Scalable Parallel Astrophysical Codes for Exascale

First report on evaluation of Energy Efficiency

Deliverable number: D2.2

Version 1.0

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Belgium, Czech Republic, France, Germany, Greece, Italy, Norway, and Spain under grant agreement No 101093441

Project Information

Project Acronym:	SPACE
Project Full Title:	Scalable Parallel Astrophysical Codes for Exascale
Call:	Horizon-EuroHPC-JU-2021-COE-01
Grant Number:	101093441
Project URL:	https://space-coe.eu

Document Information

Editor:	Lubomir Riha, Ondrej Vysocky, IT4I@VSB
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Contractual Delivery Date:	31.12.2023
Actual Delivery Date	18.12.2023
Number of pages:	30
Keywords:	Energy efficiency, DVFS, MERIC, RAPL, HDEEM
Authors:	Ondrej Vysocky – IT4I@VSB Tomas Panos – IT4I@VSB Ondrej Kozinski – IT4I@VSB
Peer review:	Gino Perna - ES, Manolis Marazakis - FORTH

History of Changes

Release	Date	Author, Organization	Description of changes
0.1	31.10.2023	Ondrej Vysocky – IT4I@VSB	Started the document
0.2	1.12.2023	Ondrej Vysocky, Tomas Panoc, Ondrej Kozinski – IT4I@VSB	Measurements results added
0.3	6.12.2023	Lubomir Riha, Ondrej Vysocky – IT4I@VSB	Version for internal review
0.4	12.12.2023	Gino Perna – ES, Manolis Marazakis – FORTH	Internal review
1.0	15.12.2023	Lubomir Riha – IT4I	Final version after review

Scalable Parallel Astrophysical Codes for Exascale

DISCLAIMER

Executive Summary

Energy efficiency is defined as the ratio of performance (floating-point operations per second) per Watt consumed by the application. We evaluated the energy efficiency of SPACE CoE applications and tried to improve it by changing selected power knobs of the underlying hardware.

In this case, we used static tuning. Static tuning in this context means that a specific hardware configuration is set at the beginning of the application execution and remains unchanged until the end. Such a static configuration is not optimal for every part of the application, which means that the static configuration cannot achieve the maximum available savings. However, such a configuration can be easily applied by the Job Scheduler at the start of the job.

All measurements were collected on the Barbora cluster CPU partition at IT4Innovations. Compute nodes are equipped with 2x Intel Xeon Gold 6240 (18-core, 2.6 GHz nominal frequency, Cascade Lake microarchitecture) processors and 192 GB DDR4 3200 MT/s memory and are interconnected through the 100 Gb/s InfiniBand HDR100 network. The network topology is the non-blocking Fat Tree.

Barbora system was selected for the energy efficiency analysis because the CPU allows to manage several knobs that influence its power consumption: 1.) CPU Core frequency, 2.) CPU uncore frequency 3.) the number of active cores and 4.) power limit. In this deliverable we use core and uncore frequency tuning only.

Since the Nehalem microarchitecture, the Intel company has been using ‘uncore‘ to refer to the frequency of subsystem in the physical processor package that is shared by multiple processor cores e.g., last-level cache, on-chip ring interconnect or the integrated memory controllers, which overall occupies approximately 30 % of a chip area [1].

Barbora’s computational nodes accommodate on board the Atos High Definition Energy Efficient Monitoring (HDEEM) system [2], which reads the power consumption from the system hardware sensors and stores the data to an integrated memory. The sensor that monitors the consumption of the entire node measures 1000 power consumption samples per second. We use MERIC, the lightweight runtime system for hardware parameters tuning and monitoring resources consumption, including energy. The library supports a wide range of tunable hardware parameters and various power monitoring systems.

Table 1 presents the summary of the energy savings achieved by all the applications. We present three scenarios, each with a different run-time penalty allowed. The runtime penalty stands for runtime extension when the processor runs on lower core and uncore frequencies. One can see that it is possible to save between 3–6 % of energy without impacting the runtime of the application (max. runtime extension of 2 %). When runtime can be extended by 5 % the savings are in the range of 6.6–9.4 %. The best results were achieved for the ChaNGa code, which can achieve 30% energy savings with almost no impact on its performance.

The maximum possible energy savings are 11–13 % and 36% for ChaNGa, but in this case the runtime is significantly longer. Still, these configurations are usable in the following scenarios: 1.) current cluster utilization is low and extending the runtime does not affect the amount of scientific results produced by it; 2.) cluster is running under strict power limit due to condition in the power distribution network; or 3.) the cost of electricity at a given moment is significantly higher, which means that OPEX is significantly higher than CAPEX.

Static tuning is only the first step in improving the energy efficiency of SPACE CoE applications. In the future, we will explore the dynamic tuning approach. Dynamic tuning means that we find optimal configurations

	Energy savings with maximum given runtime extension			
Code name	max. 2%	max. 5%	max. 10%	unlimited
Pluto	5.5 %	7.3 %	10.5 %	11.5 %
OpenGadget	5.4 %	6.8 %	7.4 %	11.8 %
iPic3D	5.7 %	9.4 %	11.0 %	12.9 %
RAMSES	7,4 %	10.1 %	11.9 %	13.4 %
BHAC	5.0 %	6.8 %	10.8 %	11.2 %
FIL	2.8 %	6.6 %	8.9 %	12.6 %
ChaNGa	30.2 %	35.8 %	36.0 %	36.0 %

Table 1: Energy savings achieved on the Barbora cluster under different maximum allowed runtime extension. The results are for static tuning of the CPU core and uncore frequency and measured using HDEEM.

for different regions of the application. Then as application moves from one region to another MERIC runtime will change the hardware settings to optimal ones for that given region.

Contents

List of Figures

List of Tables

List of Acronyms

SPACE	Scalable Parallel Astrophysical Codes for Exascale
CoE	Centre of Excellence
FLOP	Floating-point arithmetic operation
TDP	Thermal Design Power
RAPL	Running Average Power Limit
HDEEM	High Definition Energy Efficient Monitoring
CF	Frequency of CPU cores
UCF	Frequency of CPU uncore subsystem
CAPEX	Capital Expenditure (for supercomputing cluster)
OPEX	Operating Expenditure (for supercomputing cluster)

1 Introduction

Energy efficiency is defined as a fraction of performance (floating-point operations per second) per Watt consumed by the application. In this deliverable we have evaluated the energy efficiency of all SPACE applications and tried to improve it by changing selected power knobs of the underlying hardware. In this case, we performed static tuning only – static tuning in this context means that one specific hardware configuration is set at the beginning of the application run, and remains unchanged to the end of the execution. Such static configuration is not optimal for each part of the application, which means that the static configuration is not guranteed to reach maximum available savings. Please note that a static configuration can be readily applied by the HPC job scheduler at the time of launching a parallel job.

An alternative approach to static tuning is dynamic tuning, where the hardware configuration changes during application execution, which should fit the application requirements better than the single static configuration. The dynamic tuning of the SPACE applications is planned for the next phases of the project.

1.1 Hardware platform used for energy efficiency analysis

All performance data for this deliverable were collected on the Barbora cluster [1] CPU partition in IT4Innovations, i.e., using compute nodes equipped with 2x Intel Xeon Gold 6240 (18-core, 2.6 GHz nominal frequency) processors and 192 GB DDR4 3200 MT/s memory that are interconnected through the 100 Gb/s InfiniBand HDR100 network. The total theoretical peak performance of the partition (Rpeak) is 3.83 PFLOP/s. The network topology is the non-blocking Fat Tree, which consists of 60 x 40-ports HDR switches (40 Leaf HDR switches and 20 Spine HDR switches). The operating system is Red Hat Enterprise Linux 8 and the kernel versin is 4.18.0-305.25.1.el8_4.x86_64.

#nodes	CPU	RAM	Network
192	2× Intel Xeon Gold 6240 2.6 GHz nominal frequency 18 cores with AVX-512 support	192 GB 5.3 GB/core	1x 100Gbit/s InfiniBand HDR100

Table 2: Barbora cluster hardware information.

The Barbora system was selected for the energy efficiency analysis because the CPUs in its nodes allow managing several knobs that influence its power consumption, and has two power monitoring systems that can be used to measure energy consumption. The CPU power knobs are the following:

- core frequency
  - 3.9 GHz max turbo frequency
  - 3.3 GHz max turbo frequency when all cores active
  - 2.6 GHz nominal frequency
  - 1.0 GHz minimum frequency
  - 100 MHz step, some configurations in the range minimum to nominal frequency not available
- uncore frequency
  - 2.4 GHz max frequency
  - 1.2 GHz minimum frequency
  - 100 MHz step
- number of active cores
  - 18 cores
  - 2 hyper-threads (turned off on the Barbora system)
- power limit

[1] Barbora cluster docs available online https://docs.it4i.cz/barbora/introduction/

– 150 W TDP

Since Nehalem architecture, the Intel company has been using ‘uncore‘ to refer to the frequency of subsystem in the physical processor package that is shared by multiple processor cores e.g., last-level cache, on-chip ring interconnect or the integrated memory controllers, which overall occupies approximately 30 % of a chip area [1].

To keep the power consumption of the CPU below a specified limit, the CPU has a feature named Running Average Power Limit (RAPL) [3]. In default, the power limit is set to the Thermal Design Power (TDP) of the CPU (specified by the manufacturer), while its interface allows the limit to be reduced, not increased.

The RAPL also accumulates power consumption into a performance counter, so it can be used to report the energy consumption of the CPU. Barbora’s computational nodes accommodate on board the Atos High Definition Energy Efficient Monitoring (HDEEM) system [2], which reads power consumption from the system hardware sensors and stores the data to an integrated memory. The sensor that monitors the consumption of the whole node provides 1000 power consumption samples per second, and the rest of the sensors that monitor the node sub-units use 100 samples per second. Both aggregated values and power samples can be read from the user-space.

We are using the HDEEM, as well as RAPL energy consumption measurements, for energy-efficiency analysis. The HDEEM measurement provides higher precision, and most importantly, it gives the power consumption of the whole compute node (blade), while Intel RAPL monitors the power consumption of the CPU itself only. Suppose the energy consumption analysis accepts (limited) performance degradation. In that case, ignoring the power consumption of the non-monitored components leads to optimal configuration identification, which may result in longer runtime and overall energy consumption higher, in comparison to case when all components are monitored for their power consumption. From this point of view, the HDEEM values are more representative than values measured by RAPL, however to make the results comparable to other state-of-the-art research, we also present the RAPL (energy consumption of Package + DRAM power domains) measurements.

The power monitoring systems evaluate energy consumption based on power consumption samples the power monitoring system took in time, working at a specific sampling frequency. Both HDEEM blade power domain and RAPL work at 1 kHz frequency, while HDEEM not only provides the energy consumption but moreover the raw power samples. Consumed energy (in Joules) is calculated from power samples (in Watts), that are measured at a sampling frequency (in Hertz) as shows equation

$$Energy(t) = \int_0^t Power(x)\,\mathrm{d}x \approx \frac{\sum_{i=0}^{n} PowerSample_i}{SamplingFrequency} \tag{1}$$

1.2 Tools to evaluate energy efficiency

MERIC [4] is a lightweight runtime system for parallel application dynamic behavior detection and energy consumption tuning according to the READEX approach [5]. The library supports a wide range of tunable hardware parameters and various power monitoring systems.

