A Systematic Review of GPU Modelling Approaches in Datacenter Simulators

Niels Thiele
Vrije Universiteit Amsterdam, The Netherlands
n.thiele@vu.nl

ABSTRACT

Over the past decades, datacenters have become increasingly important—not only for researchers but also for society at large—by supporting services ranging from websites and video streaming platforms to, more recently, large-scale AI applications. Conducting experiments on real-world datacenter infrastructure is often infeasible due to high costs, limited accessibility, and challenges with reproducibility. Simulators provide a viable alternative, enabling users to evaluate infrastructure and workload configurations with significantly reduced carbon emissions, shorter runtimes, and lower financial overheads. However, despite the growing significance of GPUs in datacenter hardware, their representation in simulation tools remains inconsistent or entirely absent. This study presents a systematic literature review focused on how GPUs are modelled within datacenter simulators. By identifying existing tools, this work explores their underlying design decisions and evaluates the extent to which GPUs are represented in detail. Following a structured methodology, over 300 publications were screened using a combination of keyword-driven and semi-structured search strategies, from which seven simulators were selected for in-depth analysis. Our findings reveal that GPU modelling approaches vary widely across simulators. Some frameworks treat GPUs as near-autonomous subsystems, while others simplify them to CPU-like abstractions. Although the number of identified simulators is relatively small, the results highlight considerable diversity in simulator objectives, modelling granularity, and simulation techniques—ranging from binary indicators to sophisticated architectural representations. This review offers valuable guidance for developers seeking to incorporate GPU-aware functionality into simulators, and for researchers aiming to navigate the design trade-offs inherent in existing simulation frameworks. For researchers, this review provides a comprehensive overview of identified GPU use case, challenges related to GPU integration in datacenter, and which challenges have not been addressed in simulators.

KEYWORDS

Systematic Literature Review, GPU Modeling, Datacenter Simulation, Cloud Simulators, HPC Simulation, Heterogeneous Resources, Performance Modeling

1 INTRODUCTION

With announcing the digital decade policy program, the European Union(EU) has acknowledged the vital role of ICT in general, and datacenters in particular. Besides strengthening the digital skills of the population and extending the network infrastructure, the EU focuses on datacenter-specific aspects, such as climate-neutral edge-datacenters, use of AI, cloud and big data by European companies and offering all public services online [5]. To achieve these goals datacenters are required to process and store the data needed for this digitization process. But not only in the future but already now are datacenters a vital for media and entertainment, cloud service providers and training and usage of artificial intelligence. [19] These services and applications place datacenters in the front row of energy consumption, with a consumption of about 4% of the global electrical energy, with peak power demands of 7.4 GW in 2023. On top of the concerning use of energy are the CO_2 emissions, from producing and operating ICT infrastructure. [19]

Datacenter simulators serve as indispensable tools for modelling and analysing the performance of datacenter architectures and operations. These simulators typically encapsulate core components such as networking, storage, compute, and security within a software-defined framework, abstracting them from physical hardware. This abstraction enables researchers and engineers to explore a wide range of configurations and workloads without the constraints of physical infrastructure, thereby facilitating the evaluation of emerging technologies and operational strategies [23, 62].

The advantages of employing datacenter simulation are substantial. Running experiments on real-world infrastructure expensive, time-consuming and create environmental impact, even though they are producing the results closest to reality. Additionally, are they not equally accessible for everyone. Simulation, on the other hand, offers a controlled environment for testing. Datacenter simulation supports scalable experimentation involving large numbers of jobs and machines, enabling rapid what-if analyses that inform both design and operational decisions. Simulation also plays a valuable role in education and the training of future professionals in the field [62]. Datacenter simulation provides critical insights into energy efficiency, load balancing, and resource allocation—factors essential for optimizing performance and reducing operational expenses [11, 83].

As modern datacenters increasingly accommodate computationally intensive workloads, the demand for high-performance hardware has intensified. Graphics Processing Units (GPUs) have emerged as key accelerators for tasks requiring extensive parallelism, including artificial intelligence (AI), machine learning, high-performance computing (HPC), and large-scale simulations. In contrast to Central Processing Units (CPUs), which are optimized for sequential execution, GPUs are designed to excel at parallel processing, making them particularly effective for data-intensive applications [61]. This paradigm shift has led to the widespread integration of GPUs in cloud computing environments, where they significantly enhance performance across domains ranging from deep learning inference to scientific modelling [55].

The increasing reliance on GPUs in datacenter infrastructures presents both opportunities and challenges. While GPUs provide substantial acceleration for parallelizable workloads, their integration necessitates careful consideration of power consumption, memory architecture, and data transfer bottlenecks [43]. Contemporary GPU architectures, such as NVIDIA's Volta and Ampere series, emphasize advancements like high-bandwidth memory (HBM) and tensor cores to improve computational efficiency [71]. However, fully leveraging their potential requires addressing issues related to workload scheduling, interconnect bandwidth, and coordination across heterogeneous CPU-GPU systems. Efficient GPU utilization is therefore critical for minimizing operational costs and energy usage, making it a central concern in modern cloud infrastructure design [95].

In light of the increasing complexity of datacenter workloads and the pivotal role of GPUs in accelerating computation, it is imperative to examine their impact on datacenter performance and resource management. Datacenter designers, operators, and researchers need tools for optimizing, developing new approaches and generally understanding datacenters. Simulation is part of that tool set to help stakeholders achieve their goals. This study aims to investigate how GPUs are integrated into datacenter simulators. To this end, a structured, keyword-based literature review was conducted to identify and categorize existing design approaches for GPU-enabled datacenter simulators.

The remainder of this paper is structured as follows: Section 2 provides general background information on GPU technologies. Section 3 outlines the study design, including the research questions and methodology. The results are presented in two sections. Section 5 provides an overview of the identified simulators, their design approaches and their perspective on GPUs. The open challenges that have been observed during this literature survey are described in section 6. Section 7 situates this study within the context of existing literature.

2 BACKGROUND

This section provides information about datacenter simulators their overall architecture, benefits, and components. Further, a foundational overview of Graphics Processing Units (GPUs) is given, covering their historical evolution, architectural structure, processing pipeline, and virtualization techniques. The goal is to establish a shared understanding of GPU concepts relevant to this study. Finally, a small comparison to CPUs is made.

2.1 Datacenter Simulator

Datacenter simulators have become indispensable tools for the design and evaluation of cloud computing environments. These tools enable researchers and practitioners to investigate the performance, efficiency, and scalability of datacenter architectures and management policies without the prohibitive cost and complexity associated with deploying real-world infrastructures [10, 16, 62].

Contemporary datacenter simulators predominantly employ *discrete-event simulation* (DES) as their underlying methodology. In DES, system behaviour is modelled as a sequence of discrete, timestamped events, thereby enabling efficient simulation of large-scale systems [62]. Simulator architectures typically encompass representations of physical infrastructure (e.g., servers, racks, switches), virtualized resources (e.g., virtual machines, containers), and application-level workloads. A generic model is depicted in figure 1. For instance, platforms such as CloudSim and OpenDC adopt a hierarchical modelling approach, where physical hosts manage virtualized components that in turn execute user-defined tasks [10, 62].

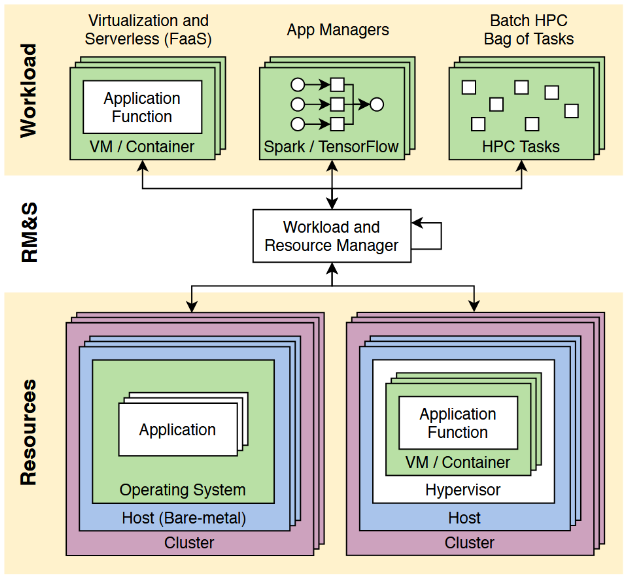

Figure 1: "Generic model for datacenter operation." [62]

The primary motivation for utilizing simulation lies in its ability to support the design, testing, and optimization of cloud systems in a flexible and cost-effective manner. Datacenter simulators are widely used for evaluating resource scheduling and allocation strategies, investigating energy-aware provisioning techniques, analysing workload behaviours, and conducting capacity planning studies under varying operational scenarios [10, 16]. Furthermore, simulators provide a controlled environment conducive to reproducible experimentation and serve as valuable pedagogical tools in both academic and industrial settings [62]. Different implementations of datacenter simulators - or in the case of CloudSim its extensions - focus on different aspects of datacenters. So provides CloudSim a general cloud datacenter implementation, with CPU and memory [17]. CloudSim's extension NetworkCloudSim implements Networking capabilities for Datacenters [28]. GreenCloud also simulates networking in a datacenter environment, with the additional focus of energy-awareness [47]. secCloudSim focuses on security in the cloud, in particular authentication [76], which is, in fact, an extension of the iCanCloud simulator [67].

Simulation offers several advantages over analytical modelling and real-world experimentation. First, it significantly reduces the financial and operational costs associated with deploying physical test beds, particularly when exploring large-scale scenarios. Second, it enhances experimental reproducibility by allowing users to replicate simulation runs under identical input conditions. Third, it enables scalability by supporting the modelling of vast datacenter topologies and high volumes of concurrent workloads, which could be achieved with analytical models, but not with real-world datacenters. Fourth, simulators offer a high degree of flexibility, allowing for rapid prototyping and evaluation of alternative architectures or policy configurations. Finally, simulation can offer a higher flexibility, with the ability to highlight different aspects of datacenters and combine them, which is more difficult to achieve in analytical modelling. [62]

Simulation models employed in datacenter simulators differ based on their level of abstraction and specific application purposes. A significant number of these simulators utilize discrete-event simulation, wherein system changes are represented as a sequence of events occurring between components over time. The discrete nature of this approach assumes that no changes occur between two successive events[62], in contrast to continuous simulation models, which presume ongoing change over time. One possible explanation is the discrete nature of the sampled data that has been obtained from real-world datacenter. Discrete-event simulation is adopted by simulators such as CloudSim [17], OpenDC [62], and Het-Sim [41], among others. Alternative modelling approaches include analytical models, which abstract the simulation into mathematical formulations; this technique is employed by, for example, MERPSYS [22] and PyPassT [68].

The network model captures the communication behaviour between simulated components. Simplified models treat bandwidth as a static, shareable resource, while more elaborate simulators incorporate datacenter network topologies (e.g., fat-tree, Jellyfish) and routing strategies. These models are essential for analysing congestion and identifying communication bottlenecks [10].

Resource scheduling and load balancing policies play a critical role in simulation accuracy. Simulators typically include built-in policies such as time-shared and space-shared scheduling, and often support user-defined extensions. These mechanisms determine how virtual machines and tasks are mapped to physical resources, influencing key performance metrics such as latency, throughput, and energy consumption [16].

Workload representation in simulations can be based on synthetic models or real-world traces. Common workload parameters include task arrival rates, resource demands, duration, and spatial or temporal distributions. Some simulators, such as CloudAnalyst, support the modelling of geographically distributed users and time-varying usage patterns, which are particularly relevant for web-scale applications [16].

2.2 GPUs

Graphics Processing Units (GPUs) have evolved from dedicated hardware for rendering graphics into powerful accelerators for high-performance computing (HPC) and artificial intelligence (AI). Their inherently parallel architecture makes them particularly well-suited for data-intensive workloads, outperforming traditional Central Processing Units (CPUs) in many computational scenarios [61, 69].

In the earliest stages of GPU development, the primary function of graphics processors was limited to rendering basic graphical elements such as dots and lines. Foundational concepts like vertex transformation and the graphics pipeline emerged during this period. Over the course of roughly six evolutionary phases, GPU programmability has increased significantly. A major milestone occurred in 1999 with the release of the Nvidia GeForce 256, the first configurable GPU. The introduction of APIs such as DirectX simplified development, shifting from hardware state modelling to programmable shaders. This marked the transition from fixed-function to fully programmable hardware, leading to the emergence of specialized processors for tasks such as audio and video (de)compression. According to Peddie et al., the current era is defined by the use of mesh shaders, which integrate multiple shader types into a unified shader architecture, emphasizing their individual strengths [69].

The GPU processing pipeline consists of several stages. Initially, data is transferred from the system's main memory into the GPU's memory. This is followed by the execution of kernels—small programs that run concurrently across thousands of cores. Threads are distributed to cores, and data flows through the hierarchical memory structure. Upon completion, the processed data is transferred back to the system's main memory [55].

Terminology for GPU components can vary across different frameworks [55, 69]. This study adheres to the terminology used within the CUDA programming model for consistency. Despite the diversity in nomenclature, GPU architectures generally follow a hierarchical structure, as illustrated in Figure 2.

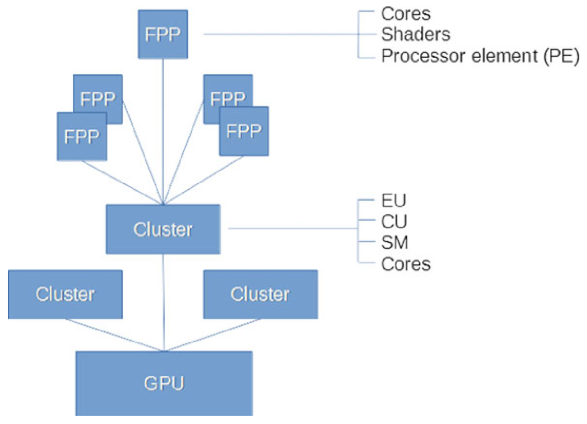

Figure 2: Structured GPU component overview [69]

At the highest level of abstraction in CUDA, the grid represents the complete problem space, composed of multiple blocks. Each block contains a group of threads and is independently assigned to a Streaming Multiprocessor (SM). Within an SM, threads from a block are executed in parallel. SMs consist of cores that are organized into groups of 32 threads called warps. A warp executes the same instruction across different data points simultaneously. Threads are mapped to individual cores, which represent the fundamental computational units of the GPU [55]. An overview of GPU components is shown in Figure 3.

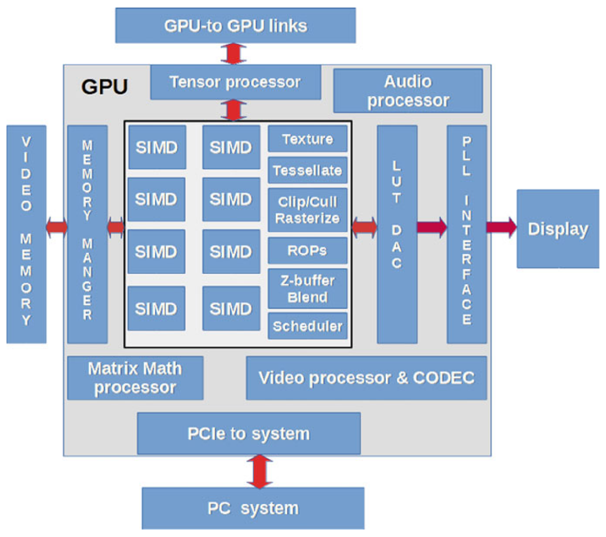

Figure 3: Component overview of a GPU [69]

Similar to computational units, the memory architecture of a GPU is also hierarchically organized. At the top of the hierarchy is global memory, which serves as the GPU's main memory. It has a large capacity and high latency, and is accessible by all threads. It is typically used for storing large datasets shared across the entire GPU. The next tier is shared memory, which is faster and lower in latency. It is accessible to all threads within the same block and facilitates intra-block communication and temporary storage during computation. At the lowest level are registers, which are the fastest memory units and are private to individual threads. If the required data exceeds the capacity of registers, local memory is used instead. Local memory is slower than registers but faster than global memory [55].

