Conceptualising Resources-aware Higher Education DigitalInfrastructure through Self-hosting: a Multi-disciplinary View

Angeli, Lorenzo ; Okur, Ö.; Corradini, Carlo; Stolin, Marcel ; Huang, Yilin; Brazier, F.M.; Marchese, Maurizio

Publication date
2022

Document Version
Final published version

Published in
Eighth Workshop on Computing within Limits

Conceptualising Resources-aware Higher Education Digital Infrastructure through Self-hosting: a Multi-disciplinary View

Lorenzo Angeli*
University of Trento
Trento, Italy
lorenzo.angeli@unitn.it

Özge Okur*
Delft University of Technology
Delft, The Netherlands
o.okur-1@tudelft.nl

Carlo Corradini*
University of Trento
Trento, Italy
carlo.corradini@studenti.unitn.it

Marcel Stolin*
University of Trento
Trento, Italy
marcelpascal.stolin@studenti.unitn.it

Yilin Huang*
Delft University of Technology
Delft, The Netherlands
y.huang@tudelft.nl

Frances Brazier*
Delft University of Technology
Delft, The Netherlands
f.m.brazier@tudelft.nl

Maurizio Marchese*
University of Trento
Trento, Italy
maurizio.marchese@unitn.it

ABSTRACT

As higher education digitalises, institutions increasingly outsource the development and management of their digital infrastructure including server hardware and services such as email, shared storage, and video conferencing, to private companies. This outsourcing trend is a change in paradigm, since universities have historically been pioneers in deploying and maintaining their own digital infrastructure, a practice also known as self-hosting. Digital infrastructure has a key role in all of a university's functions: administration, research, and education. While outsourcing infrastructure has benefits in the form of convenience and lower costs, it also erodes institutional independence, centralises points of failure, and delegates highly relevant value choices about privacy, data ownership and environmental impact to external actors.

In this article, we provide a first quantification of a potential return to self-hosting, emphasising its effect in energy reduction and avoided e-waste. We then outline some policy actions that could enable higher education institutions to re-take control over their digital infrastructure by building local services. This mode of operation reduces waste, and has the added benefit of increased resilience to scenarios of resource scarcity and collapse of external infrastructure. As an example of what could be achieved leveraging these policies, we detail the architecture of a low-impact data centre made of upcycled hardware and resource-aware software. By exploring our main structural choices we aim to showcase how, even starting from a generally heavy-weight software stack such as Kubernetes, there is significant space to reduce digital infrastructure's overall resource footprint.

*All authors contributed equally to the paper.

LIMITS '22, June 21–22, 2022, Online

KEYWORDS

digital infrastructure, self-hosting, reappropriation, resource reduction

1 INTRODUCTION

Higher education digitalisation has been happening for several decades. Over the last years, however, a growing number of voices is starting to express concern over the speed at which digitalisation is happening. This critical wave discusses in particular the field of education technologies, with structural critiques such as those of Watters [1], to platforms such as those by Selwyn [2], or more broadly to "digital capitalism" [3]. While these critical voices do exist, they are a minority in a landscape dominated by a general optimism and hype around (education) digitalisation[1].

In 2020, the COVID pandemic prompted schools and higher education institutions to adapt their educational offering to online teaching, which introduced further digital artefacts in education. In parallel, universities in Europe and US continued a trend of outsourcing digital infrastructure and services to "cloud" providers. The net effect of this trend has been a centralisation of both infrastructure and service providers, often towards private, for-profit actors (prominently "big tech" companies) in what Williamson calls new power networks [4].

As universities privatise their services and infrastructure, they achieve lower costs and increased convenience by trading off institutional autonomy, political power, and the ability to directly control their systems' embedded values. Indeed, by taking a view inspired by Actor-Network Theory, we can argue that, when universities use infrastructure provided by cloud providers, they indirectly adopt those providers' value systems.

In this article, we highlight two value categories that are affected by this change: a technical and an environmental dimension. Technically, digital infrastructure outsourcing creates a centralised

[1]Readers can look at the overwhelming number of articles discussing technological solutions in education conferences to get a first impression.

landscape where many institutions depend on few actors. Environmentally, this intermediation makes it harder to measure the infrastructure's resource footprint. making comparisons between hosting solutions harder, and limiting the ability of institutions to independently gather data and set their own environmental agendas and goals.

Through the article, we aim to state that this problem, that is largely political in nature, can be solved if looked at systemically, and if institutions are empowered to create lasting organisational change. We do so by proposing an agenda of digital infrastructure reappropriation through a form of "limits-aware" self-hosting. Our hope is that, with a broad and multi-disciplinary approach, we can show the many potentials for future work and of stimulate a systemic debate.

Existing work that addresses the issues discussed so far can range from highly conceptual [5] to quantitative [6] with, however, limited explorations on the intersections between the many disciplines working on this field. Furthermore, relevant initiatives, virtuous examples or experiments such as the LowTechMagazine website[2] [7], CoopCloud[3], commercial initiatives such as Ungleich's Data Center Light[4], or the many reflections and examples on the field of "permacomputing" [8] are anecdotally regarded, in institutional discourse, as fringe experiences that cannot be replicated, or applied at scale within institutional boundaries.

The present work is a first attempt at bridging this gap by working in two directions: on the one hand, we wish to present some policy actions that can support universities in reappropriating their digital infrastructure through self-hosting; on the other hand, we present a proof-of-concept for a low impact, "limits-aware", data centre, aiming to show what could be achieved with the policy actions we propose.

The combination of policy and technical actions are supported by an energy analysis of a prominent outsourced service, video conferencing, highlighting how infrastructural reappropriation is aligned to the energy and environmental sustainability agendas that many national and sovra-national organisations wish to pursue.

From a policy perspective, we provide three simple actions that can work, in a reappropriation logic, as institutional enablers. Technically, we propose a combination of software and hardware designed around resource reduction. Our proposal is designed to be adopted by institutional actors and, as such, it takes some trade-offs in terms of simplicity and, potentially, future-proofing, striving instead to reduce the infrastructural reappropriation barrier.

This multi-disciplinary perspective aims to showcase how higher education institutions can take a proactive role in building digital infrastructure that is more interoperable, extensible, and simpler to deploy (as per Chen's suggestion [9]) while also minimising resource waste. Fundamentally, the last three years have showed a global landscape where issues such as pandemics, wars and climate crises are a reality for the whole planet, including the global North. Infrastructure, as a consequence, needs to be redesigned so that it can thrive in situations of scarcity, both hypothetically future or indisputably present. In this sense, we see our contribution as a very literal exercise in pre- and post-apocalyptic computing [10], putting questions of resilience and resource minimisation at the forefront.

We start in Section 2 by giving an overview of the problem, and positioning it more precisely within the LIMITS discourse. We then discuss the causes and impacts of this outsourcing. We dedicate Section 3 to discussing how self-hosting could reduce resource consumption, addressing both the software and hardware dimensions, using video conferencing as a case study. In Section 4, we outline some policy interventions that could aid in creating lasting organisational change to empower institutions (in our case, universities) to once again host their own infrastructure. We dedicate a majority of the article (Section 5) to accompanying these policy reflections with a proof-of-concept of what could be achieved were those policies to be implemented, describing the architecture for a low-impact data centre. Lastly, in Section 6, we engage in a brief self-critique of our work, highlighting the many next steps and opportunities for further work that we discovered in the process of writing this contribution.

