Energy Savings in Cluster Computing with Heterogeneous Processor Architectures

Stephan Pilwax
2110781003@fh-burgenland.at

Thomas Stadler
2110781014@fh-burgenland.at

Klaus Thüringer
2110781007@fh-burgenland.at

***Abstract*—The goal of this paper is to introduce the idea of potential energy savings by using heterogeneous clusters. Different types of processors can provide advantages in certain use cases while showing lower performance in other tasks. By effectively exploiting the strengths of the processors used and thus increasing their effectiveness, it should be possible to save energy in the operation of the software. For the research on the energy-saving potential of heterogeneous clusters, an experiment is conducted. In the experiment, benchmarks are performed on devices with different processor architectures, and the power consumption is measured to calculate the energy efficiency of the tested task. A heterogeneous cluster prototype is built to demonstrate the feasibility and limitations that arise during the process. The results should provide further information about the usefulness of heterogeneous clusters for saving energy.**

***Keywords*—ARM, x86, CPU Architectures, Energy Efficiency, Heterogeneous Cluster Computing**

I. Introduction

The use of processors with Arm Big Little technology has already shown a possible reduction in power consumption of operating a device with heterogeneous Central Processing Unit (CPU) cores. Processors that use ARM Big Little architectures have both performance and efficiency cores, between which processes are distributed for maximum efficiency which leads to the lower energy consumption of the device.

Instead of integrating two different types of cores into one processor, the idea is to build a heterogeneous cluster with different CPU architectures to achieve the same effect.

The use of containers offers many advantages when it comes to running applications. They allow a high degree of independence from the base system and still offer a lower performance overhead than virtual machines. However, since they are also a form of virtualization, they are not independent of the processor architecture and must be adapted for each type of processor architecture. This is one of the problems that limit the cross-platform use of containers. With traditional software operations, it was often very difficult to move application deployments from one to another completely different system. The use of cloud technologies and especially containers can make this easier to implement. Individual containers can be moved between physical or virtual machines by container orchestration software. At the same time their computing power can be combined by using load balancing techniques.

Different processor architectures also provide advantages in certain tasks and their performance can also be tailored to specific applications. This could be used as a smart and energy-efficient way of load balancing and to reduce the energy consumption of the running applications. As a result, operations can be made more efficient and operating costs can be lowered. To answer this question, this paper will compare two processor architectures and determine their energy efficiency in specific tasks.

A useful application for saving energy by using heterogeneous clusters efficiently are data centers. By reducing the energy used to operate the server infrastructure, the active operating costs for the server hardware can be lowered. However, the reduced energy consumption also creates other benefits. Much of the energy consumed by computers is converted into heat, which in the case of a data center requires a cooling system. With a reduction in energy consumption and a higher energy efficiency, the required cooling capacity for the overall system also decreases. This in turn also reduces the required amount of cooling and thus also physical space in the data center. This can result in twice the energy saving as energy can be saved for both the operation and the cooling of the system at the same time. By using a cluster with heterogeneous CPU architectures, the efficiency of older hardware can also be increased by adding some newer systems. Older hardware usually has a significantly lower performance per watt than modern hardware. For that reason the application with the highest difference in energy efficiency should be run on the new hardware.

In a report from Shehabi et al., the data shows that servers, as well as cooling and power provision systems each make up 43 % of U.S. data center electricity use, which together are 86 %. Other minor contributors are storage with 11 % and networking with 3 %. (see Figure 1) [1].

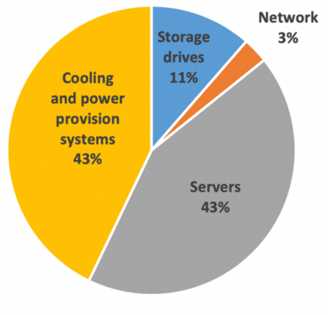


Fig. 1. Fraction of U.S. data center electricity use in 2014, by end use (Shehabi et al., 2016).

