# The OpenDC Vision: Towards Collaborative Datacenter Simulation and Exploration for Everybody

Alexandru Iosup[1,2], Georgios Andreadis[2], Vincent van Beek[2,3], Matthijs Bijman[2], Erwin van Eyk[2], Mihai Neacsu[1], Leon Overweel[2], Sacheendra Talluri[2], Laurens Versluis[1], and Maaike Visser[2]
[1] Vrije Universiteit Amsterdam, [2] Delft University of Technology, [3] Solvinity, Amsterdam, the Netherlands
Contact: a.iosup@vu.nl

***Abstract*—In the new Digital Economy, massive computer systems, often grouped in datacenters, serve as factories "producing" cloud services with massive consumption. However, to afford cloud services globally, we must address new research challenges in designing, operating, and using modern datacenters. We must also address challenges in educating and training the next generation of datacenter engineers. Addressing such challenges, in this work we present our vision on OpenDC: we envision the exploration of various datacenter concepts and technologies, using existing and new scientific methods, enabling new education practices and topics, and leading to the creation of new software and data artifacts. We present the datacenter concepts and technologies we are currently planning to explore using OpenDC. We identify the scientific methods we want to use, and explain our vision of education practices. We present the architecture and open-source program underlying the OpenDC software, and the format and open-access data we use for datacenter experiments. We conclude with an open invitation for the community to join our effort.**

***Keywords*-OpenDC, datacenter operation, scheduling, cloud computing, big data, Simulation, Experimentation.**

## I. Introduction

Our society depends today on computer systems. Numerous daily activities—from the operation of organizations of all scales, to modern governance, to consumers accessing various services—, are part of a Digital Economy worth tens of billions of euros annually and supporting millions of jobs [21]. In this new and rich economy [24], [23], massive computer systems, often grouped in *datacenters*, serve as factories "producing" *cloud services* with massive consumption. To achieve the promise of this relatively new industry, we must overcome new scientific and engineering challenges. Moreover, we must address the new demands of training human resources for this complex field, focusing on both technical and collaborative skills. Towards addressing these challenges and demands, we propose in this work our vision for `OpenDC`, a collection of scientific methods, datacenter technologies and concepts, education practices, and software and data artifacts focusing on the design, operation, and use of modern datacenters.

Designing, operating, and using modern datacenters raise new research challenges. In post-Moore computer systems [58], [18], both hardware and software have experienced rapid change from (internal) amplification towards (external) diversification, proliferation, and commoditization. This has already resulted in a diversity of radically new computer systems tuned towards efficiency and ecosystem-forming, and in customers employing complex ecosystems of cloud services as new ways to create and consume digital artifacts. At the core of these new ecosystems are datacenters. To be efficient, datacenters must use smart resource management, in particular through scheduling. Many traditional scheduling techniques work well for simple services, but only few have been shown to do so when multiple simple services are combined into modern, complex and data-intensive, services; even very successful companies in this field, such as Google, have to rely on well trained engineers and a Site Reliability Engineering business-unit, to solve the numerous problems that appear in the daily operation of their large-scale datacenters [8].

Key to this emerging field, of *massivizing computer systems* by forming efficient and controllable ecosystems out of advanced computer systems, is the exploration of design trade-offs and fundamental limitations related to challenges such as: (i) scalability[1] and, more recently, elasticity, which are today fundamental challenges of computer science; (ii) making datacenters efficient for diverse applications and services available to everyone; (iii) designing datacenters to be used by millions in their daily life, where dependability and especially reliability are equally or even more important than traditional performance; (iv) managing the risks associated with not meeting expectations and service level agreements; (v) managing unprecedented heterogeneity in resources, users, and workloads; etc.

*Which tools to use to explore massivizing computer systems?* The ecosystems exhibit many and diverse components, and their workloads and users often exhibit non-exponential [30], [36], correlated [60], [22], pro-social [45], and even bursty [42] behavior. This indicates that analytical models based on either state-space exploration or non-state-space methods[2] are not directly useful as exploration tools—they suffer from the "curse of dimensionality", are more difficult to setup when steady-state is unclear, and are much more effective under exponential

[1] Jim Gray identified scalability as a grand challenge, in his 1999 Turing Award speech, see technical report MS-TR-99-05.

[2] We use the taxonomy introduced by Kishor Trivedi at WEPPE'17.

assumptions except in limited cases [15]. Exploration through real-world experiments is also challenging in this context, due to inaccessibility of a meaningful environment, unreproducibility of experiments, and high cost of conducting experiments. Instead, we envision with OpenDC to focus on simulation, and conduct additional exploration using analytical modeling and real-world experimentation only when needed. We discuss related work, focusing especially on simulators for datacenter-like environments (in Section VII).

Even if the research and engineering challenges are met, *How to train enough human resources to make use of the capabilities offered by datacenters and cloud technology?* Due to the dire skills shortage in Europe, of over 900,000 IT specialists by 2020 [39], many organizations currently cannot find enough financial and human resources to develop and even to use cloud services. For Small and Medium Enterprises (SMEs), this is already leading to significantly lower adoption of cloud and datacenter technology than in large or rich companies (less than 50% vs. over 75% [21]).

*How to stimulate the growth of the scientific field?* Often, development of industrial products is driven by the drive to satisfy customer requirements. To replicate this sense of competition and strive for excellence, we envision OpenDC as a key component of a global competition for datacenter resource management and scheduling.

Our main contribution in this work is the OpenDC vision:

1) We envision how OpenDC can help with the exploration of datacenter technologies and concepts (in Section II).
2) We identify key scientific methods to employ and extend through work on OpenDC (in Section III).
3) We envision how OpenDC could become beneficial to a diverse set of education settings and practices (in Section IV).
4) We propose a plan to create a global competition for datacenter resource management and scheduling, using OpenDC (in Section V).
5) We describe the current development status of the open-source software and open-access data underlying our vision (in Section VI). In particular, the open-source software of OpenDC includes a web interface, a service-based architecture, and a C++-based simulator.