The Open-source MERIC library is designed to minimize the energy consumption of the HPC infrastructure running a parallel application by dynamic tuning of a wide range of hardware knobs. It supports tuning of CPUs and GPUs of various vendors, as well as several power monitoring solutions. The library and associated tools perform detailed analyses of complex application behaviour, visualization of measured data, identification of the optimal hardware settings with respect to energy consumption and runtime, and dynamic tuning during the application runtime.

MERIC traces can be visualised using the **RADAR visualizer**. Since MERIC executes parts of the application in various configurations when searching for optimal energy consumption settings, the RADAR visualizer is designed to present the application behavior in the default configuration and compare it with every other evaluated configuration, which creates a unique insight into application behavior.

1.3 Deliverable structure and organization

This deliverable strongly relates to the deliverable D2.1 (”Performance Profiling and Benchmarking”, submitted in October 2023 [6]) as it uses the same set of codes as well as the same set of use cases. Therefore, as we do not want to repeat the text and descriptions, we reference to these sections in the D2.1 which is publicly available on the SPACE CoE website: `https://www.space-coe.eu/publications.php`.

Section 2 of this document describes the methodology used to analyse energy efficiency and how to read result of each code analysis. The results itself are provided in the Section 3. Finally, Section 4 summarizes the energy savings reached by static tuning.

2 Energy efficiency analysis methodology

We have performed the energy-efficiency analysis of the SPACE applications executing the codes on 8 compute nodes of the Barbora machine (for a total of 288 cores), except for RAMSES, which was executed on 16 nodes. We were using the same input usecases as used for the performance analysis reported in the D2.1 *Performance profiling and benchmarking*. The RAMSES usecase could not fit into the memory of 8 nodes, which is why we used 16 nodes in this case.

For this deliverable, we have executed static application tuning of CPU core (CF) and uncore (UCF) frequencies that influence the power consumption of the CPU, to identify available energy savings with several performance degradation constraints – 2 %, 5 %, 10 %, and no limit. To present the applications' behavior in detail, we executed an exhaustive state-space-search, not using an algorithm directly converging from the initial to the optimal configuration. The resource consumption is visualized in heatmaps of values relative to resource consumption with the default hardware consumption.

Since energy efficiency is defined as performance per single Watt, we have evaluated the number of FLOP instructions executed by the application and energy consumption using Intel RAPL and HDEEM power monitoring systems. The following subsections also present a power consumption timeline based on HDEEM power samples taken during the test case execution. These timelines should give us a rough idea of the dynamic behavior of the applications.

2.1 Report component description

2.1.1 Power timeline

The power timeline shows the power consumption during the application runtime. It is measured by the HDEEM power monitoring system and stored in its internal memory. HDEEM has several channels to monitor individual components of the compute node which are color-coded in the legend of the figure:

- DDR_GHJ – measures power consumption of the memory DIMMs on CPU1, sampling rate is 100 Hz,
- DDR_ABC – measures power consumption of the memory DIMMs on CPU0, sampling rate is 100 Hz,
- CPU0 – measures power consumption of the CPU 1, sampling rate is 100 Hz,
- CPU1 – measures power consumption of the CPU 0, sampling rate is 100 Hz,
- Blade – measures power consumption of the entire compute node, sampling rate is 1 kHz.

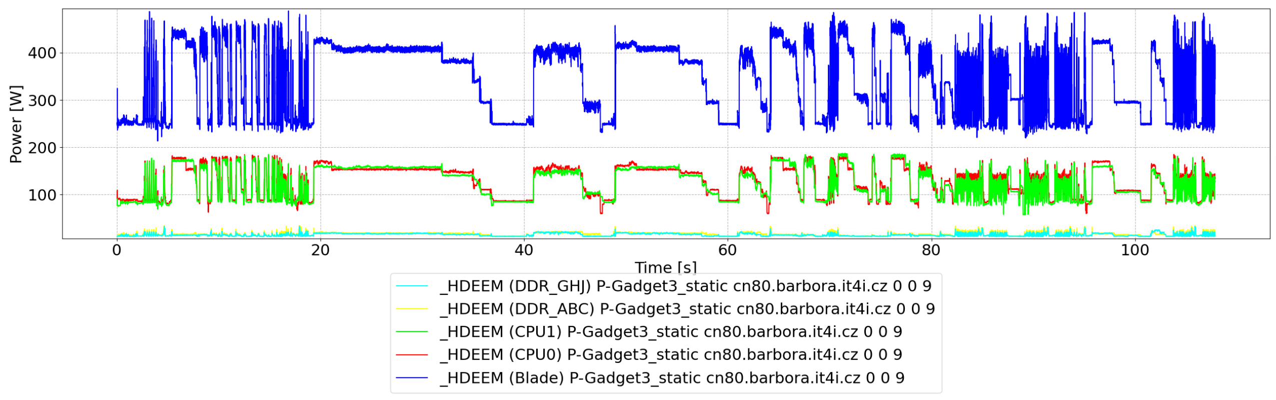

Figure 1: Example of power consumption timeline of a single node and its components (Gadget code)

The power timeline is useful for visualizing the different coarse grain phases of the application and identifying the dynamic behavior of the application from the power consumption point of view. One can see from the Figure 1 that the node's power consumption can vary significantly based on the workload. In general, one can expect that different phases have different optimal hardware settings. The exploitation of the dynamic behavior of the application is beyond the scope of this report but will be explored later in the project.

2.1.2 Floating-point operations measurements tables

For each SPACE application, we have measured its performance in FLOPs/s. Floating point operations of each supported instruction set executed by the CPU are tracked by Intel CPUs using FP_ARITH_INST_RETIRED:XX event [2], where XX represents one of 8 events associated with floating-point instructions. Half of them are dedicated to single precision and the other half to double precision. Since the Intel Xeon Cascade Lake microarchitecture used for our benchmarking supports vector instructions of length up to 512 bits, the number of events that occur must be multiplied by the number of floating point operations per single instruction. In the Application results subsections, we present tables of all eight events, so from the results, it is visible how much each code uses vectorization.

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	7240374183451	9164881641636
SCALAR_SINGLE	1	154346452896	154346452896
128B_PACKED_DOUBLE	2	557726129284	1115452258568
128B_PACKED_SINGLE	4	254594513	1018378052
256B_PACKED_DOUBLE	4	2725451737	10901806948
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	105.38	**Sum** [FLOPs]	8522093079915
		GFLOPs/s	80.87

Table 3: Example of FLOPS measurement: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

2.1.3 Heatmaps

For each application, we present 3 different heatmaps.

1.) Impact of the tuning on runtime. This heatmap shows the runtime increase (positive values) or decrease (negative values) (negative values) for different configurations of the CPU Core Frequency (denoted as *core [GHz]*) and CPU Uncore Frequency (denoted as *uncore [GHz]*). Measurement for default settings is not presented in the heatmap, but it is in the Energy savings tables. The values in the heatmaps are relative values in % with respect to the runtime for the default settings.

2.) Impact of the tuning on HDEEM energy consumption. This heatmap presents energy consumption savings (negative values) or increase (positive values). The energy consumption is measured by the HDEEM Blade sensor, which represents the energy consumption of the entire server. As applications are executed on multiple compute nodes, we sum up the energy consumption of all nodes (8 for all runs of SPACE codes, except for the RAMSES code which runs on 16 nodes). Again, the values in the heatmaps are relative values in % with respect to the HDEEM energy consumption for the default settings.

3.) Impact of the tuning on RAPL energy consumption. This heatmap presents the same type of data as the previous one. The only difference is that it uses RAPL counters to measure energy consumption. RAPL counters measure only CPU energy consumption and do not take into account energy consumption of other components of the compute node like mother-board, memory DIMMs, network cards, etc. In the case of Barbora CPU compute nodes, these other components have a power consumption of approximately 80 W. We call the power consumption of those components of the compute node as *baseline power consumption*.

CPU energy consumption is the one most affected by changes in CPU core and uncore frequencies. CPUs in this system have power consumption in the range of 45–150 W depending on the workload they execute. Therefore, the impact of tuning on the energy consumption of the CPUs only (represented by RAPL measurements) is higher, and better savings can be achieved. However, this is achieved in many cases for configuration where a significant extension of runtime is present. However, significant extension of runtime increases the energy consumption of the remaining on-node components (baseline). Since HDEEM measures both of these contributions (CPUs consumption plus baseline), the optimal settings where maximum energy savings are achieved also results in the faster runtime.

[2] List of events supported by the Intel CPU including full description available online https://perfmon-events.intel.com/cascadelake_server.html.

In production, one should always optimize for the overall compute node consumption, in this case represented by HDEEM energy consumption.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	106.64	95.07	89.08	79.95	74.68	71.86	70.56
1.5	90.81	77.83	72.3	64.95	59.81	58.26	55.46
1.7	79.73	68.37	59.76	52.62	46.43	44.1	42.27
1.9	78.24	60.6	50.98	42.75	38.39	36.25	33.26
2.1	71.83	52.24	45.23	36.16	31.4	27.74	23.78
2.3	64.06	52.49	39.01	30.3	25.97	22.34	19.35
2.5	68.28	46.39	37.38	29.44	24.35	17.48	16.44
2.6	69.78	47.34	36.22	24.38	20.38	17.05	13.32
2.7	67.6	42.46	34.08	24.44	17.59	14.37	10.33
2.8	65.64	45.95	30.55	24.79	16.31	13.2	7.78
2.9	63.26	50.33	31.94	22.67	13.99	10.46	7.53
3	63.02	46.02	27.63	21.36	13.37	8.03	2.73
3.1	59.03	45.2	27.36	22.01	12.54	6.57	1.49
3.2	56.33	43.32	28.45	19.56	13.67	5.35	4.67
3.3	56.62	42.06	34.96	16.65	13.27	4	0.57

Table 4: Example from OpenGadget: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	8.36	5.04	4.44	2.32	3.94	7.13	12.08
1.5	3.14	−1.03	−1.99	−3.04	−1.96	1.89	5.32
1.7	1.58	−1.79	−4.29	−5.89	−5.69	−2.82	0.92
1.9	3.33	−3.84	−7.02	−9.81	−8.42	−5.59	−2.62
2.1	2.43	−5.97	−8	−11.15	−10.56	−8.8	−6.5
2.3	1.66	−2.74	−8.55	−11.82	−10.85	−9.29	−7.07
2.5	8.74	−1.75	−5.89	−10.85	−9.56	−8.88	−5.21
2.6	11.55	0.27	−4.26	−9.88	−9.01	−8.1	−6.31
2.7	12.16	−1.02	−4.29	−8.45	−9.43	−7.99	−6.66
2.8	12.99	2.7	−4.8	−6.45	−8.77	−7.43	−7.43
2.9	13.87	7.94	−1.67	−5.83	−8.07	−7.52	−5.47
3	16.57	7.73	−1.85	−4.14	−6.68	−6.94	−6.82
3.1	17.16	10.53	0.39	−1.34	−4.83	−5.67	−5.43
3.2	19	12.09	3.72	−0.38	−1.55	−3.88	−1.22
3.3	22.21	14.16	10.62	0.2	0.69	−0.96	−2.36

Table 5: Example from OpenGadget: Impact of the static tuning to HDEEM energy consumption [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	−15	−16.5	−16.01	−16.62	−12.4	−7.87	−0.91
1.5	−18.03	−20.21	−20.24	−19.41	−16.43	−10.53	−5.4
1.7	−16.81	−18.35	−19.52	−19.39	−17.06	−13.21	−7.96
1.9	−14.75	−18.89	−20.62	−21.69	−18.66	−14.4	−10.33
2.1	−13.92	−19.19	−19.73	−21.53	−19.32	−16.46	−12.81
2.3	−12.68	−15.03	−19.19	−21.02	−18.76	−15.92	−12.53
2.5	−4.84	−12.34	−15.38	−16.23	−14.71	−14.07	−9.39
2.6	−1.19	−10.19	−13.3	−17.35	−15.23	−13.27	−10.09
2.7	0.11	−10.62	−12.47	−15.57	−15.12	−12.6	−9.75
2.8	1.52	−6.8	−12.33	−13.1	−14.02	−11.46	−10.36
2.9	3.07	−1.19	−8.7	−11.75	−12.68	−11.11	−7.78
3	6.65	−0.21	−8.08	−9.38	−10.74	−9.78	−8.42
3.1	8.23	3.25	−5.13	−6.09	−8	−7.8	−6.4
3.2	10.93	5.78	−1.22	−4.19	−4.3	−5.24	−1.85
3.3	15.07	8.59	6	−2.75	−1.36	−1.82	−2.24

Table 6: Example from OpenGadget: Impact of the static tuning to RAPL energy consumption [%].

