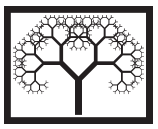

Proceedings of the Sixth International Conference on
Parallel, Distributed, GPU and Cloud Computing for Engineering,
P. Iványi and B.H.V Topping (Editors)
Civil-Comp Press, Stirlingshire, Scotland

Paper 32

Energy consumption evaluation of Blender's image renderer in HPC environment

M. JAROS, O. VYSOCKY, P. STRAKOS, M. SPETKO
IT4Innovations, VSB - Technical University of Ostrava, Czech Republic

Abstract

In our contribution we evaluate the energy consumption optimization of image rendering on a typical architecture of an HPC system. We use the renderer CyclesPhi, which is our own modified version of the Cycles renderer from the Blender 3D creation suite. CyclesPhi fits the HPC environment in such a way that it runs as a client on one or multiple nodes, and efficiently utilizes the cluster through optimal load balancing. In order to reduce the energy consumption of a scene rendering we used MERIC, our own developed library for HPC application profiling and runtime tuning. MERIC searches for the configuration of hardware, system software, and application parameters which can provide minimal energy consumption for each manually instrumented region inside the analysed application. Thusly we instrumented the Blender client and analysed the rendering task. On Haswell architecture (two Intel Xeon E5-2680v3 processors per node) we were able to reduce energy consumption by 9% while extending the rendering time by 21%. If a less energy conservative setting was applied, we would save 4.8% of energy whilest only prolonging the rendering time by 4%.

Keywords: rendering, Blender Cycles, energy efficient computing, MERIC, high performance computing

1 Introduction

Image rendering is an example of a task that can easily utilise a whole cluster, and runs for a very long time. Therefore it consumes much energy. Even minimal power reduction can result in significant energy and money savings, as well as reduced carbon footprint.

So far the main focus concerning the energy consumption in image rendering has concentrated mainly on mobile platforms. The energy consumption of computationally extensive tasks is crucial on mobile devices because it directly influences battery life. One example of a possible solution to this phenomena is presented in [1]. The authors try to find the optimal rendering setting to conserve maximum energy whilst also retaining maximal visual quality. The method through which they reach this objective sets up the Pareto frontier and solves the task as optimisation problem. Although it is a very reasonable way of achieving the goal of saving power, it is merely a compromise between power savings and lowering of visual quality.

There are several related strains of research in energy efficient High Performance Computing (HPC) working on reduction of system resources while conserving the application's time to solution. Haidar et al [2] use PAPI power capping, and Kimura et al [3] and the Adagio system [4] use dynamic voltage scaling. However, the consumed energy might be reduced despite extending the application runtime. For such a case we cannot consider only the energy consumed by the CPUs, but must measure the consumption of the whole node. We put our focus on image rendering, and in comparison with the majority of available solutions in mobile devices, we elaborate on changing the specific hardware settings of a cluster. We aim to change the CPU frequencies and consequently influence both rendering time and energy consumption. By dealing with this specific setting we can retain maximal image quality and reduce power consumption.

2 Extension of Blender Cycles renderer

Blender is an open-source 3D software that ranges from creation of 3D scenes to their photo-realistic visualizations [5]. It is equipped with several rendering engines, but the one that is mainly used for production of high quality results is Cycles. It is a path-tracing based engine that supports both off-line and interactive rendering.

Blender can be extended with a variety of plug-ins. These are usually written as a combination of C++ and Python code. C++ is used to write computationally extensive parts, while Python is used mainly for GUI implementation. In our contribution we have used an extended version of the Cycles engine in the form of Blender's plug-in to support remote utilisation of HPC (High Performance Computing) resources and to allow optimization of the energy consumption of the rendering task.

We call our rendering plug-in CyclesPhi [6]. It was originally developed to utilise HPC clusters that are equipped with regular CPU nodes and also with Intel Xeon Phi accelerated nodes. By such extension of the original Cycles engine we can achieve full utilisation of a typical HPC cluster.

The MPI version of CyclesPhi was introduced in [7]. In the current version we have added support for communication over the SSH protocol using socket technology. It allows remote off-line or interactive rendering from a personal computer. The CyclesPhi runs as a Blender client and its communication concept is depicted in Figure 1, whereas in Figure 2 the type of data being sent is shown. Using socket communication, we can send a scene directly to a cluster or we can save it to a file. This file also contains data from the scene preprocessing before the rendering.