## II. Vision on Exploring Datacenter Technologies and Concepts

In our vision, OpenDC is able to help with exploring a variety of datacenter technologies and concepts, including, but not limited to:

1) Enabling scalability and, more recently, elasticity, which remain fundamental challenges of computer science;
2) Making datacenters efficient, to make key services available to everyone, including Small and Medium Enterprises and the general public;
3) Designing datacenter-based services used by many, and thus for which dependability and especially reliability are equally or even more important than performance;
4) Understanding and managing the risks associated with not meeting expectations of service level, as needed for business-critical and other important workloads;
5) Managing heterogeneity in resources, users, and workloads. Heterogeneity is not new to computer systems, but appears in modern computer system to unprecedented extent and with novel characteristics.

### A. Exploring Resource Management and Scheduling to Improve Availability

From the scientist running a disease-spread simulation to a consumer playing an online game, availability guarantees are important to cloud users. The large-scale commodity infrastructure in each datacenter will exhibit frequent [32], often correlated [22], [60], and sometimes nearly catastrophic failures. Thus, IaaS providers such as Amazon and Microsoft sign Service-Level Agreements (SLAs), have to include availability among the Service-Level Agreements offered to their customers, and define fines upon breaking the availability-related SLAs. However, these SLAs do not include sophisticated dynamic changes and complex SLAs [27], and cloud customers may not have the technical sophistication to define their requirements [56].

To cope with this and avoid SLA fines, many IaaS providers try to achieve high availability (HA) [44]. Many HA-techniques exist [44], essentially relying on checkpointing, running the same job many times in parallel, or waiting for failed jobs to be re-computed. Modern datacenter applications are already designed as structures of smaller tasks [30], [53], which leads to new opportunities in focusing for availability only on the critical tasks. For example, Shen et al. [56] propose Availability-on-Demand (AoD), a mechanism to manage the deployment of high availability techniques. AoD allows customers to dynamically label certain tasks as high-priority using a simple Application Programming Interface (API), and an algorithm can decide which HA-techniques to deploy, when, and where. Experiments indicate that AoD is effective at minimizing failures for jobs structured as Bags of Tasks (BoT) and Master-Worker (MW). but the diversity of jobs that a datacenter can encounter requires a broader AoD system.

Our research attempts to investigate new mechanisms for achieving availability and policies for their enforcement, but also ways to allow users with different levels of sophistication to formulate their availability requirements. For example, we ask *How to design an AoD system for the diverse jobs of datacenters? Which APIs and policies to equip it with?*

### B. Exploring Scheduling of Complex Workflows with Dynamic, Per-Task Non-Functional Requirements

Scheduling complex workflows and applications such as LIGO [9], BLAST [2], and Montage [54] is already possible. Many scheduling approaches seek to improve metrics such as makespan, cost, energy consumption and resource utilization by introducing new scheduling techniques. Following up on extending the idea of availability to dynamic, per-task SLAs (Section II-A), we envision that traditional and new non-functional requirements could also become dynamic, per-task.

We ask the question: *How to design ways to specify and enforce dynamic, per-task SLAs for complex workflows?* We foresee that answers to this question will enable the

specification of non-functional requirements, such as deadlines, budgets, availability constraints, security and privacy limitations for access to and transfer of data, etc., for each task of a complex workflow and dynamically, yet where possible with minimal input from the user. This gives way for new scheduling approaches, but also to new formalisms for specifying the requirements, and new decision problems about when to change the requirements and for which task.

### C. Exploring Auto-Scaling by Application Domain

Elasticity in datacenters [27] ensures that SLAs are upheld by meeting the resource demand of running applications under variable workloads. As the workload intensity changes, an elastic infrastructure can dynamically increase or decrease the resources provisioned for an application, and thus try to minimize SLA violations or maximize resource utilization; this further finds an efficient cost-performance trade-off. Public clouds, such as Amazon AWS, offer threshold-based autoscalers where users can configure triggers and their scaling effects. However, they do not offer their users tools to explore and predict the effects of using existing (or new) autoscalers.

Prior research [29] indicates that understanding auto-scaler behavior in datacenters is a complex endeavor, in which the application domain, the optimization goals, and other factors can impact the results. We ask the basic research questions: *What laws govern the operation of datacenter autoscalers?* and *Which theories can explain the behavior of datacenter autoscalers?*, and the fundamental question *How to efficiently design datacenter autoscalers?*

### D. Exploring FaaS Management and Applications

Approximately 30% to 45% of the cloud spending of organizations is wasted, due to underutilized resources[3].The drive to reduce this waste is increasing the interest in Function-as-a-Service (FaaS), where small pieces of typically stateless application logic (the functions) are executed on specialized servers in the cloud (the service). The FaaS approach has been gaining traction in the industry: Google Cloud's Functions, Microsoft's Azure Functions, Amazon AWS' Lambda, and Platform9 Systems' Fission[4] all offer FaaS functionality. However, this emerging field also raises interesting research challenges.

In the FaaS paradigm, the cloud users solely focus on the business logic of their applications. The applications are sent to the FaaS platform packaged as cloud functions, typically, stateless functions with clearly defined inputs and outputs, and with workflow structure. The FaaS platform manages all the operational logic of these cloud functions, such as updating, scheduling, and autoscaling. This separation of concerns allows the platform to make more intelligent scheduling decisions, reducing costs, while making the applications more flexible and replicable, but raising complexity-related concerns. Among the complexity concerns, we identify *orchestrating dynamic workflows of tasks, expressing and ensuring performance and other non-functional requirements* (similarly to Section II-C, but at finer granularity), *inspecting and monitoring the functions*, and in general *auditing and controlling the processes associated with FaaS operation*.

We propose a key research question: *How to orchestrate and schedule cloud function workflows in a Function-as-a-Service environment?* To answer this question, we do not have to start from scratch. Similarities exist with the orchestration of (scientific) workflows [57], due to the stateless, short-lived characteristics of both tasks and cloud functions. Although the Workflow Management Systems (WFMS) such as Pegasus [20], ASKALON [52], and Luigi[5] do not have the desired focus on low latency or high-performance, scheduling approaches and techniques from these systems could be reused. Many of the issues important for FaaS also appear in service-oriented and micro-service-based architectures [26], but with FaaS the use of cloud resources poses additional complexity.