Cloud computing relies heavily on virtualization to enable resource sharing across users [95]. Various techniques have been developed to support GPU virtualization in cloud environments. One approach is full GPU pass-through, where an entire physical GPU is allocated to a single virtual machine (VM) or container. Another technique involves software-based sharing of a single physical GPU among multiple VMs or containers. A third method, introduced by Nvidia, employs hardware-assisted separation, enabling a physical GPU to be partitioned into multiple isolated instances capable of running distinct workloads concurrently.

2.3 CPU vs. GPUs

The main difference between a CPU and GPU lies in the purpose of use, as well as in the design philosophy. CPUs are designed to support a large range of tasks and have a more general purpose. Typically, they sequentially execute multiple tasks, ranging from running an operating system process, executing complex logic, managing input and output operations, and processing everyday applications, such as web browsing [55].

The vendor- and design-generation-independent architecture of CPUs, consists of various components. A control unit (CU) orchestrates the functioning of the CPU. The memory of the computer, the arithmetic/logic unit (ALU), and input and output devices are given the instructions by the CU. The ALU performs the operations that are of arithmetic or logical nature. To temporarily keep data and instructions within the CPU, small and fast storage is used, the so-called registers. Frequently accessed data and instructions are kept within the cache. An overview can be seen in figure 4

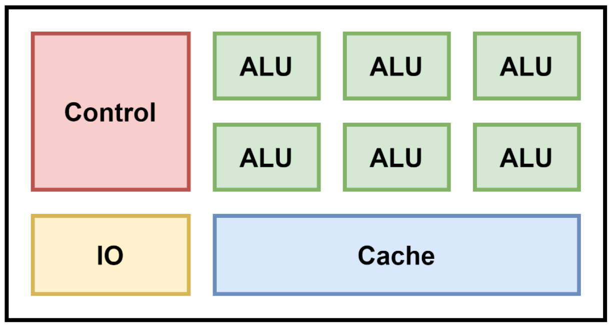

Figure 4: High-level CPU architecture overview[55]

GPUs come generally in one of the two following flavours, integrated or discrete. Integrated GPUs are within the same die as the CPU, and access the same RAM as the CPU. They come at the cost of fewer cores and therefore less computation power, compared to a discrete GPU. An architecture overview can be seen in figure 5. Discrete GPUs are connected via PCIe to the CPU and have their own main memory. [30]

As aforementioned, is the architecture of a GPU built around the philosophy of a high degree of parallelism, by employing many cores with fewer capabilities.

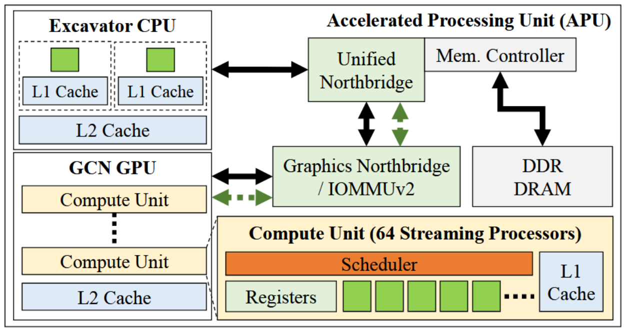

Figure 5: "Architecture of an integrated GPU" [30]

3 STUDY DESIGN

This systematic literature review adopts the methodology proposed by Carrera-Rivera et al. [18]. The process involves formulating research questions, identifying appropriate keywords and their synonyms, constructing search queries, and gathering sources from selected academic libraries. Inclusion, exclusion, and quality assessment criteria are defined to filter the results. Due to the limited number of relevant studies retrieved in the initial iteration, an additional semi-structured search was conducted. The extracted information is subsequently analysed and presented, structured according to the individual research questions. The details of each methodological step are described in the following subsections.

3.1 Research Questions

This literature review seeks to address the following research questions:

RQ1 **What are the primary application domains and computational use cases of GPU deployment in datacenters, and what operational and computational challenges do they introduce?**

RQ2 **Which existing datacenter simulators represent GPUs as resource, and what aspects do they cover?**
This primary research question is answered by identifying and characterizing relevant simulators, including their intended use cases and the terminology they employ. To narrow down the general question of modelled aspects, the following questions are stated:

RQ2.1 *What is the intended use case of the Datacenter simulator? What terminology is introduced? What is the Architecture of the Simulator?*

RQ2.2 *Which simulation approach is used for the Datacenter simulator? In which format is the input provided for the simulation?*

RQ2.3 *To what level of detail are GPUs modelled as infrastructure components, and how are they integrated into the simulation process?*

RQ3 **Which operational and computational challenges introduced by GPU use are not yet fully addressed by current datacenter simulators?**
The goal of this research questions is to disclose aspects of present challenges, that can be filled in future works.

3.2 Search Strategy

To capture the state of the art in this domain, various literature search techniques are available, including snowballing, systematic mapping, undirected search, and systematic literature review (SLR) [18, 46]. While undirected and snowballing approaches can be effective, they often lack structure and reproducibility, increasing the risk of missing relevant publications. In contrast, a systematic literature review ensures a transparent, repeatable process and helps mitigate these issues [65]. For this literature survey, a mixed approach has been chosen. First a systematic literature review was conducted, then a semi-structured, snowball-like second search was executed.

For this study, the Article Information Parser (AIP)[1], developed by the AtLarge Research Group[2] at VU Amsterdam, was employed to ensure reproducibility. AIP integrates data from DBLP [4], Semantic Scholar [7], and AMiner [2] into a unified PostgreSQL database [1].

The SQL search query, shown in Figure 6, retrieves the title, the abstract, the venue, and the year the article was published in, and the DOI. The results are then joined with the colon-separated list of authors of the article. To narrow down the search results, a filter was constructed by using relevant keywords derived from the research questions in the ‘WHERE‘ clause. These keywords, reflect different datacenter applications combined with simulation-related terms. They cover different aspects of datacenter use cases, such as cloud, grid and high-performance computing or plainly datacenter and simulation. To cover a broad spectrum of terms simulation, simulator and their related forms are abbreviated to "simulat", to perform a substring match. It was also attempted to cover different spelling variations of the other keywords.

A relevance score is computed based on the presence of target keywords in the title and abstract, prioritizing GPU-related works. If the title or abstract includes the terms "GPU modelling" 3 points are assigned, while for only mentioning "GPU" in the title 2 points are assigned. 2 points were also given for "heterogenous computing", "heterogenous resources", "accelerator-aware", "GPU", "graphics processor" or "graphics processing unit" in the abstract. If the title matches "data_center_simulat" or "cloud%sim" - where the underscores mask one character and the percent symbol any number of characters - then 1 additional point is given. The results are ordered by the descending relevance score.

The query also calculates a relative citation index (total citations divided by publication years) to measure a publication’s academic impact. To supplement citation data missing from the AIP dataset, values were retrieved via the Crossref API[3] using a custom Python script, available on GitHub[4].

[1]https://github.com/atlarge-research/AIP
[2]https://atlarge-research.com
[3]https://api.crossref.org/swagger-ui/index.html#/Works/get_works
[4]https://github.com/noleu/literature_study

```
WITH author_agg AS (
    SELECT app.paper_id, STRING_AGG(a.name, ', ') AS authors
    FROM author_paper_pairs app
         JOIN authors a ON app.author_id = a.id
    GROUP BY app.paper_id)
SELECT
    p.title,
   p.abstract,
   p.venue,
   p.year,
   p.doi,
   aa.authors,
   -- relative citation index
   0 as relative_citation_index, --n_citations is zero, will
      be filled in post-processing
   (CASE
        WHEN p.title ILIKE '% GPU modeling%'
            OR p.abstract ILIKE '% GPU modeling%' THEN 3
        WHEN p.title ILIKE '% GPU%'
            OR p.title ILIKE '%heterogeneous computing%'
            OR p.abstract ILIKE '%heterogeneous computing%'
     THEN 2
        WHEN p.abstract ILIKE '%accelerator-aware%'
            OR p.abstract ILIKE '%heterogeneous resources%'
     THEN 2
        WHEN p.abstract ILIKE '% GPU%'
            OR p.abstract ILIKE '%graphics processor%'
            OR p.abstract ILIKE '%graphics processing unit%'
      THEN 1
        else 0
        END +
    CASE
        WHEN p.title ILIKE '%data_center_simulat%' THEN 1
        WHEN p.title ILIKE '%cloud%sim%' THEN 1
        ELSE 0
        END
       )  as relevance_score

FROM publications p
         JOIN author_agg aa ON p.id = aa.paper_id
WHERE p.year >= 2015
  AND (
    (p.title ILIKE '%data_centre_simulat%'
        OR p.title ILIKE '%data_center_simulat%'
        OR p.title ILIKE '%cloud_simulat%'
        OR p.title ILIKE '%cloudsim%'
        OR p.title ILIKE '%cloud_computing_simulat%'
        OR (p.title ILIKE '%grid_computing_simulat%'
            OR p.title ILIKE '%grid_compute_simulat%')
        OR p.title ILIKE '%
     high_performance_computing_simulat%'
        OR p.title ILIKE '%HPC_simulat%')
    )
ORDER BY relevance_score DESC;
```

Figure 6: SQL Query used to identify relevant literature.

In addition to the structured AIP-based search, a semi-structured query process using Google Scholar[5] was conducted. A preliminary list of datacenter simulators was assembled based on prior literature reviews [44, 63, 64, 79, 96] and simulators identified in prior work as candidates for GPU integration [31, 42, 45, 77, 101]. For each simulator, Google Scholar was queried using the format *"$NAME_OF_SIMULATOR" GPU*, replacing `$NAME_OF_SIMULATOR` with the simulator's name (e.g., CloudSim, PICS). The first page of results for each query was reviewed to identify any additional relevant publications.

A second snowballing-based search has been conducted to answer research question RQ2. With the use of Google Scholars cited by function, follow-up articles were identified and their content scanned. Articles that are not using the simulator were then not further considered. This search was conducted, to identify topics that had not been covered in the article introducing the simulator. The articles found in this search are not accounted for in the statistics.

3.3 Selection Criteria

To ensure high quality and relevance of the included studies, the following inclusion and exclusion criteria were applied:

I1 The abstract indicates that the study implements or models GPUs in a datacenter/cloud/grid/HPC simulator or explores GPU-related aspects such as power consumption or computational performance.

I2 The full text contains evidence of datacenter simulations incorporating GPU-enabled infrastructure.

E1 Studies that are not peer-reviewed (e.g., preprints) are excluded to maintain quality standards.

E2 Publications not in English or German are excluded due to time and accuracy constraints related to translation.

E3 Duplicate entries across the datasets are removed.

E4 Studies focused solely on other datacenter topics (e.g., VM migration, load balancing) without specific reference to GPUs or heterogeneous resources are excluded.

E5 Studies published before 2015 are excluded, as 10 years of publication time are sufficient to cover state-of-the-art implementations, and pioneers of the field.

[5]https://scholar.google.com

3.4 Data Extraction

The selection and screening process was conducted in multiple rounds:

(1) **Initial filtering:** The AIP database was queried, and relative citation indices were computed. Articles with a relevance score of 0 were discarded.
(2) **Abstract screening:** Remaining articles were screened based on their abstracts. Articles that clearly did not meet the inclusion criteria were excluded. Ambiguous cases were marked for further review.
(3) **Full-text screening:** For articles flagged as undecided, a more detailed skim of the full text was performed to determine inclusion.
(4) **Semi-structured search:** Articles identified via the Google Scholar search underwent the same abstract and full-text screening procedure.
(5) **Data synthesis:** The content of the included studies was summarized, and key findings were extracted and categorized by research question.

Table 1 summarizes the number of articles retained and excluded at each stage of the screening process. Initially, the AIP dataset contained over 18 million entries. After filtering out duplicates and irrelevant entries (e.g., those with a relevance score of 0), abstract and full-text screening reduced the dataset to three selected articles. The semi-structured search added three more, bringing the total to six. Across all search iterations, approximately 500 article titles, abstracts, and previews were reviewed for potential inclusion. The articles that have been rejected, mostly matched the exclusion criteria E4, covering datacenter simulation, however a different aspect.

Phase	Rejected	Undecided	Selected
Abstract Screening	185	96	2
Full-Text Screening	280	0	3

Table 1: Selection status at the end of each screening round.

Only for the articles about Cloudy [86], GPUCloudSim [85], Het-Sim [41], and MERPSYS[22] could follow-up work be found. The search for GPUCloudSim, revealed 32 citations of the original work, Het-Sim has 2 citations, and MERPSYS has 40 citations. While reading the articles it became clear, that 1 citation accounted for GPUCloudSim, actually uses Cloudy. Summarizing work such as reviews and surveys were not included in the data extraction process.

4 GPUS IN DATACENTER AND THEIR OPERATIONAL AND COMPUTATIONAL CHALLENGES

This section of the literature survey covers the answers to Research Question RQ1 "In which scientific and industrial domains are GPUs deployed within datacenters, and what computational problems do they address?". In the first section, we discuss application domains in which GPUs play an important role. In the second half of this section, the operational and computational problems arising from the use of GPUs in datacenters will be discussed.

4.1 Applications of GPUs in Datacenters Across Domains

Graphics Processing Units (GPUs) have become indispensable in modern datacenters due to their massive parallelism and performance benefits across a diverse array of workloads. Their adoption spans several critical domains, from research applications, such as mathematical simulation, to user-facing applications such as media streaming. In this subsection of the literature survey, these application domains are discussed in more detail.

4.1.1 Deep learning and AI. GPUs have become indispensable in AI workloads, particularly for accelerating both the training and inference phases of deep learning (DL) models. During the training process, DL models undergo iterative computations involving forward and backward propagation, as well as parameter updates, which are inherently parallel and computationally intensive. GPUs are well-suited for such operations due to their high throughput capabilities and support for distributed training schemes, such as PyTorch's 'DistributedDataParallel' and TensorFlow's 'MultiWorkerMirroredStrategy' [100]. To reduce training time for large-scale datasets and complex architectures, training workloads are typically deployed across multiple GPUs. These workloads exhibit distinct characteristics, including gang scheduling, exclusive resource allocation, and sensitivity to hardware placement—particularly in relation to interconnect technologies like PCIe and NVLink [36, 100]. As such, GPU-accelerated datacenters enable the efficient development and deployment of DL models across various domains, including computer vision, natural language processing, and reinforcement learning, where resource heterogeneity and communication overhead significantly impact performance [36].

In the context of inference workloads, GPUs offer the computational efficiency necessary to meet stringent service-level objectives, such as sub-100ms response times for real-time applications in personalized recommendation, facial recognition, and language translation. Inference tasks are typically latency-sensitive and resource-light, but must be capable of scaling dynamically in response to fluctuating query loads. To maintain low-latency and high-throughput operation, inference workloads frequently utilize batching mechanisms and performance-aware scheduling strategies. However, traditional CPU-oriented orchestrators often fail to accommodate the temporal and resource-specific requirements of GPU-based inference. Studies have demonstrated that GPU-aware orchestration layers—such as Knots—can enhance both resource utilization and energy efficiency by leveraging real-time GPU telemetry and topology-aware scheduling decisions [97]. These insights underscore the critical role of GPUs in ensuring both the computational scalability of DL training and the responsiveness of AI inference services.