2 STATE OF AFFAIRS

2.1 General problematisation

Digitalisation — prominently in the Global North — is an accelerating process, and the recent COVID pandemic has often been mentioned as a catalyst. Digitalisation, however, is a complex phenomenon made of a number of macro-trends, including among many the centralisation of digital infrastructure, and of the actors running it.

Higher education has not been immune from infrastructural centralisation. Fiebig et al. [11] claim that, while they could not observe a clear "pandemic effect" accelerating the universities' reliance on cloud infrastructure, the general trend is on the rise. It should also be noted that many of these providers are private for-profit entities, and often so-called "Big Tech" companies, offering their own proprietary software solutions. A particularly emblematic case is that of video conferencing software, with the consolidation of most institutions towards proprietary solutions including Zoom, Microsoft Teams, Google Meet, or Cisco Webex.

While most institutions adopted these solutions in what Teräs et al. called a "seller's market" [12], alternatives — in the form of Free/Libre Open Source Software (FLOSS) — have also been flourishing in parallel. Remarkably, many proprietary solutions are powered, at an infrastructural or dependency level, by FLOSS[5]. The 2010s have also seen a fast development of virtualisation technologies that are nowadays at the basis of many data centres, most prominently containerisation[6]. Virtualisation technologies have had significant development contributions by "big tech" companies[7], and could represent an interesting avenue for higher education institutions to reverse the current outsourcing trend.

[2] See https://solar.lowtechmagazine.com/
[3] See https://coopcloud.tech/
[4] See https://datacenterlight.ch/

[5] For some examples, see https://discord.com/licenses, https://explore.zoom.us/en/opensource/, https://3rdpartysource.microsoft.com/
[6] For example Docker, the most popular container software suite, was launched in 2013.
[7] See, for example, Google's Kubernetes, and Microsoft, Facebook and Google's many contributions to Docker.

In parallel, the global production of computing hardware is ever increasing[8], as is the computing power of each device, thanks to Moore's law [13, 14]. These hardware advancements, however, are offset by increasing system requirements by software[9], contributing to an overall shortening of devices' life cycle [15] that has room for significant improvement only in cases of proactive tackling of the issue [16].

Summarising, the digitalisation of higher education gives rise to many challenges, including:

(1) Accelerating speed of digital infrastructure building.
(2) An outsourcing trend, with centralisation toward proprietary software providers.
(3) Increasing software system requirements.
(4) Shorter life span of computing hardware.

These challenges have political, technical, and environmental consequences, and have clear relationships in terms of what Nardi et al. defined in 2018 as the three key principles of computing within limits [17] of questioning growth, considering scarcity and reducing resource consumption.

Politically, the outsourcing trend creates ties between higher education institutions and companies that maximise growth, seek abundance, and only consider resource reduction as a means to reduce operational costs, disempowering universities from working "within limits". Externalising infrastructure also hinders institutional ability to set infrastructural R&D agendas, with procurement autonomy (by itself, a potentially virtuous feature) making it harder for institutions to achieve critical mass to demand change.

Technically, outsourcing opacises the technological stack, and while this simplifies the institutions' work, institutions consequently have limited ability to adapt and change technologies. The consolidation towards few providers also centralises points of failure, creates vendor lock-ins, and limits the ability to react or adapt to situations of crisis or of scarcity, such as universities operating in locations where bandwidth is limited or heavily bottlenecked.

Environmentally, shifting operations to centralised data centres implies longer travel distances for data, and thus increased electrical and bandwidth usage. Abstracting the data centre also reduces the institutional ability to monitor their resource use. Thus, matters of efficiency are fully delegated to service providers and removed from the institutions' space of operations.

2.2 Socio-economic causes of outsourcing

We can contextualise the current outsourcing trend as a reversal of a previous historical model. Higher education institutions have been, in the 1900s, pioneer organisations in developing digital infrastructure, from mainframe computers to the internet [18], with many protocols being born in university labs[10].

A university's digital infrastructure includes general services such as email, calendars, data storage, learning management systems and video conferencing, large sets of administrative tools, and specific tools such as high-performance computing, networking virtual labs, and more. Many such services can be — and are being — outsourced to private actors. Fiebig's work [11] gives once again an idea of the size of the outsourcing phenomenon.

Outsourcing is a widespread practice in the business world, including in the computing industry. Many of the reasons behind the use of outsourcing in business can apply to higher education. By outsourcing, system administrators are spared the many responsibilities that maintaining reliable infrastructures implies. The involvement of an external actor also creates a layer for plausible deniability in case of malfunctions or disservices.

Most importantly, outsourcing is motivated by economic reasons. Externalising services and infrastructure reduces costs both in terms of the infrastructure itself (in the form, here, of servers, bandwidth, energy, etc.) and human resources, as externalising staff can help achieve cost savings by economies of scale. A part of the cost savings however also comes from relocating staff to places with lower wages, and places of widespread resource exploitation.

Public procurement procedures are also to be kept into account. As the "Public Money, Public Code" initiative of the Free Software Foundation of Europe[11] points out, ill-designed regulation can implicitly favour the public procurement of proprietary software. When software providers further push for "cloud" versions of their software over locally-hosted versions (for example by reducing costs) institutions have fewer and fewer arguments to maintain infrastructural independence.

2.3 Ethical implications

Outsourcing digital education infrastructures to private companies may come at the expense of institutional autonomy, academic freedom and political power. In other words, by outsourcing, institutions lose control over how their digital education infrastructure is designed and how it can be used.

Widespread privatisation of services, recently in the case of video conferencing with Zoom, Microsoft Teams and Google Meet, has also raised many concerns about data privacy [19–22]. Service providers may collect personal data such as names, email addresses, phone numbers, physical addresses and IP addresses as part of their business model, and may store them in locations subject to different laws than the one the institution is based on. Also, as decisions about what services to adopt are normally taken at a university management level, end-users (including students have in the case of the COVID pandemic) are left with two choices: using these platforms and having their personal information collected by these private companies, or being excluded by activities (in the case of students, lectures — again, see [11]).

3 RESOURCE REDUCTION THROUGH SELF-HOSTING

As we discussed in the previous sections, outsourcing digital infrastructure detaches institutions from their technological choices by creating an intermediary layer that is outside of their control. A

[8] See various figures, for example, at https://www.statista.com/markets/418/topic/482/hardware/

[9] See for example how Windows 11 quadruples the minimum required RAM compared to Windows 10
https://www.microsoft.com/en-us/windows/windows-10-specifications
https://www.microsoft.com/en-us/windows/windows-11-specifications

[10] See for example IRC, a once-popular chat application, and its origins at the University of Oulu https://daniel.haxx.se/irchistory.html

[11] https://publiccode.eu/

space where this lack of control is particularly evident is that of resource consumption. In this section, we wish to provide a first quantification of how there is substantial room to reduce the resource footprint of digital infrastructure — both from a hardware and a software standpoint — by creating an environment that is more conducive to self-hosting.

