Energy and Performance Analysis of Parallel Particle Solvers from the ScaFaCoS Library

Michael Hofmann, Robert Kiesel, Gudula Rünger
Department of Computer Science, Chemnitz University of Technology
{michael.hofmann,robert.kiesel,ruenger}@cs.tu-chemnitz.de

ABSTRACT

Performance analysis in high performance computing (HPC) has traditionally focused on improving application programs, for example, by decreasing the overall runtime or increasing the throughput of floating point operations. However, the same approaches might also be used to influence the energy behavior. Since the increasing energy consumption of HPC platforms is gaining more and more attention, the identification of applications and platforms for which energy and performance measurement lead to differing results is of great importance. In this article, we analyze the energy and performance behavior of particle solvers from the ScaFaCoS library. Four different criteria are investigated with respect to their influence on the energy consumption and achieved performance. These criteria are the solution method chosen, the parameters of the solution method, the degree of parallelism, and the parameters of the hardware platform.

KEYWORDS

particle simulations; energy consumption; performance analysis

ACM Reference Format:
Michael Hofmann, Robert Kiesel, Gudula Rünger. 2018. Energy and Performance Analysis of Parallel Particle Solvers from the ScaFaCoS Library. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering, April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184407.3184409

1 INTRODUCTION

The ever increasing usage of computer systems in almost all areas of modern life goes along with increasing operational costs. An important part of these costs is caused by the energy requirements, both directly as a consumable resource as well as indirectly, for example, for power supply infrastructures or cooling facilities. Besides hardware properties, also software-based decisions such as the utilized algorithms and their parameters [12], the exploited degree of parallelism [18], and adjustable hardware parameters such as the processor frequency [10] or the mapping of workloads to processors [16] have an influence on the energy consumption. However, these decisions also affect the achievable performance of applications and their implementation. A joint analysis of energy and performance characteristics is therefore highly required.

Investigations of energy requirements on high performance computing (HPC) platforms are often performed with regard to applications in scientific computing [7]. For these kinds of applications, it is usually more important to determine the solution of a specific problem instead of utilizing a specific hardware or software. This allows for various optimizations where time-to-solution and energy-to-solution can be seen as competing goals. Particle simulations are widely used in different areas of computational scientific, such as biology, chemistry, and astrophysics. A major computational part of the simulations is spent for determining long-range particle interactions such as Coulomb or gravitational interactions. Several efficient methods with individual parameters exist for the computation of these interactions, thus providing various opportunities for investigating their energy and performance behavior.

In this article, we analyze the energy consumption and the performance of different parallel particle solvers from the ScaFaCoS library[1]. The library contains, for example, hierarchical methods such as the Fast Multipole Method (FMM) and the Barnes-Hut algorithm or mesh-based methods such as Particle-Particle-Particle-Mesh (P^3M) and Particle-Particle NFFT (P^2NFFT). All solvers are utilized through a common programming interface and compute the same particle interactions in parallel. Thus, the library allows to solve a single problem with several methods that might expose different energy and performance behaviors. Four different aspects are investigated to analyze the behavior of the particle solvers:

I. varying the utilized solver method,
II. varying parameters of the solver method,
III. varying the degree of parallelism, and
IV. varying the hardware platform.

The energy consumption and the performance of the solvers are determined experimentally using particle systems of different size and hardware platforms with different processor microarchitectures. The analyses investigate whether there are situations in which the energy behavior deviates from the performance behavior. Identifying such situations is a necessary requirement for improvements where either an optimal (i. e., lowest) energy consumption or an optimal (i. e., highest) performance is preferred.

The rest of this article is organized as follows: Section 2 introduces the ScaFaCoS library and the different variation approaches. Section 3 describes the hardware and software environment and the particle solvers. Section 4 presents the experimental results. Section 5 discusses related work and Section 6 concludes the article.

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5095-2/18/04…$15.00
https://doi.org/10.1145/3184407.3184409

[1] http://www.scafacos.de

2 ENERGY AND PERFORMANCE INFLUENCES ON PARTICLE SOLVERS

Particle solvers for Coulomb interactions are compute and data intensive program parts of particle simulations. In the following, the ScaFaCoS library with different solver methods and the different variation approaches for influencing the energy and performance behavior of the computations are described.

2.1 Scalable Fast Coulomb Solvers (ScaFaCoS)

The ScaFaCoS library is a numerical software library that contains various methods for the calculation of long-range particle interactions, i. e. Coulomb or gravitational interactions. The provided solver methods include tree-based methods, such as the Fast Multipole Method (FMM) [11] or the Barnes-Hut algorithm [5], as well as mesh-based methods, such as Particle-Particle-Particle-Mesh (P^3M) [3] or fast summations based on nonequispaced fast Fourier transforms (P^2NFFT) [15]. Parallelization is implemented using the Message Passing Interface (MPI). The library provides a common programming interface for invoking the different solver methods. Thus, alternative solver methods can be employed without additional programming efforts. However, additional solver-specific parameters can be set individually for each solver method. Additional information about the solver methods of the library and a comparison of their capabilities is presented in [4].

After building the ScaFaCoS library, the utilization within an application program proceeds in the following steps:

- Initialization of a solver method.
- Setting of particle system properties.
- Setting of solver-specific parameters.
- Tuning of the solver method.
- Computation of particle interactions.
- Termination of the solver method.

These steps have to be executed in parallel by all participating MPI processes. The input data describing the particle system consists of the three-dimensional position and the charge value of each particle. The user of the library is responsible for distributing the particle data initially among the processes such that each process contributes its local share of the overall particle system. Computing the particle interactions represents the step with the main computational effort. This step might also be executed repeatedly, for example, with slightly changing particle positions in a particle dynamics simulation.