4.1.2 *User Facing Services.*

GPUs are increasingly deployed in datacenters to meet the performance demands of User-Facing services such as real-time translation, voice recognition, and web search. These services typically involve latency-sensitive deep learning inference tasks, where the end-to-end response time must be within strict Quality-of-Service (QoS) targets, often ranging between 100–300 ms [102]. For example, applications like Apple Siri or Google Translate rely on deep neural networks (DNNs) accelerated by GPUs to provide real-time responses to user queries. The adoption of GPUs for such services is motivated not only by their high computational throughput, but also by their ability to maintain responsiveness under varying workloads. However, these services tend to follow diurnal usage patterns, leaving GPU resources underutilized during off-peak times. This has prompted efforts to co-locate user-facing services with throughput-oriented batch applications to improve resource utilization without violating QoS constraints [102].

To address the challenges of co-locating such workloads, new scheduling mechanisms and runtime systems have been proposed. For instance, C-Laius, a runtime system designed for spatial multitasking GPUs, dynamically allocates “just-enough” computational resources to user-facing tasks to meet their latency targets while assigning the remaining GPU capacity to batch jobs. Unlike traditional time-sharing models that queue tasks sequentially, spatial multitasking allows simultaneous execution of kernels from different applications. C-Laius predicts task performance, monitors progress, and reallocates resources at runtime to prevent QoS violations due to contention for shared resources such as memory bandwidth and Streaming Multiprocessors (SMs). Evaluation on NVIDIA RTX 2080Ti GPUs demonstrated that C-Laius could increase GPU utilization by up to 35.9% while ensuring that no QoS violations occur in multi-tenant scenarios [102]. Similarly, Knots, an extension of Kubernetes, introduces GPU-awareness in orchestration by monitoring GPU-specific metrics to reduce queuing delays and ensure SLA compliance for latency-critical user services [97].

4.1.3 *Batch and HPC Simulations.*

Batch and high-performance computing (HPC) workloads represent another significant category where GPUs are extensively utilized in datacenter environments. These workloads include scientific simulations, weather forecasting, data analytics, and computational modelling, all of which benefit from GPUs’ high arithmetic throughput and parallel processing capabilities. Compared to user-facing services, batch and HPC jobs are typically throughput-oriented and less sensitive to latency, enabling better scheduling flexibility. For example, organizations like Facebook have built large-scale GPU clusters tailored for machine learning (ML) workloads, supporting deep neural network training, gradient-boosted decision trees, and logistic regression tasks [99]. Such systems rely on telemetry and performance profiling tools to monitor and optimize the execution of resource-intensive batch jobs at scale.

A notable trend in this domain is the use of GPU virtualization to enhance resource utilization and management efficiency. Technologies like NVIDIA GRID vGPU and rCUDA allow multiple virtual machines to share GPU resources either through direct or mediated pass-through, making it possible to consolidate workloads and reduce idle times [39, 51]. This is particularly advantageous in HPC settings, where GPU underutilization is a persistent challenge due to fragmented job resource requirements. Studies indicate that virtual GPU systems can maintain near-native performance while enabling features like live migration, suspend/resume, and dynamic resource allocation. In some experiments, datacenters using virtualized GPUs achieved significant throughput gains and energy savings, especially when combining long-running batch jobs with short, opportunistic workloads on the same physical hardware [39]. These advances are pivotal for scaling HPC workloads in modern cloud and hybrid datacenter environments.

4.1.4 *Microservices and Cloud Services.*

Microservices and modular cloud services are increasingly leveraging GPUs to accelerate specific components of complex applications, particularly in media processing, machine learning inference, and data analytics. With the shift from monolithic to microservice architectures, cloud applications are decomposed into smaller, independently deployable components that can scale and execute concurrently. This approach aligns well with GPU-enabled spatial multitasking, where multiple application components share the same GPU through logical partitioning of resources such as Streaming Multiprocessors (SMs) and memory. For instance, applications like real-time video analytics or AI-driven content enhancement often consist of chained components—decoding, enhancement, and inference—each of which can be offloaded to the GPU [81]. By assigning GPU compute slices to individual microservices, platforms can increase throughput while maintaining responsiveness.

Supporting microservices with GPUs introduces significant scheduling and data movement challenges. In synchronous microservice chains, components depend on the timely delivery of intermediate results, and inefficient GPU resource sharing can lead to bottlenecks and queuing delays. To address these issues, recent research proposes dynamic resource management policies that monitor online performance and adapt GPU allocations in real time. These systems can categorize components based on workload characteristics and dynamically reallocate streaming multiprocessors (SMs) to balance resource utilization and reduce input loss or latency spikes [81]. Complementing this approach, He et al. [33] introduce DxPU, a disaggregated GPU infrastructure tailored for cloud environments, which addresses the scalability and flexibility limitations of conventional server-centric GPU deployments. Traditional architectures statically attach GPUs to host servers, resulting in underutilization and increased maintenance complexity. DxPU decouples GPU resources from physical hosts by utilizing proxies at the host and the GPU side, enabling more granular and efficient resource provisioning across microservice-based workloads.

4.2 Operational and Computational Challenges

Even though GPUs bring significant benefit, compared to a CPU-only approach, through their high degree of parallelization and throughput, some challenges also arise from the use of them. These challenges cover the aspects, among others, resource management and scheduling, power and energy concerns, and fault-tolerance. In this subsection, the computational and operational challenges, will be discussed.

4.2.1 Resource Management and Scheduling. Resource management and scheduling are significantly more complex for GPUs than for CPUs due to fundamental differences in architecture, usage patterns, and orchestration tooling. Unlike CPUs, which are well-supported by mature schedulers that offer fine-grained resource allocation (e.g., per-core scheduling), GPUs are typically managed as coarse-grained, monolithic resources. For instance, schedulers like Kubernetes and Mesos allocate GPUs at the device level, giving an entire GPU to a single container or virtual machine regardless of actual demand. This leads to inefficient utilization, particularly when the GPU-bound application does not saturate the GPU's compute and memory capacity. Furthermore, containers often overestimate their GPU resource requirements, exacerbating underutilization and causing queuing delays that impact latency-sensitive workloads [97].

Another complicating factor is the lack of real-time GPU visibility and dynamic control in current orchestration systems. While CPUs benefit from extensive support for metrics such as load, memory usage, and task affinity, GPUs lack similar integration in container orchestrators. For example, Kubernetes does not natively support preemption or fine-grained load balancing for GPU-bound tasks, and cannot easily query GPU-specific telemetry like Streaming Multiprocessor (SM) utilization, memory pressure, or PCIe bandwidth consumption [97]. This blind spot makes it difficult for orchestration systems to schedule GPU workloads intelligently, especially in environments where GPU and CPU tasks must co-exist or when different workloads (e.g., batch and real-time inference) are competing for limited GPU resources. Without GPU-aware scheduling extensions or custom resource managers, the system risks both performance degradation and energy inefficiency.

Simulation-based evaluation can play a vital role in improving GPU scheduling strategies. By using datacenter simulators that support GPU modelling, researchers can prototype and assess alternative resource management policies—such as fine-grained SM allocation, workload-aware scheduling, or preemption strategies—without risking service disruption or hardware inefficiency. This enables systematic exploration of how orchestration frameworks could better accommodate GPU-specific metrics and sharing mechanisms under varied workload mixes.

4.2.2 Virtualization and Disaggregation. Virtualization and disaggregation present more challenges for GPUs than for CPUs due to architectural and software limitations that hinder flexible resource sharing. While CPUs are already well-supported in virtualized environments with fine-grained control over core and memory allocation, GPUs typically require full-device pass-through virtual machines, limiting their sharing potential. For instance, in pass-through mode, a GPU is exclusively assigned to a single VM, preventing concurrent use by multiple tenants and leading to underutilization [51]. Although mediated pass-through technologies like NVIDIA GRID vGPU introduce shared access by partitioning GPU memory and scheduling compute time slices, these solutions are constrained by static vGPU profiles and limited to one vGPU per VM. Moreover, the scheduling mechanisms—such as Fixed Share or Equal Share—lack dynamic adaptability, often resulting in suboptimal GPU cycle distribution across workloads [51].

Disaggregation, where GPUs are decoupled from the host server and pooled for flexible access, introduces further complications. Traditional PCIe-based disaggregation architectures restrict the scope to rack-scale deployments due to physical cable length and bandwidth limitations. Emerging solutions like DxPU [33] aim to overcome these constraints by virtualizing GPUs at the PCIe transaction layer and enabling datacenter-scale pooling via networked proxies. However, even these advanced systems face challenges related to software compatibility, latency overhead, and limited support for diverse GPU models and frameworks. For example, while DxPU achieves less than 10% performance degradation in most cases, maintaining high compatibility across software stacks (e.g., CUDA, OpenGL) and managing dynamic workload migrations remains difficult. Compared to CPU disaggregation, where memory and I/O virtualization are mature, GPU disaggregation demands specialized hardware-software co-design and remains a nascent but critical area for scaling heterogeneous workloads in cloud datacenters [33].

Simulators provide a test bed for evaluating emerging GPU virtualization and disaggregation techniques. For example, architectural trade-offs between pass-through, vGPU, and networked GPU models can be benchmarked in silico to understand their performance implications, resource sharing efficiency, and compatibility constraints. This is particularly important for early-stage solutions like DxPU, where real-world deployment remains complex and costly.

4.2.3 Sustainability. The integration of Graphics Processing Units (GPUs) in datacenters has significantly influenced the sustainability profile of these facilities. As GPU-intensive workloads—particularly those in AI and high-performance computing—proliferate, they drive up energy consumption dramatically. Studies show that AI racks powered by GPUs can consume nearly seven times more electricity than their CPU-based counterparts, leading to substantial increases in heat generation and cooling demands [20]. This elevated thermal output, if not addressed through advanced cooling strategies, exacerbates operational inefficiencies and contributes to higher carbon footprints. Consequently, the adoption of energy-efficient cooling technologies such as direct-to-chip and immersion cooling is essential to mitigate environmental impact, especially in facilities running large-scale AI operations [52].

Beyond electricity usage, GPU-driven datacenters also pose challenges to water sustainability. Traditional cooling systems often rely heavily on water, making datacenters significant contributors to local water stress. However, innovations such as hybrid air-liquid cooling and the use of non-potable water sources (e.g., seawater) offer promising solutions. For example, studies demonstrate that facilities employing seawater-based cooling can maintain a Water Usage Effectiveness (WUE) of 0, essentially eliminating potable water consumption for cooling purposes [52]. These developments are crucial as the sector seeks to align with global climate goals and regional regulations aimed at reducing both greenhouse gas emissions and water consumption. Overall, the environmental implications of GPU adoption in datacenters underscore the need for integrated sustainability strategies that address both energy and water usage holistically.

Sustainability strategies can be validated through simulation before real-world implementation. Datacenter simulators allow modelling the power consumption and cooling requirements of various GPU deployment configurations. They can also incorporate cooling system models to evaluate how different strategies (e.g., immersion vs. direct-to-chip cooling) impact the datacenter's energy usage and water footprint, offering valuable insights for environmentally informed infrastructure planning.

4.2.4 Power and Energy Efficiency. The integration of GPUs in datacenter deployments amplifies the importance of power and energy efficiency, especially when compared to CPU-only systems. This is primarily due to the substantially higher power consumption and thermal output associated with modern GPUs. For instance, contemporary GPUs such as the NVIDIA GA100 and GV100 can consume up to 500W, and even more advanced models like the H100 can reach thermal design power (TDP) levels of 10.2 kW, significantly exceeding that of most CPUs [13, 94]. These elevated power demands challenge the capacity of existing cooling and power distribution infrastructures, especially in high-density deployment scenarios typical for large-scale inference workloads. Moreover, energy inefficiencies translate into increased operational expenses and infrastructure stress, necessitating advanced scheduling and resource management techniques that explicitly consider thermal and power profiles [94].

Additionally, the interplay between GPU power consumption and operational reliability further underscores the criticality of energy-aware design. Studies have shown that higher GPU power consumption correlates with increased temperatures and elevated soft-error rates, particularly in large-scale, long-running scientific computations [66]. Unlike CPUs, GPUs often exhibit greater thermal variability and shorter retention in thermal equilibrium states, leading to frequent transitions between thermal extremes and making them more sensitive to inefficient power management. This not only affects system stability but can also degrade performance due to thermal throttling and error correction overheads. Consequently, effective energy efficiency strategies for GPU deployments must address not just power draw, but also the dynamic thermal behaviour and its impact on performance and reliability [13, 66].

Simulated environments help quantify and mitigate the energy inefficiencies introduced by GPU deployments. By modelling GPU power profiles, workload characteristics, and thermal behaviour, simulations can guide energy-aware scheduling and provisioning strategies. Furthermore, what-if analyses can be conducted to assess the trade-offs between performance and power draw across different hardware and workload configurations.

4.2.5 Data Transfer and Memory Management. Data transfer and memory management are more problematic with GPUs than with CPUs primarily due to the separation between host (CPU) and device (GPU) memory and the overhead incurred during data movement over the PCIe bus. In GPU-accelerated applications, especially in microservice architectures, components often need to exchange intermediate data, which requires frequent and costly transfers between CPU and GPU memory. This overhead becomes particularly significant in synchronous microservice-based applications where downstream components must wait for upstream results before proceeding. The latency introduced by these transfers can degrade the overall application throughput and response time, especially when GPU memory is not efficiently reused or shared [81].

To mitigate these issues, recent research proposes shared memory-based data exchange and kernel execution overlap to reduce PCIe congestion and enhance performance. For example, in microservice-based object detection pipelines, using shared GPU memory instead of UDP-based inter-component communication has demonstrated up to 6× improvements in throughput [81]. Additionally, the default memory management techniques in GPUs often rely on static partitioning or time-sharing, which fails to adapt to varying workloads with dynamic memory demands. In contrast, CPUs benefit from unified memory spaces and OS-level paging mechanisms that are better suited for complex, interdependent applications. The lack of fine-grained, dynamic memory control in GPU environments limits the scalability of multi-component applications, emphasizing the need for enhanced GPU memory management strategies tailored for modern cloud workloads.

Simulation can reveal the bottlenecks caused by data transfer between CPU and GPU memory. This allows system designers to prototype shared memory optimizations, overlapping computation with data movement, or testing new architectural changes—such as faster interconnects or memory hierarchies—before committing to costly hardware changes.

4.2.6 Thermal Instability in GPUs: Operational Constraints and Infrastructure Implications. The thermal design power (TDP) of GPUs significantly exceeds that of CPUs, with modern GPUs reaching 700W and nearing 1000W, compared to CPUs at around 500W per socket [21]. This elevated power density leads to more frequent and extreme thermal states, creating substantial challenges for datacenter infrastructure. GPUs exhibit faster and more frequent transitions between hot and normal thermal states than CPUs, resulting in highly dynamic thermal behaviour that is difficult to manage using traditional cooling methods [66]. These conditions increase cooling demands and heighten the risk of thermally induced soft errors and reduced reliability.

To address these challenges, datacenters are adopting advanced cooling technologies that go beyond conventional air cooling. These include direct liquid cooling (DLC) with water or propylene glycol, single-phase and two-phase immersion cooling, and microfluidic systems, all offering superior thermal performance and sustainability benefits [21, 75]. For instance, cold plate-based single-phase liquid cooling has demonstrated thermal resistances as low as $0.015°C/W$ [75]. However, immersion and two-phase systems also require considerations such as contamination risks and hardware compatibility. As a result, cooling strategies must not only be efficient but also adaptable to the rapidly shifting thermal demands of high-power GPU deployments.

Thermal simulation is critical for evaluating cooling strategies under GPU-intensive workloads. Simulators that incorporate thermal models can be used to predict heat distribution, identify hotspots, and test the effectiveness of cooling technologies like immersion cooling or microfluidic systems. This enables infrastructure engineers to make data-informed decisions without risking service interruption or hardware degradation.

