Capelin

Data-Driven Compute Capacity Procurement for Cloud Datacenters using Portfolios of Scenarios

Andreadis, Georgios; Mastenbroek, Fabian Mastenbroek; van Beek, Vincent; Iosup, Alexandru

DOI
10.1109/TPDS.2021.3084816

Publication date
2021

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Parallel and Distributed Systems

Capelin: Data-Driven Compute Capacity Procurement for Cloud Datacenters using Portfolios of Scenarios

Georgios Andreadis, Fabian Mastenbroek, Vincent van Beek, and Alexandru Iosup

Abstract—Cloud datacenters provide a backbone to our digital society. Inaccurate capacity procurement for cloud datacenters can lead to significant performance degradation, denser targets for failure, and unsustainable energy consumption. Although this activity is core to improving cloud infrastructure, relatively few comprehensive approaches and support tools exist for mid-tier operators, leaving many planners with merely rule-of-thumb judgement. We derive requirements from a unique survey of experts in charge of diverse datacenters in several countries. We propose Capelin, a data-driven, scenario-based capacity planning system for mid-tier cloud datacenters. Capelin introduces the notion of portfolios of scenarios, which it leverages in its probing for alternative capacity-plans. At the core of the system, a trace-based, discrete-event simulator enables the exploration of different possible topologies, with support for scaling the volume, variety, and velocity of resources, and for horizontal (scale-out) and vertical (scale-up) scaling. Capelin compares alternative topologies and for each gives detailed quantitative operational information, which could facilitate human decisions of capacity planning. We implement and open-source Capelin, and show through comprehensive trace-based experiments it can aid practitioners. The results give evidence that reasonable choices can be worse by a factor of 1.5-2.0 than the best, in terms of performance degradation or energy consumption.

Index Terms—Cloud, procurement, capacity planning, datacenter, practitioner survey, simulation

1 Introduction

Cloud datacenters are critical for today's increasingly digital society [21, 23, 24]. Users have come to expect near-perfect availability and high quality of service, at low cost and high scalability. Planning the capacity of cloud infrastructure is a critical yet non-trivial optimization problem that could lead to significant service improvements, cost savings, and environmental sustainability [5]. This activity includes short-term capacity planning, which includes the process of provisioning and allocating resources from the capacity already installed in the datacenter, and *long-term capacity planning*, which is the process of *procuring* machines that form the datacenter capacity. This work focuses on the latter, which is a process involving large amounts of resources and decisions that are difficult to reverse.

Although many approaches to the long-term capacity-planning problem have been published [14, 54, 66], companies use much rule-of-thumb reasoning for procurement decisions. To minimize operational risks, many such industry approaches currently lead to significant overprovisioning [26], or miscalculate the balance between underprovisioning and overprovisioning [50]. In this work, as Figure 1 depicts, we approach the problem of capacity planning for mid-tier cloud datacenters with a semi-automated, specialized, data-driven tool for decision making.

We focus in this work mainly on *mid-tier providers* of cloud infrastructure that operate at the low- to mid-level tiers of the service architecture, ranging from IaaS to PaaS. Compared to the extreme-scale operators Google, Facebook, and others in the exclusive GAFAM-BAT group, the mid-tier operators are small-scale. However, they are both numerous and they are responsible for much of the datacenter capacity

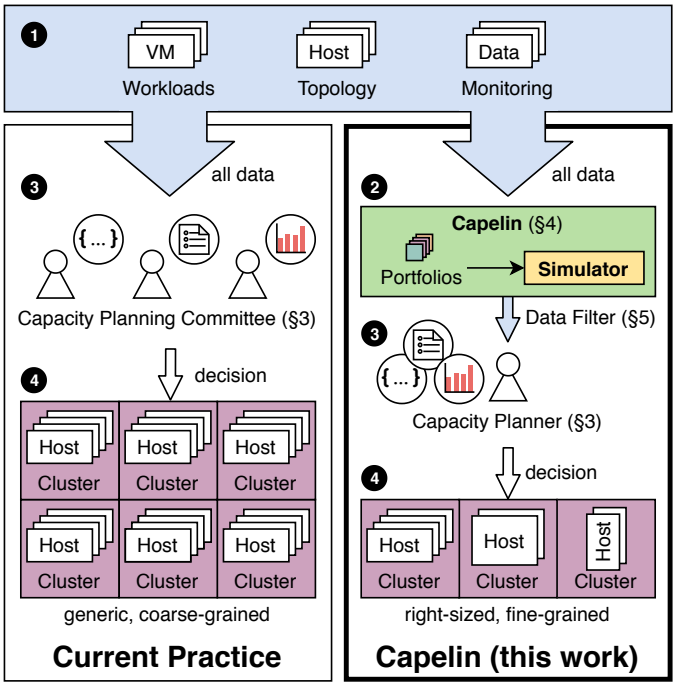

Figure 1. Capelin, a new, data-based capacity planning process for datacenters, compared against the current approach.