2.1.4 Energy savings tables

The final evaluation of an application experiment is presented in a table at the end of every section, similar to following Table 7. On the left-hand side runtime, energy consumption, and energy efficiency in terms of FLOPs/W for the default settings are presented. The CPU configuration is not presented because none have been applied.

If an application consists of mixed workloads (both memory and compute bound sections), as one decreases the CPU core and/or uncore frequency the result will be both an extension in the runtime and savings in the energy consumption. However, we must keep the runtime extension at a reasonable level. We present three scenarios where runtime can be extended up to 2 %, 5 % and 10 %, respectively. For each scenario, we find the maximum energy savings and respective CPU configuration (CPU core and uncore frequency). We present energy savings calculated from both HDEEM and RAPL energy measurements, but keep in mind that HDEEM represents more realistic values as it measures the energy consumption of the entire node.

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	105.38	**Performance penalty** [%]	1.49	2.73	7.78	30.3
Energy [kJ]		**Energy savings** [%]				
HDEEM	304.92	**HDEEM**	5.43	6.82	7.43	11.82
RAPL	225.65	**RAPL**	6.40	8.42	10.36	21.02
Eff. [MFLOPs/W]		**Eff.** [MFLOPs/W]				
HDEEM	27.95	**HDEEM**	29.55	30.00	30.19	31.69
RAPL	37.77	**RAPL**	40.35	41.24	42.13	47.82
		CPU configuration				
		Core freq. [GHz]	3.1	3.0	2.8	2.3
		Uncore freq. [GHz]	2.4	2.4	2.4	1.8

Table 7: Example from OpenGadget: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3 Results

This section provides the performance and energy measurements of all SPACE codes. For each code, we provide short description of the results.

3.1 Pluto

PLUTO [7] is designed to integrate a general system of conservation laws that we write as

$$\frac{\partial U}{\partial t} = -\nabla \cdot T(U) + S(U)\,. \tag{2}$$

Here U denotes a state vector of conservative quantities, T(U) is a rank 2 tensor (the rows of which are the fluxes of each component) and S(U) defines the source terms. Additional source terms may implicitly arise when taking the divergence of T(U) in a curvilinear system of coordinates. An arbitrary number of advection equations may be added to the original conservation law.

Pluto's power consumption time line (Figure 2) shows no coarse-grain phases; however, we still expect that fine-grain regions will be detected in the future when dynamic tuning is explored. We can also see in Table 8 that PLUTO has very limited vectorization in place. This will be evaluated during the optimization effort on the code. The heatmaps (Table 9, 10, and 11) show that runtime is more affected by the core frequency rather than the uncore one. This means that, in general, PLUTO is mostly compute-bound application. We can save up to 10.5 % of energy (HDEEM) by extending the runtime by 8.9 %. The maximum energy savings are 11.5 %, but at the cost of a major runtime extension (26 %) as presented in Table 12.

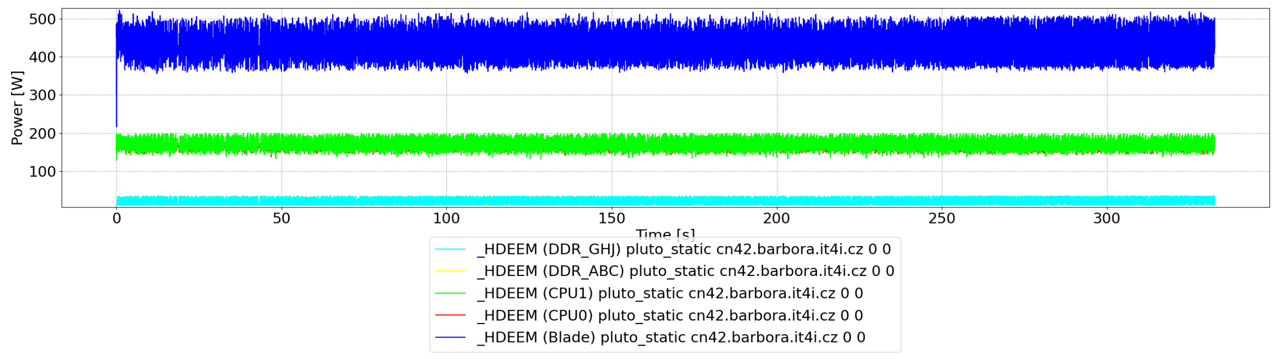

Figure 2: Pluto power consumption timeline of a single node and its components.

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	9164881641636	9164881641636
SCALAR_SINGLE	1	1441	1441
128B_PACKED_DOUBLE	2	103320	206640
128B_PACKED_SINGLE	4	0	0
256B_PACKED_DOUBLE	4	550	2200
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	257.07	**Sum** [FLOPs]	9164881851917
		GFLOPs/s	35.65

Table 8: Pluto: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	103.84	96.29	92.74	90.54	89.34	89.58	87.7
1.5	84.87	76.01	70.79	69.15	68.21	66.83	66.53
1.7	69.34	60.64	55.47	53.77	52.19	51.79	50.31
1.9	58.78	49.49	43.81	41.47	44.04	61.13	60.95
2.1	65.4	56.55	38.51	31.77	30.24	30.2	28.5
2.3	41.53	31.78	25.89	23.82	22.03	21.07	20.1
2.5	35.68	25.47	19.09	17.67	15.42	14.17	13.73
2.6	32.9	22.5	16.47	14.07	12.38	11.09	10.46
2.7	30.15	20.25	14.06	11.35	10.38	8.49	7.92
2.8	27.65	17.58	10.74	8.84	7	6.17	5.27
2.9	25.52	14.94	8.75	6.35	5.26	3.52	2.99
3	23.5	13.06	6.28	4.34	2.76	1.73	1.26
3.1	21.75	10.84	4.75	2.18	0.57	−0.55	−0.83
3.2	19.9	9.26	2.47	0.91	−1.2	−1.45	−1.6
3.3	18.49	7.83	1.02	−0.86	−1.64	−1.65	−1.56

Table 9: Pluto: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	9.27	7.86	8.48	9.52	12.78	17.55	21.76
1.5	3.25	0.77	0.08	1.32	4.45	7.73	12.26
1.7	−0.69	−3.28	−4.06	−3.09	−0.83	2.75	6.21
1.9	−3.29	−6.45	−7.76	−7.36	−2.46	12.82	17.32
2.1	−4.58	−8.1	−7.28	−9.93	−8	−5.2	−2.06
2.3	−5.86	−9.74	−11.49	−11.16	−9.55	−7.01	−4.11
2.5	−4.85	−9.3	−11.56	−10.84	−9.74	−7.62	−4.51
2.6	−4.69	−9.38	−11.42	−11.49	−10.09	−8.04	−5.16
2.7	−3.97	−8.39	−10.69	−11.09	−9.12	−7.62	−4.75
2.8	−3.14	−7.85	−10.74	−10.52	−9.34	−7.01	−4.43
2.9	−1.14	−6.46	−8.89	−9.17	−7.35	−5.87	−2.97
3	1.02	−4.35	−7.43	−7.3	−5.94	−3.81	−1.08
3.1	3.53	−2.43	−4.93	−5.44	−4.16	−2.43	−0.27
3.2	6.15	0.24	−3.04	−2.64	−2.24	−0.53	0.11
3.3	9.63	3.38	−0.46	−0.78	−0.3	0.16	0.46

Table 10: Pluto: Impact of the static tuning to HDEEM energy consumption [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	−8.76	−9.43	−8.33	−6.56	−1.95	2.96	9.12
1.5	−13.01	−14.61	−14.3	−12	−8.04	−3.5	2.15
1.7	−14.57	−15.82	−16.04	−14.2	−10.63	−6.29	−2.45
1.9	−15.57	−17.48	−17.65	−16.55	−10.91	−8.54	−5.12
2.1	−15.21	−17.44	−15.62	−17.51	−15.3	−12.26	−8.03
2.3	−14.83	−17.41	−18.42	−18.04	−15.45	−12.05	−8.48
2.5	−12.45	−15.92	−17.39	−16.46	−14.43	−11.29	−7.93
2.6	−11.65	−15.59	−16.76	−16.36	−14.07	−11.48	−8.13
2.7	−10.47	−13.48	−15.27	−14.9	−12.58	−10.58	−7.16
2.8	−8.78	−12.47	−14.52	−13.92	−12.3	−9.47	−6.29
2.9	−5.7	−10.41	−12.24	−11.96	−9.67	−7.6	−4.12
3	−3.06	−7.6	−9.98	−9.44	−7.53	−4.85	−1.63
3.1	0.24	−4.79	−6.69	−6.85	−5.11	−2.9	−0.4
3.2	3.71	−1.38	−4.17	−3.4	−2.64	−0.61	0.17
3.3	8.02	2.46	−0.97	−0.98	−0.29	0.22	0.44

Table 11: Pluto: Impact of the static tuning to RAPL energy consumption [%].

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	257.07	**Performance penalty [%]**	2.18	4.34	8.84	25.9
Energy [kJ]		**Energy savings [%]**				
HDEEM	881.28	**HDEEM**	5.44	7.30	10.52	11.49
RAPL	668.07	**RAPL**	6.85	9.44	13.92	18.42
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	10.40	**HDEEM**	11.00	11.22	11.62	11.75
RAPL	13.72	**RAPL**	14.73	15.15	15.94	16.81
		CPU configuration				
		Core freq. [GHz]	3.1	3.0	2.8	2.3
		Uncore freq. [GHz]	1.8	1.8	1.8	1.6

Table 12: Pluto: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3.2 OpenGadget

OpenGadget is a code for cosmological simulations. It solves the gravitational and hydrodynamical equations that rule the formation and evolution of cosmic structures.It computes the gravitational forces with a hierarchical tree algorithm in combination with a particle-mesh scheme for long-range gravitational forces. Then the fluid flows are computed using smoothed particle hydrodynamics (SPH) or Meshless Finite Mass (MFM). In addition, OpenGadget contains multiple sub-modules to describe various physical processes (e.g. radiative cooling, star formation, stellar feedback, magnetic fields, black holes, just to name the major ones) that shape the fundamental properties of baryons in the Universe.

The power consumption timeline (Figure 3) of OpenGadget identifies distinct phases of the application, from which the dynamic tuning may benefit since they are long enough to accommodate setting the optimal configuration. The code itself applies vectorization but does not utilize 512bit vector instructions, as shown in Table 13. Based on 3 heatmaps (Tables 14 , 15, 16) we created Table 17, which shows that by losing 1.49 % of performance, we can get 5.43 % node energy (HDEEM) savings. With static frequency tuning, we can save up to 11.82 % (21.02 % according to CPU RAPL); however, these savings would come at the cost of a 30.3 % performance loss.