II. Related Work

A. CPU Architectures

The Central Processing Unit (CPU) is the device's "brain" but it is not that smart. A CPU can only function if it is given extremely particular instructions, known as the instruction set, that tell it to move data between registers and memory or execute a computation using a certain execution unit (such as multiplication or subtraction). Distinct instructions are required by different CPU hardware blocks, which tend to scale up with more complex and powerful CPUs [2]. Computer applications are not written in CPU instructions. With today's large cross-platform apps running on a variety of chips, that would be too much of a problem. Rather, apps are written in higher-level programming languages (such as Java or C ++) which are compiled for specific instruction sets to run on arm64, amd64, and other CPUs. Within the CPU, these instructions are decoded into microcode operations that require silicon space and power. For low-performance CPUs, keeping the instruction set simple is critical. However, higher performance can be achieved at the expense of power by using more complex hardware and instructions that perform multiple operations simultaneously. This is a key distinction between Arm and x86, as well as their respective approaches to CPU design in the past[3]. RISC (Reduced Instruction Set Computing) is used by ARM, whereas CISC (Complex Instruction Set Computing) is used by Intel (x86). Arm's CPU instructions are atomic, with a close relationship between the number of instructions and the number of micro-operations. By contrast, CISC has a much larger set of instructions, many of which perform multiple tasks (optimized math and data movement) [4]. When decoding these complex instructions, this results in better performance but higher power consumption. The distinctions between RISC and CISC are becoming increasingly blurred as each borrows ideas from the other and a wide range of CPU cores is based on architectural variations. Furthermore, because Arm's architecture can be customized, partners like Apple can add their own more complex instructions [5]. When decoding these complex instructions result in better performance but higher power consumption. The distinctions between RISC and CISC are becoming increasingly blurred as each borrows ideas from the other and a wide range of CPU cores is based on architectural variations. Furthermore, because Arm's architecture can be customized, partners like Apple can add their own more complex instructions [6].

B. Energy efficiency of ARM and x86 processors

Maqbool et al. [7] state that in their benchmarks, ARM performed four times better on single-core workloads and 2.6 times better on four cores, regarding performance to power ratio. On memory-intensive database transactions, however, x86 performed 12% better for multithreaded query processing [7].

Ou et al. [8] analyzed ARM and Intel CPU cluster computing in regards to energy efficiency. They measured different applications like Web servers, in-memory databases, and video transcoding. These applications performed more energy-efficient in the ARM cluster than in the Intel workstation. Multiple ARM processors are needed to achieve performance like the Intel processor. The energy-efficiency ratio of the ARM cluster against the Intel workstation varies from 2.6 to 9.5 in in-memory database workload, to 1.3 in Web server workload, and 1.21 in video transcoding. Ou et al. [8] also state that the Intel processor uses a dynamic voltage and frequency scaling technique, which causes the power consumption to not be linear to the CPU utilization level. For the ARM processor, this is not the case. Furthermore Ou et al. [8] concluded that ARM cluster based data centers are advantageous from a cost perspective in computationally lightweight applications, however when it comes to computation-intensive applications like dynamic Web server applications or video transcoding, the advantages of ARM processors decline gradually, as the number of ARM processors needed to provide comparable performance ramps up [8].

Aroca and Gonçalves [9] compared x86 and ARM architectures power efficiency. They researched web and database servers and their power usage, CPU load, temperature, request latencies, etc. The study researched the feasibility of building server infrastructure from low power computers. It turned out that the use of ARM based systems is a good choice for increased power efficiency without losing performance. More specifically, they show ARM based Systems on a Chip (SoCs) have good performance per Watt relation. In their tests for HTTP and SQL queries, ARM machines are 3 to 4 timesmore power efficient than x86 systems, when considering requests per second per Watt relation under different load scenarios. However, in a floating point computation test, the ARM processor had superior efficiency only for small numbers - with increasing number size, the performance decreased. The results showed that x86 processors are superior for floating point computation, when considering both performance and power efficiency [9].

C. Benchmarks

There are several ways to evaluate the relative performance of CPUs. A CPU with more cores and a higher clock frequency will usually be able to perform better than others from the same product generation. But comparing two different processor architectures this way is not possible. For this reason CPU Benchmarks are used. Benchmarks allow an easy comparison between several CPUs by ranking their performance in a standardized series of tests. They can be used for in many use cases, like building a new service in the cloud environment. Benchmark results can provide information about which processor is better suited to the application. In the case of this paper, which processor consumes less power for the tested application. CPU evaluations often rely on a number of different benchmark tests to evaluate CPUs. Those tests can be divided into the following two categories, namely synthetic benchmarks, and practice oriented tests [10].