2.2 Variation approaches for energy and performance behaviors

Applications in scientific computing are usually focused on the solution of a specific problem (e. g., with numerical simulations) instead of utilizing a specific hardware platform or software implementation. This might allow to vary several aspects of the calculations and computations as long as the determined results are the acceptable, for example, in terms of correctness or accuracy. To analyze the energy and performance behavior of the particle solver methods, the following four variation approaches are investigated:

I. Variation of the solver method: For many computational problems, there exist several alternative approaches or algorithms. Their different program codes lead to a different utilization of the hardware platform, for example, due to the employed operations or memory access patterns. Thus, depending on the executed program code, individual energy and performance behaviors might be expose. The ScaFaCoS library contains several alternative algorithms for computing Coulomb interactions of particle systems. All solver methods are able to compute the same results, even though they might differ in terms of the achieved accuracy. However, in general it is possible to choose the specific solver method as part of an optimization towards energy or performance.

II. Variation of solver parameters: Algorithms and implementations usually have parameters that control their computational behavior. The resulting different kinds of computations lead to the execution of different parts of the program codes which might expose different performance and energy behaviors. Important parameters of the solver methods of the ScaFaCoS library are, for example, the tree depths for the tree-based methods or the mesh sizes for the for mesh-based methods. These parameters determine the separation of the overall computations into near-field computations (i. e., direct computations of pairwise particle interactions) and far-field computation (i. e., approximations of interactions of groups of particles). Since both kinds of computations represent different parts of the program codes, shifting the computational load between them can be adapted as part of an optimization towards energy or performance.

III. Variation of the degree of parallelism: The degree of parallelism of a parallel program code is controlled by specifying its number of processes or threads. Increasing the parallelism can lead to lower runtimes and a higher computational performance. Decreasing the parallelism can lead to idle compute units, such as cores of multi-core processors or processors of multi-processor systems. Placing these units in a standby mode might reduce the energy consumption. All solver methods of the ScaFaCoS library are parallelized with MPI. Thus, the number of utilized compute units can be adapted as part of an optimization towards energy or performance by specifying the number of MPI processes.

IV. Variation of the hardware platform: The utilized hardware platform has a strong influence on the resulting energy and performance behaviors. Furthermore, modern hardware platforms allow to control parameters, such as the frequency of processor cores, which directly influence the power consumption. Especially, for data-intensive applications, whose computational performance is limited by memory accesses or data exchanges instead of the amount of operations, it might be advantageous to operate with a lower processor core frequency. Even though this can lead to a higher runtime, a lower energy consumption can be achieved if the reduced power consumption outweighs the runtime increase. Computing particle interaction with the ScaFaCoS library is a data-intensive part of particle simulations and, thus, it can be beneficial to adapt the hardware platform parameters as part of an optimization towards energy or performance.

3 EXPERIMENTAL SETUP

The variation approaches described in the previous section are used to analyze the energy and performance behavior of the ScaFaCoS library. In the following, the solver methods and their parameters as well as the hardware and software environment are described.

3.1 Particle solver methods

The energy and performance measurements are performed with a generic test program that can invoke any solver method of the ScaFaCoS library. The test program is part of the library package and was mainly developed for the verification and comparison of the different solver methods. The measurements include only the computation of the particle interactions while all other library functions (e. g., initialization, parameter setup, tuning) are not considered. A benchmark particle system (called cloud-wall) is used for the experimental analysis. The three-dimensional base system contains 300 particles, whereas 100 particles are distributed randomly (cloud) and 200 particles are distributed on a regular grid (wall). Larger systems with $300\times8, \ldots, 300\times8^5$ particles are created by repeating the base system several times in each dimension. All parameters of the solver methods either have default values or are tuned automatically. Unless otherwise stated, these values are used.

The experimental analysis uses the following solver methods:

Direct: The Direct solver computes the particle interactions by a direct summation of the contributions of all pairs of particles. With n particles, the method has a runtime of $O(n^2)$. Periodic boundary conditions are computed by placing a layer of copies of the given particles around the original particle system. Thus, the method is only appropriate for small particle systems. The method has no parameters that can influence the behavior of the computations.

Ewald: The Ewald solver computes the particle interactions using the Ewald summation [3]. The approach separates the contributions of the particle interactions into two parts: The k-space part (i. e., far-field computations) is calculated with a reciprocal lattice. The real-space part (i. e., near-field computations) is calculated directly between particles within a given cutoff range. Both the maximum number of k-space vectors and the cutoff range are parameters of the method that influence the computational demands and the accuracy of the results.

P^{3}M: The P^{3}M solver is a parallel implementation of the Particle-Particle-Particle-Mesh algorithm [3]. The algorithm uses a grid-based approach to accelerate the time-consuming k-space part of the Ewald summation using Fast Fourier Transforms (FFT). Similar to the Ewald solver, the computational demands of the far-field computations depend on the size of the FFT grid and the computational demands of the near-field computations depend on the cutoff range. Both can be controlled by parameters.

P^{2}NFFT: The Particle-Particle NFFT (P^{2}NFFT) solver performs fast Ewald summations based on nonequispaced fast Fourier transforms (NFFT) [15]. The P^{2}NFFT solver represents a general framework for particle mesh algorithms and supports periodic and nonperiodic boundary conditions. Similar to the Ewald solver and the P^{3}M solver, the computational demands of the far-field and near-field computations can be controlled by parameters that specify the size of the FFT grid and the cutoff range.