3 Energy measurement

In order to reduce energy consumption of a scene rendering we have used MERIC library [8] developed at IT4Innovations for HPC application resources consumption evaluation. It can also tune selected hardware parameters during the application runtime. The library applies a READEX project [9] approach by searching for the optimal configuration of hardware and application parameters for separate regions inside an application, based on manual instrumentation. In the same way we manually instrumented the blender client to analyse the rendering part.

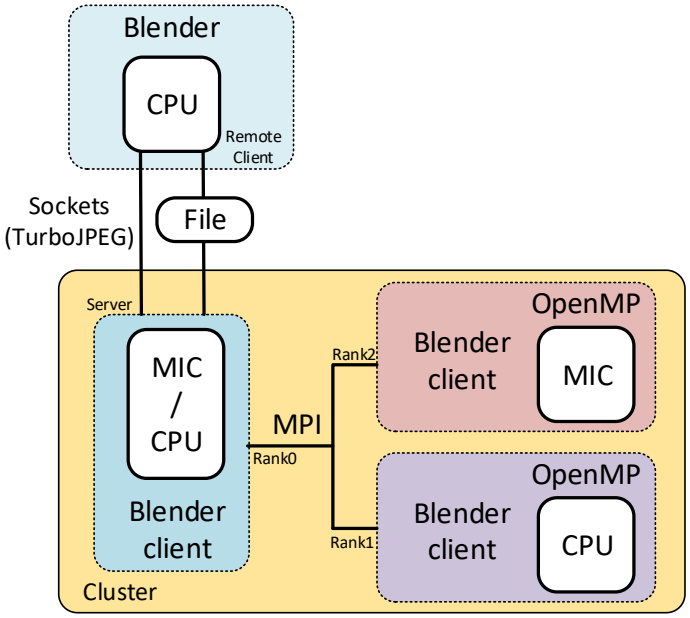

Figure 1: Concept of a client and a server utilizing one compute node of a cluster.

The library allows for Dynamic Voltage and Frequency Scaling (DVFS) and Uncore Frequency Scaling (UFS). Uncore frequency refers to frequency of subsystems in the physical processor package that are shared by multiple processor cores e.g., L3 cache or on-chip ring interconnect. In this manner we can control the separate parts of the chip more effectively in comparison to automated power capping [2]. The importance of UFS is presented in the following section.

As we proposed, we are not focusing on persistent application runtime with reduced hardware resources, so we do not consider only the CPUs energy consumption, but evaluate the whole node.

$$E = energy_{cpu} + baseline * time \tag{1}$$

The energy measurement of the whole node is defined be Equation 1, where energy consumed by CPUs is measured from Intel Running Average Power Limit (RAPL) counters, and the power baseline is defined from data provided by the Intelligent Platform Management Interface (IPMI). For evaluated Haswell architecture the power baseline is specified in Table 1.

For the energy measurements we divided the rendering task into three specific regions. We mark them as LOAD, RENDER and SAVE. They are self explanatory, LOAD and SAVE provide functionality for getting data to and from the rendering task while the RENDER region consists of the loop for path-tracing over each pixel. The load balancing within the RENDER region is improved by the OpenMP scheduler (see Figure 3). The size of x_0 and y_0 was 1 in all cases.

In our energy measurements we have used only the RENDER region, because the loading and saving of data requires almost no time compared to rendering.