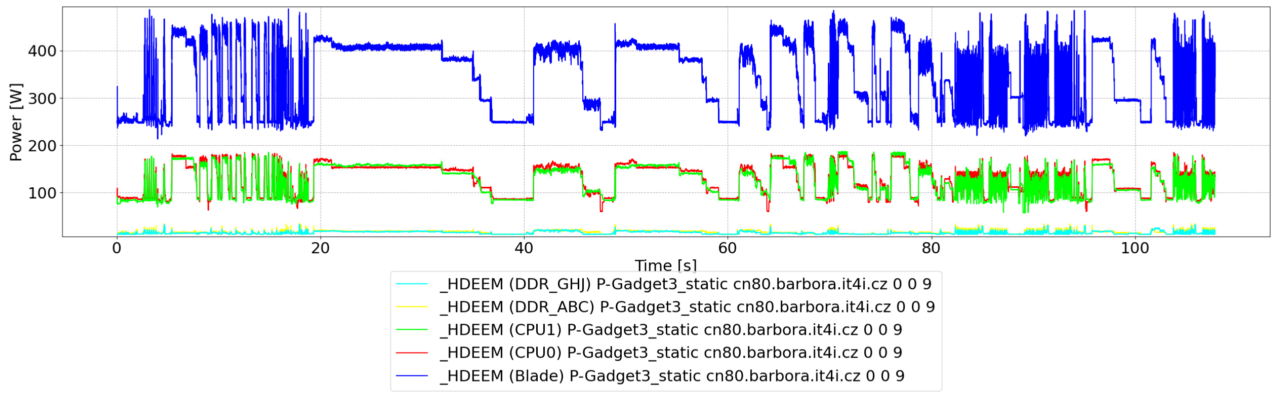

Figure 3: OpenGadget power consumption timeline of a single node and its components.

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	7240374183451	9164881641636
SCALAR_SINGLE	1	154346452896	154346452896
128B_PACKED_DOUBLE	2	557726129284	1115452258568
128B_PACKED_SINGLE	4	254594513	1018378052
256B_PACKED_DOUBLE	4	2725451737	10901806948
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	105.38	**Sum [FLOPs]**	8522093079915
		GFLOPs/s	80.87

Table 13: OpenGadget: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	106.64	95.07	89.08	79.95	74.68	71.86	70.56
1.5	90.81	77.83	72.3	64.95	59.81	58.26	55.46
1.7	79.73	68.37	59.76	52.62	46.43	44.1	42.27
1.9	78.24	60.6	50.98	42.75	38.39	36.25	33.26
2.1	71.83	52.24	45.23	36.16	31.4	27.74	23.78
2.3	64.06	52.49	39.01	30.3	25.97	22.34	19.35
2.5	68.28	46.39	37.38	29.44	24.35	17.48	16.44
2.6	69.78	47.34	36.22	24.38	20.38	17.05	13.32
2.7	67.6	42.46	34.08	24.44	17.59	14.37	10.33
2.8	65.64	45.95	30.55	24.79	16.31	13.2	7.78
2.9	63.26	50.33	31.94	22.67	13.99	10.46	7.53
3	63.02	46.02	27.63	21.36	13.37	8.03	2.73
3.1	59.03	45.2	27.36	22.01	12.54	6.57	1.49
3.2	56.33	43.32	28.45	19.56	13.67	5.35	4.67
3.3	56.62	42.06	34.96	16.65	13.27	4	0.57

Table 14: OpenGadget: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	8.36	5.04	4.44	2.32	3.94	7.13	12.08
1.5	3.14	−1.03	−1.99	−3.04	−1.96	1.89	5.32
1.7	1.58	−1.79	−4.29	−5.89	−5.69	−2.82	0.92
1.9	3.33	−3.84	−7.02	−9.81	−8.42	−5.59	−2.62
2.1	2.43	−5.97	−8	−11.15	−10.56	−8.8	−6.5
2.3	1.66	−2.74	−8.55	−11.82	−10.85	−9.29	−7.07
2.5	8.74	−1.75	−5.89	−8.22	−8.16	−8.88	−5.21
2.6	11.55	0.27	−4.26	−9.88	−9.01	−8.1	−6.31
2.7	12.16	−1.02	−4.29	−8.45	−9.43	−7.99	−6.66
2.8	12.99	2.7	−4.8	−6.45	−8.77	−7.43	−7.43
2.9	13.87	7.94	−1.67	−5.83	−8.07	−7.52	−5.47
3	16.57	7.73	−1.85	−4.14	−6.68	−6.94	−6.82
3.1	17.16	10.53	0.39	−1.34	−4.83	−5.67	−5.43
3.2	19	12.09	3.72	−0.38	−1.55	−3.88	−1.22
3.3	22.21	14.16	10.62	0.2	0.69	−0.96	−2.36

Table 15: OpenGadget: Impact of the static tuning to HDEEM energy consumption [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	−15	−16.5	−16.01	−16.62	−12.4	−7.87	−0.91
1.5	−18.03	−20.21	−20.24	−19.41	−16.43	−10.53	−5.4
1.7	−16.81	−18.35	−19.52	−19.39	−17.06	−13.21	−7.96
1.9	−14.75	−18.89	−20.62	−21.69	−18.66	−14.4	−10.33
2.1	−13.92	−19.19	−19.73	−21.53	−19.32	−16.46	−12.81
2.3	−12.68	−15.03	−19.19	−21.02	−18.76	−15.92	−12.53
2.5	−4.84	−12.34	−15.38	−16.23	−14.71	−14.07	−9.39
2.6	−1.19	−10.19	−13.3	−17.35	−15.23	−13.27	−10.09
2.7	0.11	−10.62	−12.47	−15.57	−15.12	−12.6	−9.75
2.8	1.52	−6.8	−12.33	−13.1	−14.02	−11.46	−10.36
2.9	3.07	−1.19	−8.7	−11.75	−12.68	−11.11	−7.78
3	6.65	−0.21	−8.08	−9.38	−10.74	−9.78	−8.42
3.1	8.23	3.25	−5.13	−6.09	−8	−7.8	−6.4
3.2	10.93	5.78	−1.22	−4.19	−4.3	−5.24	−1.85
3.3	15.07	8.59	6	−2.75	−1.36	−1.82	−2.24

Table 16: OpenGadget: Impact of the static tuning to RAPL energy consumption [%].

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	105.38	**Performance penalty [%]**	1.49	2.73	7.78	30.3
Energy [kJ]		**Energy savings [%]**				
HDEEM	304.92	**HDEEM**	5.43	6.82	7.43	11.82
RAPL	225.65	**RAPL**	6.40	8.42	10.36	21.02
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	27.95	**HDEEM**	29.55	30.00	30.19	31.69
RAPL	37.77	**RAPL**	40.35	41.24	42.13	47.82
		CPU configuration				
		Core freq. [GHz]	3.1	3.0	2.8	2.3
		Uncore freq. [GHz]	2.4	2.4	2.4	1.8

Table 17: OpenGadget: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3.3 iPic3D

iPic3D [8] is a general-purpose computational code developed in the kinetic domain to study collisionless plasma dynamics using the 3D Implicit Particle-in-Cell (PIC) method [9, 10]. Here in the iPic3D, i stands for scheme that is used: implicit. iPic3D employs the implicit moment method to achieve an efficient implementation of the implicit formulation that allows about a multiscale discretization in time and space of the governing equations. The use can select what scales to resolve and the unresolved scales do not cause any numerical instability by virtue of the implicit nature of the temporal discretization. Hence, we can use time steps and grid spacing that are typically order 10-100 larger than time steps and grid spacing used in traditional PIC codes.

The power consumption timeline of the iPic3D presented in Figure 4 clearly shows iterations of the solver, where we observe a variability of about 200 W node power consumption during a single iteration. With negligible performance penalty (0.8 %), we were able to reach 5.7 % node energy savings, primarily by reducing the uncore frequency of the CPUs, and improved the energy efficiency of the code by about 6.1 % (HDEEM). Relaxing the performance penalty limit leads to higher energy savings, but with a worse performance degradation trade-off. The peak of energy savings is measured with CPU configuration ”2.7 GHz core and 1.4 GHz uncore”, where energy efficiency was improved by about 14.8 %. However, with this configuration the runtime of the iPic3D is extended by about 15.5 % as shown in Table 22.

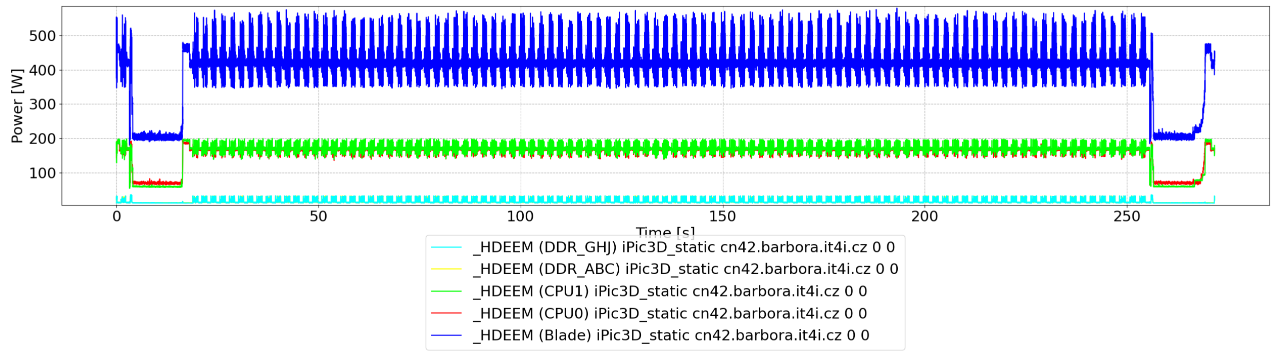

Figure 4: iPic3D power consumption timeline of a single node and its components.