Synthetic benchmarks can be used for a quick and general comparison of processors. They simulate many different tasks like 3D rendering, file compression, web browsing, floating-point calculations, and many other workloads. After measuring the CPU performance levels on each task, the numbers are weighted and combined into a single result, which represents the capability of the CPU. The results are called synthetic because the tests used to calculate them were simulated. Instead of testing the performance of the processor in a specific application such as a 3D creativity suite or a game, they simulate the workload that an application sends to the CPU under different circumstances. Accordingly, synthetic benchmarks cannot accurately predict actual performance. Synthetic scores are widely used to compare the relative performance of CPUs [11].

Here are some examples of synthetic benchmarks and their workload:

- PassMark performs heavy mathematical calculations that stress the CPU's performance on compression, encryption, and physical-related tasks.
- PCMark 10 rates the system based on how well it can handle business operations and everyday productivity tasks.

On the other hand, real-world benchmarks use realistic benchmark tests when you have specific plans for your Personal Computer (PC) and need accurate performance data for a particular application. These tests are performed by giving real programs large workloads and then measuring the time to completion. This results in a reliable prediction of system performance using the same settings.

Commonly used applications for realistic benchmark tests include: 7-Zip to measure the CPU's data compression and decompression speed. Blender to measure the 3 dimensional (3D) rendering speeds of a CPU. Handbrake to measure a CPU's video encoding speeds. In-game benchmark tools are another approach to a real-world test. These are non-interactive scenes that appear in some games. Use in-game benchmarks to check the CPU's effect on frames per second (FPS) during traditional gameplay as well as streaming. These tests provide a repeatable testing environment. As long as the system configuration remains the same, the benchmarks will show an accurate indication of power consumption [12].

For synthetic tests, the scoring system varies by program. The assessments are often measured in "points" (or other program-specific terms). A powerful CPU scores higher, or has more points, although it should be noted that different CPUs are designed for different purposes. By comparison, real-world tests use a number of different measurements:

- Dropped frames: Streaming benchmark tests count all frames dropped during video encoding. For viewers, this sometimes results in a jumpy frame change. A lower % of dropped frames is desirable.
- FPS (for video): In video encoding tests, FPS counts the number of frames your CPU encodes per second. Higher is better.
- FPS (for gaming): In in-game benchmark tests, FPS count the number of frames rendered per second. A higher FPS rate usually means more sophisticated gameplay. However, frame time should also be considered.
- GB/s (gigabytes per second): In encryption tests, GB/s measure data throughput. Higher is better.
- MIPS (million instructions per second): In data compression tests, MIPS measures the number of lower-level instructions executed per second by the CPU. Higher is better, but the value should be taken with a grain of salt when comparing different generations of CPUs, as the execution of instructions may well be different.
- Rendering Time: In rendering benchmark tests, rendering time measures the speed at which the CPU renders the geometry, lighting, and surface textures in a 3D scene. Shorter times are better.

Because some processors perform highly in certain CPU benchmark tests, multiple benchmarks should be examined rather than relying on a single value. Synthetic and real benchmarks can complement each other. Check synthetic benchmarks for an overview of the strengths of certain CPUs. With real-world benchmarks, you're more likely to find out how the CPU will perform in day-to-day use. With both, you'll get a full view of a CPU's capabilities. When selecting a CPU for gaming, for example, the benchmark scores can be used to evaluate the overall performance level of the CPU. Once the candidates are found their FPS and frame times in recently released titles can be looked up.

Benchmarks are often divided into single-core and multi-core scores. Single-core scores are more important for games and applications with a low thread count because they use a single core to process many, but not all, instructions [13]. Multi-core values are more important for software which can take advantage of higher thread counts, because they distribute their instructions over several cores [14].

Although CPU benchmarks are important, every component plays a role in determining system performance. The CPU is responsible for Games with complex AI, physical elements, and graphical post-processing that tend to be more CPU intensive. Those tasks may benefit from a CPU with a higher core and thread count and also a higher clock speed. In addition to CPU benchmarks, the examination of GPU benchmarks can also be important tor evaluating the performance. Because some programs rely more heavily on the GPU power. Dedicated graphics cards, for example, can handle most of the 3D rendering work. The used memory and data storage can also affect system responsiveness and load times which contributes to the performance of the whole system.