4.2.7 *Fault Tolerance Challenges in CPU-GPU Hybrid Systems.*

GPU failures in heterogeneous systems can significantly disrupt workloads, especially in domains like deep learning and high performance simulations. A failed GPU may crash inference services or halt distributed training, causing major performance losses and service level violations due to the GPU's central role in parallel processing. CPUs typically cannot compensate for such failures, as they lack the throughput needed for GPU-accelerated tasks. While some dynamic load balancing strategies can redistribute tasks, the performance penalty remains substantial. In contrast, CPU failures are often less severe in such systems, as other CPUs may take over control and orchestration duties more easily [56, 57, 98].

Host CPU failures, however, also have a critical impact on GPU workloads. GPUs rely on the CPU for coordination, data transfers, and memory management, including page fault servicing in systems using Unified Virtual Memory (UVM). If the host fails, GPU execution may stall entirely, regardless of the GPU's operational status. Unlike CPU-only systems, which can often recover from single-node failures using redundancy, heterogeneous CPU-GPU systems are more fragile because of this interdependence. Even advanced interconnects like NVLink cannot offset the loss of host-side orchestration, leading to a complete halt of GPU workloads [14, 91].

Simulation enables controlled exploration of failure scenarios in heterogeneous systems. Fault injection models can be used to assess the resilience of GPU-accelerated workloads to both GPU and host CPU failures. These simulations help identify failure propagation patterns, test redundancy mechanisms, and evaluate load balancing approaches that might be impractical to test in live systems due to risk or cost.

5 DESIGN OVERVIEW

In this and the next section, the contributions of this study are presented: the results of the literature review. This section focuses on answering research question RQ2, describing the different implementations. First, an overview is provided of the selected simulators, including their intended purposes and a representative subset of their features. This is followed by a comparative analysis of their respective simulation approaches and the types of input data they require. Finally, the level of granularity at which GPUs are represented and integrated within the simulations is discussed. A summary of these findings is provided in Table 2.

5.1 General overview of the simulators

Despite the relatively small number of datacenter simulators identified that support GPU simulation, their respective purposes and feature sets vary significantly.

The authors of **Cloudy** [86] recognized a gap in the ease of integrating libraries for integer linear programming and machine learning, such as TensorFlow. Given the widespread use of Python for such applications [25], it was selected as the language for implementation. Rather than offering detailed implementations for all simulation aspects, Cloudy provides abstract base classes designed to be extended by the user. Nevertheless, example and default implementations are available in the public repository[6].

Cloudy's simulation framework combines cloud component models with system control classes that govern their behaviour. The *App*, *Container*, and *Deployment* classes represent applications and their resource demands, while the *Controller* enforces operational policies within clusters. Virtual infrastructure is abstracted through the *Vm*, *Pm*, and *DataCenter* classes, representing virtual machines, physical machines, and the overall cloud environment, respectively. User behaviour and service demands are modelled using the *User*, *Action*, and *Request* classes. System operation and resource management are implemented via the *Os*, *Vmm*, *Vmp*, and *ControlPlane* classes, which coordinate application execution, VM allocation, placement strategies, and policy enforcement. These components collectively enable dynamic and adaptable cloud management.

DCSim [35], developed by Hu et al., is designed to study container-related operations such as request handling, scheduling, execution, communication, migration, and completion on heterogeneous compute infrastructures. It employs SimPy [6] for discrete-event simulation and Mininet [9] for network emulation.

An architectural overview is shown in Figure 7. DCSim consists of five core modules: *datacenter*, *network*, *container scheduling*, *discrete event driver*, and *data collection and analysis*. The datacenter module manages hosts and workloads. A workload is defined as a *job* with a specified completion time, comprising at least one *task*. Tasks are characterized by their resource requirements and may run across multiple *container* instances, which may be CPU-, memory-, or GPU-intensive. Runtime is determined by both instruction execution and communication time.

The network module utilizes Mininet objects defined by user scripts, incorporating *host*, *switch*, and *controller* components to emulate virtual network topologies and software-defined networking. The container scheduling module handles placement and execution strategies, and includes five predefined scheduling algorithms. The event driver, based on SimPy, coordinates simulation time and flow, as further detailed in Section 5.2. Lastly, the data collection and analysis module logs metrics regarding hosts, containers, network usage, and events.

[6]https://github.com/ahmad-siavashi/cloudy

Simulator Name	Use Case	Simulation Approach	Workload Input Type	Level of Granularity	Reference	Year
Cloudy	Python, Integer Linear Programming & AI libraries	Event-driven	Model-based	Number of Instructions: FLOPS	[86]	2023
DCSim	Container-Scheduling & Simulation	Event-driven	Trace-based	Number of Instructions: FLOPS	[35]	2024
GPUCloudSim	GPU-support in Cloud Simulator	Event-driven	Model-based	Cores of a Video card, Co-running application interference, PCIe Bandwidth, GPU-virtualization-impact	[85]	2019
Het-Sim	High-level heterogeneous datacenter simulation	Event-driven	Unclear	Number of Instructions	[41]	2018
MERPSYS	HPC-/Cluster Simulation	Analytical	Application-Model-based	Number of Instructions per time unit	[22, 73]	2017, 2016
PyPassT	Accurate, Performant, Scalable, performance prediction Framework for parallel applications	Analytical	Application-based	Number of Instructions per Cycle & Memory Access	[68]	2018
WCSim	Special Case of Workload: Workflow of Bags of Tasks	Event-driven	Trace-based	Boolean	[26]	2023

Table 2: Summary of the results

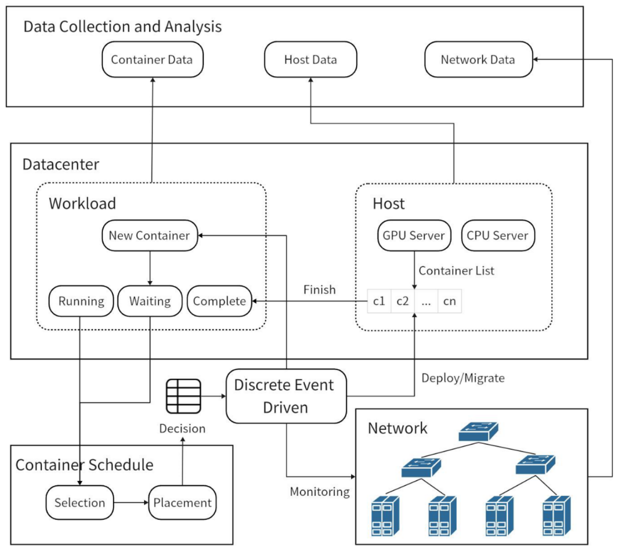

Figure 7: DCSim's architecture as depicted in [35]

GPUCloudSim [85] extends the CloudSim [17] framework to incorporate GPU support in cloud datacenter simulation. The authors identified a lack of GPU-aware simulators and proposed modular extensions layered on top of the original Java-based CloudSim implementation, without requiring changes to the base code. Figure 8 visualizes the multi-layered architecture.

The *GPUCloudSim* layer provides classes for modelling GPU-enabled infrastructures. The *interference* layer enables simulation of application co-location on GPUs. The *performance* layer captures the performance impact of GPU virtualization techniques such as API remoting, full and para-virtualization, and hardware-assisted virtualization. Finally, the *power-aware* layer models GPU energy consumption. The layers are depicted in figured 8.

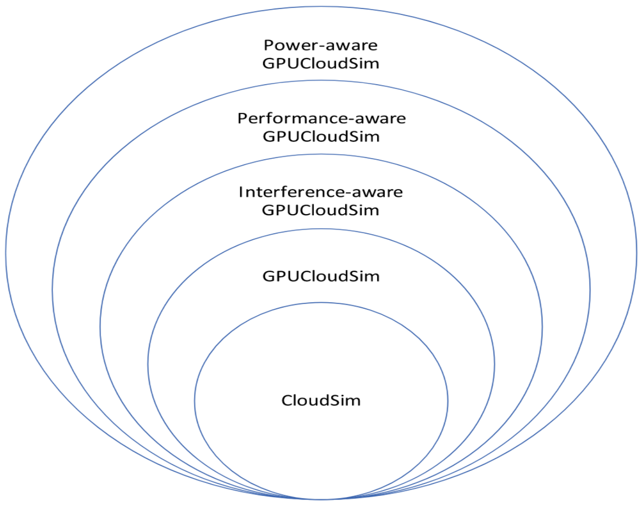

Figure 8: "GPUCloudSim Architecture. Layers are added gradually so that a GPU-enabled cloud computing data center can be simulated based on research requirements. Upper layers add new features to lower layers; hence, lower layers can be used independently from upper layers" [85]

The main terminology is defined by CloudSim's architecture, which is visualized in figure 9. It comprises several fundamental components: *Data Centers*, overseeing multiple *Hosts* (physical machines) which allocate *Virtual Machines* (VMs) to execute computational tasks. VMs host *Cloudlets*, representing application workloads, and are managed by *VmSchedulers* using either space-shared or time-shared allocation policies. The *DatacenterBroker* (Cloud Broker) facilitates VM provisioning and workload distribution across cloud providers, while the *CloudCoordinator* manages inter-cloud federation and resource sharing mechanisms. Additionally, CloudSim integrates a *NetworkTopology* model to simulate communication latencies and a *PowerModel* to assess energy consumption. [17]

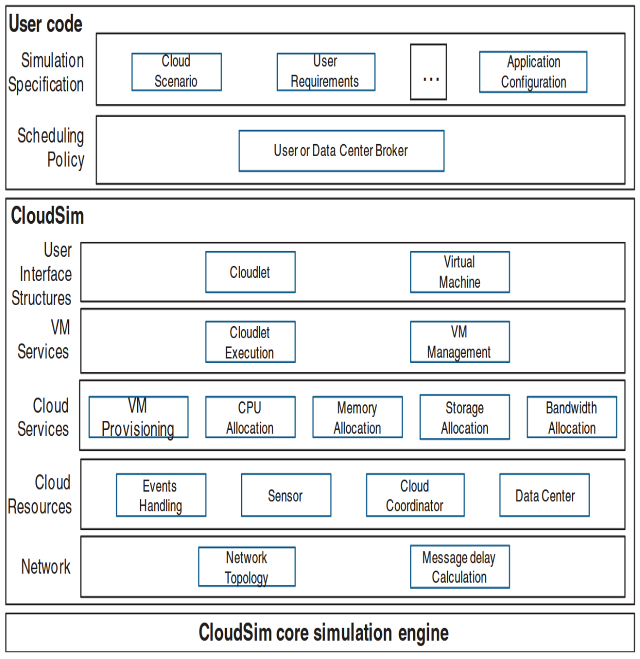

Figure 9: CloudSim Architecture Overview [17]

In contrast to previously named simulators, **Het-Sim** [41] focuses not on simulating the cloud environment, but a broader datacenter environment. This simulator enables users to model a diversity of user requests on hosts with heterogeneous resources on a high-level. The (theoretical) model on which Het-Sim is based, starts on the highest-level with a *datacenter*, containing numbered *rack cabinets* with *shelves* which are numbered as well. Each shelve contains either a *storage unit* or a *processing unit*, each unit is identified by the tuple of rack and shelve number. A storage unit is typically a hard disk as permanent data storage. A processing unit has a type, which is either CPU, GPUs, application-specific FPGA accelerator, or application specific integrated circuit (ASIC), and is generally just referred to as *accelerators*. It manifested that the majority of processing units are not CPUs, but other types of accelerators. Communication within the datacenter can occur either, within the same shelve *(direct)*, the same rack *(intra-rack)* or between racks *(inter-rack)*. The *bandwidth* is assumed to decrease from direct communication over intra-rack communication and ending with inter-rack communication. The computational element of work in Het-Sim is a task. Each task has a type, describing the work that is done. The type is any of the following: integer computing *(INT)*, floating-point computation which is inefficient on a GPU *(GPU-averse)*, floating-point computation which is efficient on a GPU *(GPU-Affine)*, memory-intensive computation *(MEM)*, I/O-intensive computation *(I/O)*, arbitrary computation suitable for an FPGA *(ARB)* or none of it and therefore unspecified *(UNSPEC)*. A set of typed tasks is a *job*, which can be scaled across multiple processing units and is requested by a client. A *workload* combines the job attributes with the information of the submitting user, and the location of the stored *input* data. A schematic outline of Het-Sim is given in figure 10.

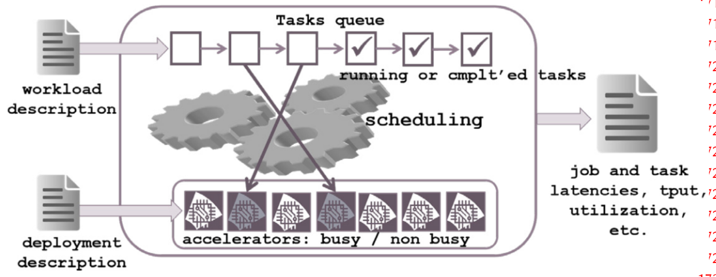
Tasks queue
workload description
running or cmplt'ed tasks
scheduling
job and task latencies, tput, utilization, etc.
deployment description
accelerators: busy / non busy

Figure 10: "Schematic outline of Het-Sim" [41]

MERPSYS by Czarnul et al. [22] is the next simulator not focusing on cloud simulation, but in this case, on high-performance computing simulation of parallel applications on large-scale systems, including systems of Volunteers. Parallel application paradigms that are supported include *Master-Worker*, *geometric parallelism* aka Single Program Multiple Data (*SPMD*), and divide-and-conquer. The main element in MERPSYS is called a *Component*, and is the base of all entities. Entities that can be connected, either by network or system bus, have the base class *Connectable*. Connectables are combined into *machines* or *group*s, which then form a *cluster*. A group is connected through LAN or WAN. Components can also either be *Computational* or *Storage*. Computationals can either be *CPU*s, *GPU*s or *Volunteer* systems. Storage is either *Memory*, *Disc*, or a *DiscArray*. Workloads in MERPSYS are generally application models written in a Java-like language, their details are explained in subsection 5.2. An overview of the class-diagram selection, can be seen in figure 11. At least one aggregator with at least one component and a network, and a specific computational model is required for simulation. To simulate the application, it is suggested by the authors to run a subset on a real cluster to calibrate the computation model. As a simulation result, the execution time, the consumed energy and the fault probability are returned.

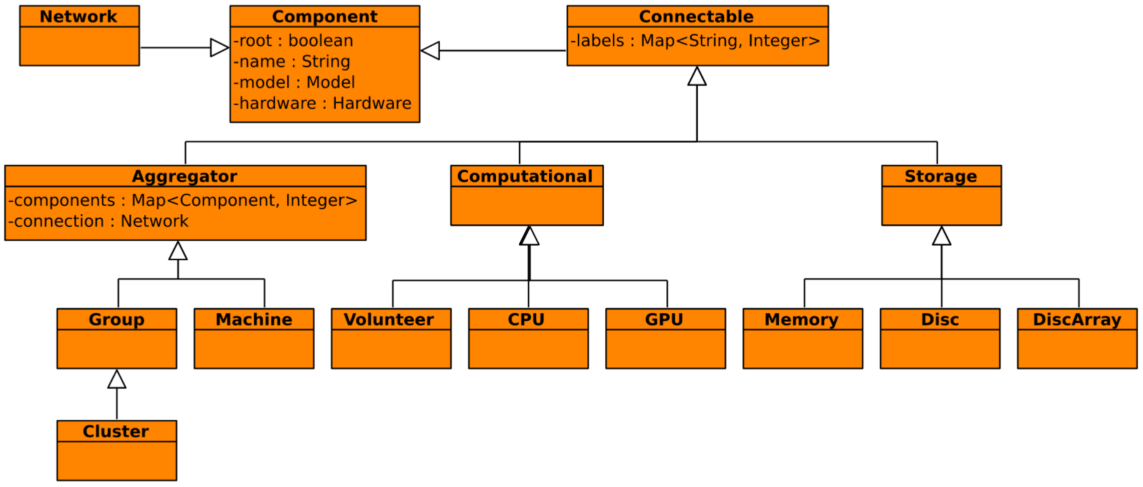

Figure 11: Selection of classes in MERPSYS [22]

PyPassT is another simulator not focusing directly on cloud scenarios, but on simulation of High-performance-computing. The foundation of the analytical simulation, is OpenACC annotated C-code, which is then transformed into an *PPT application model*. To achieve this, Obaida et al. have extended and then combined several libraries, such as COMPASS[3] and PPT[8]. With PyPassT it is possible to simulate different *hardware configurations*, *job scheduling* techniques, *networking and MPI* in the form of network topology and interference by other applications and rate limiting as*background traffic*. The actual simulation is then handled by PPT. An overview of the prediction framework can be found in figure 12, and the description of model generation can be found in 5.2.