We will illustrate these benefits by discussing a case that has been under substantial public scrutiny: video conferencing. Video conferencing is used in all of a university's functions: research, administration, and teaching. Video conferencing also involves transferring large data volumes, and requires substantial computational power both on the server side (if present) and on the devices of all end users, making it prone to high resource usage.

What seems to be a purely software matter, however, also has impacts in terms of hardware: when institutions choose a video conferencing platform, they delegate the choice of user hardware requirements. The more institutions are detached from these choices — most prominently, when adopting a proprietary solution — the more they are letting service providers choose what devices their users should have. Beyond OS compatibility, if a provider pushes an update that significantly increases system requirements, devices may be rendered obsolete, shortening their life cycles.

The impacts are therefore broader than they initially appear. In this section we will analyse this problem from both the software and hardware perspectives. We quantify how self-hosting might reduce energy consumption on the software side, and electronic-waste (e-waste) on the hardware side, creating digital infrastructure that is less wasteful and more suitable to scenarios of resource scarcity.

3.1 Software: an energy analysis of video conferencing

In the literature, in-person and online meetings/conferences are compared in terms of their energy consumption and carbon footprint. Findings from [23] show that having an online meeting consumes at most 7% of the energy consumed for an in-person meeting. In a more recent study, [24] finds that transitioning from in-person to online conferencing can lessen the carbon footprint by 94% and energy use by 90%. Based on these studies, online meetings may be seen as an effective strategy to reduce greenhouse gas emissions and thus mitigate climate change.

Despite emitting less greenhouse gases compared to in-person meetings, however, an online meeting/conference consumes a considerable amount of energy in three elements: (1) electricity required for the participants' devices; (2) electricity required for the network infrastructure; and (3) electricity required for servers.

Several studies estimated the electricity consumption of a virtual conference: [25] calculated the energy consumption of a virtual conference of five days with 1777 participants, which took place on Zoom. The results from this paper indicate that the total electrical energy consumption is as follows: (a) 1173 kWh from user laptops, (b) 1263 kWh from the network, and (c) 15 kWh from the servers. Similarly, the authors in [26] estimate electricity consumption of an online academic conference on Zoom, as well as its carbon footprint.

The papers mentioned focus on the energy consumption of video-calling via external companies like Zoom. However, self-hosting the software required for an online meeting/conference may significantly reduce the physical distance between users and servers, thus reducing the energy consumption required for the network, i.e., data moving across the routers. This is an especially likely scenario for online classes, since students and teachers are more likely to be in physical proximity of each other.

To the best of our knowledge, a comprehensive energy analysis of self-hosting online meetings has not been carried out. [25] however, places network energy consumption in the bracket of 50% of total energy consumed for online video conferencing. It should also be noted that the geographical situation discussed in the article is almost ideal, as data centres tend to be positioned in geographically central places, leading to close to optimal routing in the case of a global conference. In the case of online classes, however, routing data through a data centre might imply substantially longer distances, and therefore an even more skewed energy consumption balance toward the network[12].

As a promising first step, Suga [27] compared three video conferencing systems in terms of bandwidth consumption: two proprietary solutions using their own data centre (Zoom and MS Teams), and an open source, self-hosted system (BigBlueButton). Suga's results show a significantly lower bandwidth consumption for BigBlueButton (approximately 40% lower compared to Teams and 15% lower compared to Zoom). Combined with reduced network distances, self-hosting appears to be a promising avenue to greatly reduce the energy footprint of video conferencing.

3.2 Hardware: rethinking institutional e-waste

In addition to reducing energy consumption through software, self-hosting can contribute to reducing e-waste in higher education institutions. As we will discuss in Section 5.1, advancements in virtualisation technologies open the opportunity to reduce the need for dedicated server hardware, running the server software on suitably re-purposed unused general hardware that would otherwise constitute e-waste.

E-waste is one of the fastest-growing global waste streams, owing to high consumer demand and a parallel issue of shortening lifespan of electronic devices. In 2016, the total amount of e-waste generated globally was 44.7 million metric tonnes [28]. Higher education anecdotally has significant amounts of unused hardware in the form of desktop and laptop computers, tablets, smartphones, etc, which can be seen as a form of e-waste. While the amount of unused and not recycled hardware at higher education institutes has not been studied to provide concrete numbers, some articles survey how higher education institutions implement a waste management plan that should includes e-waste [29, 30].

Studies in [31] and [32] surveying education institutes in Australia and in the US, respectively studied institutional awareness regarding e-waste materials, and their impact on the environment. The results of the surveys suggest that there is a lack of awareness in the e-waste management among university students and staff.

[12]For example, as of the time of writing, the University of Trento hosts its online lectures via Zoom, through a data centre in Germany. If students and teachers were to be in the vicinity of Trento, all of their data would need to be routed from Italy to Germany and back — a substantial increase in distance.

In light of these, upcycling unused hardware at university for self-hosting services can be a promising avenue to manage e-waste at higher education institutes in a more effective manner.

4 POLICY ACTIONS

In the previous sections, we outlined the ethical implications of outsourcing, as well as how energy and e-waste reduction can be achieved through self-hosting. In this section, we propose several policy recommendations which can support universities in reappropriating and strengthening their digital education infrastructures through self-hosting.

4.1 FLOSS funds in universities

Self-hosting digital education infrastructures may provide a viable solution to deal with the privacy and autonomy concerns which are pointed out in Section 2.3. Higher education institutions can further use Free/Libre Open Source Software (FLOSS) to have higher transparency in the software they use, compounding the effects of self-hosting to get additional benefits such as service interoperability[13].

Because universities can contribute to FLOSS, they can be enabled to solve issues with the software (especially related to governance). This, combined with a good self-hosting backbone, can reduce the need for external companies. Nonetheless, it should be noted that universities can still achieve significant benefits by self-hosting proprietary solutions, especially if they had strong internal capacity to contributing to the (FLOSS) technologies that serve as the back-end of many data centres (see also Section 2.1).

FLOSS-based digital infrastructure can be considered as a commons, where communities work collectively to manage the software for shared benefits [33]. Education institutions, with their high capacity for R&D, are well-positioned actors to use FLOSS as a way to create a concrete and lasting benefit for their community, rather than providing gains for individual organisations.

A first policy recommendation that we propose for infrastructural reappropriation is to establish a permanent fund for FLOSS development at universities. That is to say, universities should receive (or at least allocate) a dedicated budget to hire developers that contribute as part of their job to the development of selected FLOSS. To maximise internal synergy, this FLOSS should be of high value to the institution contributing to it, and the staff hired in this way should be also in charge of running the selected FLOSS within their university and providing user support.

This model is gaining in popularity in the technology industry, where companies hire high contributors to FLOSS that is powering their solutions, so that they can benefit from increased internal capacity. Further synergies can be achieved by making use of research capacity to improve software in dimensions such as UI/UX.

4.2 Circular procurement

As mentioned previously, hardware for self-hosting can be run on re-purposed devices. Institutions, however, need to have strong circular procurement policies in place. The purpose of circular procurement is to ensure that acquired goods can be effectively repurposed at the end of their originally-intended use. In the context of electronic devices at universities, this implies reusing devices for new applications, repairing them, and disposing them only when they can no longer be used for other purposes.