D. ARM Big Little Architecture

The ARM Big Little architecture was developed for applications in areas where energy efficiency is very important. The architecture combines weaker energy-saving cores with more powerful, but less efficient cores in one CPU. Processes can then be assigned to one of the two core types depending on the application. According to the developer, ARM Holding, up to 70 % of the energy consumption can be saved in some use cases [15]. CPUs that are designed for the highest possible performance are different from CPUs that are designed to save power. When very powerful processor cores operate at low clock speeds, they still consume a lot of power. These tasks could be taken over by weaker processor cores, which can perform these tasks with less power consumption. These CPU cores can be made more efficient by many factors. For example, the number of transistors and the cache can be reduced or the CPU pipeline can be simplified [16].

Depending on the use case there are a few different models of how the two types of processor cores can be combined with each other.

- There are also different possible arrangements of both core types. With the cluster switching method, one cluster of each core type is created. The process scheduler can only run processes on one of the clusters at the same time. This means if one process changes its needed performance status the whole cluster has to be switched. This means all processes have to be switched over to the other cluster.
- The method of using In Kernel Switcher (IKS) tries to make all processing cores more independent from one another. In this model, one performance core is bundled with one efficiency core which is then called virtual cores. During operation, only one of both cores is working. Depending on the currently needed performance the processor type of each virtual core can be switched independently from other virtual cores.
- The most flexible and most powerful model is heterogeneous multi-processing. Compared to both previously explained models this model allows to run on all processing cores at the same time and thus provide the highest processing power of the mentioned models. This is achieved by running all processing cores in parallel. Tasks are then scheduled by their needed performance to the matching cores [17].

The advancements of processors by using Arm Big Little-based Processors have brought significant advancements in power savings. With smartphones being one of the greatest users of this technology the battery life has increased a lot using this technology [15]. The Use of ARM Big Little is restricted to just trying to run one single processor as efficient as possible, which makes it only relevant for the application in fields that don't need more than one processor. By scaling the idea behind ARM Big Little up to fit for a larger data center a lot of energy could be saved. In Comparison, to ARM Big Little the scheduling in a data center would happen between different machines with different capabilities instead of just single cores. By using multiple processors instead of just multiple cores the problem of compatibility arises. The cores of one CPU are specifically built to work and function together whereas different CPU Architectures are not created with this functionality in mind. This field of application is to be researched in this paper.

III. Methodology

A. Performance per Watt Benchmarking and Calculation

For the experiment, it was decided to set up a series of benchmarks with devices with different CPU types, in which the computing power of devices is determined. While measuring the computing power by means of benchmarks, the energy consumption of the hardware is measured. The two values obtained can then be used to calculate the ratio of the computing power to the energy consumed in the process.

$$\frac{Score\, of\, Benchmark}{Power\ consumption\,[W]} = Score\, per\, Watt\,[W] \qquad (1)$$

The resulting ratios of the two architectures can then be used to compare the processor types with each other and draw conclusions.

With the experiment, measurable data is created that simulates a potential real-world use case. The resulting data of the experiment could later be used to draw conclusions and create a power-saving plan. With a power-saving plan, it can be decided which applications should run on which hardware in order to achieve the most power-efficient usage. The creation of a dedicated power-saving plan and applying it is not the goal of this paper. It is limited to showing potential real-world benefits of choosing the best-suited CPU for certain tasks.

A Raspberry Pi 4 with 8 GB RAM is used as a representative for ARM CPUs. It was released in May 2020 and is a single-board computer with limited expansion options. The Random-Access Memory (RAM) can not be expanded and an SD card serves as the device's main storage. In addition, it offers four USB ports that can be used for expansions, as well as an Ethernet port. There are also GPIO pins for connecting sensors and other devices. It has a 64-bit 4-core Broadcom BCM2711 CPU with a maximum clock speed of 1,5GHz. [18]

As the system contains an Intel-based CPU, an old Optiplex 7010 office PC is chosen. This is a small form factor PC and was released in the year 2012. It has an Intel i7-3770 processor and 16 GB of RAM. The Intel i7-3770 is a CPU released in 2012 with 4 cores and 8 threads and has a clock speed of 3.4GHz which can be boosted to 3.9GHz. [19]

TABLE I
COMPARISON OF SYSTEMS

	Optiplex 7010	Raspberry Pi 4 8GB
Release	2012	2020
CPU	Intel i7-3770	Broadcom BCM2711
Architecture	x86	ARM 64
Cores	4	4
Threads	8	4
RAM	16 GB	8 GB
Max Clock	3,9 GHz	1,5 GHz

The significant difference between the hardware of both devices is not ideal. The great differences between both systems introduce a lot of aspects, that can influence the outcome of the experiment. For an optimal setup, both systems would share most of their components and only differ from the used CPU and motherboard. This would greatly simplify the interpretation of the experiment and make it easier to draw conclusions about the CPU. In the case of this experiment, it was not possible to gain access to such similar hardware. Nevertheless, this test setup can be used to obtain meaningful measured values, and conclusions can be drawn from the results.