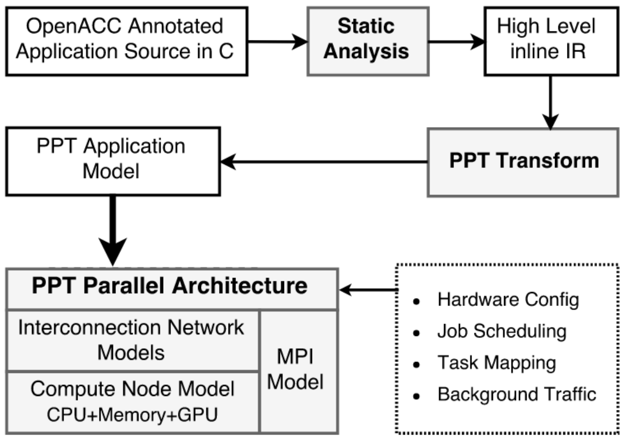

Figure 12: "An overview of PyPassT prediction framework." [68]

The main focus of **WCSim** [26] is to model a specific type of workload called directed acyclic graph (DAG) of bags of tasks (*DoB*). This new workload type combines the attributes of workflow tasks, with dependencies between them and bags of tasks. The application was developed under the assumptions that DoBs are launched on *VMs* which are owned by *Users*, which are virtualized on physical infrastructure. Physical infrastructure is composed of *datacenters* being a set of *processing servers*. Processing servers are connected by a 2-layer network. The first layer being the connection within a datacenter and the second layer the connection between multiple datacenters. Each connection is bidirectional. Further, a processing server consists of CPU cores, RAM, and storage and can host one or more GPUs and has an ID. Processing power is defined in "million instructions per second"(*MIPS*), memory and storage are measured in GB. Additionally, are costs associated with a PS for each unit used by the respective resource. The unitization of a processing server is divided into 8 brackets, increasing in steps of 25% ranging from 0 to 200. For each bracket, the percent of simulation time spent in that utilization bracket is given as output. The bracket (0, 25%] is considered idle time. As aforementioned, are users creating computational demand through the submission of *applications*. The assigned group and priority influence the scheduling decisions for a user. Groups determine where tasks may be executed, either one specific VM in one datacenter, marked as the home datacenter, may migrate their tasks across different datacenters, or may use cloud bursting. Cloud bursting describes the process of offloading overloading applications to a public cloud. To use the infrastructure, the user needs to spend credits stored in their wallet. An excerpt of the UML class diagram can be seen in figure 13.

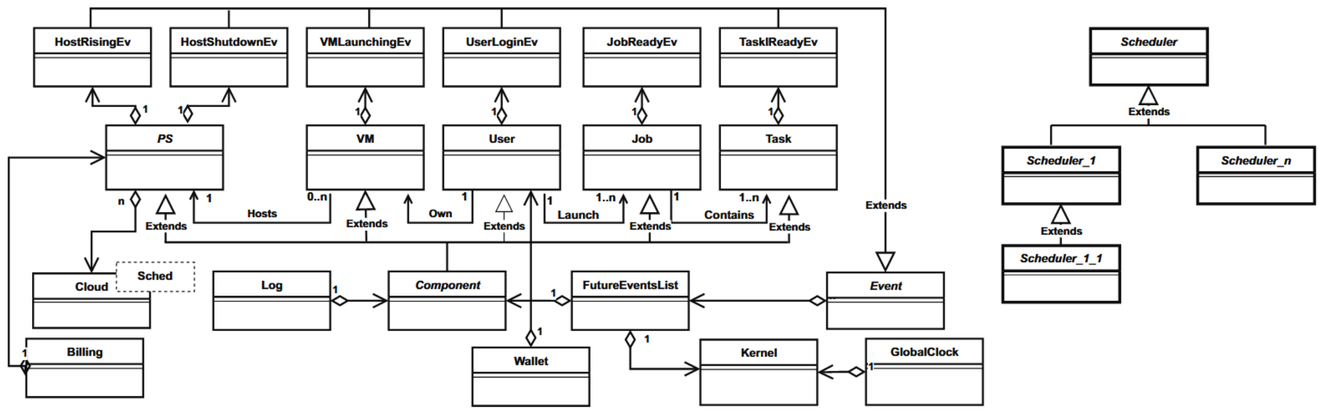

Figure 13: An excerpt of the UML class diagram of WCSim [26]

5.2 Simulator's control model & Input format

Similar to Gallia in Asterix & Obelix seems, the world of datacenter simulator ruled by an event-driven simulation approach, with some strongholds following an analytical approach. More diversity can be seen other the hand in the matter of how simulation parameters and especially workload is provided, including various file-formats.

Cloudy utilizes a queue system as the core component of the simulation. In publish-subscribe-pattern, messages are exchanged between various components involved in the simulation. The relevance of an event message for a simulation component is determined by the message's topic. The topics are concerning requests, apps, containers, deployments among other things, and can be extended if required. The simulation input is given purely as python code, defining physical infrastructure, virtual machines and application and how and when they should be scheduled, finished by the call to simulate the given scenario and report the results. This model-like input, allows the definition of used threads per application and their duration in cycles and giving that application a name, such as NGINX. VMs have resource requests defined, in the form of CPUs core, RAM in MB and a GPU compute engines and memory blocks, and OS. Physical machines are defined with the same parameters, with the exception that the OS is replaced with a virtual machine monitor. The Virtual machine placement policy is attached to the datacenter creation. The example of an input, given by the authors, can be seen in Figure 14.

As indicated in, belongs**DCSim** also to the family of event-driven datacenter simulators. It consists of 4 key components inherited from SimPy. The *environment*, concerned with scheduling processes, events, the progression of simulation, as well as the advancement and completion of the simulation. A *process* implements a simulation model, uses the standard python generator function to yield events, which blocks the yielding process, until completion of said event. An *event* is described as an asynchronous occurrence used to control the operation, it can only be triggered once. The *store* component models producer-consumer-relationships within the system, helping to maintain dependency relationships. To start a simulation, the user needs to provide the datacenter configuration, including performance specification of the hosts, in a CSV format. Additionally, in an INI format are the workload task flows and simulation parameters provided. The network is defined in custom python scripts.

The next event-driven datacenter simulator is **GPUCloudSim**, due to the use CloudSim. GPUCloudSim just extends the abstract class SimEntity, to create additional event-aware classes such as GPUDataCenter and GPUDataCenterBroker. To model GPU-specific events, the GPUCloudSimTags is implemented, but not further explained. GPUCloudSim takes the simulation definition in the form of code. Neither the CloudSim nor the GPUCloudSim paper provide any detail about the simulation input, a code inspection of the GPUCloudSim repository[7] revealed this detail. In the examples, a datacenter with hardware specification, including the construction of videocards in their details. Further, the respective VMs and Cloudlets are created via looping. Finally, the Simulation is started, stopped and the cloudlets retrieved for further analysis.

[7]https://git.ceit.aut.ac.ir/lpds/gpucloudsim

```
app = App(NAME='Nginx', LENGTH=(1, 1, 1))
vm = Vm(NAME='WebServer', CPU=1, RAM=1024, GPU=(2, 2), OS=OsTimeShared)
vm.OS.schedule([app])
request = Request(ARRIVAL=0, VM=vm)
user = User(NAME='Portal', REQUESTS=[request])
pm = Pm(NAME='HPE', CPU=(2, 2), RAM=2048, GPU=((7, 8),), VMM=VmmSpaceShared)
datacenter = DataCenter(NAME='Tehran', HOSTS=[pm], VMP=VmpFirstFit)
Simulation(NAME='Example', USER=user, DATACENTER=datacenter).run().report()
```

Figure 14: Example simulation input for Cloudy given by [86]

In figure 15, can the theoretical high-level view of **Het-Sim**'s core component be seen, the *time machine*. While the simulation is running, events are continuously pushed and popped onto the queue. These events include the arrival of tasks, the completion of computation and optionally the current utilisation of the simulated system. It is mentioned that the simulation input consists of files describing the workload and the deployment. No further information is given on how the input is exactly structured, only that in the future realistic workload traces should be supported.

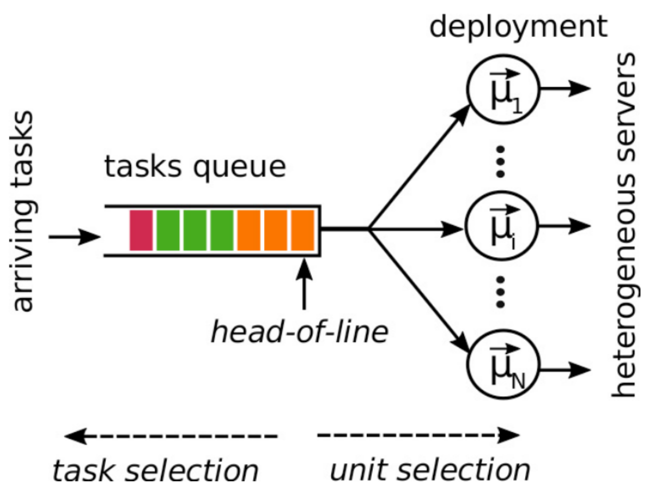

Figure 15: Theoretical model of Het-Sim's time machine [41]

The simulation model of **MERPSYS** appears to be extendable via coefficients in the simulation definition. Further details could not be found in [22], but in [73]. Generally does, the MERPSYS appears to be following an analytical approach, rather than an event-driven approach. The functions compromising the *computational model* utilize the number of threads, number of operations and parameters of the hardware involved in the simulation. MERPSYS is used via a Web application interface, where simulation infrastructure is defined via a Graphical interface. The workload is defined via Java-like pseudocode, defining the *application model*. The code consists of *computational blocks* and *communication blocks*. Besides, not mentioned parameters, are the parameters for the data size, the computation type, a complexity function based on the input size, and number of threads or processes used. The communication blocks can use the methods for point to point, one to one and scatter communication, to model communication being similar to MPI, Hadoop. Different software stacks - such as programming languages - can be considered in the simulation, by naming them in the respective blocks.

Just as MERPSYS, does **PyPassT** not follow an event-driven, but rather an analytical approach. The computation time is calculated by addition of *instruction execution time* and main *memory access timing* at one core. The memory access timing is calculated via a probabilistic state graph, forecast the chances of a code block to be executed. Communication is included by letting the sending core sleep for the predicted computation time at the remote core. The given C source code with openAcc annotations is compiled into a *high-level inline intermediate representation*, by static analysis. The goal is to identify computation blocks, communication blocks, memory access and GPU operations and model these blocks in ASPEN[93] code. The ASPEN code consists of blocks which are defined by flops, loads, stores, other memory operations, and intra- and internode communication. The abstracted communication is represented as MPI operations. The abstract code representation is then transformed into a PPT model via the use of COMPASS++, an extension of COMPASS to model computation and communication. How other input, such as infrastructure, is required to be presented is not further determined.

WCSim does not only close the list of simulators in this literature review, but also the long list of event-driven datacenter simulators. The produced *events* can either be *exogenous* or *endogenous*. Exogenous events are provided via the simulation input at launch time and could be the booting of a processing server or job submission by a user. The input is provided in 3 different files, leading to a trace-based approach. All files are CSV-formatted, with different extensions. The .dob extensions describes the submitted workload, with an arrival time, the task owner, required resources, number of tasks in this job, and the number of tasks that need to be completed before this job can be started. The file with extension .pas describes the Users during the simulation, their assigned home datacenter, their group, the time the user will log onto the system. When a file has the extension .inf, then it describes the infrastructure during simulation with WCSim. The description contains an ID, the starting time, a family (of which the meaning is not further elaborated), and the communication cost to each following instance.

5.3 Level of Granularity

As the different datacenter simulators vary in their feature sets and intentions, so does the level of granularity of how GPUs are considered in terms of hardware simulation. Further, do the simulators differ in how they perceive GPUs, ranging from "yet another CPU" to "a nearly independent system". Not only the information provided by the article has been used for this subsection, but also some code inspection, to clarify details. A high-level overview of the findings can be seen in figure 16.

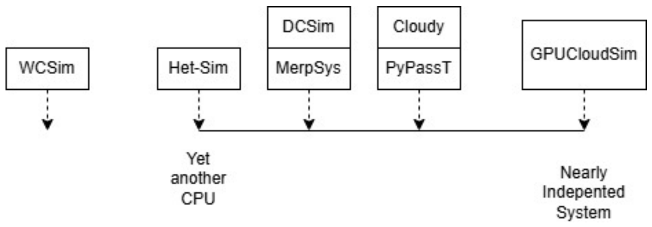

Figure 16: Approximate ranking of granularity levels simulating GPUs in datacenter

Similar to the majority of the findings, does cloudy tend to be closer to the "yet another CPU" end of the range, utilizing the number of FLOPs of Nvidia A100, the number of compute engines and the number of memory blocks to define a GPU. A code inspection revealed[8], that the GPU instructions cycles are not considered, to determine if a task is finished, This could lead to an inaccuracy during simulation, if some applications require less CPU cycles than GPU cycles to complete.

In the summary of related work, it is stated that **DCSim** uses FLOPs as the metric for GPU specification. In the evaluation section of the same article, the GPU is defined by the number of cores and a single-digit inter representing the GPU speed. A public code repository could not be found, but two other simulators with the same name, one datacenter simulator[9] and one high-energy physics workload simulator[10]. Therefore, DCSim is placed with uncertainty on coarse end of simulation granularity, leading to "yet another CPU".

GPUCloudSim defines one end of the given range of classification. GPUs in GPUCloudSim form "a nearly independent system". On the top abstraction level is a *videocard* placed, which is the component attached to a physical host, via high-speed *PCIe*. The PCIe connection interface is shared by all videocards which are attached to the same host. In a video card are one or more *GPU*s placed. Multiple *Streaming multiprocessors*, which are "Single Instructions Multiple Data" cores, form a GPU. Virtual machines have a virtual GPU (*vGPU*) attached, they are implemented via CloudSims *Pe* class. How the processing power is shared among resident vGPUs is defined via the implementation of the abstract class *VgpuScheduler*. Per host exists a *VideoCardAllocationPolicy* determining the provisioning policy of video cards to vGPUs, this is an abstract class. The provisioning policy of PCIe bandwidth needs to be implemented via the abstract class *VideoCardBwProvisioner*. The mapping of vGPUs onto physical GPUs is determined by implementing a policy in the abstract class *PgpuSelectionPolicy*. The virtualization overhead can be modelled via the *PerformanceModel* interface, an interface of a scheduler for this layer is provided via *PerformanceScheduler*. Power-awareness of videocard is created via the *VideoCardPowerModel* interface. Allocation of processing power in the form of Pe is defined via the abstract class *PeProvisioner*. GDDR Ram provisioning policies are represented by the *GpuGddramProvisioner* abstract class, and how the bandwidth to that GPU memory is allocated can be provided via *GPUBwProvisioner*. How vGPUs execute their tasks is defined with *GpuTaskScheduler* abstract class. Interference of co-running GPU tasks, can be modelled by implementing the *Interference model* interface. From this high-level description, one can get an intuition of how many aspects of a GPU are considered in GPUCloudSim, and how many more ways own models can be incorporated. Some default implementations for the respective models and policies have been provided, their details are omitted in this systematic literature review.