### E. Exploring the Combined Memory-Storage Stack

One of the leading contributors to the rise of the digital economy is the massive amount of data generated by human and machine interaction in the digital world. IBM reports that approximately 2.5 Exabytes of data is generated every day [28]. The problem of storing and using this data is relevant to all enterprises. Thus, it becomes necessary for datacenters, which have become the primary avenues of computing for almost all enterprises, to be equipped with the ability to store and access large amounts of data in time.

Modern datacenters combine various types of storage systems, often in complex, distributed storage systems. Traditionally, datacenter storage consisted of main memory of the individual servers, a set of storage appliances (NAS, SAN), and an off-line backup system (Tapes). With the recent increase in the quantity of data, and in the diversity of processing needs and patterns, the setup is increasingly diverse and dynamic. Data which was only occasionally accessed before, such as metrics, is now stored in memory and accessed frequently [50]. Main memory is being used for storage [49] while at the same time the case for hard disk drives in datacenters is being made [11]. It is this diversity in storage systems that needs to be studied: although the storage subsystem in a machine or even in an storage appliance is relatively well understood, we still need to study the behavior of storage systems at datacenter scale, and the complex interplay between various storage devices and the behavior of their distributed, multi-layered software. We ask as key research question: *How to understand the complex interplay between the different layers of the datacenter storage system?* and *How to auto-tier data across the entire memory and storage stack, ensuring an efficient cost-performance trade-off?* To answer these questions, we envision a simulation-based approach, where we could explore the dynamic behavior of workloads, storage software, and storage devices; this contrasts to analytical models, and with the existing real-world investigations [17], [49], [59], both of which suffer in generality and ability to scale.

[3]RightScale presentation "State of the Cloud Report 2017", http://www.rightscale.com/lp/2017-state-of-the-cloud-report

[4]https://github.com/fission/

[5]https://github.com/spotify/luigi

The primary concern of storage is not to just to store vast amounts of data efficiently to be used occasionally, such as when a customer performs a transaction. It is to be able to process it using systems such as Apache Hadoop and Apache Spark and derive value out of it. This adds an additional dimension to the problem of data storage. Also, the popularity of stream processing systems and massive machine learning systems [43] has required storage systems to be more performant than ever. With more data produced every day, the trade-off between cost of storage and cost of not being able to process the data becomes more apparent. This has lead to the use of caching [1] and tiering techniques to efficiently use the performance provided by the storage systems. Several key questions arise, among which: *which performance, reliability and consistency trade-offs are acceptable for the data that is rushing into a datacenter?* and *How can the heterogeneous hardware in a datacenter be used to make these trade-offs?*

### F. Exploring Energy Management and Scheduling

Recent years have seen a significant increase in the amount of energy used by datacenters around the world. In 2012, the demand was around 270 TWh worldwide, with a 4.4% Compound Annual Growth Rate since 2007. This means that datacenters account for around 1.4% of energy consumption worldwide [25]. Reducing the amount of energy consumed is desirable for several reasons, including lower cost of datacenter operation, and reduced carbon footprint associated with datacenters. To minimize cost and environmental impact, and thus maximize profit, making datacenters more energy efficient is imperative. However, exploring energy-related scenarios is expensive and can be very time-consuming [19]. Thus, focusing in the `OpenDC` project on using a data center or cloud simulator to explore and to validate theoretical results is timely and important.

*How can we analyze in simulation the energy-aware decision ecosystem, that is, the interplay between different energy saving techniques, operating at different levels in the datacenter or across datacenters?* A variety of techniques exist [6]. Static Performance Management (SPM) focuses mainly on the design of new hardware, with inherently more efficient circuitry and architectures. Dynamic Performance Management (DPM) covers typically software-based methods that allow for more efficient use of the available resources: scaling down, whenever possible, the performance of individual servers or of server components to reduce energy consumption; putting resources in sleep-mode; virtualizing tasks so that multiple jobs can be consolidated on as few machines as possible; task scheduling and resource sharing, across all types of resources; making algorithms more infrastructure-aware (e.g., where energy can be produced cheaply [47]) and energy-proportional [4]. finding ways to combine different energy saving techniques in such a way that they do not negate each other's positive effects, and data enter performance does not suffer unduly [7]. Such techniques form complex decision ecosystems. For entire datacenters and for multi-datacenter clouds, DPM efforts focus also on algorithms for thermal scheduling, reducing the cost of cooling [47] and improving equipment performance [7].

Decision-making regarding datacenters must be energy-aware, yet precious few tools exist for assisting and facilitating these decisions. Queue-based mathematical models are versatile, but difficult to configure and requiring specialist understanding of the results. Monitoring energy consumption in real data centers can be difficult and expensive [19]. Declarative models, which may be useful for management-engineering, and DevOps-customer definitions of goals, are still under-developed. In contrast, the development of simulators for energy consumption in data centers could offer a mix between tools under a friendly interface, providing a key set of features for energy-aware decision making. We ask the following research questions: *How can we model energy consumption in data centers so that we can simulate it both accurately and fast?*, and *How can simulation be used by datacenter engineers, managers, and other stakeholders to facilitate energy-aware decision making?*

## III. Vision on `OpenDC` Scientific Methods

We envision `OpenDC` will guide and focus our efforts to improve the body of the scientific methods for datacenter research, including, but not limited to:

1) Conducting comprehensive, systematic surveys. The large body of related work for many important topics in resource management and scheduling in datacenters makes it difficult for newcomers to start understanding the field, and for experts to continue mastering the state-of-the-art.
2) Characterizing and modeling the three-dimensional search space: input (workload), platform (datacenter characteristics, especially the hardware and software ecosystem that is not under study), and output (the performance and other non-functional metrics).
3) Characterizing the design space of architectures, mechanisms, policies, and other concepts explored in resource management and scheduling, through reference concepts, e.g., a reference scheduling architecture.
4) Understanding and managing the variability of the systems under test, and finding ways to control experiments and make them reproducible.