A number of strategies can be employed at universities to realise this:

- Keeping a detailed inventory of unused hardware
- Arranging effective communication between staff in charge of waste management and procurement, or between users and waste management so that leftover products can be linked with a person who plans to use this specific product
- Running surveys to ask employees about the state of their hardware

These strategies might be more likely to be adopted, or might already being implemented, by universities that aim to be more sustainable[14]. Achieving as much circularity as possible in device procurement procedures can have major impacts in achieving carbon neutrality at institutional scale.

Universities should keep detailed inventories of unused (or even broken) hardware, to identify where circularity can be improved. For example, laptops with broken screens or underperforming batteries that are deemed uneconomical to be repaired might still be perfectly workable as platforms for running server software upon.

4.3 Aggregating demand and staff sharing

A counterargument for hiring staff to self-host digital infrastructure is that this might be economically unsound. This is compounded by the fact that support staff workload tends to be uneven, with significant peaks (e.g., times where classes start) and downtimes.

A way to overcome this is to use university consortiation as a way to aggregate digital infrastructure management. Institutions participating in project consortia and requiring same classes of service can hire shared staff to work on support and contribute to selected FLOSS (see Section 4.1) projects.

In many ways, this again replicates practices from the business sector. Companies under the same management might delegate to specific spin-offs the running of niche functions, or complex organisations might create divisions to internally create enough critical mass to justify the running of a service. Similarly, universities can share staff to achieve some level of economies of scale, and have enough critical mass to hire staff that can meaningfully contribute to upstream FLOSS.

5 PROOF OF CONCEPT — THE "RECLUSTER"

The policy actions we described in Section 4 aim to be enablers for institutional reappropriation of digital infrastructure. As a counterpart to these, we wish to present here the a proof-of-concept for a technological artefact that showcases what could exist, were those policies implemented.

We present here the "reCluster", a data centre and computing platform inspired by the Right to Repair movement[15] values of reducing, reusing, repairing, and recycling computing resources. The reCluster is an architecture for a data centre that actively reduces its impact and minimises its resource utilisation.

[13] See also https://joinup.ec.europa.eu/

[14] For instance, TU Delft has the target to become carbon neutral by 2030 [34]

[15] https://www.repair.org/

Most commercial or university-controlled data centres are designed to maximise performance, responsiveness, and uptime. Indeed, a main point of competition between hosting providers is the uptime metric, expressed in their Service-Level Agreements (SLAs)[16]. There might be, however, a trade-off and diminishing returns effect between maximising uptime, i.e., a resource's availability, and its resource utilisation[17] [18]. The industry-standard requirement of fast responsiveness further means that data centres need to provision computing resources based on load peaks, which may result in wasted resources in the average case.

In this section, we propose an alternative to this model. The reCluster revisits many trade-offs taken in the running of a data centre, attempting to prioritise the reduction of resources as much as possible, even (potentially) at the cost of performance. Our design of the reCluster attempts to identify all high-level decision points where we can reasonably perform technological choices to reduce resource utilisation and provide some ways to tackle these issues.

5.1 General Architecture

The reCluster is made of multiple "nodes". Each node is a physical machine (hardware), and a software stack. The general architecture of a node is summarised in Figure 1.

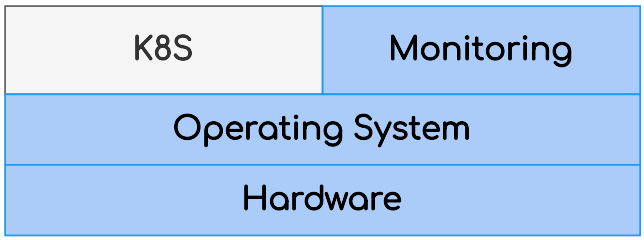

Figure 1: A block diagram describing the main architectural components of a node in the reCluster. The figure highlights where choosing the appropriate component can help reduce resource consumption.

All reCluster nodes are made of "upcycled" hardware, in the form of unused institutional property. This includes devices that have been shelved, discarded as unusable, uneconomical to repair (e.g., broken screens or degraded batteries), formerly in use by staff that has since changed jobs, or any otherwise unused device.

The software stack includes an Operating System (OS), which is always a Linux distribution selected to be as simple, durable, and resource-aware as possible. On top of the OS, we run monitoring suites that assess the state of each node and the whole cluster, and tailored Kubernetes (K8S) software, either in a "worker" or a "control plane" configuration. For Kubernetes, we start from a "vanilla" distribution, since this gives us maximum ability to adjust (and minimise) its internal workings.

[16] See, for example, SLAs from the "big three" providers such as *Amazon Web Services* (https://aws.amazon.com/eks/sla), *Microsoft Azure* (https://azure.microsoft.com/support/legal/sla/kubernetes-service) and *Google Cloud Platform* (https://cloud.google.com/kubernetes-engine/sla)

[17] See also how Low-Tech magazine achieved various uptime percentages at the expense of increased energy and resource requirements (https://solar.lowtechmagazine.com/2020/01/how-sustainable-is-a-solar-powered-website.html)

5.2 Hardware

While we do not yet have hard data on the amount of unused devices typically accumulated within universities, and we are not aware of any studies doing such estimation, large stockpiles of unused devices are an anecdotally significant occurrence in universities.

At this stage, we see six primary devices classes that could make nodes for the reCluster:

(1) Laptop computers
(2) Desktop computers
(3) Server hardware
(4) Board computers
(5) Android-native mobile devices
(6) iOS-native mobile devices

The first three categories are normally powered by x86-based CPUs, that perform well and natively run most server software. The latter three, instead, use various 32- and 64-bits ARM CPU architectures, that may not run natively common server software. Interestingly for our work, ARM architectures are known to have significantly lower power draws at equivalent performance levels [35], though best-in-class ARM-based devices struggle to reach the performance of best-in-class x86-based devices [35, 36]. As the reCluster aims to reduce energy draw as much as possible, ARM devices should be prioritised whenever peak performance is not critical.

Different devices also have — sometimes radically — different life cycles, with ARM devices (smartphones and tablets) generally having a shorter lifespan [15]. Different device classes may also have different "energy balances" with respect to their embedded energy [37], with some devices being discarded before they have used (and hopefully performed useful work) even a fraction of the energy used in that device's manufacturing. This suggests a simple goal for hardware used in the reCluster architecture: to maximise the useful life of each device before it is discarded, and to **aim for each device to perform at least as much useful work (in the form of energy used in software tasks) as it took to manufacture it** before it is discarded.

5.3 Software

Each node is composed of several software components, represented as blocks in the Figures. To choose the overall best fitting software component for a specific task, we separated them in two categories.

A first category, represented by blocks with a red background in the figures, groups software components where customising the existing implementation may achieve significant resource savings.

The second category, represented by blocks with a blue background, groups components where multiple softwares can be chosen in the same role, and there is space to reduce resources by selecting the lowest-footprint implementation. The evaluation of what implementation to choose happens along three main criteria:

(1) License — require software to be FLOSS.
(2) Performance — favour software that consumes less resources and, when possible, runs faster.
(3) Simplicity — favour software that is as simple as possible.