The perspective on GPUs in **Het-Sim** is on the opposite end of the spectrum, than GPUCloudSim, and considers GPUs as "yet another CPU" or as Kanellou et al. call it an accelerator. [41] So are GPUs defined via their amount of FLOPs, which should be significantly higher than of CPUs. The processing speed is the main factor of determining the completion of a task on a processing unit and is calculated by the number of operations, that a task requires, divided by the *affinity mapping*. The affinity mapping is the number of operations per time unit an accelerator can perform.

Similar to the majority of simulators, are GPUs not simulated in great detail in **MERPSYS**. Given by the general and overview-like description in [22] and the formulas in [73], is that GPUs are also seen "yet another CPU". In the given computation model are the number of instructions multiplied by the time needed to execute a single instruction. The time per instruction, varies depending on if less threads than cores are used, with the minimal time. A time overhead is added if hyperthreading is used, and a linear increase in time is assumed for each instruction if more threads than cores are needed. The level of detail in the simulation may vary with the chosen computational model; however, it is not clear how such a computational model can be defined. Or if it is pre-defined as the hardware.

After reading the article about **PyPassT** the reader was left with the impression that the usage of GPUs in PyPassT falls close to "yet another CPU". This impression was created due to displaying the GPU as an example of an accelerator and not explicitly (enough) mentioning the difference in treatment of CPUs and GPUs. A code inspection on the PPT repository[11] revealed that the simulation is actually closer to the end of "a nearly independent system", by

[8]https://github.com/ahmad-siavashi/cloudy/blob/212b5addbd694edf08b1d88747f97e2b4253d9f0/src/policy/os.py#L15
[9]https://github.com/digs-uwo/dcsim/tree/master
[10]https://github.com/HEPCompSim/DCSim
[11]https://github.com/lanl/PPT/blob/9fd25471a9623f301ab0e15f9459f647cda7d32a/code/hardware/processors_new.py#L2423

considering GPU cycles, memory access in the advancement of the simulation. This allows a (cycle) accurate simulation of the given parallel application.

In the article of **WCSim** itself it is mentioned that GPUs are supported as part of the infrastructure, however an analysis of the code[12] revealed that the GPUs of the simulated infrastructure have the data type boolean[13]. Meaning that the GPUs can be signalled to be existing, but have no input on the acceleration power, nor on the advancement of simulation workload. Therefore, is GPU support in WCSim considered to be not implemented.

6 COVERED AND REMAINING CHALLENGES

This section of the literature study presents the findings related to Research Question 3, which addresses unresolved challenges in GPU deployments within datacenter simulations. As outlined in Section 3, a secondary snowballing search was conducted to identify articles based on the original datacenter simulator implementations. This section builds on the challenges identified and discussed in Section 4.2 in response to Research Question 1. It examines which of these challenges have already been addressed by the datacenter simulators identified through Research Question 2 (Section 5). These addressed challenges are discussed first, followed by the remaining open problems, each accompanied by a brief motivation.

6.1 Problems Addressed Through Experiments

Section 5 offered an overview of the supported features, intended use cases, and simulation approaches of the respective datacenter simulators. This subsection outlines the experiments and use cases previously explored, hereafter collectively referred to as experiments for simplicity. All articles introducing a datacenter simulator include at least one experiment, which can generally be classified into one of three categories: "proof of accuracy", "comparison to other simulators", or "use case demonstration".

Both MERPSYS [22] and PyPassT [68] fall into the "proof of accuracy" category. In their respective evaluations, the authors compare simulation outcomes against results from executing the same applications on actual hardware. In the case of MERPSYS, three applications—each representing a distinct parallel programming paradigm—are developed: Master-Worker[14], Geometric Single Program Multiple Data, and Divide-and-Conquer. The experiment does not involve GPUs as a hardware component. Obaida et al. evaluate PyPassT [68] in three steps: firstly, by assessing the simulator's accuracy in predicting resource utilisation (FLOPS, loads, stores) and runtime; secondly, by verifying the simulator's runtime prediction in scenarios involving GPU usage; and thirdly, by validating the accuracy of predicted byte transfers for different MPI commands.

Both Cloudy [86] and WCSim [26] present experiments comparing their simulators to other datacenter simulators. In both cases, CloudSim is used as a baseline for comparison—CloudSim Plus for WCSim, and the original CloudSim for Cloudy. WCSim additionally includes SimGrid in the comparative analysis. Dos Santos et al. compare the simulation and execution times of WCSim with those of other simulators across 100, 1000, and 10000 tasks, executed on two different hardware configurations. Cloudy evaluates execution time, CPU utilisation, and memory usage of the host system during the simulation of an Alibaba Cluster trace. This simulation excludes GPUs due to the absence of GPU support in CloudSim. It is possible that GPUCloudSim was not selected for comparison because both Cloudy and GPUCloudSim were authored by Siavashi et al.

The published introductions of DCSim [35], GPUCloudSim [85], Het-Sim [41], and WCSim [26] all include demonstrations of use cases relevant to their respective simulators. Each article compares various schedulers, including at least one newly adapted variant of a pre-existing scheduler. These schedulers pursue different optimisation objectives, tailored to the intended use case. For example, DCSim aims to co-locate dependent containers; GPUCloudSim examines the interaction between GPU and host allocation; Het-Sim considers task preferences to varying degrees, with one scheduler attempting to minimise the distance between computation and data; and WCSim analyses the cost implications of offloading tasks to the cloud. Additionally, GPUCloudSim evaluates its scalability in terms of application and infrastructure size. Notably, the WCSim experiments do not incorporate GPUs.

6.2 Problems Addressed Through Follow-Up Work

Some simulators discussed in this study have been extended through subsequent publications. To account for these developments and avoid mistakenly identifying previously addressed issues as open challenges, a follow-up literature search was conducted. This section discusses such follow-up work where applicable. No follow-up work was identified for DCSim, PyPassT, or WCSim. The citations of Het-Sim [32, 54] do not include practical usage or extensions of the simulator. Citations of MERPSYS often express an intent to transfer prior insights into the simulator, as seen in [48, 49, 72, 74, 78].

For Cloudy, a follow-up study [88] proposes an Integer Linear Programming-based scheduler aimed at reducing GPU fragmentation through migration, leveraging NVIDIA's Multi-Instance GPU (MIG) technology.

Follow-up studies using GPUCloudSim as a platform mostly focus on the implementation of various scheduling algorithms. Smith et al. [90] develop a thermal-aware scheduling algorithm to reduce datacenter energy consumption, combining GPUCloudSim with CloudSim Plus. Siavashi et al. introduce GPU API remoting and a scheduler named gVMP [87], also examining network traffic overheads. Zhao et al. [103] propose a scheduling firework algorithm optimising VM placement with respect to energy consumption. Kulkarni et al. [50] develop a scheduler that considers GPU memory during task placement, optimising both make span and energy use. Gao et al. [27] implement a grey wolf-based scheduling algorithm targeting electricity cost reduction in cloud gaming datacenters.

[12]There was no public repository given in the article, a search revealed nearly identical repositories from different authors of this study. The newer one has been chosen.
[13]https://github.com/GersonCavalheiro/WCSim/blob/d9e32eb36db05fc65e36f31fa1b00ec97f1613b3/host.cpp#L19
[14]Name is adjusted

6.3 Remaining Problems

Building on the discussion in Section 4.2 addressing Research Question 1, this subsection outlines and motivates the open challenges identified in the reviewed literature.

6.3.1 Interplay Between Scheduling and GPU Allocation. While some studies have explored the relationship between GPU allocation at the host level and workload scheduling at the datacenter level, these remain limited. The use-case demonstrations in GPU-CloudSim [85] include different datacenter-level scheduling and GPU allocation policies, but the applied strategies—Breadth-First Search and Depth-First Search—are relatively rudimentary. The authors conclude that GPU selection policies had minimal impact due to GPU overprovisioning relative to workload demands. Further developments in this area, such as those by Siavashi et al. [88], aim to reduce GPU fragmentation, yet additional combinations of datacenter- and host-level allocation policies are still unexplored [38]. Overall, GPU allocation remains underrepresented in datacenter simulators. A deeper understanding of its effects can lead to improvements in performance, cost-efficiency, and energy usage [53, 100].

6.3.2 Carbon Footprint. Although datacenters contribute an estimated 1–2% of global carbon emissions [15], none of the reviewed simulators consider carbon footprint in their models. Incorporating both embodied and operational carbon costs of ICT infrastructure could inform decisions on whether to retain existing hardware deployments or upgrade to newer, potentially more sustainable configurations.

6.3.3 Thermal Modelling. Thermal effects—key to sustainability and energy consumption—are largely absent from current simulators. Only one study [90] considers GPU thermal output when designing a scheduler. None of the simulators reviewed can perform thermal modelling natively; the aforementioned study required the integration of multiple CloudSim extensions. Future research should encompass thermal effects alongside other environmental concerns such as water usage to present a more comprehensive and realistic simulation environment [29].

6.3.4 Failure Modelling. The reviewed simulators lack experimental evaluations involving failure scenarios—be it GPU, host, or component-level failures—despite growing research interest in failure characteristics of GPU-intensive workloads, such as AI [34, 37]. Integrating failure modelling capabilities would enable more robust investigations under varying reliability conditions.

6.3.5 Integration of Challenges. While each of the above challenges warrants investigation in its own right, datacenters are complex systems composed of interacting components. A holistic approach that simultaneously addresses multiple challenges could yield insights that isolated investigations cannot [89, 92]. Although it may be technically feasible to achieve such integration by combining existing CloudSim extensions, doing so often requires in-depth programming knowledge and may involve updating unmaintained codebases. A simulator that natively supports a subset of these challenges would enhance usability and accessibility for researchers [82].

7 RELATED WORK

Several surveys and systematic literature reviews have examined the modelling of computing systems and datacenter simulation. Most of these studies analyse a subset of simulators and offer guidance primarily for end-users [10, 16, 60, 70, 80, 84], often supplemented by popularity rankings based on citations in academic literature [58]. Other reviews focus on specific aspects of simulation, such as energy and power consumption [24, 40, 59].

A comprehensive overview of energy modelling in datacenter components is provided by Dayarathna et al. [24]. Their survey introduces energy models for a wide range of components, from low-level hardware such as single-core CPUs and various storage types, to higher-level abstractions including operating systems, virtualization technologies, and data-intensive applications. While the coverage is extensive, the focus is solely on energy consumption rather than the modelling of the components themselves, which is the focus of the present study.

Makaratzis et al. [59] examine how five different simulators represent hardware components during workload execution. However, their work does not provide a systematic characterization of the identified models. Moreover, the study is limited to CPUs, storage, networking, and energy aspects, with no discussion of GPU modelling.

The survey by Bambrik [16] offers a broad analysis of the features found in various cloud simulators. In addition to discussing architectural characteristics and the motivations behind simulator development, it presents varying levels of implementation detail, depending on the relevance to each simulator. Although informative, the survey does not classify or compare implementation strategies, and GPU modelling is not addressed, as it falls outside the scope of their investigation.

Ismail's work [40] extends the findings of Makaratzis et al. [59] by analysing which IT (e.g., CPU, memory, servers) and non-IT components (e.g., cooling, lighting, heating) are incorporated into energy models within datacenter simulators. However, further functional or architectural modelling aspects of these simulators are not discussed.

Mansouri et al. [60] conduct a broad evaluation of 33 datacenter simulators to guide researchers and practitioners in selecting appropriate tools. Their analysis considers simulator features, architectural design, and supported models. Nonetheless, architectural discussions are largely limited to high-level structural descriptions, typically presented through figures or brief textual summaries. The study serves primarily as a user's guide, rather than a developer-oriented resource.

Sanjalawe et al. [80] present a similar guide with a narrower focus on fewer simulators. While this work provides a more detailed examination compared to [60], it still omits modelling details of individual components. The emphasis remains on performance and usability from an end-user perspective, rather than on simulator development.

Alaei et al. [12] offer a focused survey on heterogeneous CPU-GPU architectures and associated simulators. Their work begins with an introduction to GPU functionality and proceeds to identify key challenges in heterogeneous systems, such as shared resource management, task scheduling, and energy efficiency. Notably, the survey reviews five cycle-accurate simulators and evaluates them against these challenges, concluding with design recommendations for simulator implementation. Among existing literature, this study is most closely related to the present work. The key distinction lies in scope: while Alaei et al. focus on cycle-accurate GPU simulators that could potentially be extended to multi-node datacenter settings, this study directly investigates datacenter simulators with GPU modelling as a central concern.

Although prior studies—such as [24, 59]—address certain aspects of component modelling in datacenters, there remains a significant gap in developer-oriented overviews and discussions of design decisions. This systematic literature review aims to bridge that gap and provide an up-to-date understanding of the current state of datacenter modelling, particularly regarding GPU integration.

8 CONCLUSION

In this systematic literature review, the different designs decisions of datacenter simulators implementing GPU-enabled simulations. The respective purpose of the simulators have been studied, an introduction to their terminology, as well as the simulation model and the input requirements have been described. Finally, the different level considered details in the simulating the hardware and the influence of GPUs in the simulation process have been ranked.

To identify datacenter simulators with simulated GPUs, a systematic literature search has been conducted with the use of AIP. Due to limited results, a keyword-based search along existing datacenter simulator implementations has been conducted with Google Scholar. Approximately 500 articles have been screened, from which 7 made it into the final selection.

The findings showcase, that only a few datacenter simulators have the desired implementations. Despite the small number show the implementations a diverse set of design decisions. Ranging from cycle-accurate modelling to a boolean representation of GPUs in the simulated hardware. The majority based their GPU-simulation design on the number of instructions or floating-point operations. Inspecting the publicly available code repositories suggests a non-complete implementation of a selection of simulators.

While progress in GPU modelling for datacenter simulators is evident, the challenges outlined in this literature review suggest that the field remains in an early stage of maturity. The lack of coordination across scheduling layers, limited architectural fidelity, and the tendency to address isolated aspects rather than system-wide interactions point to clear limitations in current approaches. These gaps highlight the potential for more integrated and modular simulation frameworks that can better reflect the complexity of modern heterogeneous infrastructures. By addressing these issues, future work can support more realistic and flexible experimentation, helping to bridge the divide between simulation-based research and the practical needs of operational datacenters.

Interested readers could use this guide to either extend their existing datacenter simulator with GPU-support in simulation, or implement a new one filling a gap that the existing implementations have not done yet. Further, could this survey be the foundation to test the difference implementations extensively to showcase their difference in behaviour and test the made claims by the authors. A different spin on the last idea is to select one or two simulators and implement various multi-objective scheduling algorithms.

