On the other hand, incorporating communication metrics into the energy model improves the mean error of the predicted values, indicating a better overall fit to the observed data. However, this model is slightly less successful at predicting the optimal number of processes for energy efficiency. Notably, even when the model does not predict the ideal process count, the difference in energy consumption between the predicted and optimal configurations is typically minimal and may be attributed to system noise.

In conclusion, the energy model with communication metrics provides more accurate predictions of energy consumption overall and is more likely to be adaptable to a wider variety of applications. Although it is slightly less precise in predicting the optimal process count, the small differences in energy consumption are not likely to be practically significant and could be influenced by factors beyond the model's control. Therefore, the energy model with communication metrics is the more robust and reliable option for predicting energy consumption and optimizing process distribution.

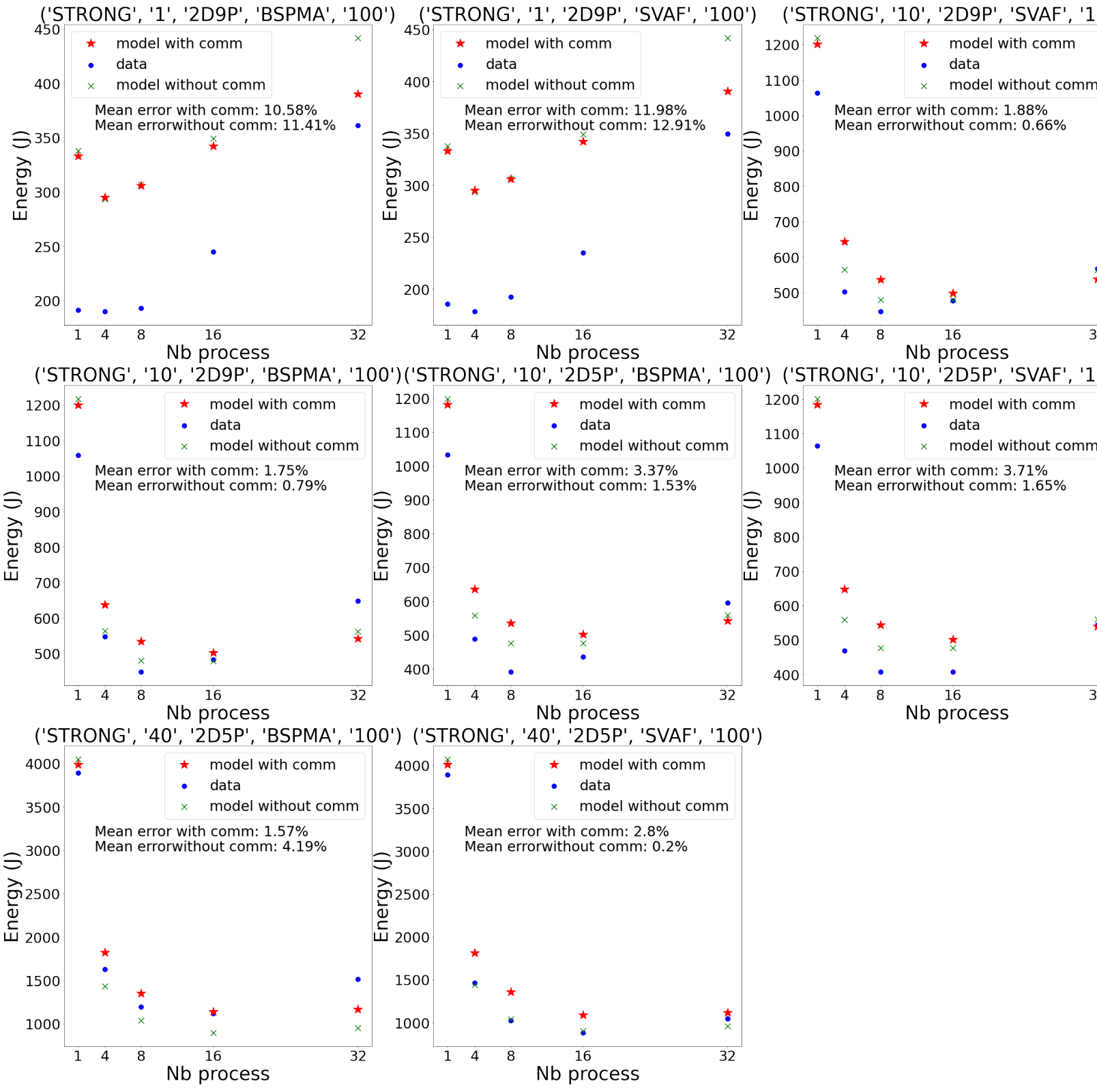

Figure 7: energy model

5.4 Communication Model

During the process of developing a communication model, it became evident that predicting communication time accurately is a complex and multifaceted task. Multiple models were constructed and evaluated, but none were found to provide satisfactory predictions of communication time.

Several models that fit well to the data were identified, but upon closer examination, the calculated parameters for these models were nonsensical and appeared to result from overfitting. In an effort to develop a model that had a more logical basis, the number of messages sent was incorporated as a predictor. Unfortunately, this approach also failed to yield satisfactory results, with the size of the grid in the x dimension, a parameter that have not influence on the number of messages sent, having a significant impact on the total communication time.

Considering the challenges encountered during the modeling process and the influence of seemingly unrelated parameters, it seems that communication time may be influenced more by the probability of a process needing to wait for data, rather than the sheer number of messages exchanged. This complexity makes accurate prediction difficult with the current available information and modeling approaches.

Further exploration of the factors affecting communication time, including considerations of network congestion, priority of messages, and the internal workings of the communication library, may be necessary to develop a reliable and meaningful model. For the time being, the communication model remains a challenging problem in need of further research and analysis.

6 Conclusion

In the quest for optimal performance in high-performance computing (HPC) systems, the assumption often made is that optimizing for performance will also naturally lead to energy efficiency. However, our study reveals a more nuanced reality. In certain scenarios, particularly in small-scale applications, optimizing solely for performance can result in disproportionately high energy consumption. This highlights the need for a more holistic approach to HPC optimization that takes into account both performance and energy efficiency.

A key factor that influences the relationship between performance and energy efficiency is the communication overhead incurred by processes. Our experiments with the MiniGhost application underscore the significance of communication overhead in HPC systems. As the number of processes increases, so does the need for inter-process communication. This communication can introduce substantial time overheads, which in turn can impact energy efficiency. By analyzing the communication patterns and overheads in HPC applications, we can gain valuable insights into the trade-offs between performance and energy consumption.

In conclusion, our study underscores the importance of a comprehensive approach to HPC optimization that considers both performance and energy efficiency. By taking into account factors such as communication overhead, we can better understand the intricate relationships between various HPC metrics and make more informed decisions in the quest for optimal HPC performance and sustainability.

References

[1] “Optimizing performance and energy across problem sizes through a search space exploration and machine learning”. In: (). DOI: https://doi.org/10.1016/j.jpdc.2023.104720.

[2] "Combining Thread Throttling and Mapping to Optimize the EDP of Parallel Applications". In: (). DOI: 10.1109/PDP52278.2021.00035.

[3] "Multicore processor computing is not energy proportional: An opportunity for bi-objective optimization for energy and performance". In: (). DOI: https://doi.org/10.1016/j.apenergy.2020.114957.

[4] "Process mapping on any topology with TopoMatch". In: (). DOI: https://doi.org/10.1016/j.jpdc.2022.08.002.