The operating system used to run both devices is Ubuntu Server 22.04 LTS without a desktop environment. This GNU/Linux distribution was chosen because it is one of the few that is available in the exact same version for both systems. [20] The base configuration of the system is not altered, and only the benchmark software is installed additionally. The two benchmark tools selected for the test are Sysbench and 7zip. These simulate real-world use cases, which can be used to evaluate the performance of the processors. Each of the benchmarks is run 5 times on each device. Afterward, the average values of all measurements are created for creating a more stable end result of the tests.

The first benchmark to be tested on the systems is the 7-Zip benchmark [21]. This Benchmark tool evaluates CPU capabilities based on the performance of compressing and decompressing data. The benchmark uses input data which is stored on the main storage of the device. The used data is then processed in the compression algorithm of 7-Zip and its output is stored back on the main storage. Because of the involvement of other components in this test, those could also influence the outcome of the benchmark. For simplicity reasons only the values of the compression step were taken into account and the average of the test results was used as the final result of each run.

The second benchmark tool is the CPU benchmark of sysbench [22]. Sysbench provides many benchmark tests for various components of a system. The CPU benchmark was chosen for this experiment. This benchmark program runs an algorithm to calculate prime numbers within a certain time frame. The result of the test is the achieved events per second of the system. Which represents the score of the system.

For the measurement of the consumed wattage during the tests, a wattage and current meter were used. The device used for this was a Brennstuhl PM 231 E. This device is able to measure voltage, frequency, current, power factor, and power of the device plugged into it. [23]

IV. Results

For the results of the experiment, each benchmark was run five times in order to achieve a more stable outcome. The formula used for the calculation of the score per watt can be found in the section III. Methodology.

A. Results of Benchmarks using 7-Zip

The value of the Score shown in the tables of the 7-Zip benchmark is the average score the system achieved in the compression part of the benchmark. For simplicity reasons, the scores of the decompression benchmark were not taken into account.

TABLE II
7-Zip Benchmark Optiplex 7010

Optiplex 7010			
Run	Score	Watt[W]	Score/Watt[W]
1	22502	136,0	165,5
2	22562	136,4	165,4
3	23431	137,2	170,8
4	22890	137,6	166,4
5	22071	137,3	160,8
Average	22691,2	136,9	165,8

TABLE III
7-Zip Benchmark Raspberry Pi 4 8GB

Raspberry Pi 4 8GB			
Run	Score	Watt[W]	Score/Watt[W]
1	3841	10,7	359
2	3963	10,8	367
3	4018	10,9	368,6
4	4006	10,9	367,5
5	4007	11	364,3
Average	3967,8	10,86	365,3

B. Results of Benchmarks using Sysbench

The scores of the Sysbench benchmark listed below are the total number of events the system achieved in the test.

TABLE IV
Sysbench Benchmark Optiplex 7010

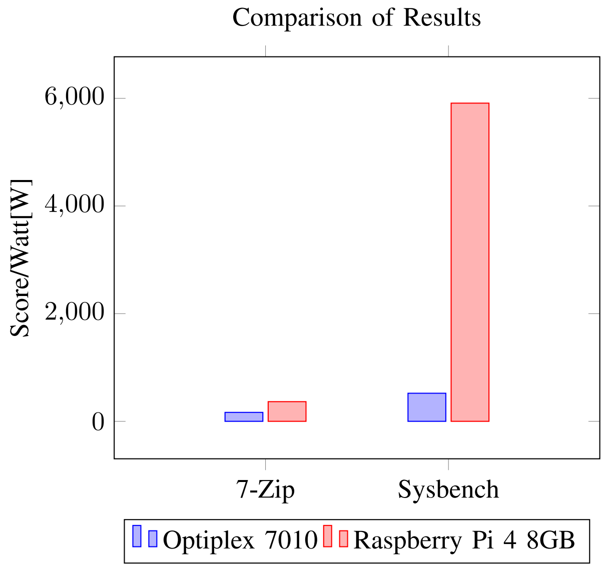
Optiplex 7010			
Run	Score	Watt[W]	Score/Watt[W]
1	67884	129,8	523
2	67841	130,3	522,8
3	67844	129,9	522,3
4	67843	130,2	521,1
5	67840	130,6	519,4
Average	67859,43	9,9	521,5