REFERENCES

[1] [n.d.]. AIP/docs/pdfs/5C_group_final_report.pdf at master · atlarge-research/AIP. https://github.com/atlarge-research/AIP/blob/master/docs/pdfs/5C_group_final_report.pdf
[2] [n.d.]. AMiner - AI-----. https://www.aminer.cn/
[3] [n.d.]. COMPASS | Proceedings of the 29th ACM on International Conference on Supercomputing. https://dl.acm.org/doi/abs/10.1145/2751205.2751220
[4] [n.d.]. dblp: How can I download the whole dblp dataset? https://dblp.uni-trier.de/faq/How+can+I+download+the+whole+dblp+dataset.html
[5] [n.d.]. Digital Decade - Policy programme | Shaping Europe's digital future. https://digital-strategy.ec.europa.eu/en/policies/digital-decade-policy-programme
[6] [n.d.]. Overview — SimPy 4.1.2.dev8+g81c7218 documentation. https://simpy.readthedocs.io/en/latest/
[7] [n.d.]. Semantic Scholar Academic Graph API | Semantic Scholar. https://www.semanticscholar.org/product/api
[8] 2024. lanl/PPT. https://github.com/lanl/PPT original-date: 2017-10-06T22:52:06Z.
[9] 2025. mininet/mininet. https://github.com/mininet/mininet original-date: 2011-04-21T17:46:06Z.
[10] Mohamed Abu Sharkh, Ali Kanso, Abdallah Shami, and Peter Öhlén. 2016. Building a cloud on earth: A study of cloud computing data center simulators. *Computer Networks* 108 (Oct. 2016), 78–96. https://doi.org/10.1016/j.comnet.2016.06.037
[11] Baris Aksanli, Jagannathan Venkatesh, and T. Š Rosing. 2012. Using Datacenter Simulation to Evaluate Green Energy Integration. *Computer* 45, 9 (Sept. 2012), 56–64. https://doi.org/10.1109/MC.2012.249 Conference Name: Computer.
[12] Mohammad Alaei and Fahimeh Yazdanpanah. 2025. A Survey on Heterogeneous CPU–GPU Architectures and Simulators. *Concurrency and Computation: Practice and Experience* 37, 1 (2025), e8318. https://doi.org/10.1002/cpe.8318 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.8318.
[13] Ghazanfar Ali, Mert Side, Sridutt Bhalachandra, Nicholas J. Wright, and Yong Chen. 2023. An automated and portable method for selecting an optimal GPU frequency. *Future Generation Computer Systems* 149 (Dec. 2023), 71–88. https://doi.org/10.1016/j.future.2023.07.011
[14] Tyler Allen, Bennett Cooper, and Rong Ge. 2024. Fine-grain Quantitative Analysis of Demand Paging in Unified Virtual Memory. *ACM Trans. Archit. Code Optim.* 21, 1 (Jan. 2024), 14:1–14:24. https://doi.org/10.1145/3632953
[15] Thomas Anderson, Adam Belay, Mosharaf Chowdhury, Asaf Cidon, and Irene Zhang. 2023. Treehouse: A Case For Carbon-Aware Datacenter Software. *SIGENERGY Energy Inform. Rev.* 3, 3 (Oct. 2023), 64–70. https://doi.org/10.1145/3630614.3630626
[16] Ilyas Bambrik. 2020. A Survey on Cloud Computing Simulation and Modeling. *SN Computer Science* 1, 5 (Aug. 2020), 249. https://doi.org/10.1007/s42979-020-00273-1
[17] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1 (2011), 23–50. https://doi.org/10.1002/spe.995 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.995.
[18] Angela Carrera-Rivera, William Ochoa, Felix Larrinaga, and Ganix Lasa. 2022. How-to conduct a systematic literature review: A quick guide for computer science research. *MethodsX* 9 (Jan. 2022), 101895. https://doi.org/10.1016/j.mex.2022.101895
[19] Michael Chrysostomou, Nicholas Christofides, and Stelios Ioannou. 2024. Turning weakness into strength - A feasibility analysis and comparison of datacenter deployment in hot and cold climates. *Solar Energy Advances* 4 (Jan. 2024), 100068. https://doi.org/10.1016/j.seja.2024.100068
[20] Cloud Infrastructure Engineering, Oracle Corporation, Raleigh, USA, Krishna Chaitanya Sunkara, Krishnaiah Narukulla, and Core Platform Engineering, Roku, San Jose, USA. 2025. Power Consumption and Heat Dissipation in AI Data Centers: A Comparative Analysis. *International Journal of Innovative Research in Science, Engineering and Technology* 14, 02 (Feb. 2025). https://doi.org/10.15680/IJIRSET.2025.1402015
[21] Robert Curtis, Tim Shedd, and Emily B. Clark. 2023. Performance Comparison of Five Data Center Server Thermal Management Technologies. In *2023 39th Semiconductor Thermal Measurement, Modeling & Management Symposium (SEMI-THERM)*. 1–9. https://doi.org/10.23919/SEMI-THERM59981.2023.10267908 ISSN: 2577-1000.
[22] Paweł Czarnul, Jarosław Kuchta, Mariusz Matuszek, Jerzy Proficz, Paweł Rościszewski, Michał Wójcik, and Julian Szymański. 2017. MERPSYS: An environment for simulation of parallel application execution on large scale HPC systems. *Simulation Modelling Practice and Theory* 77 (Sept. 2017), 124–140. https://doi.org/10.1016/j.simpat.2017.05.009
[23] Ala Darabseh, Mahmoud Al-Ayyoub, Yaser Jararweh, Elhadj Benkhelifa, Mladen Vouk, and Andy Rindos. 2015. SDDC: A Software Defined Datacenter Experimental Framework. In *2015 3rd International Conference on Future Internet of Things and Cloud*. 189–194. https://doi.org/10.1109/FiCloud.2015.127
[24] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2016. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 732–794. https://doi.org/10.1109/COMST.2015.2481183 Conference Name: IEEE Communications Surveys & Tutorials.
[25] Mrs Smita Desai and Ms Shreya Desai. 2017. Python and Machine Learning. (2017). https://www.academia.edu/download/117517675/10848.pdf
[26] Maicon Ança dos Santos, Gabriel J. A. Grabher, Matheus F. Kovaleski, Cláudio F. R. Geyer, and Gerson Geraldo H. Cavalheiro. 2023. WCSim: A Cloud Computing Simulator with Support for Bag of Tasks Workflows. In *2023 IEEE 35th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. 230–241. https://doi.org/10.1109/SBAC-PAD59825.2023.00032 ISSN: 2643-3001.
[27] Yongqiang Gao, Lin Wang, and Jiantao Zhou. 2019. Cost-Efficient and Quality of Experience-Aware Provisioning of Virtual Machines for Multiplayer Cloud Gaming in Geographically Distributed Data Centers. *IEEE Access* 7 (2019), 142574–142585. https://doi.org/10.1109/ACCESS.2019.2944405
[28] Saurabh Kumar Garg and Rajkumar Buyya. 2011. NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. 105–113. https://doi.org/10.1109/UCC.2011.24
[29] Wedan Emmanuel Gnibga, Andrew A. Chien, Anne Blavette, and Anne Cécile Orgerie. 2024. FlexCoolDC: Datacenter Cooling Flexibility for Harmonizing Water, Energy, Carbon, and Cost Trade-offs. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems (e-Energy '24)*. Association for Computing Machinery, New York, NY, USA, 108–122. https://doi.org/10.1145/3632775.3661936
[30] Younghwan Go, M. Jamshed, Younggyoun Moon, Changho Hwang, and KyoungSoo Park. 2017. APUNet: Revitalizing GPU as Packet Processing Accelerator. https://www.semanticscholar.org/paper/APUNet%3A-Revitalizing-GPU-as-Packet-Processing-Go-Jamshed/bb3e259eae0d70d7bc53746baedf6a4e01861b31
[31] Mohamed Hadi Habaebi, Yaçine Merrad, Md Rafiqul Islam, Elfatih A A Elsheikh, F M Sliman, and Mokhtaria Mesri. 2023. Extending CloudSim to simulate sensor networks. *SIMULATION* 99, 1 (Jan. 2023), 3–22. https://doi.org/10.1177/00375497221105530 Publisher: SAGE Publications Ltd STM.
[32] Adel Hatami-Marbini, John Otu Asu, and Pegah Khoshnevis. 2024. Environmental performance assessment in the transport sector using nonparametric frontier analysis: A systematic literature review. *Computers & Industrial Engineering* 189 (March 2024), 109968. https://doi.org/10.1016/j.cie.2024.109968
[33] Bowen He, Xiao Zheng, Yuan Chen, Weinan Li, Yajin Zhou, Xin Long, Pengcheng Zhang, Xiaowei Lu, Linquan Jiang, Qiang Liu, Dennis Cai, and Xiantao Zhang. 2023. DxPU: Large-scale Disaggregated GPU Pools in the Datacenter. *ACM Trans. Archit. Code Optim.* 20, 4 (Dec. 2023), 55:1–55:23. https://doi.org/10.1145/3617995
[34] Yi He, Mike Hutton, Steven Chan, Robert De Gruijl, Rama Govindaraju, Nishant Patil, and Yanjing Li. 2023. Understanding and Mitigating Hardware Failures in Deep Learning Training Systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3579371.3589105
[35] Jinlong Hu, Zhizhe Rao, Xingchen Liu, Lihao Deng, and Shoubin Dong. 2024. DCSim: Computing and Networking Integration based Container Scheduling Simulator for Data Centers. https://doi.org/10.48550/arXiv.2411.13809 arXiv:2411.13809 [cs].
[36] Qinghao Hu, Peng Sun, Shengen Yan, Yonggang Wen, and Tianwei Zhang. 2021. Characterization and Prediction of Deep Learning Workloads in Large-Scale GPU Datacenters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15. https://doi.org/10.1145/3458817.3476223 arXiv:2109.01313 [cs].
[37] Qinghao Hu, Zhisheng Ye, Zerui Wang, Guoteng Wang, Meng Zhang, Qiaoling Chen, Peng Sun, Dahua Lin, Xiaolin Wang, Yingwei Luo, Yonggang Wen, and Tianwei Zhang. 2024. Characterization of Large Language Model Development in the Datacenter. https://doi.org/10.48550/arXiv.2403.07648 arXiv:2403.07648 [cs].
[38] Jiahua Huang, Weiwei Lin, Wentai Wu, Yang Wang, Haocheng Zhong, Xinhua Wang, and Keqin Li. 2025. On Efficiency, Fairness and Security in AI Accelerator Resource Sharing: A Survey. *ACM Comput. Surv.* 57, 9 (April 2025), 221:1–221:35. https://doi.org/10.1145/3721427
[39] Sergio Iserte, Javier Prades, Carlos Reaño, and Federico Silla. 2021. Improving the management efficiency of GPU workloads in data centers through GPU virtualization. *Concurrency and Computation: Practice and Experience* 33, 2 (2021), e5275. https://doi.org/10.1002/cpe.5275 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5275.
[40] Azlan Ismail. 2020. Energy-driven cloud simulation: existing surveys, simulation supports, impacts and challenges. *Cluster Computing* 23, 4 (Dec. 2020), 3039–3055. https://doi.org/10.1007/s10586-020-03068-4
[41] Eleni Kanellou, Nikolaos Chrysos, and Angelos Bilas. 2018. A Flexible Datacenter Simulator. *Procedia Computer Science* 136 (Jan. 2018), 72–81. https://doi.org/10.1016/j.procs.2018.08.239