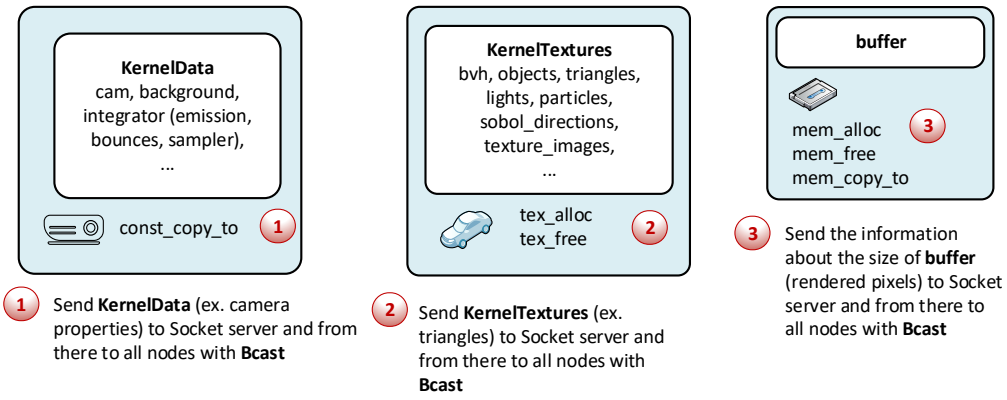

Figure 2: Communicated date within rendering task in Blender.

Architecture	Haswell
Baseline [W]	70
TDP [W]	240 (2x E5-2680v3)
Total power [W]	310

Table 1: Baseline and Thermal Design Power (TDP) for Haswell Intel platforms measured by IPMI.

We have also skipped rendering over more than one node, since rendering has a linear scalability, see Figure 4. Therefore to obtain the approximate energy consumption of the whole cluster we can multiply the consumption of one node by the total number of cluster nodes.

4 Experiment

For the experiment we have selected four different scenes of different complexity. Scene previews in high sample rates are depicted in Figure 5. Detailed descriptions of chosen scenes are in Table 2. Relatively small sample rates during the measurements of energy consumption are used to set the average rendering time around 60 seconds per one HW configuration. The detailed measurement contains about 90 HW combinations for one scene.

4.1 Energy consumption measurement and optimization

Analysis of the energy consumption was done on IT4Innovations' Salomon cluster [10], which is based on nodes with two Intel Xeon E5-2680v3 (Haswell-EP) processors with 12 cores each.

The advantage of the Haswell processor in terms of our analysis is the availability of DVFS (core freq.) and UFS (uncore freq.). Another benefit is the wide frequency range of both. Intel Haswell uncore frequency can be set in the range of 1.2–3.0 GHz and core frequency in the range of 1.2–2.5 GHz. For the evaluated Haswell processor there is a CPU core turbo frequency available. CPUs in general have a wide range for the turbo frequency (Haswell 2.5–

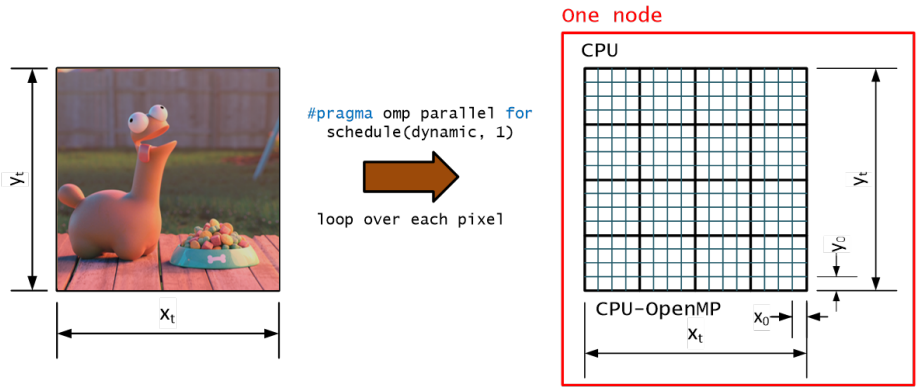

Figure 3: Image distribution using OpenMP on multiple threads of a compute node.