TABLE V
Sysbench Benchmark Raspberry Pi 4 8GB

Raspberry Pi 4 8GB			
Run	Score	Watt[W]	Score/Watt[W]
1	58543	9,9	5913,4
2	58478	9,8	5906,9
3	58553	9,9	5914,4
4	58488	9,9	5907,9
5	58495	9,9	5908,7
Average	3967,8	9,9	5910,2

C. Comparison of Results

The results of both benchmarks show that the Raspberry Pi 4 performed significantly more energy-efficient in both tests. In the 7-Zip benchmark the Raspberry Pi 4 was able to achieve 2.2 times the energy efficiency of the Optiplex 7010. In comparison to that, the achieved score per watt of the Sysbench benchmark of the Raspberry Pi 4 was more than 11 times higher compared to the Optiplex 7010.



V. Conclusion

This paper gives an overview of possible application areas and advantages of cleverly used heterogeneous clusters. However, many of the points raised still need to be discussed further, and the current possibilities of implementing a heterogeneous cluster need to be evaluated in a real-world scenario.

The results have shown significant differences in efficiency in the tested use cases in the experiment. Where in the 7-Zip benchmark the difference in efficiency was by a factor of 2.2, the difference in the Sysbench benchmark was more than 11 times higher. For the interpretation of the results, this means that certain types of software can be run significantly more efficiently if the right hardware is chosen. This means having the ability to choose which system an application runs can decrease the overall power consumption.

An important thing to note about the experiment is the less than optimal hardware that was used for the benchmarks. The comparison of a single-board computer with a full desktop PC is not completely representative of a comparison between two processor architectures. Also, the additional hardware in a desktop PC may introduce influences and disturbing factors during the experiment.

To further investigate the operation of heterogeneous clusters in more detail, a prototype is presented in the future work section of this paper. The structure and goal of the investigations were described in this paper and can be implemented and documented in a subsequent paper so that conclusions can be drawn from the results about the usefulness of using heterogeneous clusters with today's technology.

VI. Future Work

The experiment results suggest significant benefits from leveraging the ARM processor architecture for computing. In use cases, it is possible for ARM-based processors to achieve multiples of the performance achieved by an x86 CPU while consuming the same amount of energy.

As an add-on to this research paper, as well as for possible future work in this field, A prototype framework for running containers in heterogeneous clusters is created.

A. *Prototype heterogeneous cluster*

The prototype presents a guideline for running containers in a heterogeneous Kubernetes [24] cluster, which contains worker nodes with the x86 architecture, as well as the ARM architecture. A graphical representation of this architecture design can be seen in Fig. 2.

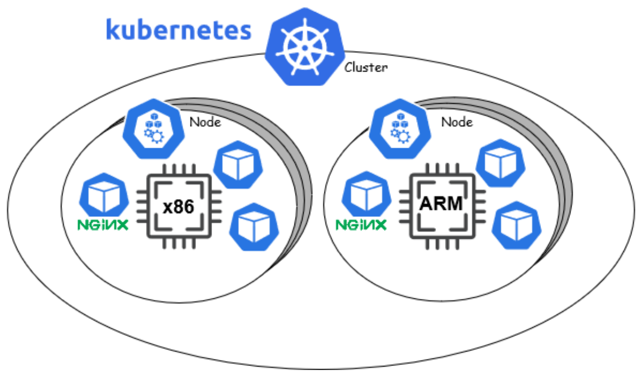


Fig. 2. Prototype heterogeneous Kubernetes cluster.

The virtual machines are provisioned on Amazon AWS's Elastic Compute Cloud (EC2) service [25]. The AWS Marketplace [26] offers operating system images for x86 and ARM, as well as many other machine types. For the prototype setup, the following Ubuntu Linux images from Canonical [27] are used for x86 resp. ARM nodes:

- Ubuntu 22 (Ubuntu 22.04 LTS) [28]
- Ubuntu 22 (ARM) (Ubuntu 22.04 LTS) [29]

The EC2 machines are joined together with kubeadm [30] to form a Kubernetes cluster.