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	18965546807918	18965546807918
SCALAR_SINGLE	1	1441	1441
128B_PACKED_DOUBLE	2	30338972604	60677945208
128B_PACKED_SINGLE	4	0	0
256B_PACKED_DOUBLE	4	370	1480
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	280,33	**Sum** [**FLOPs**]	19026224756047
		GFLOPs/s	67,87

Table 18: iPic3D: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	123.43	119.4	116.29	115.66	114.72	113.81	115.27
1.5	94.15	91.92	87.98	86.77	87.77	86.92	87.19
1.7	73.06	71.8	68.58	68.43	66.98	65.76	68.67
1.9	57.89	55.24	53.37	52.36	51.41	51.66	51.33
2.1	46.05	42.27	40.4	40.66	38.74	39.08	38.49
2.3	35.85	31.6	30.6	29.37	28.65	28.75	28.99
2.5	27.54	24.16	21.06	20.67	20.14	23.66	21.47
2.6	24.49	19.67	18.23	18	15.88	15.9	16.01
2.7	20.48	15.54	17.15	14.98	14.78	13.95	13.75
2.8	17.4	13.23	10.86	10.41	10.22	8.93	10.72
2.9	14.34	9.8	8.07	7.04	6.9	6.75	5.48
3	11.99	8.49	5.16	5.39	4.04	4.61	3.8
3.1	9.38	4.22	2.51	1.64	3.1	1.31	2.22
3.2	6.96	3.31	0.84	0.55	0.15	0.07	1.07
3.3	4.8	2.21	−0.65	−0.67	−0.32	0.42	0.43

Table 19: iPic3D: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	12.45	12.82	13.72	15.88	19.92	24.53	31.56
1.5	3.59	4.43	4.42	5.97	10.52	14.63	20.13
1.7	−2.92	−2.47	−1.52	0.43	3.08	6.46	13.23
1.9	−7.24	−6.99	−6.26	−5.06	−2.35	1.51	5.82
2.1	−9.81	−10.43	−9.82	−7.89	−6.21	−2.45	1.31
2.3	−10.76	−11.77	−10.87	−10.05	−7.67	−4.29	−0.24
2.5	−11.14	−11.71	−12.34	−11.11	−8.81	−2.71	−0.99
2.6	−10.84	−12.48	−11.95	−10.61	−9.6	−6.64	−3.05
2.7	−11.02	−12.93	−10.03	−10.38	−8.1	−6.88	−2.96
2.8	−10.08	−11.85	−12.07	−11	−8.52	−6.9	−1.89
2.9	−9.26	−11.03	−10.74	−10.19	−7.77	−5	−2.93
3	−7.28	−8.13	−9.42	−7.67	−6.46	−3.4	−1.17
3.1	−5.44	−8.05	−7.86	−7.18	−3.39	−3.12	−0.2
3.2	−3.65	−4.82	−5.71	−4.87	−3.55	−2.11	0.15
3.3	−1.75	−2.31	−4.27	−3.48	−2.12	−0.48	0.19

Table 20: iPic3D: Impact of the static tuning to HDEEM energy consumption [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	−11.96	−10.71	−8.65	−6.65	−1.67	4.8	13.51
1.5	−16.58	−14.89	−14.26	−12.12	−6.45	−1.09	5.94
1.7	−20.02	−20.02	−17.33	−14.59	−10.7	−6.42	1.41
1.9	−21.3	−20.48	−19.24	−16.8	−13.64	−8.91	−3.72
2.1	−22.07	−21.63	−20.14	−18.11	−15.4	−11.03	−6.38
2.3	−20.38	−20.77	−19.84	−18.59	−15.25	−10.94	−6.02
2.5	−19.16	−19.05	−19.27	−17.53	−14.7	−7.85	−5.22
2.6	−18.14	−18.91	−17.98	−16.41	−14.73	−11.05	−6.7
2.7	−17.31	−18.69	−15.53	−15.53	−12.65	−10.33	−5.7
2.8	−14.51	−16.98	−16.67	−15.23	−12.2	−9.87	−4.19
2.9	−13.88	−15.19	−14.43	−13.51	−10.53	−7.11	−4.35
3	−10.96	−11.33	−12.26	−10.24	−8.41	−4.78	−1.86
3.1	−8.29	−10.38	−9.78	−8.82	−4.48	−3.78	−0.5
3.2	−5.6	−6.24	−6.87	−5.82	−4.18	−2.39	0.1
3.3	−2.92	−3.08	−4.91	−3.97	−2.4	−0.56	0.25

Table 21: iPic3D: Impact of the static tuning to RAPL energy consumption [%].

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	280.33	**Performance penalty [%]**	0.84	5.16	9.80	15.54
Energy [kJ]		**Energy savings [%]**				
HDEEM	893.53	**HDEEM**	5.71	9.42	11.03	12.93
RAPL	652.56	**RAPL**	6.87	12.26	15.19	18.69
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	21.29	**HDEEM**	22.58	23.51	23.93	24.45
RAPL	29.16	**RAPL**	31.30	33.23	34.38	35.86
		CPU configuration				
		Core freq. [GHz]	3.2	3.0	2.9	2.7
		Uncore freq. [GHz]	1.6	1.6	1.4	1.4

Table 22: iPic3D: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3.4 RAMSES

RAMSES [11, 12] is an Adaptive-Mesh-Refinement (AMR) code which is used to study astrophysical fluid dynamics and the formation of structures in the Universe. It is based on an oct-tree structure, where parent cells are refined into children cells on a cell-by-cell basis following some user-defined criteria. RAMSES can deal with 1D, 2D and 3D Cartesian grids.

The power timeline of the RAMSES presented in Figure 5 clearly highlights the iteration behavior of the code. There are regions where memory consumption is high, and shorter regions where it is lower and at the same time CPU power consumption is higher. This means that the code contains both compute- and memory-bound regions. From Table 23 we can see that limited vectorization is present, only in form of SSE instructions. No AVX/2 or AVX512 instructions are present.

From the heatmaps (Figures 24, 25 and 26), we can see that runtime is mainly affected by changing the core frequency, which means that in this configuration the code is mainly compute-bound. There is only a negligible performance penalty (on average 5%) when the uncore frequency is fully under-clocked. This is the main reason we are able to save up to 7.4% of energy with less than 2% of runtime extension.

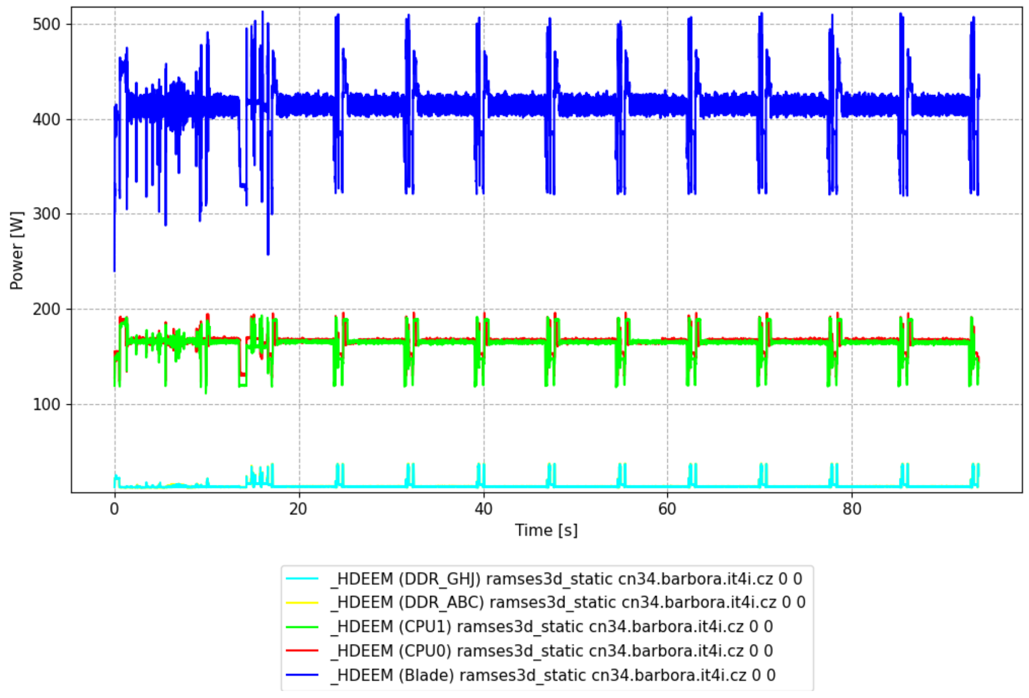

Figure 5: RAMSES power timeline

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	29131616712906	29131616712906
SCALAR_SINGLE	1	1305081	1305081
128B_PACKED_DOUBLE	2	1014686798816	2.0293736×10^{12}
128B_PACKED_SINGLE	4	0	0
256B_PACKED_DOUBLE	4	0	0
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	93.74	**Sum [FLOPs]**	3.1161×10^{13}
		GFLOPs/s	332.42

Table 23: RAMSES: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	127.13	121.77	124.6	119.56	118.92	119.03	117.62
1.5	98.54	94.92	93.03	91.51	91.23	91.66	89.81
1.7	75.76	73.54	71.39	75.53	70.58	69.52	69.03
1.9	60.61	57.11	54.96	53.69	52.98	53.2	52.63
2.1	46.53	44.4	41.48	41.38	42.19	39.94	39.36
2.3	35.29	34.04	31	29.76	28.9	28.33	27.98
2.5	26.75	23.3	21.42	21.42	19.79	19.33	19.23
2.6	23.84	19.6	17.69	17.16	15.4	15.36	14.75
2.7	18.76	15.65	14.34	12.74	12.76	11.88	11.17
2.8	16.37	12.55	12.48	10.11	10.11	10.76	8.67
2.9	13.72	9.25	7.17	6.64	6.33	5.66	4.54
3	10.3	7.41	4.76	3.28	2.26	1.84	3.46
3.1	7.22	3.63	1.85	0.67	0.06	0.21	0.02
3.2	4.18	1.7	1.97	−0.43	0.73	−0.38	−0.76
3.3	4.69	1.72	−0.41	−0.24	−0.01	0.54	1.72

Table 24: RAMSES: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	9.58	9.33	12.93	13.11	16.97	22.13	27.13
1.5	1.09	1.22	2.35	3.76	7.35	11.94	16.02
1.7	−4.73	−4.13	−3.42	0.77	1.41	4.74	9
1.9	−8.45	−8.71	−8.19	−7.07	−4.39	−0.79	3.01
2.1	−11.63	−11.23	−11.35	−9.87	−6.65	−4.57	−1.11
2.3	−13.05	−12.3	−12.66	−11.87	−9.77	−7.12	−3.82
2.5	−12.71	−13.47	−13.22	−11.81	−10.4	−7.86	−4.57
2.6	−11.83	−13.33	−13.23	−12.15	−10.97	−8.19	−5.55
2.7	−12.48	−13.17	−12.69	−12.43	−10.01	−8.02	−5.39
2.8	−11.05	−12.41	−11	−11.42	−9.11	−6.21	−4.47
2.9	−9.45	−11.13	−11.32	−10.29	−8.28	−6.14	−4.06
3	−7.59	−8.86	−9.23	−9.07	−7.61	−5.45	−1.52
3.1	−5.9	−7.33	−7.39	−7.09	−5.45	−3.27	−1.87
3.2	−3.99	−4.71	−3.17	−4.39	−1.98	−1.92	−1.41
3.3	0.17	−1.6	−2.85	−2.21	−1.16	−0.08	1.59

Table 25: RAMSES: HDEEM energy savings [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	−13.83	−12.91	−9.21	−8.86	−3.61	2.38	8.97
1.5	−18.16	−18.08	−15.77	−14.04	−9.32	−3.85	2.03
1.7	−20.79	−19.42	−17.93	−13.73	−11.84	−7.45	−2.23
1.9	−21.72	−21.59	−20.71	−18.88	−15.11	−10.37	−6.46
2.1	−22.89	−21.94	−21.08	−19.24	−15.7	−12.9	−8.85
2.3	−21.81	−20.84	−20.55	−19.65	−17	−14.07	−9.11
2.5	−19.82	−20.38	−19.4	−17.74	−15.61	−12.4	−8.43
2.6	−18.09	−18.94	−18.74	−17.12	−15.44	−11.85	−9.3
2.7	−17.81	−18.14	−17.3	−16.68	−13.61	−11.51	−8.11
2.8	−15.83	−16.24	−14.96	−15.17	−12.18	−8.94	−6.55
2.9	−13.21	−14.64	−14.54	−13.08	−10.41	−8	−5.2
3	−10.66	−11.41	−11.49	−10.95	−9	−6.28	−2.14
3.1	−7.88	−8.78	−8.68	−8.05	−6.15	−3.56	−1.99
3.2	−5.06	−5.54	−3.73	−4.75	−2.25	−1.96	−1.29
3.3	−0.35	−1.9	−2.97	−2.27	−1.1	0.09	1.81

Table 26: RAMSES: RAPL energy savings [%].