FMM: The FMM solver is a parallel implementation of the Fast Multipole Method [11]. This tree-based algorithm uses a recursive subdivision of the particle system into smaller boxes. Contributions from interactions between particles inside each box and between neighboring boxes belong to the near-field and are calculated directly. All other contributions belong to the far-field and are approximated with multipole expansions. Using these expansions allows for a hierarchical grouping of the contributions and enables efficient operations for calculating interactions between entire boxes of particles at once. The level up to which the subdivision into boxes is performed can be set by a parameter that controls the computational costs of the near-field and far-field computations.

Table 1: Overview of the utilized hardware platforms.

Name	Sandybridge	Haswell	Skylake
Processor	Xeon E5-2650	Xeon E5-2683 v3	Core i7-6700
Cores	2×8	2×14	1×4
Frequency	1.2–2.0 GHz	1.2–2.0 GHz	0.8–3.4 GHz
L3 cache	20 MB	35 MB	8 MB
Memory	32 GB	128 GB	16 GB

3.2 Hardware platforms

The experimental analysis uses three hardware platforms with different processor microarchitectures as shown in Table 1. The platforms comprise of two Intel Xeon server systems and one Intel Core desktop system. The server systems have dual sockets, a high number of cores, and large L3 caches, but only a narrow range of frequencies with a low maximum frequency. In contrast, the desktop system has only a single socket, a low number of cores, and a smaller L3 cache. However, the range of processor frequencies is wider and includes a significantly higher maximum frequency.

3.3 Software environment

The utilized hardware platforms use the Debian GNU/Linux 8.6 operating system with kernel version 4.7. The processor frequencies are controlled manually through the CPUFreq kernel interface using the userspace governor. Energy measurements are performed with the Running Average Power Limit (RAPL) interface by accessing model specific registers of the processor. Instead of reading the appropriate results directly from the registers, the Performance Application Programming Interface (PAPI)[2] is used. A dedicated RAPL component allows for transparent power and energy readings for Intel Sandy Bridge processors and its successors [23]. The PAPI library of version 5.5.1 was used for the measurements.

[2] http://icl.utk.edu/papi

3.4 Measurement methodology

The measurements were conducted by performing *multiple consecutive trials* [1]. For a specific configuration of the settings and parameters to be varied, ten consecutive trials are executed and measured. This process is repeated for the trials of the next configurations to be investigated. Average values of the consecutive trials are used for the analyses. The hardware platforms were used exclusively, i. e. no other users or processes except from the operating system have utilized the platform at the same time.

The PAPI library provides a uniform interface for starting, stopping, and querying performance-relevant measurements for the execution of program codes at runtime [6]. Measuring the energy consumption of the processors is performed using the hardware counters `rapl:::PACKAGE_ENERGY:PACKAGE0` (first socket) and `rapl:::PACKAGE_ENERGY:PACKAGE1` (second socket if present). The program code for the measurements is integrated into the library function that performs the step for computing the particle interactions (see Sect. 2.1). However, since this library function is executed in parallel with MPI, only a single process per compute node performs the measurements and determines the energy consumption of all processors of the compute node. The overall execution of the library and its solver methods is performed with a generic test program that is part of the ScaFaCoS library.

4 ENERGY AND PERFORMANCE ANALYSES

The energy and performance behavior of the different solvers of the ScaFaCoS library is analyzed experimentally as described in the previous section. In the following, the results of the four variation approaches from Sect. 2.2 are shown and the findings are discussed.

4.1 Variation of the solver method

To investigate the influence of the specific particle solver method, the five methods Direct, Ewald, P^3M, P^2NFFT, and FMM are compared. Figure 1 shows the runtime (left), the energy consumption (middle), and the power consumption (right) depending on the number of particles for the different methods. All methods are executed sequentially on the Haswell platform. The runtimes of all solvers increase strongly for increasing numbers of particles. However, especially the Direct method and the Ewald method show a very steep increase and their runtimes are even for the smallest number of particles at least one order of magnitude higher than the other methods. Both methods represent reference methods which are not designed for efficient computations and thus, results with larger numbers of particles are omitted. The fast methods P^3M, P^2NFFT, and FMM show a very similar runtime with the FMM method being slightly slower and the P^3M being slightly faster. The runtime of the P^2NFFT method varies between them.

The behavior of the energy consumption of all methods corresponds to their runtimes. This indicates that the operations performed by the different methods lead to almost the same utilization of the hardware. Even though the methods are based on different mathematical and computational approaches, neither of them provides a significant increase or reduction of the energy consumption during the course of the computations. However, the power consumption shows a more different behavior. All methods are within a range of about $58.5\,\mathrm{J\,s^{-1}} \pm 5\,\%$, but the lowest power consumption is now achieved with the P^2NFFT method. Furthermore, also the two reference methods Direct and Ewald are able to achieve a lower power consumption than the two fast methods P^3M and FMM. However, in general these differences occur mainly for short computations with small numbers of particles, while during longer running computations the power consumption of all methods approaches similar values of $58\,\mathrm{J\,s^{-1}}$ to $59\,\mathrm{J\,s^{-1}}$.

4.2 Variation of solver parameters

To investigate the influence of method parameters, the separation of the computations into near-field computations and far-field computation is varied for the methods P^2NFFT and FMM. For the P^2NFFT method, an increase of the far-field computations is achieved by increasing the grid size parameter. At the same time, the near-field computations are reduced by reducing the utilized cutoff radius as far possible while still achieving a specific accuracy of the results. For the FMM method, the far-field computations are increased and the near-field computations are decreased by using a larger tree depth for the subdivision into boxes. The methods are executed sequentially on the Haswell platform using particle systems of size $300{\times}8^2$ and $300{\times}8^5$. Total results as well as separate results for the near-field and far-field computations of the methods are shown.