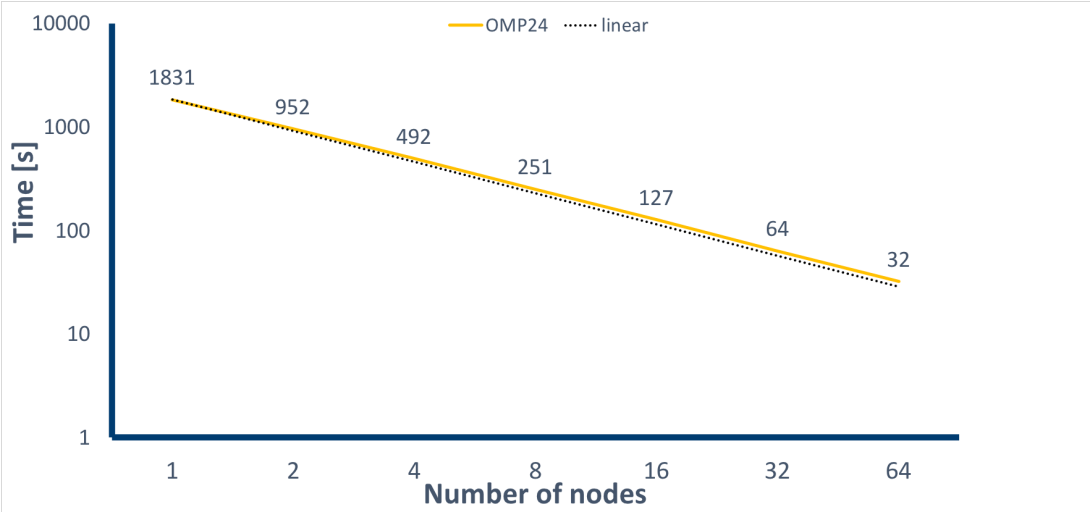

Figure 4: The multi-node scalability of the rendering engine. The test was made on the Classroom scene with 1200 samples using 2× Haswell with 24 OpenMP threads.

3.3 GHz) because of the rapid temperature rise when running at the available maximum, and the governor reducing the frequency to reduce the chip temperature accordingly.

In our tests we measured the hardware performance counters using PAPI and we watched the PAPI_TOT_CYC counter, which counts the number of CPU cycles. According to this counter, the average CPU frequency of a Haswell processor when in turbo mode was 2.79 GHz. In the graphs and heat-maps that follow the 2.8 GHz CPU core frequency represents Haswell's turbo frequency.

In Table 3 we show the default and optimal hardware settings to reach the minimum energy consumption and its impact on rendering time. Reducing the number of OpenMP threads would not have a positive impact on energy consumption in the case of a linearly scaling application because it prolongs the application runtime accordingly. Therefore all presented results are using all available threads (24 threads/node). We were able to gain 7.81–9.08 %

Figure 5: Test scenes - The Daily Dweebs by the Blender Foundation , Classroom by Christophe Seux, Fishy Cat by Manu Jarvinen, and Pabellon Barcelona by Claudio Andres (from left to right and top to bottom).

Scene	Dweebs	Classroom	Fishy Cat	Pabellon B.
Frame	150	1	1	1
Verts	4643383	127812	218761	22432
Faces	4160837	126231	326855	19910
Tris	8066390	242474	436998	40189
Objects	239	301	27	102
Lamps	9	4	2	1
Mem	6738.78MB	797.11MB	908.02MB	303.13MB
Resolution	1920x1080	1920x1080	1002x460	1280x720
Samples	12	6	3	7

Table 2: Description of evaluated scenes.

energy savings when using 1.6 or 1.8 GHz uncore and 2.4 or 2.5 GHz core CPU frequency. The runtime in such a scenario was extended by approximately 16.19–22.51 %.

We have selected the Classroom scene that provided maximal energy savings to show in detail the impact of the applied hardware configuration. The influence of reduced resources on the runtime is shown in Figure 6 while the influence on the consumed energy is depicted in Figure 7.

Based on our findings we realize that extending the runtime by about 20 % might be considered unacceptable in some cases, hence one may set a configuration that does not effect the runtime as much. All other available configurations with the respective impact on energy consumption and runtime can be read from the Table 4 and Table 5. From the values in the aforementioned tables it is obvious that the highest impact on the rendering runtime the the CPU core turbo frequency. So obviously ignoring DVFS and using UFS only is a solution for how to gain about 4.8 % energy savings whilst only extending the runtime by a more acceptable 4 %.

So far the presented results have been measured on nodes with a Direct Liquid Cooling (DLC) system, however the Salomon cluster also has Haswell nodes with an Air Cooling (AC) system. The cooling system does not effect the rendering runtime, but it has a distinct impact on the energy consumption of the node. The systems in the default configuration and with

Scene	Default run	Best HW configuration	Optimized run
Classroom	18698.64 J 65.38 s	1.6 GHz (UnCF), 2.4 GHz (CF)	17000.55 J (9.08 %) 79.06 s (-20.93 %)
Dweebs	18541.11 J 64.03 s	1.8 GHz (UnCF), 2.4 GHz (CF)	17093.31 J (7.81 %) 77.80 s (-21.51 %)
Fishy Cat	18210.75 J 63.23 s	1.8 GHz (UnCF), 2.5 GHz (CF)	16671.43 J (8.45 %) 73.47 s (-16.19 %)
Pabellon B.	17067.56 J 60.09 s	1.8 GHz (UnCF), 2.4 GHz (CF)	15731.53 J (7.83 %) 73.02 s (-21.51 %)

Table 3: Overall application evaluation of different scenes on compute node with Haswell processors from the energy consumption point of view.