### A. Survey of Resource Management and Scheduling Techniques Applied in Datacenters

In the past two decades, much research has been conducted towards scheduling in datacenters, and in related systems, such as clusters, grids, and clouds. Numerous surveys already outline the scheduling algorithms and optimization metrics most commonly developed; however, it is unclear how comprehensive or systematic these surveys were. Recently, research effort has been put into the topics of energy consumption and fairness, on security and privacy aspects of resource management and scheduling, etc. Current surveys do not or partially address these emerging topics, leaving gaps that are not addressed by current survey approaches.

We focus on the methodological question: *How to conduct scientific surveys of resource management and scheduling*

*techniques in datacenters?* In fields such as software engineering, surveys have started to receive proper methodological consideration, for example through the seminal work of Kitchenham et al. [37], [10], and have started to be done using sophisticated software tools that provide automatic filtering based on deep text inspection [40]. We aim to bring to datacenter research the same level of quality and confidence in the results that software-engineering surveys have.

### B. Design of Datacenter Metrics

Metrics are commonly used to quantify the performance of systems under test, allowing their users to understand how well their SLAs are met, their designers focus the testing and tuning processes, and the market to fairly compare products. However, traditional performance metrics, such as response time and resource utilization, and even relatively newer metrics, such as cost per use, may not capture the most sought-after features of modern datacenter operation. For example, performance isolation (and its opposite, performance variability [35]) is yet not fully understood and not transparently reported by cloud providers, scalability remains an elusive reporting target and lacks a universal and practical metric [6], and elasticity provides a multi-faceted optimization target that is difficult to quantify [27], [29].

We raise two important research questions: *How to provide a useful yet reduced set of metrics for modern datacenter operation?*, and *How to design a deep yet practical methodological apparatus for obtaining such metrics?* We envision that conceptual advancements in the design of `OpenDC` can facilitate the design and testing of metrics through a controlled, easy-to-instrument environment. This would complement the conceptual work done currently by the community, for example in the SPEC Research Group[7].

### C. Creating a Reference Scheduling Process for Datacenters

Stakeholders across industry, government, and academia require the services of datacenters, expecting high speed and low cost. To be able to fulfill their growing demands, managing existing resources efficiently and fairly poses many online workload-resource scheduling challenges. The high rate of incoming jobs, the limited amount of precise information available on the system, and the constraint to ensure fairness among submitters, are just some of the issues datacenter schedulers must address.

Consequently, datacenter scheduling is becoming increasingly complex. This development, however, limits the ability to innovate in and improve the scheduling process, and severely restricts the ability of newcomers to the field of datacenter scheduler design to understand and learn about it. Addressing these issues, we propose to answer the question *how to design a reference architecture for cloud schedulers?* We foresee answers to this question as proposing a series of stages common to most scheduling approaches currently in practice, which could be interconnected to form scheduling pipelines and workflows. Such a reference architecture could lead to creating a framework for the development of new approaches, where pragmatically limiting the design space can lead to focus for the community; it can also ease learning about scheduling by making structure more easily understandable. We anticipate the results of this line of research to be very useful for defining a global datacenter-scheduling competition (see Section V).

The notion of a scheduling formalism is not new: Schopf [55] has proposed such a formalism for grid schedulers, dividing the grid scheduling process into 10 stages. Whereas the core principles are easily translated to the datacenter domain, cloud computing brings a number of new challenges with it: new job structures, new stakeholders, and new non-functional requirements such as reliability, elasticity, and scalability need to be ensured.

### D. Reproducibility, Performance, and Accuracy in Simulating Datacenters

Simulation as tool of research offers interesting challenges in proving the performance and validating the instrument, and in showing that the results are reproducible [51]. As demonstrated by other sciences, and by preliminary work in distributed computer systems, building provenance records [3], that is, being able to further account for the paths taken by raw data and results, is also important for modern scientific processes; yet, this is still not provided by datacenter simulation tools.

We ask the following research questions: *How to efficiently validate datacenter simulations, that is, showing they are accurate and precise?*, *How to build datacenter-simulation environments where reproducibility is ensured by the instrument?*, and *What is the performance-validity trade-off for datacenter simulation?* The problems in this space are complex, so we envision that even simple steps forward, such as conducting experiments on the same topic, but with multiple non-communicating groups that reveal their results only at the end of the experimentation phase, could shed light into the problems and lead to improvements.

## IV. Vision on Education Practices

We envision the use of `OpenDC` for educational purposes, from enthusing high-school students about computer systems, through educating researchers as young as B.Sc.-level honors-program students, to giving M.Sc.-level students material for scientific inquiry and engineering practice.

Our aim with the `OpenDC` project is to make datacenter design, operation, and research more appealing, by visualizing the effects of different design choices in datacenter design and scheduler design, and by simplifying the API to access and control the operation of the system.

We conjecture that `OpenDC` also lends itself to gamification in higher education, which in our experience leads to activating education and high engagement of students [31]. To this end, we envision `OpenDC` as supporting the formulation of a serious game, in which individual and group exercises with:

[6] Bill Gropp, Acceptance Speech at SC'16.
[7] https://research.spec.org

1) Achievement-oriented goals such as "Complete the following list of steps in building a datacenter, and explain for each why it works."
2) Socially oriented goals such as "Discuss in the classroom the elements of datacenter design, and collaborate in groups of 3-5 to create simple designs that work."
3) Exploration-oriented goals such as "Optimize the datacenter (get the best score) to run big data workloads for a scientific group with specific deadline constraints, under CapEx and OpEx budget constraints."
4) Competition-oriented goals such as "Win the scheduling competition for this year's course." (see also Section V).