Where we do not have research results to guide us, we assess performance and simplicity "by proxy". For the performance criterion, we refer to documentation and benchmarks when available or, when this is not available, we base our choice on implementation language, as some languages tend to have significantly lower resource consumption [38]. For the simplicity criterion, we look at the size of binaries and number of external dependencies.

Many additional software components in the reCluster architecture are, in the figures and this section, omitted. These omissions might include ancillary services such as remote access (e.g., SSH) or core parts of the Kubernetes architecture (e.g., control web interfaces and CLI tools). We omit them because, from our understanding, no significant energy savings can be achieved by applying our two strategies of customisation or implementation selection.

All nodes (see Figure 1) include an OS and monitoring software.

For OSs, we investigated *K3OS*, *Fedora CoreOS*, and *Alpine Linux*.

K3OS is a minimal OS designed specifically to run *k3s*, a "lightweight" Kubernetes distribution[18], developed by Rancher and part of its ecosystem. There is therefore a risk that choosing this OS might contribute to a vendor lock-in of reCluster to the Rancher ecosystem. Furthermore, *K3OS* is bound exclusively to the usage of *k3s*, and overall designed to remove many aspects of the OS, like the ability to install OS packages[19].

Fedora CoreOS promotes itself as a minimal, immutable, and container-native OS. It comes from a long legacy as an independent Linux distribution that was eventually adopted within the Red Hat product portfolio. Much of what applies to *K3OS* applies to *CoreOS*. In addition, *CoreOS* features an immutable filesystem and, as a container-native OS, it forces any software execution in containers, hindering node benchmarking.

Alpine Linux has been chosen as the OS for all nodes. *Alpine* is general-purpose, designed to be lightweight, and proven to run on low-performance devices[20].

Monitoring via real-time metrics can help detect when the cluster is in a sub-optimal scaling position (i.e., too many or too few active nodes), so that actions can be taken to improve performance. To achieve this we need a monitoring system that periodically observes the cluster. Critically, our cluster needs to observe and analyse containers and node hardware.

Prometheus is the de-facto Open Source standard for collecting metrics, and is the central point of the monitoring stack. It saves all metrics as time-series data in its own database, and it scrapes metrics from components called exporters. *Container Advisor* (*cAdvisor*) is one such exporter, analysing and exposing container resource and performance metrics. *cAdvisor* supports the Prometheus format out of the box, and is already integrated into Kubernetes. For node observation, we rely on *Node Exporter*, an exporter for hardware and OS kernel metrics (specifically *NIX kernels). As part of the *Prometheus* project, it supports its formats natively.

5.3.1 Worker nodes. In a Kubernetes-managed cluster, workloads are executed on worker nodes. Each worker node runs the *kubelet* component, that communicates with the control plane to register the node as a worker in the cluster. Figure 2 shows the high-level architecture of a worker node. Assigned workloads are executed in containers, and bundled as single units called Pods. To run containers, worker nodes additionally need (i) an Open Container Initiaitve runtime (OCI)[21] and (ii) a Container Runtime Interface[22] (CRI).

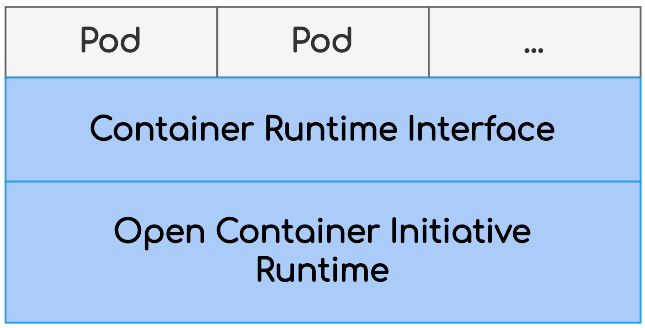

Figure 2: A diagram of the high-level architecture of the Kubernetes component within a "worker" node that can be selected to reduce resources.

The OCI runtime is the low-level software responsible for managing the lifecycle of a container. It starts, stops, and deletes containers, as well as executing commands within containers. Overall, it abstracts the usage of Linux primitives, like cgroups (for specifiying container resources) and Namespaces (isolation of containers).

Multiple OCI runtimes exist, using different programming languages and features. *runc*[23] is an OCI runtime written in Go, that was developed by Docker, specifically for the Docker engine. *Youki*[24] is another OCI runtime written in Rust. Because of its different implementation language, and according to performance benchmarks on its website[24], it is supposed to be lighter and faster than *runc*. *crun*[25], finally, is an OCI runtime written in C, mainly maintained by Red Hat. Because of the choice of C, and according to available benchmarks[24], *crun* provides the best overall performance while using less resources of all three OCI runtimes.

The CRI, instead provides additional functionality on top of the OCI runtime, like managing container persistent storage, container image management, and container networks.

There are two CRIs suitable for reCluster, namely *containerd* and *CRI-O*. *containerd*[26] was developed as the CRI for the Docker engine using the Go programming language. *containerd*'s default OCI runtime is runc, but it can leverage other OCI runtimes, like *Youki* or *crun*. *CRI-O*[27] instead is a CRI specifically designed for Kubernetes as a lightweight alternative for *containerd*. It is written in Go as well, and mainly maintained by Red Hat. Most importantly, *CRI-O* is compatible with all OCI runtimes.

Both Espe et al. [39] and our own previously-mentioned evaluation criteria suggest the usage of *CRI-O*. In combination with *crun*,

[18] See https://k3s.io/

[19] This will become relevant when we initialise worker nodes in Algorithm 1, as we need to install packages in order to gather performance and energy benchmarks.

[20] See the Alpine Linux system requirements at https://wiki.alpinelinux.org/wiki/Requirements.

[21] https://github.com/opencontainers/runtime-spec/blob/v1.0.2/spec.md

[22] https://kubernetes.io/docs/concepts/architecture/cri/

[23] https://github.com/opencontainers/runc

[24] https://github.com/containers/youki

[25] https://github.com/containers/crun

[26] https://containerd.io/

[27] https://cri-o.io/

the best overall choice for reCluster, as the two components are the lightest and provide the best overall performance.

Worker nodes within reCluster are intrinsically heterogeneous, with different hardware, performance, capabilities, and resource consumption. Therefore, when adding a node to reCluster, we need to profile it, gathering relevant data and performance metrics.

To do so, we have implemented a bootstrap procedure, summarised in Algorithm 1, that lets us perform all the necessary steps to add a new worker node. At a high level, it consists of two steps: node initialisation (in the *main* function) and, if the node is being added to reCluster for the first time, node registration (in the *node_registration* function).

The initialisation phase simply starts the worker's baseline services (SSH, monitoring software, and Kubernetes worker set) and notifies the control plane that the worker is ready.

The registration phase handles the profiling of each worker, storing it in what we call *node facts*. *Node facts* include system information such as CPU, RAM, and GPU. Most importantly, they include a variety of performance and energy consumption benchmarks.

The registration starts by acquiring basic information on the node's hardware and its capabilities. Then, the procedure performs benchmarks on all resources. We illustrate, on Algorithm 1's lines 24–30, how one such benchmark might work.