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	−132.8	−130.6	−129.4	−128.2	−127	−126.3	−125.7	−125.6	−124.8	−124.7
1.4	−101.8	−99.4	−97.9	−96.7	−96	−95.2	−94.3	−94.2	−93.7	−93.2
1.6	−78.8	−76.5	−74.8	−73.9	−72.7	−71.7	−71.2	−70.6	−70.2	−69.9
1.8	−61	−58.6	−56.9	−55.2	−54.5	−53.4	−52.9	−52.5	−52	−51.7
2	−46.9	−44.5	−42.2	−41	−39.7	−39.3	−38.4	−37.9	−37.6	−37.1
2.2	−35.2	−32.6	−30.9	−29	−28	−27.3	−26.7	−26.1	−25.8	−25.3
2.4	−25.5	−22.9	−20.9	−19.6	−18.5	−17.5	−16.7	−16.2	−15.8	−15.3
2.5	−21.3	−18.8	−16.8	−15.4	−14.1	−13.2	−12.4	−11.9	−11.5	−11
2.8	−10.3	−7.7	−5.6	−4	−2.9	−2.2	−1.4	−0.5	−0.1	0

Table 4: Heat-map representing classroom rendering **runtime extension** [%] in different hardware configurations.

the applied optimal settings are compared in the Table 6. The comparison is made against the Haswell (AC) default configuration. From this perspective, reducing the hardware resources on an AC Haswell processor may gain 3–6 % energy savings, but upgrading the cooling system to DLC increases the saving to 11–14%.

5 Conclusions

Image rendering presents an application with a single compute intensive kernel that may significantly reduce its power consumption due to static resources throttling.

We have carried out the Blender client analysis on an Intel Haswell processor using the MERIC library for DVFS and UFS to reach maximum energy savings. The applied approach can reduce the energy consumption by up to 9 %, although with a significant runtime extension. To reduce the impact on runtime we propose to use UFS only, which results in a 4.8 % energy saving with a 4 % runtime extension.

More significant energy savings were achieved when we compared the results in terms of different types of cooling systems on the same Haswell architecture. Directly Liquid Cooled technology gives energy savings of 14 % with a 22 % longer runtime compared to Air Cooled HSW technology.