We have already used the collaborative OpenDC software (see Section VI) in various educational settings. The following project are already using or are planning to use OpenDC for educational purposes:

- Already used to deliver key learning experiences in classroom-based (traditional) courses, as part of the Honors Programme course on Distributed Systems, Cloud Computing, and Big Data.
- Already used in project-based learning that is part of credited activities in the B.Sc. Honors Programme at TU Delft, to engage top-level young B.Sc. students in scientific topics. Specifically, the practice of developing and using OpenDC is the focus of project-based learning focusing on datacenters.
- Already used in project-based learning that is part of various M.Sc.-level credited activities at TU Delft and Vrije Universiteit Amsterdam, to engage several M.Sc. students that develop new scheduling techniques and conduct thorough datacenter research.
- We plan to integrate OpenDC in our periodic workshops held as part of the Restart.network[8] education network for refugees in the Netherlands.
- We plan to use OpenDC to support activities of the Royal Academy of Arts and Sciences of the Netherlands, in their effort to promote science in schools.
- We further plan to engage high-school students in IT-education through workshops organized in collaboration with the the Royal Dutch Engineers Society, KIVI[9].

## V. Vision on a Global Scheduling Competition for Datacenters

The resource management system and in particular its scheduler are vital to any cloud system, and key to achieving service and business objectives in datacenters. Although their non-functional requirements are often competing, stakeholders expect high-speed, near-optimal decisions. Consequently, datacenter schedulers tend to become complex, barely accessible systems, and few people actually get the opportunity to familiarize themselves with the scheduling process in real datacenters. This is where a simulation platform such as OpenDC can help: it can provide the opportunity for low-barrier experimentation with datacenter-scheduling concepts, by facilitating the creation of custom schedulers and the exploration of their behavior, in low-cost simulated environments. We envision OpenDC at the center of a global competition for improving scheduling in datacenters.

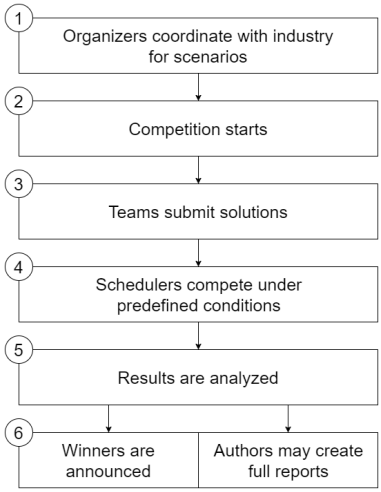


Fig. 1: The structure of a global scheduling competition for datacenters, with the OpenDC platform.

In our vision, users define their own schedulers using the software provided by OpenDC, and especially the modular and flexible reference architecture for cloud schedulers proposed in Section III-C. For each stage of the scheduling process in the reference architecture, users will be able to choose pre-built implementations from a library of components, or write their own components conforming to our specification. By allowing users to focus on specific stages or modules in the complex scheduling process, we conjecture OpenDC can facilitate new competitive designs.

Figure 1 depicts our vision for the global competition between datacenter schedulers. Competition sessions can be organized in conjunction with scientific venues dedicated to topics related to datacenter scheduling, e.g., HPDC, IPDPS, JSSPP, and SC. Each competition session begins with a collection of scenarios, assembled by the competition's Program Committee with the help of companies and research groups affiliated with the community (this step is labeled 1 in the figure). After deciding on scheduling scenarios to model, the competition starts with a public announcement (2), and research labs and other participants can enter their own scheduler in competition (3). Subsequently, the platform will run the submitted schedulers against a workload and topology, in pre-defined, reproducible experiments (4). Based on that year's goals, effectively a selection of relevant metrics, the competition tools select automatically the best-performing scheduler(s) (5), and the Program Committee decides on the winner, and offers a public analysis and explanation (6). Finally, authors can create full reports of their performance, explaining in scientific and engineering detail the design and performance evaluation of their schedulers (6).

[8]https://restart.network/
[9]www.kivi.nl

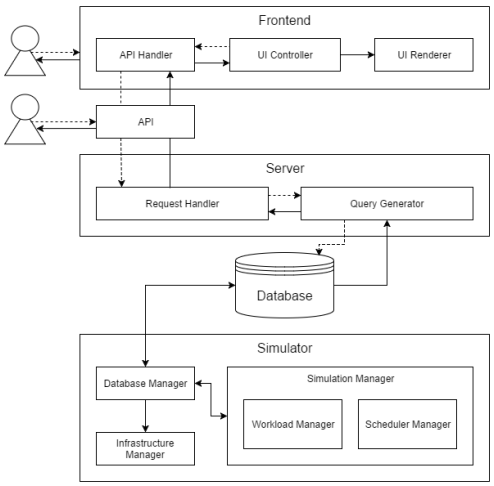


Fig. 2: The OpenDC architecture.

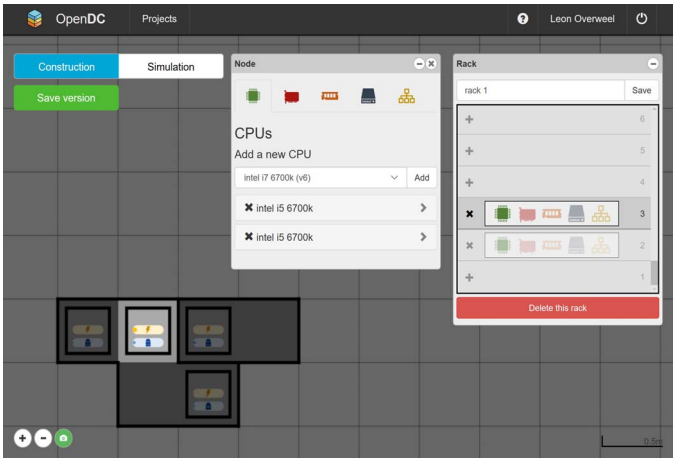


Fig. 3: Example of the web frontend showing the datacenter construction phase.

## VI. Software and Data Artifacts for OpenDC

In this section, we discuss the diverse set of artifacts produced and extended in the OpenDC project.

### A. Overview of the OpenDC Software Artifacts

At the core of OpenDC is a software artifact, developed using modern software engineering methods (e.g., CI/CD) and distributed as open-source software through a public GitHub repository[10].

*1) Overview:* The architecture of OpenDC consists of four components: a frontend, a web server, a database, and a simulator. Figure 2 depicts this architecture.