Kubernetes itself automatically assigns the label "kubernetes.io/arch" to nodes with the value corresponding to the processor architecture.

- "amd64", which equates to x86 architecture with a 64-bit instruction set.
- "arm64", which equates to ARM architecture with a 64-bit instruction set.

The container image nginx [31] from Docker Hub is used for demonstration. The image is offered by the maintainer in several versions for different processor architectures. For the prototype setup, the images "linux/amd64" [32] and "linux/arm64/v8" [33] in version 1.23.0 are used.

The specific images for the different processor architectures are transparently pulled by Kubernetes to the nodes when specifying "nginx:1.23.0". The user does not need to specify the architecture explicitly.

Kubernetes pods can be constrained to specific nodes based on node labels with the concept of node affinity. There are two types of node affinity:

- **requiredDuringSchedulingIgnoredDuringExecution**: The scheduler can not schedule the Pod unless the rule is met.
- **preferredDuringSchedulingIgnoredDuringExecution**: The scheduler tries to find a node that meets the rule. If a matching node is not available, the scheduler still schedules the Pod.

The ”requiredDuringSchedulingIgnoredDuringExecution” affinity type can be used to constrain pods strictly to specific nodes.

With the latter affinity type ”preferredDuringSchedulingIgnoredDuringExecution”, the key ”kubernetes.io/arch” can be set to match ”amd64”. This expression can be supplied a weight of e.g. ”1”, while an accompanying expression can be set to match ”arm64” with a weight of e.g. ”50”. This results in pods being scheduled first to arm64 nodes whenever possible, since the weight is higher. An example configuration in YAML format can be seen below. Long lines are shortened and marked with ”[...]”.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnor[...]
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/os
          operator: In
          values:
          - linux
    preferredDuringSchedulingIgnor[...]
    - weight: 1
      preference:
        matchExpressions:
        - key: kubernetes.io/arch
          operator: In
          values:
          - amd64
    - weight: 50
      preference:
        matchExpressions:
        - key: kubernetes.io/arch
          operator: In
          values:
          - arm64
```

The prototype framework introduced in this section may be extended by other researchers in future work. We propose the following possible topics:

- Power efficiency driven workload scheduling in cluster computing.
- Dynamic pod scheduling and node selection in Kubernetes, based on key performance indicators and other metrics.

References

[1] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. United states data center energy usage report. 2016.

[2] Ulan Degenbaev. Publikationen der uds: Formal specification of the x86 instruction set architecture. https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/26394, march 2012. (Accessed on 05/22/2022).

[3] Abdullah Yıldız. Cpu design simplified — ieee conference publication — ieee xplore. https://ieeexplore.ieee.org/abstract/document/8566475, 2018. (Accessed on 05/22/2022).

[4] T. Jamil. Risc versus cisc — ieee journals & magazine — ieee xplore. https://ieeexplore.ieee.org/abstract/document/464688, 1995. (Accessed on 05/22/2022).

[5] Rechnerarchitektur : Einführung in den aufbau moderner computer. https://ebookcentral.proquest.com/lib/fh-burgenland/reader.action?docID=4691453, 2016 august. (Accessed on 05/22/2022).

[6] AlexeyLastovetsky. Heterogeneity in parallel and distributed computing - sciencedirect. https://www.sciencedirect.com/science/article/pii/S0743731513001652, 2013 . (Accessed on 05/22/2022).

[7] Jahanzeb Maqbool, Sangyoon Oh, and Geoffrey C Fox. Evaluating arm hpc clusters for scientific workloads. *Concurrency and Computation: Practice and Experience*, 27(17):5390–5410, 2015.

[8] Zhonghong Ou, Bo Pang, Yang Deng, Jukka K Nurminen, Antti Ylä-Jääski, and Pan Hui. Energy-and cost-efficiency analysis of arm-based clusters. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 115–123. IEEE, 2012.