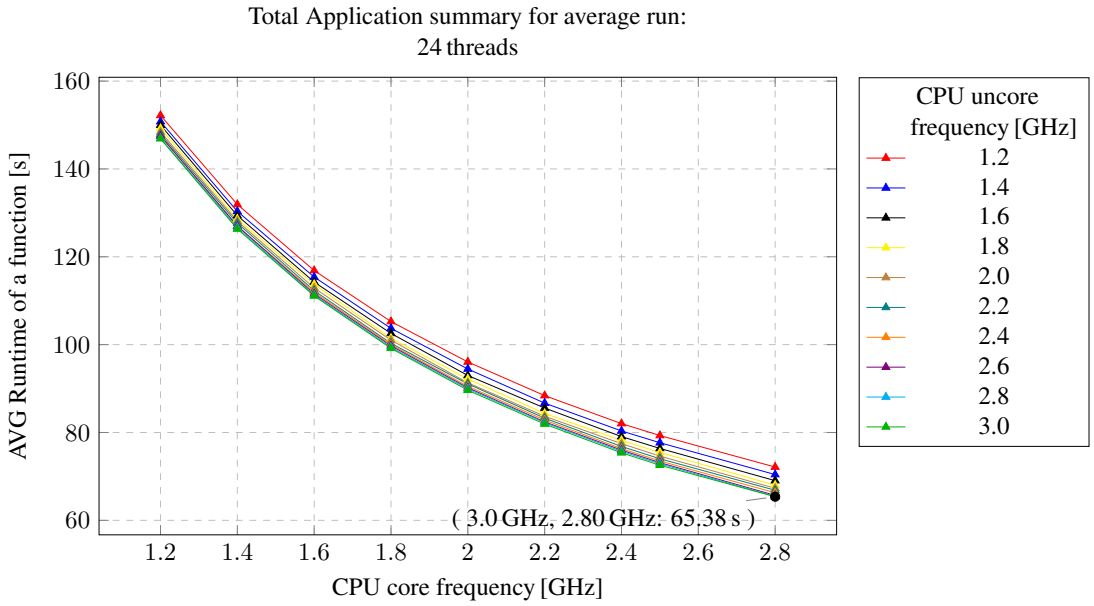

Figure 6: Graph of the rendering runtime [s] of the Classroom scene in different hardware configurations.

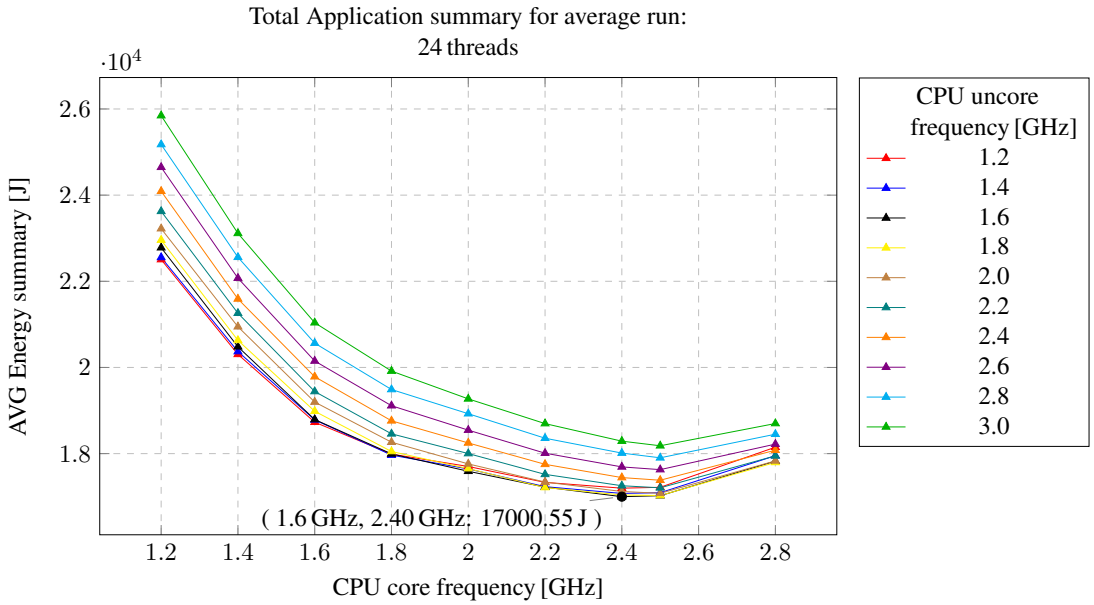

Figure 7: Graph of the energy [J] consumed during rendering of the Classroom scene in different hardware configurations.

Acknowledgements

This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project “IT4Innovations National Supercomputing Center – LM2015070”. This work was also supported by The Min-

uncore [GHz] / core [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	−20.3	−20.6	−21.8	−22.8	−24.2	−26.3	−28.8	−31.8	−34.6	−38.2
1.4	−8.6	−8.9	−9.5	−10.3	−12	−13.7	−15.5	−18	−20.6	−23.6
1.6	−0.2	−0.5	−0.5	−1.5	−2.7	−4	−5.8	−7.8	−10	−12.5
1.8	3.8	3.9	3.8	3.5	2.3	1.3	−0.3	−2.2	−4.2	−6.5
2	5.3	5.6	5.9	5.6	5	3.7	2.4	0.8	−1.2	−3.1
2.2	7.3	7.8	7.9	7.9	7.3	6.3	5.1	3.7	1.8	0
2.4	8	8.7	9.1	8.9	8.4	7.7	6.7	5.4	3.7	2.2
2.5	7.9	8.6	9	8.9	8.7	8	7	5.7	4.3	2.8
2.8	2.9	4	4.7	4.8	4.6	4	3.3	2.6	1.3	0

Table 5: Heat-map representing classroom rendering **energy savings** [%] in different hardware configurations.