Figure 2 shows the sequential runtime (left) and the energy consumption (right) of the P^2NFFT method depending on the grid size. It can be observed that the grid size has a significant influence on the runtime of the P^2NFFT method. This behavior is caused by the near-field and far-field computations whose runtimes behave as expected. The minimum of the total runtime is achieved with a grid size of 32 for $300{\times}8^2$ particles and with a grid size of 448 for $300{\times}8^5$ particles. The energy consumption reflects the general behavior observed for the runtime. For $300{\times}8^2$ particles, the minimum of the total energy is achieved with the same grid size that also leads to the minimum of the total runtime (i. e., 32). Choosing slightly lower or higher grid sizes leads to an increase of both the runtime and the energy consumption. For $300{\times}8^5$ particles, the minimum of the total energy is achieved with a grid size of 384. The corresponding runtime is about 5 % higher than the runtime minimum. However, using the optimal grid size of the runtime (i. e., 384) leads to an increase of the energy consumption of only about 1 %. Even though the potential savings in runtime or energy are relatively small, the results indicate a difference between the energy and runtime requirements of the two computational parts of the P^2NFFT method.

Figure 3 shows the sequential runtime (left) and the energy consumption (right) of the FMM method depending on the maximum tree depth. It can be observed that the tree depth has a significant influence on the runtime of the FMM method. However, the parameter controls only the maximum tree depth while the actual tree depth is determined automatically by the specific FMM solver [9]. Thus, using a larger maximum tree depth (i. e., larger than 3 for $300{\times}8^2$ particles and larger than 6 for $300{\times}8^5$ particles) do not lead to further differences. The minimum of the total runtime is achieved when the runtimes of the near-field and far-field computations are about the same. The same behavior is observed for the energy consumption. The minimum of the total runtime and the total energy is achieved with the same maximum tree depth.

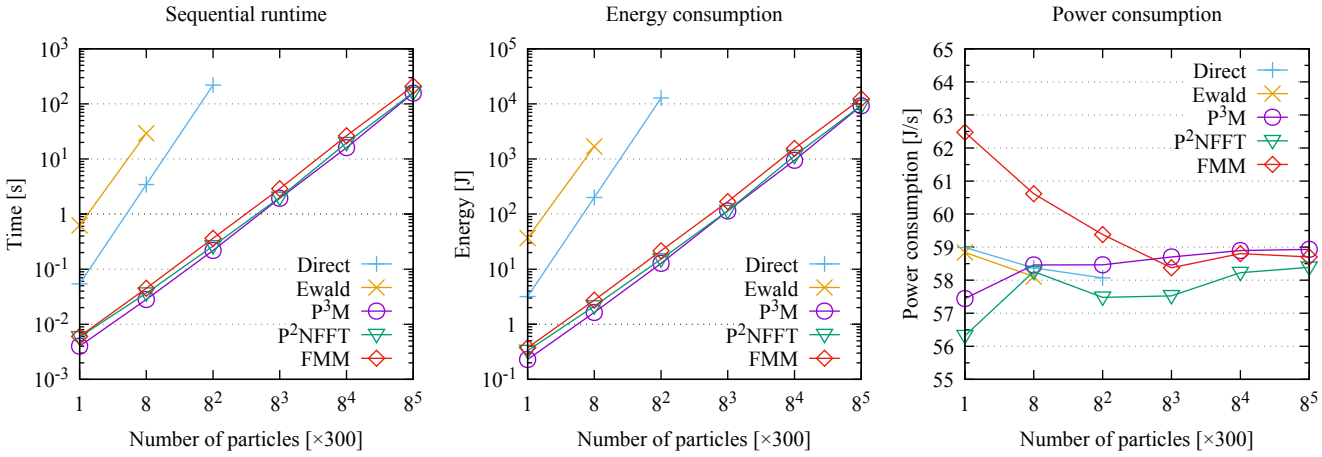

Figure 1: Sequential runtime (left), energy consumption (middle), and power consumption (right) depending on the number of particles for different particle solver methods.

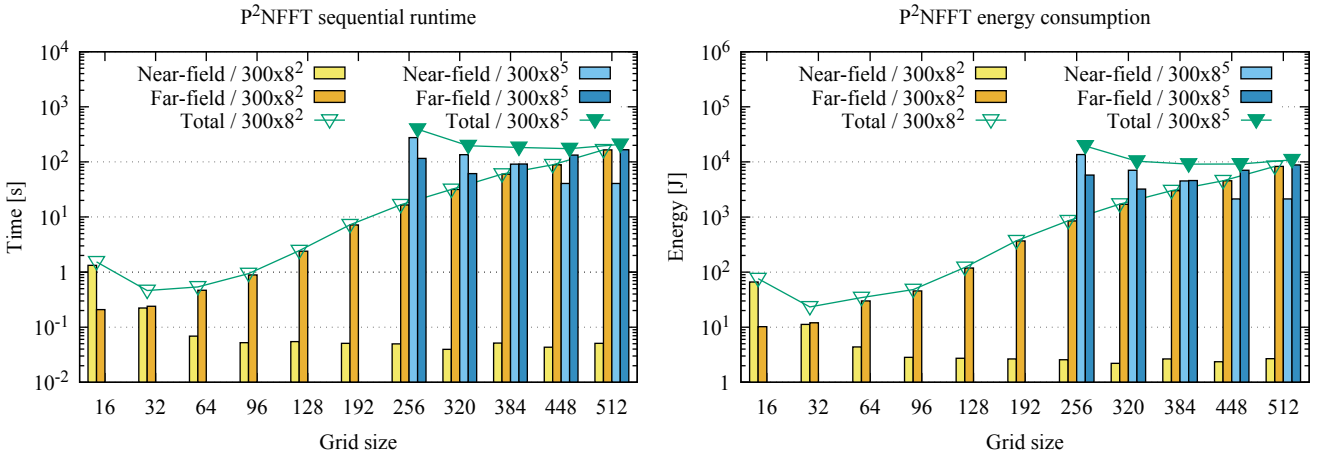

Figure 2: Sequential runtime (left) and energy consumption (right) of the P^{2}NFFT method depending on the grid size.