Using the TypeScript-based GUI ("Frontend" in Figure 2), users can construct a topology by specifying a datacenter's rooms, racks, and machines, and create experiments to see how the datacenter completes a workload. The GUI communicates with the web server over SocketIO, through a custom REST request/response layer. The OpenAPI-compliant API specification (available on GitHub) specifies what requests the GUI can make to the web server.

The Python Flask-based web server ("Server" in Figure 2) receives these API requests and processes them in the database ("Database" in Figure 2). When the GUI requests to run a new experiment, the web server creates an experiment in the database and marks it as QUEUED. Currently, we have implemented the database using SQLite, and are testing various other databases, such as MariaDB and MySQL, to meet the scalability and performance requirements of OpenDC.

The C++-based simulator ("Simulator" in Figure 2) monitors the experiments table, and simulates experiments as they are submitted. It loads the topology, workloads, and scheduler, all as defined by the user. It writes the resulting machine_states and task_states to the database, which the GUI can then again retrieve via the web server.

[10]OpenDC documentation and source code are available on GitHub: https://github.com/atlarge-research/opendc

*2) GUI:* A core aim of OpenDC is to make datacenter technology accessible to a diverse group of users, and in particular to students. The GUI must allow users to create their own datacenters, and to run experiments in these simulated environments while understanding the inner workings of a datacenter, but without significant technological barriers. To address this complex requirement, we have designed a GUI as a browser application, and spent considerable design effort to make the interface useful.

We use web technologies to build the frontend UI, which makes the tool easily accessible to potential stakeholders and inherently cross-platform. To keep the developed artifacts maintainable, we built the UI with TypeScript, a typed superset of the JavaScript language. The main map component is built with the HTML5 canvas element, controlled with the CreateJS library. Finally, the connection with the server is established with web-sockets, leading to testable low overhead when (many) requests are made. The use of web-sockets also allows the server to communicate back with the client, enabling collaborative datacenter design.

The OpenDC GUI supports different levels of granularity: starting at an overall view of the entire datacenter, users can dive into the system by clicking on rooms, racks, and machines. The traversal of the hierarchy is achieved through zooming in and out, and dynamically fading out irrelevant parts of the map, depending on the level. The main navigation view is key to these goals, offering users visual access to the datacenter, at different levels of granularity. Inspired by commercial simulation games, which like OpenDC have to balance creative goals with the cost/time needed to produce content and complex navigational software, we have designed OpenDC's GUI to use a top-down, map-like view for the main navigation view (see Figure 3 for an example).

In the construction phase, users can compose individual machines from CPUs/GPUs, memory, and other elements predefined by the database specification. After constructing the datacenter, users can design and run experiments on it, using the *Simulation mode*. While simulating, they get a live, color-coded view of load metrics and statistics for each

room/rack/machine, and a list of tasks currently in the system and their progress towards completion.

*3) Simulator:* The simulator provides the operational information for students to dissect, researchers to analyze, and DevOps to consider in their designs. To support these diverse stakeholders of OpenDC, the simulator must be fast and scalable, and its code-base must be extensible and comprehensible. Our conjecture is that modern C++, as a native mix between low- and high-level programming, and modern software engineering practices (including good documentation and testing processes), make it possible to achieve these goals.

*4) Roadmap for the* OpenDC *Software:* The roadmap of simulator includes:

- The OpenDC software is continuously extended with the concepts resulting from each research project (Section II).
- An extensive test suite. Our goal is to create a test-suite that not only tests functional requirements, but also profiles the performance of the system and consequently fails if there is a regression.
- An extensive, comprehensive documentation. We are actively documenting the software, but our goal is to also test the usefulness of the documentation in practice.

### B. Overview of Data Artifacts for OpenDC

OpenDC benefits from our long-term activity in collecting and sharing data with the community. Among the workload and operational traces representative for datacenters, OpenDC has enriched or is actively working on extending:

- The Grid Workloads Archive [33] is a source of traces representative for datacenter workloads. Since the start of OpenDC, the Grid Workloads Archive has been enriched with datacenter datasets, collected for us by companies such as Solvinity and Materna.
- The Failure Trace Archive [36] is a source of operational traces collecting resource failures from datacenters and other large-scale environments.

## VII. Related Work

We survey in this section work related to ours, primarily from the field of computer-system simulation, for which we summarize the comparison in Table I. Although the existing simulators provide a rich set of features to build upon, the goal of OpenDC is to serve a more diverse set of stakeholders. In particular, we strive to provide a visual platform to teach, explain, and explore datacenters visually while also supplying researchers with a valuable tool that serves as a backbone for simulation research. Although the existing frameworks provide an adequate base for research, the tools for visualization are often lacking. The new research proposed in Sections II–III also leads to significant R&D effort, which in our view offsets the benefits of using existing technology.

The MONARC [41] and MONARC II simulators focus on CERN-like environments and applications. They have a process-based simulation approach, that is, they allocate one thread of execution for each component and run real-time. The framework includes many scalability and speed improvements.

The SimGrid [16] framework combines the simulation of cluster, grid, and peer to peer architectures. Its key features include a scalable and extensible simulation engine, allowing simulation of arbitrary network topologies, dynamic compute tasks, network availabilities, and resource failures. Code extensibility is provided through high-level interfaces, both C and Java. The framework also includes APIs for simulating distributed applications.

The GridSim [12], [14] framework is an event-based grid simulator with support for modeling heterogeneous resources and applications, resource capability, time zones, network speed, static and dynamic schedulers, failures, etc. A key feature of GridSim is the per-grid-user private resource-broker, focusing on per-user goals instead of the global focus of the typical shared scheduler.

The CloudSim [13] framework focuses on simulating cloud system components including virtual machines, data centers, and resource provisioning policies. Its extensible architecture provides a way for all levels cloud users to test their various configurations and scenarios, from top level users to brokers and cloud providers. The simulator is used by HP Labs to investigate energy-efficient management of dc resources.

FederatedCloudSim [38] is an extension on top of the CloudSim framework. It adds the capability of simulating federated cloud scenarios including support for SLAs.

The DGSim [34] framework is a grid resource management simulator, capable of generating realistic grid and workload scenarios. These features facilitate the setting up and running experiments. DGSim also supports modeling typical resource-management components, including topologies, resources, service level agreements, and resource availability in grids.