Platform	Default settings	Optimal settings	Energy and time savings
Classroom scene			
HSW AC	**19318 J; 65 s**	18477 J; 79 s	E+4%; T-22%
HSW DLC	18699 J; 65 s	**17001 J; 79 s**	E+14%; T-22%
Dweebs scene			
HSW AC	**19072 J; 64 s**	18249 J; 78 s	E+4%; T-22%
HSW DLC	18541 J; 64 s	**17093 J; 78 s**	E+12%; T-22%
Fishy Cat scene			
HSW AC	**18794 J; 63 s**	17755 J; 73 s	E+6%; T-16%
HSW DLC	18211 J; 63 s	**16672 J; 73 s**	E+13%; T-16%
Pabellon B. scene			
HSW AC	**17833 J; 60 s**	17220 J; 73 s	E+3%; T-22%
HSW DLC	17068 J; 60 s	**15732 J; 73 s**	E+11%; T-22%

Table 6: Runtime and energy consumption comparison of Haswell nodes with Air Cooling (HSW AC) and with Direct Liquid Cooling system (HSW DLC) in the default and optimal settings.

istry of Education, Youth and Sports from the National Programme of Sustainability (NPS II) project "IT4Innovations excellence in science - LQ1602".

References

[1] R. Wang, B. Yu, J. Marco, T. Hu, D. Gutierrez, H. Bao, "Real-time rendering on a power budget", *ACM Transactions on Graphics*, 35(4), 2016, URL `www.scopus.com`, Cited By :2.

[2] A. Haidar, H. Jagode, P. Vaccaro, A. YarKhan, S. Tomov, J. Dongarra, "Investigating power capping toward energy-efficient scientific applications", *Concurrency and Computation: Practice and Experience*, 0(0): e4485, URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4485`.

[3] H. Kimura, M. Sato, Y. Hotta, T. Boku, D. Takahashi, "Emprical study on Reducing Energy of Parallel Programs using Slack Reclamation by DVFS in a Power-scalable High Performance Cluster", *2006 IEEE International Conference on Cluster Computing*, pages 1–10, 2006.

[4] B. Rountree, D.K. Lowenthal, B.R. de Supinski, M. Schulz, V.W. Freeh, T.K. Bletsch, "Adagio: making DVS practical for complex HPC applications", in *ICS*, 2009.

[5] B.O. Community, "Home of the Blender project - Free and Open 3D Creation Software", 2018, URL `http://www.blender.org`.

[6] M. Jaros, L. Riha, "CyclesPhi", 2016, URL `https://blender.it4i.cz`.

[7] M. Jaros, L. Riha, T. Karasek, P. Strakos, D. Krpelik, "Rendering in Blender Cycles Using MPI and Intel&Reg; Xeon Phi&Trade;", in *Proceedings of the 2017 International Conference on Computer Graphics and Digital Image Processing*, CGDIP '17, pages 2:1–2:5. ACM, New York, NY, USA, 2017, ISBN 978-1-4503-5236-9, URL `http://doi.acm.org/10.1145/3110224.3110236`.

[8] O. Vysocky, M. Beseda, L. Riha, J. Zapletal, V. Nikl, M. Lysaght, V. Kannan, "Evaluation of the HPC Applications Dynamic Behavior in Terms of Energy Consumption", in P. Ivanyi, B.H.V. Topping, G. Varady (Editors), *Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*. Civil-Comp Press, Stirlingshire, UK, Paper 3, 2017. doi:10.4203/ccp.111.3.

[9] Y. Oleynik, M. Gerndt, J. Schuchart, P.G. Kjeldsberg, W.E. Nagel, "Run-Time Exploitation of Application Dynamism for Energy-Efficient Exascale Computing (READEX)", in C. Plessl, D. El Baz, G. Cong, J.M.P. Cardoso, L. Veiga, T. Rauber (Editors), *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*, pages 347–350. IEEE, Piscataway, Oct 2015.

[10] "Hardware Overview - IT4Innovations Documentation", 8 2017, URL `https://docs.it4i.cz/salomon/hardware-overview/`.