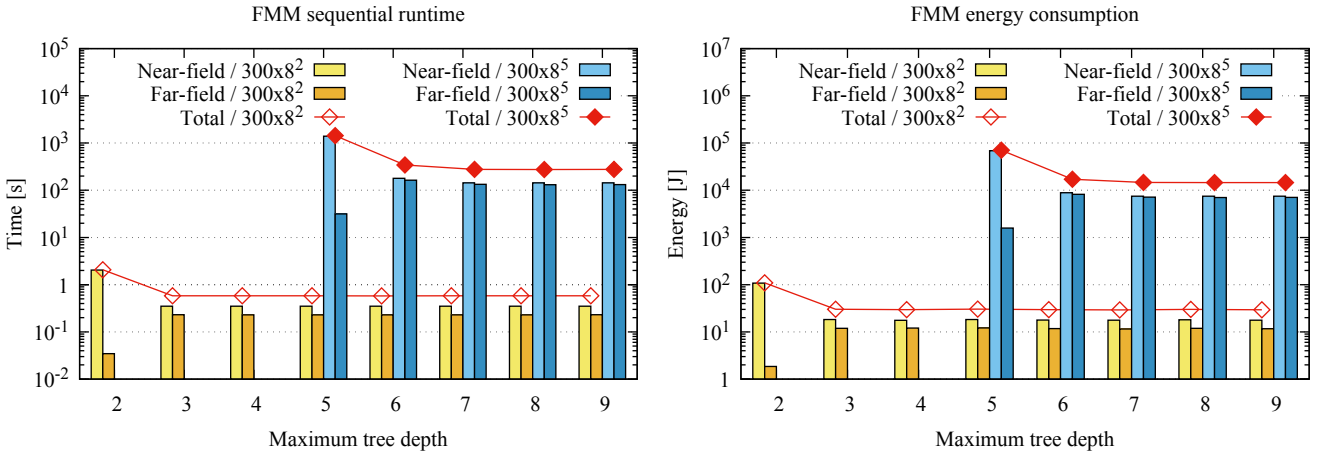

Figure 3: Sequential runtime (left) and energy consumption (right) of the FMM method depending on the maximum tree depth.

4.3 Variation of the degree of parallelism

To investigate the parallel behavior of the particle solver methods, the degree of parallelism represented by the number of utilized MPI processes is varied. The two methods P^{2}NFFT and FMM are considered. The methods are executed on the Haswell platform using particle systems of size 300×8^2 and 300×8^5.

Figure 4 shows the parallel runtime (left), the parallel speedup (middle), and the energy consumption (right) depending on the number of MPI processes. The results show that the parallel runtimes of all solver methods decrease for increasing numbers of MPI processes. However, for 300×8^2 particles, this runtime improvement deteriorates when using more than 14 MPI processes (i. e., more than the number of cores of one processor). For the larger number of particles, the runtime improvement becomes less significant when using more than 28 MPI processes (i. e., more than the total number of physical cores of both processors). Nevertheless, the P^{2}NFFT method still achieves a reduction of the parallel runtime when using the Hyper-Threading capabilities of processors (i. e., with 56 MPI processes). The runtime results are reflected by achieved parallel speedups. While with the smaller number of particles, a maximum speedup of about 8 is achieved, the larger number of particles lead to a maximum speedup close to the total number of physical cores available. Only for the P^{2}NFFT method, a further increase of the parallel speedup is achieved using Hyper-Threading.

The energy consumption corresponds to the general runtime behavior. However, there are quantitative differences, especially for high numbers of MPI processes. When using Hyper-Threading for the P^{2}NFFT method, the resulting reduction of the energy consumption of about 6 % to 7 % is smaller than the runtime reduction of about 14 % to 34 % occurring at the same time. Furthermore, for larger numbers of MPI processes (i. e., 28 and 56), the parallel runtime of the FMM method increases only slightly while the increase of the energy consumption is stronger at the same time. Nevertheless, these results do not indicate significant differences between optimal energy and performance behaviors. The number of MPI processes with the smallest parallel runtime leads also to the smallest energy consumption for the considered measurements.

4.4 Variation of the hardware platform

To investigate the influence of parameters of the hardware platform on the energy and performance behavior, the utilized hardware platform as well as the processor frequency is varied. The two methods P^{2}NFFT and FMM and the particle system of size 300×8^5 are considered. All methods are executed by using one MPI process per physical processor core of the specific hardware platform, i. e., 16 MPI processes for Sandybridge, 28 MPI processes for Haswell, and 4 MPI processes for Skylake (see Table 1).