The GroudSim [48] framework focuses on scalable grid and cloud simulation, with features including simulation of background load and resource failure, and support for real and realistic cost models.

The OptorSim [5] framework was developed to optimize job scheduling and file replication algorithms. Simulation input contains the network topology and simulated jobs, where jobs are defined by their data-access pattern. OptorSim is extensible, allowing its users to create new data-aware scheduling and replication algorithms.

The PeerSim [46] framework was developed to support protocol simulation on millions of concurrent but simple nodes. The simulator is extensible through Java Reflection—classes defined in a configuration file. PeerSim supports now a variety of protocols, including Pastry, BitTorrent, etc.

## VIII. Conclusion and Ongoing Work

Datacenters "produce" cloud services in the Digital Economy. To afford cloud services globally, we must address many new research, technical, and human-resource challenges. Addressing such challenges, OpenDC aims to change datacenter operation, science, and education.

In this work, we have envisioned how OpenDC can help in exploring datacenter concepts and technologies, in refining and extending scientific methods for the field, and in exploring and deploying education practices. Multiple ongoing projects

| | Simulation | Resources | Stakeholders | Scheduling | Additional Features |
|---|---|---|---|---|---|
| MONARC (2000–2004) | Process-based | Hosts, CPU, Netw., Storage | Comp.sci. | Dynamic local scheduling | CERN DCs and workload |
| SimGrid (2000–) | Event-/Process-based | Hosts, VMs, CPUs, Network, Storage | Comp.sci., Grad stud. | Dynamic local scheduling, networking | Resource failures, APIs for distributed application benchmarking |
| GridSim (2002-2010) | Event-/Process-based | Hosts, VMs, CPUs | Comp.sci., Grad stud. | Dynamic scheduling, parallel workloads | Private resource broker |
| CloudSim (2011–) | Event-based | DCs, Hosts, VMs | Comp.sci., Grad stud. | (Custom) resource and application scheduling policies | Inter-networked clouds, Power, extensibility through interfaces |
| Federated CloudSim (2014–) | Event-based | DCs, Hosts, VMs | Comp.sci., Sys.adm. | (Custom) resource and application scheduling policies | Based on CloudSim, adding SLAs |
| DGSim (2008-2012) | Event-based | Hosts, CPUs | Comp.sci. | (Custom) resource management and scheduling policies | Availability, resource evolution |
| GroudSim (2010–) | Event-based | Hosts, CPUs | Comp.sci. | (Custom) resource management and scheduling policies | Availability, background load, cost models |
| OptorSim (2002-2004) | Step/Event-based | Storage, Compute units | Comp.sci., Grad stud. | data scheduling, replication | File replication |
| PeerSim (2004-2011) | Step/Event-based | P2P nodes, communication | Comp.sci., Grad stud. | distributed protocols (Chord, Kad., etc.) | Many plugin protocols, including BitTorrent, etc. |
| **OpenDC (2016–)** | Step-/Event-based | Hosts, CPUs, GPUs | **Comp.sci., Stud., DevOps, sys.adm.** | Dynamic global scheduling | **Section VI-A, *Sections II–III** |

TABLE I: Datacenter technologies and concepts to explore using `OpenDC`. (A star ("*") denotes our planned capabilities.)

direct `OpenDC` to new science and engineering in datacenter scheduling and resource management. Ongoing projects that use and create new capabilities in `OpenDC` include:

- Ph.D. research project on resource management in datacenters for business-critical workloads, focusing on reducing operational and disaster-recovery risks.
- Ph.D. research project on resource management and scheduling in datacenters running business-critical workloads, focusing on complex workflows with strict SLAs that the customer can change over time.
- Over 10 M.Sc. and B.Sc.(!) research projects on scheduling in datacenters across the space workload-environment-optimization goals. These focus on, e.g., big data, FaaS, portfolio scheduling, auto-scaling, auto-tiering, etc.
- Over 5 B.Sc. engineering projects for creating the software tools and data artifacts of `OpenDC`.
- Several international collaborations, through the SPEC Cloud Group.

We have also introduced processes and tools through which `OpenDC` could become useful for the entire community: a global competition for datacenter resource management and scheduling, open-source software, and open-access data. We are also creating tools to visualize the impact of design choices related to scheduling, workloads, datacenter setup, resource management, etc.

## ACKNOWLEDGMENTS

This research and development process is conducted by a multicultural, inclusive, diverse team. Our work is supported by the Dutch STW/NWO personal grants Veni @large (#11881) and Vidi MagnaData (#14826), by the Dutch national program COMMIT and its funded project COMMissioner, and by the Dutch KIEM project KIESA.

## REFERENCES

[1] Albrecht et al. Janus: Optimal flash provisioning for cloud storage workloads. In *USENIX ATC*, 2013.
[2] Altschul et al. Basic local alignment search tool. *Journal of molecular biology*, 215(3), 1990.
[3] Barga et al. Provenance for scientific workflows towards reproducible research. *IEEE Data Eng. Bull.*, 33(3), 2010.

[4] Barroso et al. The case for energy-proportional computing. *IEEE Computer*, 40(12), 2007.

[5] Bell et al. Simulation of dynamic grid replication strategies in optorsim. In *IEEE/ACM GRID*, 2002.

[6] Beloglazov et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Adv. in Comput.*, 82, 2011.

[7] Beloglazov et al. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Comp. Syst.*, 28(5), 2012.

[8] Beyer et al. *Site Reliability Engineering: How Google Runs Production Systems*. 2016.

[9] Bharathi et al. Characterization of scientific workflows. In *WORKS*, 2008.

[10] Brereton et al. Lessons from applying the systematic literature review process within the software engineering domain. *J. of Sys. and Softw.*, 80(4), 2007.

[11] Brewer et al. Disks for data centers. Technical report, Tech. rep., Google, 2016.

[12] Buyya et al. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concur. and Comput.: Pract. and Exper.*, 14(13-15), 2002.

[13] Calheiros et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw., Pract. Exper.*, 41(1), 2011.

[14] Caminero et al. Extending gridsim with an architecture for failure detection. In *ICPADS*, 2007.