[42] Gábor Kecskeméti. 2015. Scheduler hierarchies to aid peta-scale cloud simulations with DISSECT-CF. Krakow, Poland, 71–82. http://www.nesus.eu/proceedings
[43] Mahmoud Khairy, Zhesheng Shen, Tor M. Aamodt, and Timothy G. Rogers. 2020. Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 473–486. https://doi.org/10.1109/ISCA45697.2020.00047
[44] Khaled M Khalil, M Abdel-Aziz, Taymour T Nazmy, and Abdel-Badeeh M Salem. 2017. CLOUD SIMULATORS – AN EVALUATION STUDY. *International Journal* 6, 1 (2017).
[45] In Kee Kim, Wei Wang, and Marty Humphrey. 2015. PICS: A Public IaaS Cloud Simulator. In *2015 IEEE 8th International Conference on Cloud Computing*. 211–220. https://doi.org/10.1109/CLOUD.2015.37 ISSN: 2159-6190.
[46] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology* 51, 1 (Jan. 2009), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009
[47] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. 2012. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing* 62, 3 (Dec. 2012), 1263–1283. https://doi.org/10.1007/s11227-010-0504-1
[48] Tomasz Kołodziej and Paweł Rościszewski. 2021. Towards Scalable Simulation of Federated Learning. In *Neural Information Processing*, Teddy Mantoro, Minho Lee, Media Anugerah Ayu, Kok Wai Wong, and Achmad Nizar Hidayanto (Eds.). Springer International Publishing, Cham, 248–256. https://doi.org/10.1007/978-3-030-92307-5_29
[49] Adam Krzywaniak, Paweł Czarnul, and Jerzy Proficz. 2022. DEPO: A dynamic energy-performance optimizer tool for automatic power capping for energy efficient high-performance computing. *Software: Practice and Experience* 52, 12 (2022), 2598–2634. https://doi.org/10.1002/spe.3139 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3139.
[50] Ashwin Kumar Kulkarni and B. Annappa. 2021. GPU-aware resource management in heterogeneous cloud data centers. *The Journal of Supercomputing* 77, 11 (Nov. 2021), 12458–12485. https://doi.org/10.1007/s11227-021-03779-4
[51] Uday Kurkure, Hari Sivaraman, and Lan Vu. 2018. Virtualized GPUs in High Performance Datacenters. In *2018 International Conference on High Performance Computing & Simulation (HPCS)*. 887–894. https://doi.org/10.1109/HPCS.2018.00142
[52] Imran Latif, Muhammad Mubashar Ashraf, Umaima Haider, Gemma Reeves, Alexandrina Untaroiu, Fábio Coelho, and Denis Browne. 2025. Advancing Sustainability in Data Centers: Evaluation of Hybrid Air/Liquid Cooling Schemes for IT Payload Using Sea Water. *IEEE Transactions on Cloud Computing* 13, 1 (Jan. 2025), 184–197. https://doi.org/10.1109/TCC.2024.3521666
[53] Francesco Lettich, Emanuele Carlini, Franco Maria Nardini, Raffaele Perego, and Salvatore Trani. 2024. Power- and Fragmentation-aware Online Scheduling for GPU Datacenters. https://doi.org/10.48550/arXiv.2412.17484 arXiv:2412.17484 [cs].
[54] Danyang Li, Jie Song, Hui Liu, and Jingqing Jiang. 2024. Simulators for Conversing Power to Thermal on Green Data Centers: A Review. *Energies* 17, 22 (Jan. 2024), 5631. https://doi.org/10.3390/en17225631 Number: 22 Publisher: Multidisciplinary Digital Publishing Institute.
[55] Ming Li, Ziqian Bi, Tianyang Wang, Yizhu Wen, Qian Niu, Junyu Liu, Benji Peng, Sen Zhang, Xuanhe Pan, Jiawei Xu, Jinlang Wang, Keyu Chen, Caitlyn Heqi Yin, Pohsun Feng, and Ming Liu. 2024. Deep Learning and Machine Learning with GPGPU and CUDA: Unlocking the Power of Parallel Computing. https://doi.org/10.48550/arXiv.2410.05686 arXiv:2410.05686 [cs].
[56] Heting Liu, Zhichao Li, Cheng Tan, Rongqiu Yang, Guohong Cao, Zherui Liu, and Chuanxiong Guo. 2022. Prediction of GPU Failures Under Deep Learning Workloads. https://doi.org/10.48550/arXiv.2201.11853 arXiv:2201.11853 [cs].
[57] Shuo Liu, Bin Zou, Lamei Zhang, and Shulei Ren. 2020. Heterogeneous CPU+GPU-Accelerated FDTD for Scattering Problems With Dynamic Load Balancing. *IEEE Transactions on Antennas and Propagation* 68, 9 (Sept. 2020), 6734–6742. https://doi.org/10.1109/TAP.2020.2990486
[58] Theo Lynn, Anna Gourinovitch, James Byrne, P. J. Byrne, Sergej Svorobej, Konstantinos Giannoutakis, David Kenny, and John P. Morrison. 2017. A preliminary systematic review of computer science literature on cloud computing research using open source simulation platforms.. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*. ScitePress, Porto, Portugal. https://doi.org/10.5220/0006351805650573
[59] Antonios T. Makaratzis, Konstantinos M. Giannoutakis, and Dimitrios Tzovaras. 2018. Energy modeling in cloud simulation frameworks. *Future Generation Computer Systems* 79 (Feb. 2018), 715–725. https://doi.org/10.1016/j.future.2017.06.016
[60] N. Mansouri, R. Ghafari, and B. Mohammad Hasani Zade. 2020. Cloud computing simulators: A comprehensive review. *Simulation Modelling Practice and Theory* 104 (Nov. 2020), 102144. https://doi.org/10.1016/j.simpat.2020.102144
[61] Tshilidzi Marwala. 2024. CPUs Versus GPUs. In *The Balancing Problem in the Governance of Artificial Intelligence*, Tshilidzi Marwala (Ed.). Springer Nature, Singapore, 137–152. https://doi.org/10.1007/978-981-97-9251-1_9
[62] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin van Eyk, Laurens Versluis, Vincent van Beek, and Alexandru Iosup. 2021. OpenDC 2.0: Convenient Modeling and Simulation of Emerging Technologies in Cloud Datacenters. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 455–464. https://doi.org/10.1109/CCGrid51090.2021.00055
[63] Neo Motlhabane, Naison Gasela, and Michael Esiefarienrhe. 2018. Comparative Analysis of Cloud Computing Simulators. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. 1309–1316. https://doi.org/10.1109/CSCI46756.2018.00254
[64] Preethi Josephina Mudialba. 2016. A Study on the Fundamental Properties, Features and Usage of Cloud Simulators. In *2016 International Conference on Platform Technology and Service (PlatCon)*. 1–5. https://doi.org/10.1109/PlatCon.2016.7456782
[65] Mahmood Niazi. 2015. Do Systematic Literature Reviews Outperform Informal Literature Reviews in the Software Engineering Domain? An Initial Case Study. *Arabian Journal for Science and Engineering* 40, 3 (March 2015), 845–855. https://doi.org/10.1007/s13369-015-1586-0
[66] Bin Nie, Ji Xue, Saurabh Gupta, Christian Engelmann, Evgenia Smirni, and Devesh Tiwari. 2017. Characterizing Temperature, Power, and Soft-Error Behaviors in Data Center Systems: Insights, Challenges, and Opportunities. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 22–31. https://doi.org/10.1109/MASCOTS.2017.12 ISSN: 2375-0227.
[67] Alberto Núñez, Jose L. Vázquez-Poletti, Agustin C. Caminero, Gabriel G. Castañé, Jesus Carretero, and Ignacio M. Llorente. 2012. iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator. *Journal of Grid Computing* 10, 1 (March 2012), 185–209. https://doi.org/10.1007/s10723-012-9208-5
[68] Mohammad Abu Obaida, Jason Liu, Gopinath Chennupati, Nandakishore Santhi, and Stephan Eidenbenz. 2018. Parallel Application Performance Prediction Using Analysis Based Models and HPC Simulations. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '18)*. Association for Computing Machinery, New York, NY, USA, 49–59. https://doi.org/10.1145/3200921.3200937
[69] Jon Peddie. 2022. What is a GPU? In *The History of the GPU - Steps to Invention*, Jon Peddie (Ed.). Springer International Publishing, Cham, 333–345. https://doi.org/10.1007/978-3-031-10968-3_7
[70] David Perez Abreu, Karima Velasquez, Marilia Curado, and Edmundo Monteiro. 2020. A comparative analysis of simulators for the Cloud to Fog continuum. *Simulation Modelling Practice and Theory* 101 (May 2020), 102029. https://doi.org/10.1016/j.simpat.2019.102029
[71] Jason Power, Joel Hestness, Marc S. Orr, Mark D. Hill, and David A. Wood. 2015. gem5-gpu: A Heterogeneous CPU-GPU Simulator. *IEEE Computer Architecture Letters* 14, 1 (Jan. 2015), 34–36. https://doi.org/10.1109/LCA.2014.2299539 Conference Name: IEEE Computer Architecture Letters.
[72] Jerzy Proficz. 2020. Process arrival pattern aware algorithms for acceleration of scatter and gather operations. *Cluster Computing* 23, 4 (Dec. 2020), 2735–2751. https://doi.org/10.1007/s10586-019-03040-x
[73] Jerzy Proficz and Paweł Czarnul. 2016. Performance and Power-Aware Modeling of MPI Applications for Cluster Computing. In *Parallel Processing and Applied Mathematics*, Roman Wyrzykowski, Ewa Deelman, Jack Dongarra, Konrad Karczewski, Jacek Kitowski, and Kazimierz Wiatr (Eds.). Springer International Publishing, Cham, 199–209. https://doi.org/10.1007/978-3-319-32152-3_19
[74] Jerzy Proficz, Piotr Sumionka, Jarosław Skomiał, Marcin Semeniuk, Karol Niedzielewski, and Maciej Walczak. 2020. Investigation into MPI All-Reduce Performance in a Distributed Cluster with Consideration of Imbalanced Process Arrival Patterns. In *Advanced Information Networking and Applications*, Leonard Barolli, Flora Amato, Francesco Moscato, Tomoya Enokido, and Makoto Takizawa (Eds.). Springer International Publishing, Cham, 817–829. https://doi.org/10.1007/978-3-030-44041-1_72
[75] Gamal Refai-Ahmed, Md Malekkul Islam, H. Kabbani, T. Cader, Husam Alissa, and Hoa Do. 2022. Holistic Understanding of Best Engineering Practice to Extend Cooling Limit of Next Generations of GPU. In *2022 IEEE 24th Electronics Packaging Technology Conference (EPTC)*. 890–897. https://doi.org/10.1109/EPTC56328.2022.10013227
[76] Ubaid Ur Rehman, Amir Ali, and Zahid Anwar. 2014. secCloudSim: Secure Cloud Simulator. In *2014 12th International Conference on Frontiers of Information Technology*. 208–213. https://doi.org/10.1109/FIT.2014.47
[77] Yifei Ren, Himanshu Agrawal, Nasim Ferdosian, and Reza Nejabati. 2023. PyCloudSim: Modernized Cloud Computing Simulation Framework with the Incorporation of SFC. In *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 92–98. https://doi.org/10.1109/NFV-SDN59219.2023.10329606 ISSN: 2832-2231.
[78] Paweł Rościszewski, Michał Iwański, and Paweł Czarnul. 2019. The impact of the AC922 Architecture on Performance of Deep Neural Network Training. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*. 666–673. https://doi.org/10.1109/HPCS48598.2019.9188164

[79] Dilshad Hassan Sallo and Gabor Kecskemeti. 2021. A Parallel Event System for Large-Scale Cloud Simulations in DISSECT-CF. *Acta Cybernetica* 25, 2 (Aug. 2021), 469–484. https://doi.org/10.14232/actacyb.287937 Number: 2 Publisher: University of Szeged.

[80] Yousef Sanjalawe and Salam Al-E'mari. 2023. Cloud Computing Simulators: A Review. In *2023 24th International Arab Conference on Information Technology (ACIT)*. 1–14. https://doi.org/10.1109/ACIT58888.2023.10453820 ISSN: 2831-4948.

[81] Hoda Sedighi, Daniel Gehberger, Amin Ebrahimzadeh, Fetahi Wuhib, and Roch H. Glitho. 2025. Efficient Dynamic Resource Management for Spatial Multitasking GPUs. *IEEE Transactions on Cloud Computing* 13, 1 (Jan. 2025), 99–117. https://doi.org/10.1109/TCC.2024.3511548

[82] Muhammad Asim Shahid, Muhammad Mansoor Alam, and Mazliham Mohd Su'ud. 2023. A Systematic Parameter Analysis of Cloud Simulation Tools in Cloud Computing Environments. *Applied Sciences* 13, 15 (Jan. 2023), 8785. https://doi.org/10.3390/app13158785 Number: 15 Publisher: Multidisciplinary Digital Publishing Institute.

[83] S. Shanmugapriya and N. Priya Lowast. 2024. Examination of Cloud Simulation Platforms and Implementation of Load Balancing in CloudAnalyst. *Indian Journal of Science and Technology* 17, 33 (Aug. 2024), 3424–3436. https://doi.org/10.17485/IJST/v17i33.1751 Publisher: The Indian Society of Education and Environment.

[84] S. R. Shishira, A. Kandasamy, and K. Chandrasekaran. 2018. Comparative Study of Simulation Tools and Challenging Issues in Cloud Computing. In *Smart Secure Systems – IoT and Analytics Perspective*, Guru Prasadh Venkataramani, Karthik Sankaranarayanan, Saswati Mukherjee, Kannan Arputharaj, and Swamynathan Sankara Narayanan (Eds.). Springer, Singapore, 3–11. https://doi.org/10.1007/978-981-10-7635-0_1

[85] Ahmad Siavashi and Mahmoud Momtazpour. 2019. GPUCloudSim: an extension of CloudSim for modeling and simulation of GPUs in cloud data centers. *The Journal of Supercomputing* 75, 5 (May 2019), 2535–2561. https://doi.org/10.1007/s11227-018-2636-7

[86] Ahmad Siavashi and Mahmoud Momtazpour. 2023. Cloudy: A Python-based Simulator for Modern Cloud Environments. https://doi.org/10.21203/rs.3.rs-3665148/v1 ISSN: 2693-5015.

[87] Ahmad Siavashi and Mahmoud Momtazpour. 2023. gVMP: A multi-objective joint VM and vGPU placement heuristic for API remoting-based GPU virtualization and disaggregation in cloud data centers. *J. Parallel and Distrib. Comput.* 172 (Feb. 2023), 97–113. https://doi.org/10.1016/j.jpdc.2022.10.008

[88] Ahmad Siavashi and Mahmoud Momtazpour. 2025. A Multi-Objective Framework for Optimizing GPU-Enabled VM Placement in Cloud Data Centers with Multi-Instance GPU Technology. https://doi.org/10.48550/arXiv.2502.01909 arXiv:2502.01909 [cs].

[89] Manoel C. Silva Filho, Raysa L. Oliveira, Claudio C. Monteiro, Pedro R. M. Inácio, and Mário M. Freire. 2017. CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 400–406. https://doi.org/10.23919/INM.2017.7987304

[90] Matthew Smith, Luke Zhao, Jonathan Cordova, Xunfei Jiang, and Mahdi Ebrahimi. 2023. Energy-Efficient GPU-Intensive Workload Scheduling for Data Centers. In *2023 International Conference on Machine Learning and Applications (ICMLA)*. 1735–1740. https://doi.org/10.1109/ICMLA58977.2023.00263 ISSN: 1946-0759.

[91] Peitao Song, Zhijian Zhang, Qian Zhang, Liang Liang, and Qiang Zhao. 2020. Implementation of the CPU/GPU hybrid parallel method of characteristics neutron transport calculation using the heterogeneous cluster with dynamic workload assignment. *Annals of Nuclear Energy* 135 (Jan. 2020), 106957. https://doi.org/10.1016/j.anucene.2019.106957

[92] Vijayaraghavan Soundararajan and Kinshuk Govil. 2010. Challenges in building scalable virtualized datacenter management. *SIGOPS Oper. Syst. Rev.* 44, 4 (Dec. 2010), 95–102. https://doi.org/10.1145/1899928.1899941

[93] Kyle L. Spafford and Jeffrey S. Vetter. 2012. Aspen: A domain specific language for performance modeling. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–11. https://ieeexplore.ieee.org/abstract/document/6468530/?casa_token=FpHVsuqa2o8AAAAA:efiNHkO6-qhG_1BcMYprZWU8-p9tJJ56oULL5SLDss09uPOrm2Liu0d3NN_VeB-3TVfLbeQ3SGazJQ

[94] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Esha Choukse, Haoran Qiu, Rodrigo Fonseca, Josep Torrellas, and Ricardo Bianchini. 2025. TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms. https://doi.org/10.48550/arXiv.2501.02600 arXiv:2501.02600 [cs].

[95] Ningning Sun, Yong Li, Lihua Ma, Wenyong Chen, and Dunn Cynthia. 2019. Research on cloud computing in the resource sharing system of university library services. *Evolutionary Intelligence* 12, 3 (Sept. 2019), 377–384. https://doi.org/10.1007/s12065-018-0195-8

[96] Pericherla S. Suryateja. 2016. A Comparative Analysis of Cloud Simulators. *International Journal of Modern Education & Computer Science* 8, 4 (2016). https://www.researchgate.net/profile/Pericherla-Suryateja/publication/301241369_A_Comparative_Analysis_of_Cloud_Simulators/links/579352e608aed51475bcfc29/A-Comparative-Analysis-of-Cloud-Simulators.pdf

[97] Prashanth Thinakaran, Jashwant Raj, Bikash Sharma, Mahmut T. Kandemir, and Chita R. Das. 2018. The Curious Case of Container Orchestration and Scheduling in GPU-based Datacenters. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '18)*. Association for Computing Machinery, New York, NY, USA, 524. https://doi.org/10.1145/3267809.3275466

[98] Harshavardhana A. Uranakara, Shivam Barwey, Francisco E. Hernández Pérez, Vijayamanikandan Vijayarangan, Venkat Raman, and Hong G. Im. 2023. Accelerating turbulent reacting flow simulations on many-core/GPUs using matrix-based kinetics. *Proceedings of the Combustion Institute* 39, 4 (Jan. 2023), 5127–5136. https://doi.org/10.1016/j.proci.2022.07.144

[99] Lukasz Wesolowski, Bilge Acun, Valentin Andrei, Adnan Aziz, Gisle Dankel, Christopher Gregg, Xiaoqiao Meng, Cyril Meurillon, Denis Sheahan, Lei Tian, Janet Yang, Peifeng Yu, and Kim Hazelwood. 2021. Datacenter-Scale Analysis and Optimization of GPU Machine Learning Workloads. *IEEE Micro* 41, 5 (Sept. 2021), 101–112. https://doi.org/10.1109/MM.2021.3097287

[100] Zhisheng Ye, Wei Gao, Qinghao Hu, Peng Sun, Xiaolin Wang, Yingwei Luo, Tianwei Zhang, and Yonggang Wen. 2024. Deep Learning Workload Scheduling in GPU Datacenters: A Survey. *ACM Comput. Surv.* 56, 6 (Jan. 2024), 146:1–146:38. https://doi.org/10.1145/3638757

[101] Muhammad Zakarya, Lee Gillam, Ayaz Ali Khan, and Izaz Ur Rahman. 2021. PerficientCloudSim: a tool to simulate large-scale computation in heterogeneous clouds. *The Journal of Supercomputing* 77, 4 (April 2021), 3959–4013. https://doi.org/10.1007/s11227-020-03425-5

[102] Wei Zhang, Quan Chen, Ningxin Zheng, Weihao Cui, Kaihua Fu, and M. Guo. 2021. Toward QoS-Awareness and Improved Utilization of Spatial Multitasking GPUs. *IEEE Trans. Comput.* 71 (March 2021), 866–879. https://doi.org/10.1109/TC.2021.3064352

[103] Hui Zhao, Fanxin Meng, Nanzhi Feng, Quan Wang, Bo Wan, and Jing Wang. 2021. An Energy-Efficient Task Scheduling Strategy Based on Improved Fireworks Algorithm in Heterogeneous Cloud. In *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. 197–204. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00052