Figure 5 shows the parallel runtime (left) and the energy consumption (right) depending on the processor frequency. The results show that the processor frequency has a significant influence on the parallel runtime of the solver methods. For the two server platforms (i. e., Sandybridge and Haswell), the rather small range of supported processor frequencies of 1.2 GHz to 2.0 GHz leads to a reduction of the parallel runtime by a factor of about 1.6. For the Skylake desktop platform, the larger range of 0.8 GHz to 3.4 GHz leads to a runtime reduction by a factor of about 4. This corresponds to an almost linear improvement for both solver methods, thus showing that their performance behavior is mainly determined by the computational speed of the processors. Furthermore, even for equal processor frequencies, there is a strong difference between the hardware platforms resulting from the different numbers of processor cores available. The minimum of the parallel runtime is achieved by choosing the maximum processor frequency on each platform as well as the Haswell platform among the three available platforms.

The energy consumption shows a very different behavior than the parallel runtime. For all measurements, there occurs a U-shape where the energy consumption first decreases and later increases for increasing processor frequencies. Thus, the minimum of the energy consumption is achieved by choosing an intermediate processor frequency of about 1.6 GHz to 1.7 GHz. The optimal choice of the processor frequency is almost the same for the two solver methods and the three hardware platforms. In comparison to using the maximum frequency (i. e., with the minimum parallel runtime), the optimal frequency with respect to the energy consumption leads to an increase of the parallel runtime of about 22 % to 23 % for the Sandybridge and Haswell platform and 90 % to 98 % for the Skylake platform. At the same time, the achieved reduction of the energy consumption is only up to 6 % for the Sandybridge and Haswell platform but up to 46 % for the Skylake platform. The results demonstrate that optimizations towards energy or performance require differing approaches. Even though the loss of performance observed is larger than the energy improvement obtained, a significant reduction of the energy consumption can still be achieved especially for platforms with large ranges of processor frequencies.

4.5 Discussion

The measurements indicate that all variation approaches investigated in the previous subsections have a significant influence on the energy and performance behavior. However, varying the solution methods or their parameters often requires additional programming efforts and a deep knowledge of the specific solution methods. The common programming interface of the ScaFaCoS library and the parameterized implementations of their solvers lower the required efforts for these kinds of investigations significantly, thus demonstrating that this library design is very advantageous for subsequent analyses and optimization approaches. Varying the degree of parallelism or the processor frequency is already common practice and can be performed without significant efforts. However, the findings have shown that especially the achievable reduction of the energy consumption with an optimal processor frequency depends strongly on the range of frequencies supported by the processor. The desktop platform was much more affected by this effect, thus showing that the priorities of the analyses might have to incorporate, for example, differences between desktop and server platforms or the direction of future hardware platform developments.

So far, the analysis has considered only selected configurations of particle solver methods, parameters, and input data. However, an investigation encompassing the entire space of possible configurations leads to large amounts of measurement results that would be very time consuming to obtain and to analyze. This would require a generalization of the approach, for example, with consecutive iterations parameter spaces and automated detection of

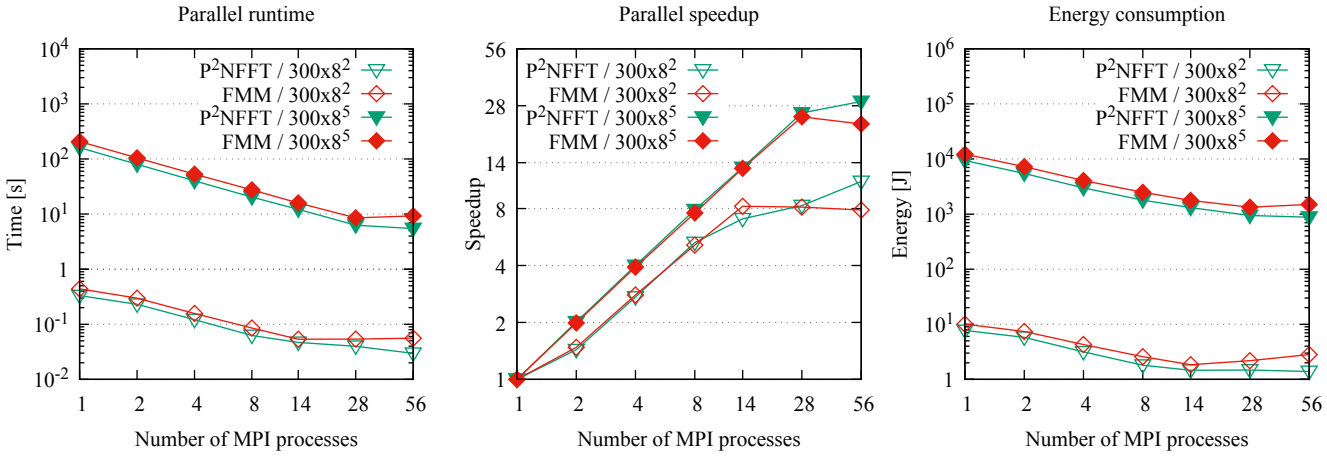

Figure 4: Parallel runtime (left), parallel speedup (middle), and energy consumption (right) depending on the number of MPI processes for different particle solver methods.