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	93.74	**Performance penalty [%]**	1.85	4.76	9.25	23.3
Energy [kJ]		**Energy savings [%]**				
HDEEM	38.89	**HDEEM**	-7.39	-9.23	-11.13	-13.47
RAPL	28.99	**RAPL**	-8.68	-11.5	-14,64	-20.38
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	100.15	**HDEEM**	108.00	111.31	112.68	115.73
RAPL	134.36	**RAPL**	144.01	159.01	157.41	168.75
		CPU configuration				
		Core freq. [GHz]	3.1	3.0	2.9	2.5
		Uncore freq. [GHz]	1.6	1.6	1.4	1.4

Table 27: RAMSES: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3.5 BHAC

The Black Hole Accretion Code (BHAC) [13, 14, 15, 16] is a multidimensional General Relativistic Magneto-hydrodynamics (GRMHD) code that solves the equations of ideal GRMHD in one, two or three dimensions in order to perform (magneto)hydrodynamical simulations of accretion flows onto compact objects in arbitrary stationary space-times (Cowling approximation) using an efficient block based approach. BHAC is build upon the MPI-Adaptive Mesh Refinement-Versatile Advection Code (MPI-AMRVAC). MPI-AMRVAC [17, 18] is a parallel adaptive mesh refinement framework aimed at solving (primarily hyperbolic) partial differential equations (PDEs) by a number of different numerical schemes. The framework supports 1D to 3D simulations, in a number of different geometries (Cartesian, cylindrical, spherical). MPI-AMRVAC is written in Fortran 90 and uses MPI for parallelization.

From the power time-line in Figure 6 we can clearly identify several stages of execution with significantly different power consumption. These show promising potential for dynamic tuning in the next phases. BHAC also shows good vectorization (Table 28) of the code with both the SSE and AVX instruction set, but no usage of AVX-512.

From the heat-maps (Tables 29, 30 and 31) we can see that the code is sensitive to both core and uncore frequency tuning, which means that it includes both memory and compute bound sections. Therefore, in most of the cases energy savings are achieved at the cost of increased runtime.

The maximum savings are 11.2 % at a cost of 18 % runtime extension. In the case of the max. 2 % runtime extension, the energy savings are 5 %.

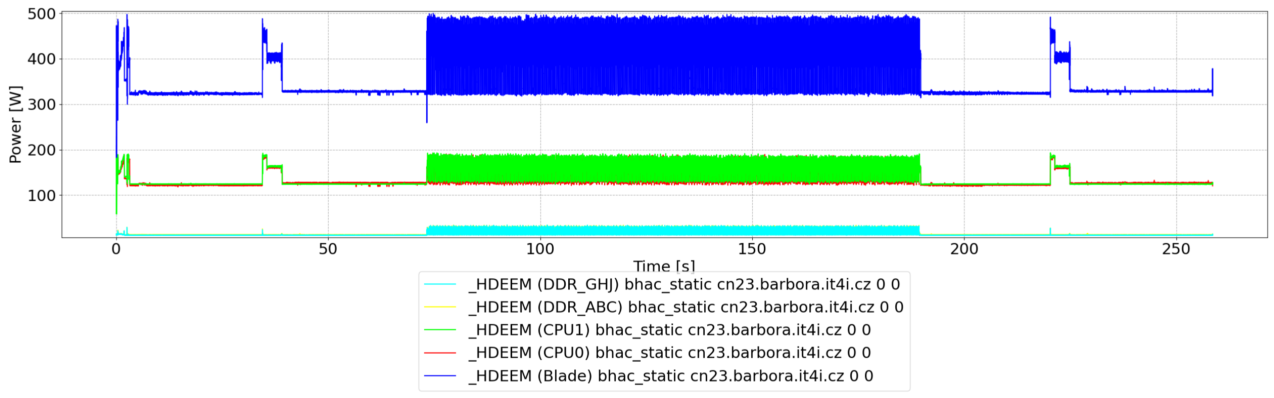

Figure 6: BHAC power timeline

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	4048122412149	4048122412149
SCALAR_SINGLE	1	19178542	19178542
128B_PACKED_DOUBLE	2	2900537466898	5.801074934*10e12
128B_PACKED_SINGLE	4	36010281682	144041126728
256B_PACKED_DOUBLE	4	6841735384269	2.7367*10e13
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	261.32	**Sum [FLOPs]**	3.73*10e13
		GFLOPs/s	142.96

Table 28: BHAC: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	105.3	103.16	105.22	101.7	99.9	99.92	100.84
1.5	83.38	81.1	78.96	77.74	76.45	77.04	76.9
1.7	66.72	64.42	61.99	60.52	60.6	59.51	60.18
1.9	52.7	49.82	46.68	46.21	45.9	45.74	44.81
2.1	42.21	36.4	37.97	36.11	36.25	33.94	33.82
2.3	32.36	28.87	26.52	25.64	24.49	24.71	23.94
2.5	25.39	22.08	21.72	18.87	17.9	17.7	16.58
2.6	21.31	18.27	17.06	14.78	15.19	14.27	15.09
2.7	18.72	14.96	13.6	12.44	11.82	10.73	10.01
2.8	16.12	11.86	9.97	8.42	8.87	6.78	7.42
2.9	13.17	8.07	6.15	5.11	4.23	3.86	4.24
3	11.29	8.07	6.15	5.11	4.23	3.86	4.24
3.1	9.5	6.58	5.37	3.61	3.34	2.48	2.22
3.2	8.66	6.1	4.33	2.89	2.68	1.48	1.49
3.3	6.48	4.63	1.87	0.78	0.8	−0.04	0.84

Table 29: BHAC: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	13.99	15.01	18.51	19.1	22.68	27.82	34.62
1.5	6.81	7.51	8.42	10.01	13.14	18.41	23.73
1.7	0.99	1.45	1.93	3.13	7.11	10.48	15.85
1.9	−4.02	−4.08	−4.1	−2.32	0.71	4.45	8.52
2.1	−6.83	−8.6	−5.95	−5.43	−2.2	−0.06	4.07
2.3	−9.02	−8.97	−8.95	−7.89	−5.54	−2.11	1.18
2.5	−10.06	−10.81	−9.54	−9.79	−7.7	−4.73	−1.85
2.6	−10.66	−11.21	−10.54	−10.53	−7.63	−5.24	−0.9
2.7	−10	−11.05	−10.47	−9.88	−7.73	−5.45	−2.49
2.8	−9.19	−10.87	−10.79	−10.43	−7.61	−6.06	−2.24
2.9	−9.02	−9.7	−10.22	−9.75	−6.85	−4.57	−1.35
3	−8.07	−8.97	−8.88	−8.3	−6.38	−3.81	−0.42
3.1	−6.76	−7.47	−6.88	−7.01	−4.85	−2.81	−0.29
3.2	−4.78	−5.37	−5.51	−5.32	−3.15	−1.57	0.61
3.3	−4.05	−4.12	−5.02	−4.59	−2.41	−1	1.66

Table 30: BHAC: HDEEM energy savings in [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	−8.16	−5.94	−2.02	0.01	4.23	10.21	19.16
1.5	−11.22	−10.2	−8.68	−6.09	−2.06	4.18	10.9
1.7	−14.61	−13.36	−12.04	−10.75	−5.5	−1.11	5.12
1.9	−17.06	−16.89	−15.97	−14	−9.93	−4.78	0.03
2.1	−18.14	−19.31	−16.46	−14.98	−11.04	−8.08	−2.86
2.3	−18.4	−17.99	−17.3	−15.68	−12.51	−8.39	−4.13
2.5	−18.59	−18.5	−18.47	−17.12	−13.8	−11.17	−6.58
2.6	−18.2	−18	−15.63	−16.56	−13.7	−10.66	−6.47
2.7	−16.46	−17.03	−16.57	−15.02	−12.53	−9.89	−4.95
2.8	−15.57	−15.53	−15.27	−14.8	−12	−8.37	−5.38
2.9	−13.95	−14.92	−14.1	−13.18	−10.38	−7.88	−3.75
3	−12	−13.17	−12.66	−11	−8.89	−5.62	−3.28
3.1	−10.88	−11.65	−10.98	−9.68	−7.34	−4.06	−0.43
3.2	−8.42	−9.06	−9.02	−7.62	−4.23	−2.53	−0.22
3.3	−5.7	−6.96	−6.77	−5.97	−3.11	−0.88	2.53

Table 31: BHAC: RAPL energy savings in [%].

	default		-2 % limit	-5 % limit	-10 % limit	unlimited
Runtime [s]	261.32	**Performance penalty [%]**	1.86	4.22	9.97	18.26
Energy [kJ]		**Energy savings [%]**				
HDEEM	774.96	**HDEEM**	5.02	6.85	10.79	11.2
RAPL	584.25	**RAPL**	6.77	10.38	15.27	18
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	48.21	**HDEEM**	50.75	51.75	54.04	54.29
RAPL	63.94	**RAPL**	68.59	71.34	75.47	77.97
		CPU configuration				
		Core freq. [GHz]	3.3	2.9	2.8	2.6
		Uncore freq. [GHz]	1.6	2.0	1.6	1.4

Table 32: BHAC: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3.6 FIL

FIL is a module of the Einstein Toolkit (ET). The ET is a publicly available evolution framework that is designed, maintained and extended to enable numerical relativity simulations. Modules of the ET are known as Thorns. FIL is used in conjunction with other Thorns of the ET to solve the General Relativistic Magneto-Hydrodynamic (GRMHD) equations in 3-dimensions. FIL can therefor simulate magnetic fields in dynamical space-times allowing for the study of binary neutron star and black hole-neutron star mergers. Further information on both FIL and the ET can be found here [19] and here [20] respectively.

FIL uses the CPUs extensively, while most of the run is formed by a solver with a dynamic behaviour according to Figure 7. FIL is vectorized, however, it does not use 512bit vector instructions. The heatmaps show that energy savings cannot be achieved without a performance loss in the case of static frequency tuning. At the cost of 1.98 % in performance, we only save 2.83 % of compute node energy. This is not a worthwhile cost-benefit trade-off. It is possible to save up to 12.66 % of node energy (21.93 % of CPU energy, respectively), but this configuration would reduce the performance by about 27.44 %. These results are summarized in Table 37.

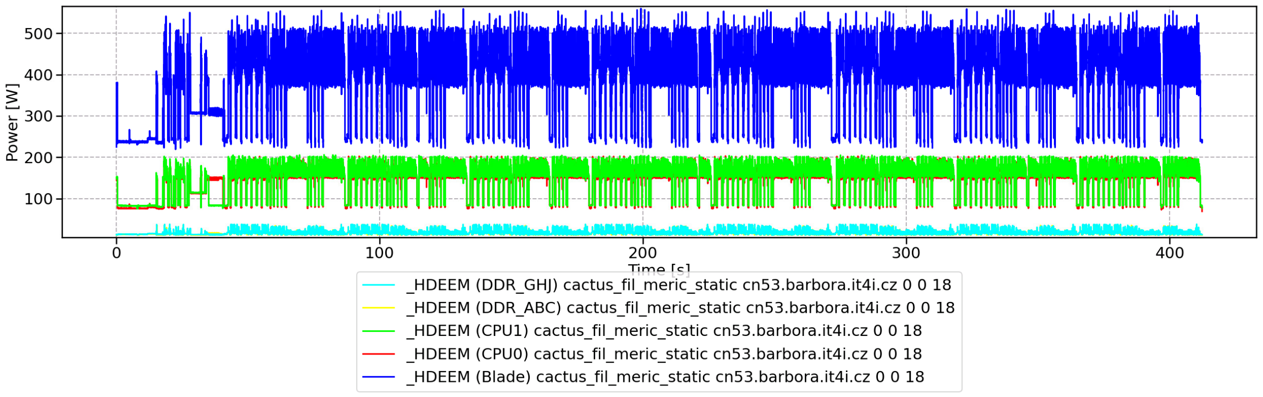

Figure 7: FIL power consumption timeline of a single node and its components.