Hardware resources are benchmarked at various load levels, and for each load level, the component's performance and the system's power draw are gathered and stored in the *node facts*. This is done because the power consumption might not be linearly correlated with the resource load. For example, devices might include energy-saving features that allow it to operate most efficiently below a certain utilisation threshold[28], or devices might be degraded, leading to disproportionately high energy draw at high loads.

Tested resources include the system's CPU, RAM, GPU (if present), I/O (including storage and networking), and the system's wake-up time. Energy draw readings can be gathered either using on-board software or via external power meters.

Gathering such rich data is a crucial step in achieving a system that is able to conserve resources. Indeed, this knowledge is later used by the control plane to make decisions about which worker nodes should be switched on or off.

5.3.2 Control Plane nodes. Control plane nodes are responsible for managing all nodes in a Kubernetes cluster. The control plane keeps a record of all information about cluster nodes in a database, and schedules new Pods on available workers using controllers[29].

There are various database backends that could store the cluster's information. Using vanilla Kubernetes, *etcd*[30] a distributed key-value pair database is the only option. SQLite[31], a very small relational database, however, would be the ideal initial choice for reCluster, and could be used through either the *Kine*[32] or *k8s-dqlite*[33] plugins. Another solution is to use *k3s* instead of vanilla Kubernetes, as it supports *SQLite* by default. We have discussed already, however, why we deem this choice not to be viable. *SQLite* might eventually be insufficient on a larger deployment, at which point further analysis should be conducted if the goal is to maintain a relational database, including implementations such as *MySQL*, *PostgreSQL* and *MariaDB*.

The control plane uses "controllers", control loops that observe and change the state of the Kubernetes cluster, to keep it able to optimally fulfil incoming requests. There are three different controllers that are responsible to scale, either the cluster or Pods, to match the performance demand: the Horizontal Pod Autoscaler alters computing resources assigned to each Pod, while the Vertical Pod Autoscaler scales the number of containers in a Pod. Most importantly for resource reduction purposes, the Cluster Autoscaler increases or reduces the number of worker nodes in a cluster.

Pseudocode:

```
1  Function main():
2      if node not registered then
3          node_registration()
4      end
5      init_services(SSH, MONITORING, K8S)
6      recluster_node_ready()
7  end
8  Function node_registration():
9      node_facts ← {}
       // Info(s)
10     node_facts += read_sys_info()
11     node_facts += read_cpu_info()
12     if node has gpu then
13         node_facts += read_gpu_info()
14     end
       // Benchmark(s)
15     node_facts += read_cpu_bench()
16     if node has gpu then
17         node_facts += read_gpu_bench()
18     end
19     node_facts += read_ram_bench()
20     node_facts += read_io_bench()
21     node_facts += read_waketime_bench()
22     recluster_node_add(node_facts)
23 end
24 Function read_cpu_bench():
25     measure, power_draw ← 0, 0
26     for pct ← 0 to 100 by 25 do
27         measure, power_draw += cpu_bench(pct)
28     end
29     return measure, power_draw
30 end
```

Algorithm 1: Worker node initialization and registration

[28] See, for example, ARM's big.LITTLE (https://www.arm.com/technologies/big-little) that combines high-performance and high-efficiency cores in the same CPU.
[29] https://github.com/kubernetes/autoscaler
[30] https://etcd.io/
[31] https://sqlite.org/
[32] https://github.com/k3s-io/kine
[33] https://github.com/canonical/k8s-dqlite

The implementation of the Cluster Autoscaler is, in Kubernetes, left to each individual provider. Cloud providers such as AWS or Microsoft Azure implement their own Cluster Autoscaler components, that can be inspected in the upstream Kubernetes repository[34]. Similarly, the reCluster should implement its own Cluster Autoscaler. Since in our case adding or removing nodes results in turning on or off physical machines, this is a highly critical component, and represents our only "red" block in Figure 3.

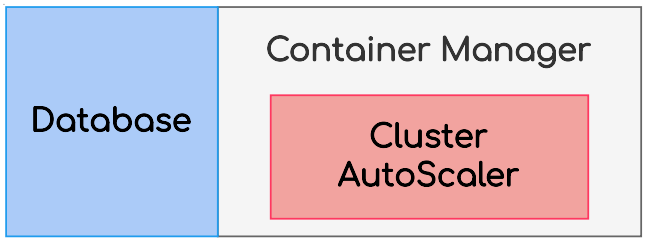

Figure 3: A block diagram of the high-level architecture of the Kubernetes component within a "control plane" node that can be adapted to reduce resources.

As per the official Kubernetes documentation[35], *[the] Cluster Autoscaler automatically adjusts the size of the Kubernetes cluster when one of the following conditions [verifies]*:

(1) *[Pods] failed to run in the cluster due to insufficient resources.*
(2) *[Nodes] in the cluster have been underutilized for an extended period of time, and their pods can be placed on other nodes.*

In reCluster, we propose a strategy where, if we need to switch on additional nodes (scenario 1), the decision always falls on nodes that use the least energy possible. reCluster's Autoscaler selects by default the node that draws the least energy. This behaviour is maintained even when the administrator requires a minimum amount of resources (such as CPU, RAM, or wake time).

We give a high-level overview of our Autoscaler in Algorithm 2. The procedure shows the selection of a worker node that matches a set of requirements.

The scaling procedure starts by initialising a list of worker nodes that are not already active, named *candidates*. This variable will ultimately hold, in its first position, the node to be switched on.

If there are no requirements, we sort the list by power consumption in ascending order, thus selecting the node that draws the least power. Otherwise, the procedure continues by filtering *candidates* by the given requirements. The filter functions gradually remove workers from the *candidates* list, removing nodes that do not fulfil each individual requirement.

At the end of the procedure, we check that there is at least one candidate available. If not, an error is returned and the procedure exits without switching on additional nodes. This might happen under the following conditions:

- `WORKER_NODES` is empty (i.e., no registered worker nodes).
- All worker nodes are already switched on.
- No worker node fulfils the given requirements.

[34] https://github.com/kubernetes/kubernetes
[35] https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler

Pseudocode:

```
// Start from inactive worker nodes
candidates ← filter_by_availability(WORKER_NODES)
if reqs is empty then
    sort_by_power_consumption(candidates, ASCENDING)
else
    if CPU in reqs then
        candidates ← filter_by_cpu(candidates, cpu_thr)
    end
    if GPU in reqs then
        candidates ← filter_by_gpu(candidates, gpu_thr)
    end
    if RAM in reqs then
        candidates ← filter_by_ram(candidates, ram_thr)
    end
    if WAKETIME in reqs then
        candidates ←
            filter_by_waketime(candidates, waketime_thr)
    end
    sort_by_power_consumption(candidates, ASCENDING)
end
if candidates is empty then
    return "ERROR: No suitable worker node found"
else
    // Selected node is the first in candidates
    selected_node ← candidates[0]
    return selected_node
end
```

Input:

- **`reqs`**: Pairs composed by required resource(s) and a minimum threshold value for that resource.
- **`WORKER_NODES`**: List of all available worker nodes.

Output: A worker node fulfilling the given requirements.

Algorithm 2: Cluster Autoscaler upscaling strategy

If the procedure ends successfully, the reCluster's Autoscaler will consistently select nodes that have minimal resource consumption, going towards our goal of creating a more resource-aware self-hosting platform.