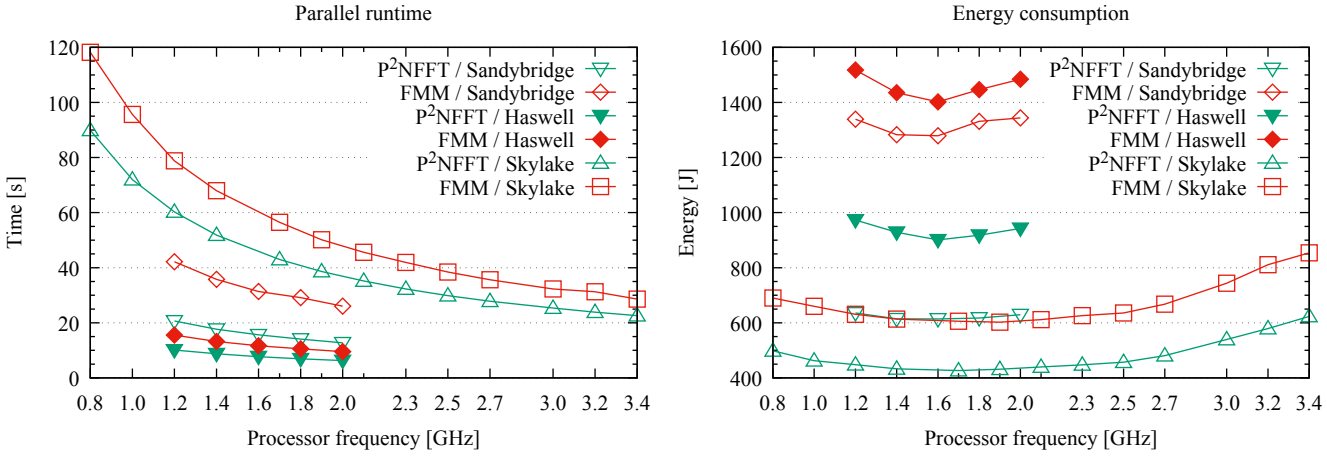

Figure 5: Parallel runtime (left) and energy consumption (right) depending on the processor frequency for different particle solver methods and hardware platforms.

optimal energy and performance behaviors. The analysis based on the four considered variation approaches is independent from the considered library of particle solver methods and can be applied to other libraries as well. However, single investigations might not be applicable to all kinds of libraries, for example, a variation of the solution method can only be exploited if there are several alternative methods available for solving a single problem. Furthermore, the overall approach is not limit to software libraries at all, but can be generalized to other computational problems where different algorithms or implementations are available.

5 RELATED WORK

In the past, studies about the energy and power consumption in scientific computing have focused on linear algebra operations [21] as well as on general benchmarks, such as NPB and PARSEC [22]. However, only few investigate specifically the differences between energy/power and performance [2, 13]. Commonly analyzed influences on the energy requirements include code optimizations, dynamic concurrency throttling (DCT), and dynamic voltage and frequency scaling (DVFS). The overall results show large variations in the achieved improvements of the energy requirements, thus demonstrating that dedicated analyses for individual applications and their various influences are necessary. This article provides a dedicated analysis of solver methods for particle simulations, which represent an important class of applications that have not been investigated in detail so far. Furthermore, we have extended the analysis to include novel influences, such as the usage of alternative solution methods and the selection of method-specific parameters.

For an older version of the FMM solver method, an autotuning approach for its method parameters, such as the utilized tree depth, was developed [9]. The presented results showed the significant influence of the method parameters on the performance of the

FMM method. While the optimization approach was also based on experimental measurements, it did not consider the parallelization or the energy consumption. An analysis of the energy behavior of the FMM with DVFS is presented in [8]. The work focuses on the the prediction of energy-efficient settings using a system-on-chip platform. The results showed that due to a large percentage of energy consumed by constant power of the platform, a maximum frequency setting minimized both the execution time and the energy consumption. For the hardware platforms and measurements presented in this article, the minimum of the energy consumption is achieved with an intermediate processor frequency.

Several approaches were developed to measure, model, and predict the energy behavior of parallel hardware platforms. Various benchmark suites were used to include a representative set of applications codes into the studies. For example, the ASC Sequoia benchmark suite and the multi-zone version of the NAS Parallel Benchmarks were used to analyze the effects of DCT and DVFS in hybrid MPI/OpenMP applications [14]. In [20], the SPEC CPU2006 benchmark suite is used to compare the measurements obtained with external power-meters and with internal hardware counters on multi-core platforms. Benchmark suites, such as SPLASH-2 or PARSEC, also include particle solver methods. These benchmark suites are used, for example, to investigate different power and energy models with respect to their prediction capabilities [19] or to quantify the amount of energy required for shared memory programming of an energy-efficient system-on-chip many-core system [17]. The results presented in these works also include energy measurements of particle solver methods. However, a dedicated analysis of these methods is not included as they represent only a small part of the utilized applications. The analysis provided in this article, focuses solely on particle solver methods and investigates especially the relation between energy and performance behaviors.

6 CONCLUSIONS

In this article, we have analyzed the energy and performance behavior of different particle solver methods from the ScaFaCoS library. Four different aspects were presented to systematically vary the computations used to solve a specific problem. The results showed that the energy consumption and the required runtime often exhibit the same general behavior. Small deviations occurred for the power consumption when varying the particle solver method as well as for the energy consumption when varying the grid size parameter of the P^2NFFT solver method. However, much larger differences between the energy and the parallel behavior were observed for varying the processor frequency. While the minimum of the parallel runtime is always achieved with the maximum processor frequency, the minimum of the energy consumption requires to use significantly smaller processor frequencies. The findings indicate that there can be situations in which the energy behavior deviates from the performance behavior. Due to the large number of potential influences for particle solvers, the presented grouping allows to identify specific aspects that might be used for further analyses or optimizations. Furthermore, the implementation of the particle solvers as a library has demonstrated to be of great advantage for the analyses, because all investigated variations of the solvers could be performed without additional programming efforts.

ACKNOWLEDGMENTS

This work was supported by the German Ministry of Science and Education (BMBF) under Grant No. 01IH16012B.