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	105327876242346	105327876242346
SCALAR_SINGLE	1	32597447946	32597447946
128B_PACKED_DOUBLE	2	5877320250325	11754640500650
128B_PACKED_SINGLE	4	3875897448	15503589792
256B_PACKED_DOUBLE	4	320	1280
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	413.56	**Sum** [**FLOPs**]	117130617782014
		GFLOPs/s	283.23

Table 33: FIL: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	102.5	93.39	88.96	85.64	83.31	80.6	79.38
1.5	85.59	75.4	70.24	68.68	65.2	61.06	59.77
1.7	72.72	63.63	56.85	53.76	50.18	47.71	45.78
1.9	62.49	54	46.66	42.11	39.03	36.68	45.57
2.1	55.52	45.17	37.48	33.66	31.02	28.28	26.31
2.3	48.55	39.03	32.19	27.44	23.4	21.65	19.34
2.5	44.66	33.23	26.71	21.98	18.32	15.54	13.71
2.6	43.3	31.99	24.31	19.65	16.09	13.52	11.49
2.7	41.2	30.79	22.39	17.24	14.32	11.4	8.79
2.8	39.07	27.54	20.27	15.02	11.99	9.5	6.8
2.9	37.31	27.09	16.68	13.65	10.48	7.06	5.02
3	35.71	25.86	16.87	12.76	7.88	4.93	2.89
3.1	33.75	23.16	14.89	9.65	7.02	3.41	1.71
3.2	31.99	21.51	13.28	8.05	4.97	1.98	0.67
3.3	30.26	19.8	12.98	7.65	3.04	0.44	−0.25

Table 34: FIL: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	9.05	6.63	6.66	7.34	10.17	13.39	18.22
1.5	3.33	0.33	−0.28	0.98	2.94	4.85	9.13
1.7	0.34	−2.54	−4.3	−3.96	−2.49	−0.04	3.37
1.9	−2.74	−5.49	−7.72	−8.32	−6.96	−4.71	−1.34
2.1	−5.62	−9.52	−11.96	−12.36	−10.78	−8.91	−6.06
2.3	−5.67	−9.26	−11.47	−12.66	−12	−9.78	−7.39
2.5	−3.75	−8.83	−11.09	−12.28	−11.65	−10.21	−7.79
2.6	−2.88	−7.96	−11	−12.25	−11.59	−10.07	−7.93
2.7	−1.83	−6.52	−10.12	−11.74	−10.65	−9.55	−7.63
2.8	−0.87	−6.45	−9.37	−11.15	−10.36	−8.89	−7.33
2.9	1.1	−3.71	−8.97	−9.2	−8.62	−8.03	−5.9
3	3.35	−1.41	−5.82	−6.91	−7.6	−6.6	−4.78
3.1	5.46	0.05	−4.01	−6.15	−5.12	−4.79	−2.74
3.2	7.9	2.3	−1.96	−3.92	−3.54	−2.83	−0.87
3.3	11.02	5.15	1.96	−0.6	−1.42	−0.94	0.8

Table 35: FIL: Impact of the static tuning to HDEEM energy consumption [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4
1.3	−7.98	−9.28	−8.12	−6.63	−2.74	2.06	9.36
1.5	−12.05	−12.91	−12.98	−11.27	−7.3	−3.91	2.25
1.7	−12.5	−13.78	−14.82	−13.52	−10.56	−6.47	−2.04
1.9	−14.2	−15.53	−16.2	−16.27	−13.75	−10.22	−8.48
2.1	−20.14	−22.47	−23.35	−22.91	−20.24	−17.22	−13.18
2.3	−18.32	−20.39	−21.44	−21.93	−20.15	−16.85	−13.43
2.5	−14.91	−18.4	−19.73	−19.9	−18.26	−15.99	−12.61
2.6	−13.41	−17.14	−18.9	−19.37	−17.56	−15.34	−12.2
2.7	−11.72	−14.89	−17.38	−17.86	−16.31	−14.5	−11.52
2.8	−10.01	−14.06	−15.77	−16.9	−15.39	−12.91	−10.35
2.9	−6.98	−10.6	−14.61	−14.22	−12.81	−11.21	−8.22
3	−3.87	−7.46	−10.8	−11.18	−10.95	−9.05	−6.38
3.1	−0.8	−4.93	−8.06	−9.54	−7.73	−6.5	−3.61
3.2	2.66	−1.82	−5.22	−6.49	−5.36	−3.78	−1.11
3.3	6.85	2.02	−0.41	−2.34	−2.27	−1.14	1.12

Table 36: FIL: Impact of the static tuning to RAPL energy consumption [%].

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	413.56	**Performance penalty [%]**	1.98	4.93	9.5	27.44
Energy [kJ]		**Energy savings [%]**				
HDEEM	1278.75	**HDEEM**	2.83	6.6	8.89	12.66
RAPL	934.48	**RAPL**	3.78	9.05	12.91	21.93
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	91.60	**HDEEM**	94.27	98.07	100.53	104.87
RAPL	125.34	**RAPL**	123.27	127.81	143.93	160.56
		CPU configuration				
		Core freq. [GHz]	3.2	3.0	2.8	2.3
		Uncore freq. [GHz]	2.2	2.2	2.2	1.8

Table 37: FIL: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

3.7 ChaNGa

ChaNGa [21][22][23] is an N-body and smoothed particle magneto-hydrodynamics (SPMHD) code which is used to study a wide array of astrophysical systems. While the gravity and SPMHD algorithms are based on the gasoline [24] and pkdgrav [25] codes, the unique feature of ChaNGa is it's implementation of the Charm++ framework, which enables highly efficient parallel scaling. Charm++ employs overdecomposition to achieve this. That is to divide the work into many more pieces (chares/tree pieces) than you have processors and let the Charm++ runtime system load balance by appropriately assigning pieces to real processors.

ChaNGa's power timeline (Figure 8) shows coarse-grain regions where power consumption varies between approximately 320 W and 480 W. This clearly shows the dynamic behavior of the application's solver. From Table 41 one can see a large portion of SSE instructions, with very few AVX2 instructions. The most interesting results are presented in heatmaps (Tables 39, 40, and 41). Runtime (see Table 39) is very little affected by underclocking the uncore frequency all the way to 1.2 GHz. In addition, lowering the core frequency also has very little impact on the runtime if it stays above 2.3 GHz. This gives us a great window of opportunity for under-clocking the CPU and reaching good energy savings of 30 % with less than 2 % of runtime extension. The maximum energy savings are 36 %.

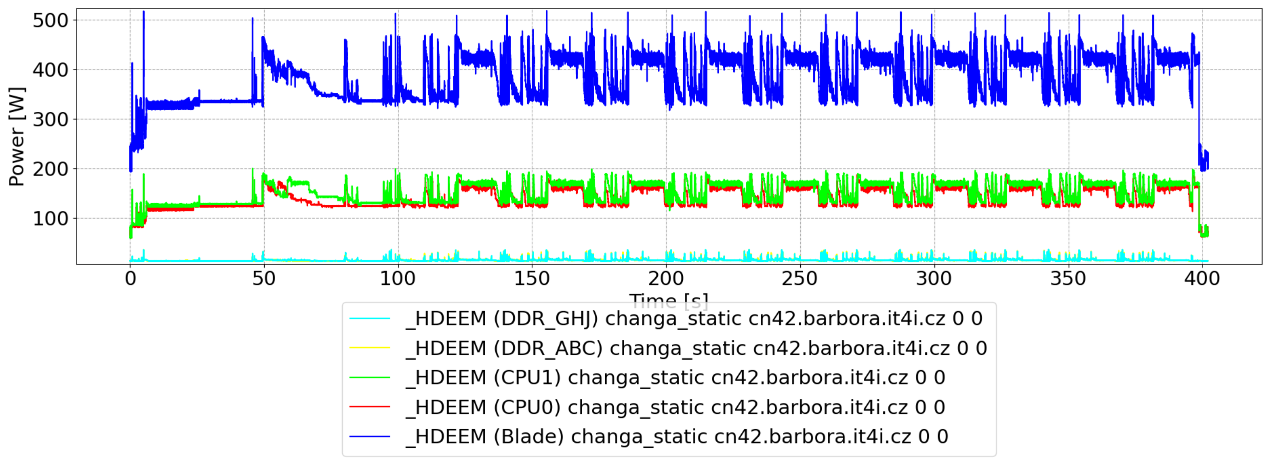

Figure 8: ChaNGa power consumption timeline of a single node and its components.

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	19197682911907	19197682911907
SCALAR_SINGLE	1	43537222949	43537222949
128B_PACKED_DOUBLE	2	70213594138546	140427188277092
128B_PACKED_SINGLE	4	62945292	251781168
256B_PACKED_DOUBLE	4	412	1648
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	402.07	**Sum** [**FLOPs**]	159668660194764
		GFLOPs/s	397.12

Table 38: ChaNGa: Floating-point operations measurement using various FP_ARITH_INST_RETIRED events of the Intel Xeon processor.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	14.76	15.27	13.27	13.25	13.53	12.44	12.78
1.5	9.94	8.39	8.64	7.24	9.82	9.31	8.78
1.7	7.27	4.52	4.82	4.15	5.86	4.12	3.4
1.9	9.05	4.46	4.12	3.81	1.52	3.11	2.35
2.1	3.07	3.4	3.53	3.24	1.58	1.98	1.7
2.3	2.64	2.79	1.22	1.4	1.09	1.12	1.16
2.5	2.81	1.37	1.26	1.01	0.31	0.69	0.8
2.6	2.79	1.8	1.47	1.16	−0.11	0.2	−0.14
2.7	3.22	1.51	0.55	0.49	−0.41	1.29	0.34
2.8	2.82	1.98	1.05	0.9	0.04	0.47	−0.33
2.9	2.52	1.51	0.7	1.19	0.1	0.95	−0.46
3	2.31	0.99	−0.1	−0.43	−1.14	−0.29	−1.37
3.1	2.29	0.41	−0.21	−0.43	0.92	−0.87	−0.32
3.2	2.6	0.7	−0.09	0.99	−0.24	−1.14	−2.26
3.3	2.86	0.11	−0.55	−0.25	−0.43	−0.33	−1.18

Table 39: ChaNGa: Impact of the static tuning to overall runtime [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	−35.78	−34.22	−33.49	−32.13	−29.62	−26.95	−23.38
1.5	−36.03	−35.26	−33.88	−33.12	−28.94	−26.63	−23.78
1.7	−35.57	−35.76	−34.11	−32.86	−30.14	−27.82	−25.15
1.9	−32.81	−34.2	−32.82	−31.43	−30.26	−26.49	−24.66
2.1	−34	−31.32	−30.93	−29.53	−28.09	−25.59	−22.7
2.3	−31.3	−29.79	−29.32	−27.84	−25.34	−22.3	−19.62
2.5	−28.51	−27.84	−26.46	−24.81	−22.87	−19.76	−16.47
2.6	−28	−26.18	−25.13	−23.17	−21.81	−18.9	−16.07
2.7	−27.85	−27.64	−26.53	−25	−22.82	−19.46	−16.9
2.8	−24.52	−25.64	−24.33	−22.86	−21.24	−17.74	−15.11
2.9	−23.87	−23.35	−22.06	−20.08	−18.67	−14.79	−12.55
3	−21.47	−20.82	−19.92	−18.84	−16.5	−13.34	−10.42
3.1	−18.46	−18.08	−16.88	−14.02	−12.11	−10.83	−6.68
3.2	−15.86	−15.61	−14.17	−12.05	−10.18	−7.93	−6.39
3.3	−12.21	−12.51	−10.9	−10.86	−7.94	−5.95	−4.4

Table 40: ChaNGa: Impact of the static tuning to HDEEM energy consumption [%].