[9] Rafael Vidal Aroca and Luiz Marcos Garcia Gonçalves. Towards green data centers: A comparison of x86 and arm architectures power efficiency. *Journal of Parallel and Distributed Computing*, 72(12):1770–1780, 2012.

[10] M; Campbell J Sirrell, B; Holliday. The benchmark test 1995 (conference) — osti.gov. https://www.osti.gov/biblio/227824, 1995. (Accessed on 05/22/2022).

[11] Lizy K. John Ajay M. Joshi, Lieven Eeckhout. The return of synthetic benchmarks. https://biblio.ugent.be/publication/679247/file/713352. (Accessed on 05/22/2022).

[12] R.P. Weicker. An overview of common benchmarks — ieee journals & magazine — ieee xplore. https://ieeexplore.ieee.org/abstract/document/62094, 1990. (Accessed on 05/22/2022).

[13] From single core to multi-core — proceedings of the 2006 ieee/acm international conference on computer-aided design. https://dl.acm.org/doi/abs/10.1145/1233501.1233516. (Accessed on 05/22/2022).

[14] Balaji Venu. [1110.3535] multi-core processors - an overview. https://arxiv.org/abs/1110.3535. (Accessed on 05/22/2022).

[15] big.little processing - arm. https://web.archive.org/

web/20121022055646/http://www.arm.com/products/processors/technologies/bigLITTLEprocessing.php, . (Accessed on 06/30/2022).

[16] Ten things to know about big.little - arm community. https://web.archive.org/web/20130910163539/http://blogs.arm.com/soc-design/1009-ten-things-to-know-about-biglittle/. (Accessed on 06/30/2022).

[17] big.little software update — linaro - open source software for arm socs. https://web.archive.org/web/20131004230806/http://www.linaro.org/linaro-blog/2013/07/10/big-little-software-update/, . (Accessed on 06/30/2022).

[18] Raspberry pi 4 model b specifications – raspberry pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. (Accessed on 06/18/2022).

[19] Intel core i73770 processor 8m cache up to 3.90 ghz produktspezifikationen. https://ark.intel.com/content/www/de/de/ark/products/65719/intel-core-i73770-processor-8m-cache-up-to-3-90-ghz.html. (Accessed on 06/18/2022).

[20] Canonical ubuntu 22.04 lts is released — ubuntu. https://ubuntu.com/blog/ubuntu-22-04-lts-released. (Accessed on 06/19/2022).

[21] 7-zip benchmark. https://sevenzip.osdn.jp/chm/cmdline/commands/bench.htm, 2022. (Accessed on 06/28/2022).

[22] Scriptable database and system performance benchmark. https://github.com/akopytov/sysbench, 2022. (Accessed on 06/28/2022).

[23] Primera-line wattage and current meter pm 231 e — brennenstuhl®. https://www.brennenstuhl.com/en-DE/products/travel-adapters-adapter-plugs/primera-line-wattage-and-current-meter-pm-231-e. (Accessed on 06/20/2022).

[24] Production-grade container orchestration. https://kubernetes.io/, 2022. (Accessed on 06/28/2022).

[25] Elastic compute cloud. https://aws.amazon.com/ec2/, 2022. (Accessed on 06/28/2022).

[26] Aws marketplace. https://aws.amazon.com/marketplace, 2022. (Accessed on 06/28/2022).

[27] Canonical group limited. https://aws.amazon.com/marketplace/seller-profile?id=565feec9-3d43-413e-9760-c651546613f2, 2022. (Accessed on 06/28/2022).

[28] Ubuntu 22 (ubuntu 22.04 lts). https://aws.amazon.com/marketplace/pp/prodview-cs32nfwh3hkvi?sr=0-1&ref_=beagle&applicationId=AWSMPContessa, 2022. (Accessed on 06/28/2022).

[29] Ubuntu 22 (arm) (ubuntu 22.04 lts). https://aws.amazon.com/marketplace/pp/prodview-vb4u3b7aoczd4?sr=0-2&ref_=beagle&applicationId=AWSMPContessa, 2022. (Accessed on 06/28/2022).

[30] Creating a cluster with kubeadm. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/, 2022. (Accessed on 06/28/2022).

[31] nginx. https://hub.docker.com/_/nginx, 2022. (Accessed on 06/28/2022).

[32] nginx:1.23.0. https://hub.docker.com/layers/nginx/library/nginx/1.23.0/images/sha256-3536d368b898eef291fb1f6d184a95f8bc1a6f863c48457395aab859fda
context=explore, 2022. (Accessed on 06/28/2022).

[33] nginx:1.23.0. https://hub.docker.com/layers/nginx/library/nginx/1.23.0/images/sha256-ec2290b7c5d15abb4b3384ad66a89e9c523a4668c057898f3114fa61df
context=explore, 2022. (Accessed on 06/28/2022).