6 DISCUSSION, CONCLUSIONS AND FUTURE WORK

As we said in the introduction, the ultimate goal of this work is to enable actionable and lasting change within the organisations we operate in, namely universities. With our analyses and the reCluster proof of concept, we wished to showcase how a systemic approach to infrastructural reappropriation can bring this target within reach in a relatively short time frame.

The policy actions we proposed aim to show that outsourcing is not the only solution, nor it is inevitable. Furthermore, we wanted

to highlight how policies and coordinated action can be powerful enablers for universities to retake control of their infrastructure, institutionalising problems, such as contrasting increasing system requirements, that are normally relegated to individuals.

The actions we proposed are intentionally broad and high-level: they need to be further segmented in smaller elements so that the desired goals can be achieved incrementally. A powerful avenue to do so in the authors' operational context is to leverage university collaboration frameworks that aim to distil and push bottom-up "best practices".

The reCluster, if and when implemented, could represent such a practice. This type of architecture is particularly relevant in contexts of resource scarcity, where bootstrapping infrastructure from locally-sourced hardware can be an advantage. In this sense, the reCluster could also be seen as a "crisis response technology", following up on Chen's invitation to look for technologies that are *"easily deployable, self-sustainable, extensible, and compatible with existing infrastructure"* [9]. A particularly promising avenue for future work in this sense is to investigate the use of Android-native devices, since they are compact, rich in hardware, and easy to relocate if/when needed.

A key bottleneck in such a scarcity scenario, however, is software fragility. The choice of Kubernetes requires special self-critique, as Kubernetes relies on many design choices that are not fully aligned with the values we have sought throughout this article. Prominently, Kubernetes is a complex software system (understanding its structure and workings is a non-trivial task even for trained computer scientists), and has substantial external dependencies. The underlying containerisation technologies also rely on frequent updates and the existence of other infrastructure such as container registries.

At the same time, Kubernetes is extremely complete, having strategies in place for redundancy, scheduling, and many other useful features. The present article hopefully showed how Kubernetes can be leveraged in a short time frame to make a significant first step towards a limits-aware infrastructural reappropriation, showing how many of the obstacles that originally brought institutions away from self-hosting can be nowadays circumvented, paving the way for further action.

Future work will reflect on how to bootstrap the software stack and distribute container images without relying as heavily on the internet, and eventually aims to replace Kubernetes with a different, more minimal, software stack.

The methodology we are adopting is to identify what components in the Kubernetes stack can be removed or replaced by simpler software. Indeed, software optimisation has been described even as "a replacement for Moore's law" [40]. As an an example of simplifiable components, much of Kubernetes' internal communication happens through web APIs. Using HTTP seems to be excessive, and introduces significant overhead in the main use case that we foresee for reCluster-like systems, where all nodes would be physically proximal, and located in the same LAN.

It should also be noted that, while Kubernetes is open source, it started as a Google project. We find the idea of basing our work on technology built by Google to be stimulating: on the one hand, one could claim that by using Kubernetes, we are embedding in our system the very source of the problem we wish to address, namely Google's "big tech" value system; on the other hand, this decision can be seen as a first step towards technological reappropriation that starts by leveraging the work of large technology companies as tools for common good.

The authors are well-aware of this trade-off, and decided to take it to showcase that there is significant margin for action and improvement by taking problematic actors in the status quo and enrolling them to a limits-aware value system.

Because of its use of computing resources that are normally already considered "end-of-life", the reCluster also has a number of potential uses that go beyond self-hosting services, and could co-exist together with existing infrastructure. The reCluster could be used as security testing infrastructure, a distributed computing platform, an infrastructure-as-a-service testbed, or for continuous integration/continuous deployment. The reCluster, where potential mistakes are less costly (since they would be done on containers running over discarded hardware) has also substantial educational use.

As a last remark, we wish to highlight how much the multidisciplinary approach that we took while writing this article shed light on many research gaps and avenues for future work. Many of the issues discussed here are ill-quantified, and would benefit from conducting literature reviews and more precise surveying. Among many, we especially foresee the need to systematise issues of software energy consumption, providing simple means to quantify and benchmark different solutions. Future work will also go in the direction of exploring further the implications of the policy proposals we have outlined here: reversing outsourcing, indeed, would have wide-reaching consequences from a management, HR, and operations point of view. Our immediate next step, however, will go in the direction of creating a way for institutions to survey and monitor the state of unused institutional hardware.

At the end of this article, we are left with the impression that the ambition of having more "limits-aware" digital infrastructure is not only a technical challenge, but is primarily social and political in nature. Many technological enablers are already present, placing the bottleneck at a non-technical level. With this contribution, we tried to make an argument for this process to have a strong contribution by universities. In a context that is as resource-intensive as the global North, it might just be that we already have all the necessary resources to substantially challenge the status quo.

ACKNOWLEDGMENTS

Part of this article has been written in the context of the Erasmus+ "C-FLEX" project, awarded with grant number 2021-1-IT02-KA220-HED-000032115. This project has been funded with support from the European Commission. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

The authors also wish to thank the reviewers for their encouraging and insightful comments.

All links have been accessed 2022-03-31.

REFERENCES

[1] A. Watters, *Teaching Machines: The History of Personalized Learning.* Cambridge, MA, USA: MIT Press, Aug. 2021.

[2] N. Selwyn, T. Hillman, R. Eynon, G. Ferreira, J. Knox, F. Macgilchrist, and J. M. Sancho-Gil, "What's next for Ed-Tech? Critical hopes and concerns for the 2020s," *Learning, Media and Technology*, vol. 45, no. 1, pp. 1–6, Jan. 2020.

[3] J. de Rivera, "A Guide to Understanding and Combatting Digital Capitalism," *tripleC: Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society*, pp. 725–743, Nov. 2020.

[4] B. Williamson, "New Pandemic Edtech Power Networks," *TECHLASH*, vol. 01, p. 17.26, Jun. 2020.

[5] N. Selwyn, "Ed-Tech Within Limits: Anticipating educational technology in times of environmental crisis," *E-Learning and Digital Media*, p. 204275302110229, Jun. 2021.

[6] J. Switzer, R. McGuinness, P. Pannuto, G. Porter, A. Schulman, and B. Raghavan, "TerraWatt: Sustaining Sustainable Computing of Containers in Containers," *arXiv:2102.06614 [cs]*, Feb. 2021.

[7] R. R. Abbing, "'This is a solar-powered website, which means it sometimes goes offline': A design inquiry into degrowth and ICT," in *LIMITS Workshop on Computing within Limits*, Online, Jun. 2021.

[8] M. de Valk, "A pluriverse of local worlds: A review of Computing within Limits related terminology and practices," *LIMITS Workshop on Computing within Limits*, Jun. 2021.

[9] J. Chen, "A strategy for limits-aware computing," in *Proceedings of the Second Workshop on Computing within Limits*. Irvine California: ACM, Jun. 2016, pp. 1–6.

[10] M. S. Silberman and B. Tomlinson, "Precarious infrastructure and postapocalyptic computing," in *CHI '10: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Atlanta, Georgia, USA: ACM, Apr. 2010, p. 3.