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2	2.2	2.4
1.3	−41.31	−40.23	−38.49	−35.93	−32.56	−28.63	−23.81
1.5	−41.01	−39.51	−37.37	−35.96	−31.27	−28.11	−23.82
1.7	−38.64	−38.3	−36.2	−34.25	−31.06	−27.57	−24.28
1.9	−35.97	−36.43	−34.66	−33.02	−31.03	−25.83	−23.37
2.1	−36.36	−33.11	−32.03	−29.68	−28.11	−24.58	−20.53
2.3	−31.85	−30.26	−29.82	−27.45	−24.34	−20.25	−16.66
2.5	−29.16	−27.66	−25.72	−23.44	−20.79	−16.66	−12.58
2.6	−27.12	−25.65	−24.16	−21.8	−19.7	−15.76	−11.94
2.7	−28.13	−27.79	−26.04	−23.93	−21.07	−17.02	−13.3
2.8	−24.52	−25.19	−23.26	−21.41	−19.17	−14.38	−10.51
2.9	−23.16	−22.28	−20.44	−17.89	−15.49	−10.51	−7.27
3	−19.92	−18.73	−17.04	−15.56	−12.22	−8.42	−4.3
3.1	−15.84	−14.82	−12.85	−9.79	−6.86	−4.94	0.37
3.2	−12.32	−11.61	−9.46	−7.08	−4.63	−1.11	1.14
3.3	−7.45	−7.3	−5	2.35	−0.11	5.83	3.96

Table 41: ChaNGa: Impact of the static tuning to RAPL energy consumption [%].

	default		**-2 % limit**	**-5 % limit**	**-10 % limit**	**unlimited**
Runtime [s]	402.07	**Performance penalty [%]**	1.52	4.52	9.94	9.94
Energy [kJ]		**Energy savings [%]**				
HDEEM	1221.90	**HDEEM**	30.26	35.76	36.03	36.03
RAPL	886.78	**RAPL**	31.03	38.30	41.01	41.01
Eff. [MFLOPs/W]		**Eff. [MFLOPs/W]**				
HDEEM	130.05	**HDEEM**	187.37	203.40	204.25	204.25
RAPL	180.06	**RAPL**	261.05	291.78	305.23	305.23
		CPU configuration				
		Core freq. [GHz]	1.9	1.7	1.5	1.5
		Uncore freq. [GHz]	2.0	1.4	1.2	1.2

Table 42: ChaNGa: energy savings for various performance degradation trade-offs, and without any runtime limit. CPU configuration represents a configuration applied to each socket.

4 Conclusion

Table 43 presents the summary of the energy savings achieved by all the applications. We present three scenarios, each with different run-time penalty allowed. The runtime penalty stands for runtime extension when the processor runs on lower core and uncore frequencies. One can see that it is possible to save between 3–6 % of energy without impacting the runtime of the application (max. runtime extension of 2 %). When runtime can be extended by 5 % the savings are in the range of 6.6–9.4 %. The best results were achieved for the ChaNGa code, which is able to achieve 30% of energy savings with almost no impact on its performance.

The maximum possible energy savings are 11–13 % and 36% for ChaNGa, but in this case the runtime is significantly extended. Still, these configurations are usable in the following scenarios: 1.) current cluster utilization is low and extending the runtime does not affect the amount of scientific results produced by it; 2.) cluster is running under strict power limit due to condition in the power distribution network; or 3.) the cost of electricity at a given moment is significantly higher, which means that OPEX is significantly higher than CAPEX.

	Energy savings with maximum given runtime extension			
Code name	max. 2%	max. 5%	max. 10%	unlimited
Pluto	5.5 %	7.3 %	10.5 %	11.5 %
OpenGadget	5.4 %	6.8 %	7.4 %	11.8 %
iPic3D	5.7 %	9.4 %	11.0 %	12.9 %
RAMSES	7,4 %	10.1 %	11.1 %	13.5 %
BHAC	5.0 %	6.8 %	10.8 %	11.2 %
FIL	2.8 %	6.6 %	8.9 %	12.6 %
ChaNGa	30.2 %	35.8 %	36.0 %	36.0 %

Table 43: Energy savings achieved on the Barbora cluster under different maximum allowed runtime extension. The results are for static tuning of the CPU core and uncore frequency and measured using HDEEM.

4.1 Future work

Static tuning is only the first step in improving the energy efficiency of SPACE CoE applications. In the future, we will explore the dynamic tuning technique. Dynamic tuning means that we find optimal configurations for different regions of the application. Then, as an application moves from one region to another, the MERIC runtime will change the hardware settings to the optimal ones for that given region. Similar tuning can be also performed for GPU accelerators. We support NVIDIA GPUs for both hardware tunning and energy consumption measurements. Some of these evaluations will be described in more detail and presented in the next deliverable related to energy optimization (D2.4, due Dec. 2024).

Table 44 shows the statistics of the code vectorization that has been measured for the execution of the application under the default system settings (no frequency tuning was performed). We can see that only one code (BHAC) uses the AVX/AVX2 instruction set. Other applications use SSE or scalar floating-point operation instructions. This is an important analysis for future inter-node optimizations.

	PLUTO	OpenGadget	iPic3D	BHAC	FIL	ChaNGa	RAMSES
Scalar	100,0%	89,2%	99,7%	10,8%	90,0%	12,1%	93,5%
SSE	0,0%	10,7%	0,3%	15,9%	10,0%	87,9%	6.5%
AVX/AVX2	0,0%	0,1%	0,0%	73,3%	0,0%	0,0%	0,0%
AVX512	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%

Table 44: Distribution of FLOPs per instruction set. This represents the level of vectorization of each application.

References

[1] D. L. Hill, D. Bachand, S. Bilgin, R. Greiner, P. Hammarlund, T. Huff, S. Kulick, and R. Safranek, "The uncore: A modular approach to feeding the high-performance cores," *Intel Technology Journal*, vol. 14, no. 2, pp. 30–49, 2010.

[2] D. Hackenberg, T. Ilsche, J. Schuchart, R. Schöne, W. Nagel, M. Simon, and Y. Georgiou, "Hdeem: High definition energy efficiency monitoring," in *Energy Efficient Supercomputing Workshop (E2SC)*, 11 2014, pp. 1–10.

[3] C. Gough, I. Steiner, and W. Saunders, *CPU Power Management*. Berkeley, CA: Apress, 2015, pp. 21–70. [Online]. Available: https://doi.org/10.1007/978-1-4302-6638-9_2

[4] O. Vysocky, M. Beseda, L. Riha, J. Zapletal, M. Lysaght, and V. Kannan, "Meric and radar generator: Tools for energy evaluation and runtime tuning of hpc applications," in *High Performance Computing in Science and Engineering*, T. Kozubek, M. Cermak, P. Tichy, R. Blaheta, J. Sistek, D. Lukas, and J. Jaros, Eds. Cham: Springer International Publishing, 2018, pp. 144–159.

[5] J. Schuchart, M. Gerndt, P. G. Kjeldsberg, M. Lysaght, D. Horák, L. Říha, A. Gocht, M. Sourouri, M. Kumaraswamy, A. Chowdhury, M. Jahre, K. Diethelm, O. Bouizi, U. S. Mian, J. Kružík, R. Sojka, M. Beseda, V. Kannan, Z. Bendifallah, D. Hackenberg, and W. E. Nagel, "The readex formalism for automatic tuning for energy efficiency," *Computing*, vol. 99, no. 8, pp. 727–745, Aug 2017.

[6] SPACE, "Deliverable D.2.1, Performance profiling and benchmarking," December 2023, https://www.space-coe.eu/files/SPACE_D2_1_Performance_profiling_and_benchmarkingfinal.pdf.

[7] "The pluto code for astrophysical gasdynamics," http://plutocode.ph.unito.it/, accessed: 28.10.2023.

[8] G. Lapenta, "ipic3d," 2015, accessed: 2023-10-29. [Online]. Available: https://github.com/CmPA/iPic3D.git

[9] S. Markidis, G. Lapenta *et al.*, "Multi-scale simulations of plasma with ipic3d," *Mathematics and Computers in Simulation*, vol. 80, no. 7, pp. 1509–1519, 2010.

[10] G. Lapenta, "Exactly energy conserving semi-implicit particle in cell formulation," *Journal of Computational Physics*, vol. 334, pp. 349–366, 2017.

[11] R. Teyssier, "Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES," , vol. 385, pp. 337–364, Apr. 2002.

[12] "Reposiroty of the ramses code on bitbucket," https://bitbucket.org/rteyssie/ramses/src/master/, accessed: 29.10.2023.

[13] O. Porth, H. Olivares, Y. Mizuno, Z. Younsi, L. Rezzolla, M. Moscibrodzka, H. Falcke, and M. Kramer, "The Black Hole Accretion Code," *Computational Astrophysics and Cosmology*, vol. 4, 2017.

[14] H. Olivares, O. Porth, J. Davelaar, E. R. Most, C. M. Fromm, Y. Mizuno, Z. Younsi, and L. Rezzolla, "Constrained transport and adaptive mesh refinement in the black hole accretion code," *Astronomy & Astrophysics*, vol. 629, p. A61, 2019.

[15] "The Black Hole Accretion Code – Documentation," https://bhac.science, accessed: 2023-11-01.

[16] "Repository of BHAC on GitLab," https://gitlab.itp.uni-frankfurt.de/BHAC-release/bhac, accessed: 2023-11-01.

[17] "The MPI - Adaptive Mesh Refinement - Versatile Advection Code – Documentation," https://amrvac.org/, accessed: 2023-11-01.

[18] "Repository of MPI-AMRVAC on GitHub," https://github.com/amrvac/amrvac, accessed: 2023-11-01.

[19] Z. B. Etienne, V. Paschalidis, R. Haas, P. Mösta, and S. L. Shapiro, "IllinoisGRMHD: An Open-Source, User-Friendly GRMHD Code for Dynamical Spacetimes," *Class. Quant. Grav.*, vol. 32, p. 175009, 2015.

[20] M. Zilhão and F. Löffler, "An Introduction to the Einstein Toolkit," *Int. J. Mod. Phys. A*, vol. 28, p. 1340014, 2013.

[21] "Repository of the changa code on github," https://github.com/N-BodyShop/changa, accessed: 28.10.2023.

[22] P. Jetley, F. Gioachin, C. Mendes, L. V. Kale, and T. R. Quinn, "Massively parallel cosmological simulations with ChaNGa," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium 2008*, 2008.

[23] H. Menon, L. Wesolowski, G. Zheng, P. Jetley, L. Kale, T. Quinn, and F. Governato, "Adaptive techniques for clustered N-body cosmological simulations," *Computational Astrophysics and Cosmology*, vol. 2, p. 1, Mar. 2015.

[24] J. W. Wadsley, B. W. Keller, and T. R. Quinn, "Gasoline2: a modern smoothed particle hydrodynamics code," , vol. 471, no. 2, pp. 2357–2369, Oct. 2017.

[25] J. G. Stadel, "Cosmological N-body simulations and their analysis," Ph.D. dissertation, University of Washington, Seattle, Jan. 2001.