REFERENCES

[1] A. Abedi and T. Brecht. 2017. Conducting repeatable experiments in highly variable cloud computing environments. In *Int. Conf. on Performance Engineering (ICPE'17)*. ACM, 287–292.
[2] J.I. Aliaga, M. Barreda, M.F. Dolz, and E.S. Quintana-Ortí. 2015. Are our dense linear algebra libraries energy-friendly? *Computer Science-Research and Development* 30, 2 (2015), 187–196.
[3] A. Arnold. 2011. Fourier transformed-based methods for long-range interactions: Ewald, P^3M and more. In *Fast Methods for Long-Range Interactions in Complex Systems*. IAS Series, Vol. 6. Forschungszentrum Jülich, 39–64.
[4] A. Arnold, F. Fahrenberger, C. Holm, O. Lenz, M. Bolten, H. Dachsel, R. Halver, I. Kabadshow, F. Gähler, F. Heber, J. Iseringhausen, M. Hofmann, M. Pippig, D. Potts, and G. Sutmann. 2013. Comparison of scalable fast methods for long-range interactions. *Physical Review E* 88 (2013), 063308. Issue 6.
[5] J. Barnes and P. Hut. 1986. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature* 324, 6096 (1986), 446–449.
[6] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. 2000. A portable programming interface for performance evaluation on modern processors. *Int. J. of High Performance Computing Applications* 14, 3 (2000), 189–204.
[7] J. Carretero, S. Distefano, D. Petcu, D. Pop, T. Rauber, G. Rünger, and D.E. Singh. 2015. Energy-efficient algorithms for ultrascale systems. *Supercomputing Frontiers and Innovations* 2, 2 (2015), 77–104.
[8] J.W. Choi and R.W. Vuduc. 2016. Analyzing the energy efficiency of the fast multipole method using a DVFS-aware energy model. In *Int. Parallel and Distributed Processing Symposium Workshops (IPDPSW'16)*. IEEE, 79–88.
[9] H. Dachsel, M. Hofmann, J. Lang, and G. Rünger. 2012. Automatic tuning of the fast multipole method based on integrated performance prediction. In *Int. Conf. on High Performance Computing and Communication (HPCC'12)*. IEEE, 617–624.
[10] M. Etinski, J. Corbalán, J. Labarta, and M. Valero. 2012. Understanding the future of energy-performance trade-off via DVFS in HPC environments. *J. of Parallel and Distributed Computing* 72, 4 (2012), 579–590.
[11] L. Greengard and V. Rokhlin. 1987. A fast algorithm for particle simulations. *J. of Computational Physics* 73 (1987), 325–348.
[12] T. Jakobs, J. Lang, G. Rünger, and P. Stöcker. 2017. Tuning linear algebra for energy efficiency on multicore machines by adapting the ATLAS library. *Future Generation Computer Systems* (2017). (to appear).
[13] E.A. León, I. Karlin, R.E. Grant, and M. Dosanjh. 2016. Program optimizations: The interplay between power, performance, and energy. *Parallel Comput.* 58 (2016), 56–75.
[14] D. Li, B.R. de Supinski, M. Schulz, K. Cameron, and D.S. Nikolopoulos. 2010. Hybrid MPI/OpenMP power-aware computing. In *Int. Symposium on Parallel Distributed Processing (IPDPS 2010)*. IEEE, 1–12.
[15] M. Pippig and D. Potts. 2013. Parallel three-dimensional nonequispaced fast Fourier transforms and their application to particle simulation. *SIAM J. on Scientific Computing* 35, 4 (2013), C411–C437.
[16] A. Podzimek, L. Bulej, L.Y. Chen, W. Binder, and P. Tuma. 2015. Analyzing the impact of CPU pinning and partial CPU loads on performance and energy efficiency. In *Int. Symposium on Cluster, Cloud and Grid Computing (CCGrid'15)*. IEEE, 1–10.
[17] M. Puzović, S. Manne, S. GalOn, and M. Ono. 2016. Quantifying energy use in dense shared memory HPC node. In *Int. Workshop on Energy Efficient Supercomputing (E2SC 2016)*. IEEE, 16–23.
[18] T. Rauber and G. Rünger. 2015. Modeling and analyzing the energy consumption of fork-join-based task parallel programs. *Concurrency and Computation: Practice and Experience* 27, 1 (2015), 211–236.
[19] T. Rauber, G. Rünger, and M. Schwind. 2014. Energy measurement and prediction for multi-threaded programs. In *High Performance Computing Symposium (HPC 2014)*. Society for Computer Simulation International, 20:1–20:9.
[20] T. Rauber, G. Rünger, M. Schwind, H. Xu, and S. Melzner. 2014. Energy measurement, modeling, and prediction for processors with frequency scaling. *J. of Supercomputing* 70, 3 (2014), 1451–1476.
[21] L. Tan, S. Kothapalli, L. Chen, O. Hussaini, R. Bissiri, and Z. Chen. 2014. A survey of power and energy efficient techniques for high performance numerical linear algebra operations. *Parallel Comput.* 40, 10 (2014), 559–573.
[22] S. Wang, B. Luo, W. Shi, and D. Tiwari. 2016. Application configuration selection for energy-efficient execution on multicore systems. *J. of Parallel and Distributed Computing* 87 (2016), 43–54.
[23] V.M. Weaver, D. Terpstra, H. McCraw, M. Johnson, K. Kasichayanula, J. Ralph, J. Nelson, P. Mucci, T. Mohan, and S. Moore. 2013. PAPI 5: Measuring power, energy, and the cloud. In *Int. Symposium on Performance Analysis of Systems and Software (ISPASS'13)*. IEEE, 124–125.