[11] T. Fiebig, S. Gürses, C. H. Gañán, E. Kotkamp, F. Kuipers, M. Lindorfer, M. Prisse, and T. Sari, "Heads in the Clouds: Measuring the Implications of Universities Migrating to Public Clouds," Jul. 2021.

[12] M. Teräs, J. Suoranta, H. Teräs, and M. Curcher, "Post-Covid-19 Education and Education Technology 'Solutionism': A Seller's Market," *Postdigital Science and Education*, Jul. 2020.

[13] R. Schaller, "Moore's law: Past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, Jun. 1997.

[14] E. Mollick, "Establishing Moore's Law," *IEEE Annals of the History of Computing*, vol. 28, no. 3, pp. 62–75, Jul. 2006.

[15] V. Forti, C. P. Baldé, R. Kuehr, and G. Bel, "The Global E-waste Monitor 2020: Quantities, flows and the circular economy potential," United Nations University (UNU)/United Nations Institute for Training and Research (UNITAR), Bonn/Geneva/Rotterdam, Tech. Rep., 2020.

[16] K. Parajuly, R. Kuehr, A. Kumar Awasthi, C. Fitzpatrick, J. Lepawski, E. Smith, R. Widmer, and X. Zeng, "Future E-Waste Scenarios," The StEP Initiative, UNU ViE-SCYCLE, and UNEP IETC, Bonn/Bonn/Osaka, Tech. Rep., 2019.

[17] B. Nardi, B. Tomlinson, D. J. Patterson, J. Chen, D. Pargman, B. Raghavan, and B. Penzenstadler, "Computing within limits," *Communications of the ACM*, vol. 61, no. 10, pp. 86–93, Sep. 2018.

[18] R. Balodis and I. Opmane, "History of Data Centre Development," in *Reflections on the History of Computing*, A. Tatnall, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 387, pp. 180–203.

[19] Chaim Gartenberg, "Google Meet, Microsoft Teams, and WebEx are collecting more customer data than they appear to be - the verge," https://www.theverge.com/2020/5/1/21244058/google-meet-microsoft-teams-webex-personal-data-collection-privacy-policy-concerns. Last accessed online: 01-03-2022.

[20] Matthew Finnegan, "Zoom hit by investor lawsuit as security, privacy concerns mount," https://www.computerworld.com/article/3537193/zoom-hit-by-investor-lawsuit-as-security-privacy-concerns-mount.html. Last accessed online: 01-03-2022.

[21] A. Aiken, "Zooming in on privacy concerns: Video app zoom is surging in popularity. in our rush to stay connected, we need to make security checks and not reveal more than we think," *Index on Censorship*, vol. 49, no. 2, pp. 24–27, 2020.

[22] D. Kagan, G. F. Alpert, and M. Fire, "Zooming into video conferencing privacy and security threats," *arXiv preprint arXiv:2007.01059*, 2020.

[23] D. Ong, T. Moors, and V. Sivaraman, "Comparison of the energy, carbon and time costs of videoconferencing and in-person meetings," *Computer communications*, vol. 50, pp. 86–94, 2014.

[24] Y. Tao, D. Steckel, J. J. Klemeš, and F. You, "Trend towards virtual and hybrid conferences may be an effective climate change mitigation strategy," *Nature communications*, vol. 12, no. 1, pp. 1–14, 2021.

[25] L. Burtscher, D. Barret, A. P. Borkar, V. Grinberg, K. Jahnke, S. Kendrew, G. Maffey, and M. J. McCaughrean, "The carbon footprint of large astronomy meetings," *Nature Astronomy*, vol. 4, no. 9, pp. 823–825, 2020.

[26] S. Jäckle, "The carbon footprint of travelling to international academic conferences and options to minimise it," in *Academic Flying and the Means of Communication*. Palgrave Macmillan, Singapore, 2022, pp. 19–52.

[27] H. Suga, "A comparison of bandwidth consumption between proprietary web conference services and bigbluebutton, an open source webinar system," *Bioresource Science Reports*, vol. 13, pp. 1–11, 2021.

[28] C. P. Baldé, V. Forti, V. Gray, R. Kuehr, and P. Stegmann, *The global e-waste monitor 2017: Quantities, flows and resources*. United Nations University, International Telecommunication Union, and International Solid Waste Association (ISWA), Bonn/Geneva/Vienna, 2017.

[29] C. E. Saldaña-Durán and S. R. Messina-Fernández, "E-waste recycling assessment at university campus: a strategy toward sustainability," *Environment, Development and Sustainability*, vol. 23, no. 2, pp. 2493–2502, 2021.

[30] A. Disterheft, S. S. F. da Silva Caeiro, M. R. Ramos, and U. M. de Miranda Azeiteiro, "Environmental management systems (ems) implementation processes and practices in european higher education institutions–top-down versus participatory approaches," *Journal of Cleaner Production*, vol. 31, pp. 80–90, 2012.

[31] M. T. Islam, P. Dias, and N. Huda, "Young consumers'e-waste awareness, consumption, disposal, and recycling behavior: A case study of university students in Sydney, Australia," *Journal of Cleaner Production*, vol. 282, p. 124490, 2021.

[32] A. Arain, R. Pummill, J. Adu-Brimpong, S. Becker, M. Green, M. Ilardi, E. Van Dam, and R. Neitzel, "Analysis of e-waste recycling behavior based on survey at a Midwestern US university," *Waste Management*, vol. 105, pp. 119–127, 2020.

[33] C. M. Schweik and R. English, "Tragedy of the FOSS commons? Investigating the institutional designs of free/libre and open source software projects," *First Monday*, Feb. 2007.

[34] Delft University of Technology, "Sustainability at TU Delft," https://www.tudelft.nl/en/sustainability. Last accessed online: 02-03-2022.

[35] R. V. Aroca and L. M. G. Gonçalves, "Towards green data centers: A comparison of x86 and ARM architectures power efficiency," *Journal of Parallel and Distributed Computing*, vol. 72, no. 12, pp. 1770–1780, Dec. 2012.

[36] M. Yuan, "Performance Analysis for Arm vs x86 CPUs in the Cloud," https://www.infoq.com/articles/arm-vs-x86-cloud-performance/, Jan. 2021.

[37] J. F. Ordoñez Duran, J. M. Chimenos, M. Segarra, P. A. de Antonio Boada, and J. C. E. Ferreira, "Analysis of embodied energy and product lifespan: The potential embodied power sustainability indicator," *Clean Technologies and Environmental Policy*, vol. 22, no. 5, pp. 1055–1068, Jul. 2020.

[38] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. a. P. Fernandes, and J. a. Saraiva, "Energy efficiency across programming languages: How do energy, time, and memory relate?" in *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, ser. SLE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 256–267. [Online]. Available: https://doi.org/10.1145/3136014.3136031

[39] L. Espe, A. Jindal, V. Podolskiy, and M. Gerndt, "Performance Evaluation of Container Runtimes:," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science*. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2020, pp. 273–281.

[40] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, "There's plenty of room at the Top: What will drive computer performance after Moore's law?" *Science*, vol. 368, no. 6495, p. eaam9744, Jun. 2020.