[15] Yonghuan Cao et al. System availability with non-exponentially distributed outages. *IEEE Trans. Reliability*, 51(2), 2002.

[16] Casanova, Legrand, and Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Comput. Model. and Sim.*, 2008.

[17] Fay Chang et al. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2), 2008.

[18] Cusumano and Yoffie. Extrapolating from moore's law. *Commun. ACM*, 59(1):33–35, December 2015.

[19] Dayarathna, Wen, and Fan. Data center energy consumption modeling: A survey. *IEEE Commun. Surv. and Tut.*, 18(1):732–794, 2016.

[20] Deelman et al. Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Comput.*, 20(1):70–76, 2016.

[21] European Commission. Uptake of Cloud in Europe. Digital Agenda for Europe report. Digital Agenda for Europe report. Publications Office of the European Union, Luxembourg., Sep 2014.

[22] Gallet et al. A model for space-correlated failures in large-scale distributed systems. In *Euro-Par*, pages 88–100, 2010.

[23] Gartner Inc. Infrastructure and Operations (I&O) Leadership Vision for 2017, section CIO Technology Priorities. Tech.Rep., 2017.

[24] Gens et al. Worldwide and Regional Public IT Cloud Services 20132017 Forecast. Tech.Rep. by IDC, Doc #242464, Feb 2014.

[25] van Heddeghem et al. Trends in worldwide ICT electricity consumption from 2007 to 2012. *Computer Comm.*, 50:64–76, 2014.

[26] Heinrich et al. Performance engineering for microservices: Research challenges and directions. In *ACM/SPEC ICPE*, pages 223–226, 2017.

[27] Herbst et al. Ready for rain? A view from SPEC research on the future of cloud metrics. *CoRR*, abs/1604.03470, 2016.

[28] IBM. What is big data? Official marketing page. https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html.

[29] Ilyushkin et al. An experimental performance evaluation of autoscaling policies for complex workflows. In *ICPE*, pages 75–86, 2017.

[30] Iosup and Epema. Grid computing workloads. *IEEE Internet Computing*, 15(2):19–26, 2011.

[31] Iosup and Epema. An experience report on using gamification in technical higher education. In *ACM SIGCSE*, 2014.

[32] Iosup et al. On the dynamic resource availability in grids. In *GRID*, pages 26–33, 2007.

[33] Iosup et al. The grid workloads archive. *FGCS*, 24(7):672–686, 2008.

[34] Iosup, Sonmez, and Epema. DGSim: Comparing grid resource management architectures through trace-based simulation. In *Euro-Par*, 2008.

[35] Iosup, Yigitbasi, and Epema. On the performance variability of production cloud services. In *CCGRID*, pages 104–113, 2011.

[36] Javadi et al. The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems. *J. Parallel Distrib. Comput.*, 73(8):1208–1223, 2013.

[37] Kitchenham and Lawrence Pfleeger. Principles of survey research: Part 1–6. *ACM SIGSOFT*, 2001–2003.

[38] Kohne et al. Federatedcloudsim: A sla-aware federated cloud simulation framework. In *CCB*, pages 3:1–3:5, 2014.

[39] Korte et al. e-Skills for Jobs in Europe: Measuring Progress and Moving Ahead. European Commission report., Sep 2014.

[40] Landman, Serebrenik, and Vinju. Challenges for static analysis of java reflection literature review and empirical study. In *ICSE*, 2017.

[41] Legrand and Newman. The MONARC toolset for simulating large network-distributed processing systems. In *WSC*, pages 1794–, 2000.

[42] Li. Realistic workload modeling and its performance impacts in large-scale escience grids. *IEEE Trans. Parallel Distrib. Syst.*, 21(4), 2010.

[43] Lin and Kolcz. Large-scale machine learning at twitter. In *SIGMOD*, pages 793–804, 2012.

[44] S. Loveland. Leveraging virtualization to optimize high-availability system configurations. *IBM Syst. J.*, 2008.

[45] Lu Jia et al. Socializing by gaming: Revealing social relationships in multiplayer online games. *TKDD*, 10(2):11:1–11:29, 2015.

[46] Montresor and Jelasity. PeerSim: A scalable P2P simulator. In *P2P*, pages 99–100, 2009.

[47] Orgerie, de Assunção, and Lefèvre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–31, 2013.

[48] Ostermann et al. Groudsim: An event-based simulation framework for computational grids and clouds. In *Euro-Par WS.*, pages 305–313, 2010.

[49] Ousterhout et al. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. *SIGOPS Oper. Syst. Rev.*, 43(4), 2010.

[50] Pelkonen et al. Gorilla: A fast, scalable, in-memory time series database. *PVLDB*, 8(12):1816–1827, August 2015.

[51] Pérez et al. Reproducibility of execution environments in computational science using semantics and clouds. *Future Generation Comp. Syst.*, 67:354–367, 2017.

[52] Prodan. Online analysis and runtime steering of dynamic workflows in the ASKALON grid environment. In *IEEE CCGrid*, 2007.

[53] Reiss et al. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *SoCC*, page 7, 2012.

[54] Rodriguez and Buyya. A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments. *Concur. and Comput.: Pract. and Exper.*, 2016.

[55] Schopf. Ten actions when grid scheduling. In Nabrzyski, Schopf, and Weglarz, editors, *Grid Resource Management: State of the Art and Future Trends*, pages 15–23. Springer US, 2004.

[56] Shen et al. An availability-on-demand mechanism for datacenters. In *IEEE/ACM CCGrid*, pages 495–504, 2015.

[57] Taylor, Deelman, Gannon, and Shields. *Workflows for e-Science: Scientific Workflows for Grids*. Springer, 2014.

[58] Vardi. Is moore's party over? *Commun. ACM*, 54(11):5–5, 2011.

[59] Weil et al. Ceph: A scalable, high-performance distributed file system. In *USENIX OSDI*, pages 307–320, 2006.

[60] Yigitbasi et al. Analysis and modeling of time-correlated failures in large-scale distributed systems. In *IEEE/ACM GRID*, pages 65–72, 2010.