- *G. Andreadis, F. Mastenbroek, V. van Beek, and A. Iosup are with Electrical Engineering, Mathematics & Computer Science, Delft University of Technology, 2628 CD Delft, Netherlands.*
- *V. van Beek is also with Solvinity, 1100 ED Amsterdam, Netherlands.*
- *A. Iosup is also with Computer Science, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, Netherlands.*

Manuscript received MMMM DD, YYYY; revised MMMM DD, YYYY.

in modern, service-based and knowledge-driven economies. This work addresses four main capacity planning challenges for mid-tier cloud providers. First, the **lack of published knowledge about the current practice of long-term cloud capacity planning**. For a problem of such importance and long-lasting effects, it is surprising that the only studies of how practitioners make and take long-term capacity-planning decisions are either over three decades old [45] or focus on non-experts deciding how to externally procure capacity for IT services [7]. A survey of expert capacity planners could reveal new requirements.

Second, we observe the **need for a flexible instrument for long-term capacity planning, one that can address various operational scenarios**. State-of-the-art tools [31, 35, 63] and techniques [14, 25, 58] for capacity-planning operate on abstractions that match only one vendor or focus on simplistic problems. Although single-vendor tools, such as VMware's Capacity Planner [63] and IBM's Z Performance and Capacity Analytics tool [35], can provide good advice for the cloud datacenters equipped by that vendor, they do not support real-world cloud datacenters that are heterogeneous in both software [2][5, §2.4.1] and hardware [11, 19][5, §3]. Yet, to avoid vendor lock-in and licensing costs, cloud datacenters acquire heterogeneous hardware and software from multiple sources and could, for example, combine VMware's, Microsoft's, and open-source OpenStack+KVM virtualization management technology, and complement it with container technologies. Although linear programming [64], game theory [58], stochastic search [25], and other optimization techniques work well on simplistic capacity-planning problems, they do not address the multi-disciplinary, multi-dimensional nature of the problem. As Figure 1 (left) depicts, without adequate capacity planning tools and techniques, practitioners need to rely on rules-of-thumb calibrated with casual visual interpretation of the complex data provided datacenter monitoring. This state-of-practice likely results in overprovisioning of cloud datacenters, to avoid operational risks [27]. Even then, evolving customers and workloads could make the planned capacity insufficient, leading to risks of not meeting Service Level Agreements [1, 8], inability to absorb catastrophic failures [5, p.37], and even unwillingness to accept new users.

Third, we identify the **need for comprehensive evaluations of long-term capacity-planning approaches, based on real-world data and scenarios.** Existing tools and techniques have rarely been tested with real-world scenarios, and even more rarely with real-world operational traces that capture the detailed arrival and execution of user requests. Furthermore, for the few thus tested, the results are only rarely peer-reviewed [1, 54]. We advocate comprehensive experiments with real-world operational traces and diverse scaling scenarios to test capacity planning approaches.

Fourth and last, we observe the **need for publicly available, comprehensive tools for long-term capacity planning**. However, and in stark contrast with the many available tools for short-term capacity planning, few procurement tools are publicly available, and even fewer are open-source. From the available tools, none can model all the aspects needed to analyze cloud datacenters from §2.

We propose in this work Capelin, a data-driven, scenario-based alternative to current capacity planning approaches. Figure 1 visualizes our approach (right column of the figure) and compares it to current practice (left column). Both approaches start with inputs such as workloads, current topology, and large volumes of monitoring data (step ❶ in the figure). From this point on, the two approaches diverge, ultimately resulting in qualitatively different solutions. The current practice expects a committee of various stakeholder to extract meaning from all the input data (❸), which is severely hampered by the lack of decision support tools. Without a detailed understanding of the implications of various decisions, the final decision is taken by committee, and it is typically an overprovisioned and conservative approach (❹). In contrast, Capelin adds and semi-automates a data-driven approach to data analysis and decision support (❷), and enables capacity planners to take fine-grained decisions based on curated and greatly reduced data (❸). With such support, even a single capacity planner can make a tailored, fine-grained decision on topology changes to the cloud datacenter (❹). More than a purely technical solution, this approach can change organizational processes. Overall, our main contribution is:

1) We design, conduct, and analyze community interviews on capacity planning in different cloud settings (Section 3). We use broad, semi-structured interviews, from which we identify new, real-world requirements.
2) We design Capelin, a semi-automated, data-driven approach for long-term capacity planning in cloud datacenters (Section 4). At the core of Capelin is an abstraction, the capacity planning portfolio, which expresses sets of "what-if" scenarios. Using simulation, Capelin estimates the consequences of alternative decisions.
3) We demonstrate Capelin's ability to support capacity planners through experiments based on real-world operational traces and scenarios (Section 5). We implement a prototype of Capelin as an extension to OpenDC, an open-source platform for datacenter simulation [37]. We conduct diverse trace-based experiments. Our experiments cover four different scaling dimensions, and workloads from both private and public clouds. They also consider different operational factors such as the scheduler allocation policy, and phenomena such as correlated failures and performance interference [43, 61, 65].
4) We release our prototype of Capelin, consisting of extensions to OpenDC 2.0 [47], as Free and Open-Source Software (FOSS), for practitioners to use. Capelin is engineered with professional, modern software development standards and produces reproducible results.

2 A System Model for DC Operations

In this work we assume the generic model of cloud infrastructure and its operation depicted by Figure 2.

Workload: The workload consists of applications executing in *Virtual Machines (VMs)* and *containers*. The emphasis of this study is on *business-critical workloads*, which are long-running, typically user-facing, and back-end enterprise services at the core of an enterprise's business [56, 57]. Their downtime, or even just low Quality of Service (QoS), can incur significant and long-lasting damage to the business. We also consider virtual *public cloud workloads* in this model, submitted by a wider user base.

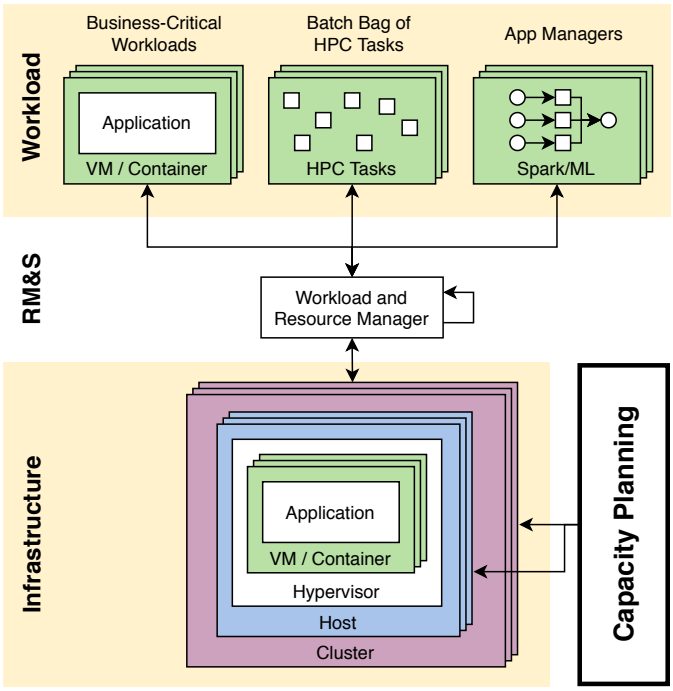

Figure 2. Generic model for datacenter operation.

The business-critical workloads we consider also include virtualized High Performance Computing (HPC) parts. These are primarily comprised of conveniently (embarrassingly) parallel tasks, e.g., Monte Carlo simulations, forming *batch bags-of-tasks*. Larger HPC workloads, such as scientific workloads from healthcare, also fit in our model.

Our system model also considers app managers, such as the big data frameworks Spark and Apache Flink, and machine learning frameworks such as TensorFlow, which orchestrate virtualized workflows and dataflows.

Infrastructure: The workloads described earlier run on physical datacenter infrastructure. Our model views datacenter infrastructure as a set of physical clusters of possibly *heterogeneous hosts* (machines), each host being a node in a datacenter rack. A host can execute multiple VM or container workloads, managed by a *hypervisor*. The hypervisor allocates computational time on the CPU between the workloads that request it, through *time-sharing* (if on the same cores) or *space-sharing* (if on different cores).

We model the CPU usage of applications for discretized time slices. Per slice, all workloads report requested CPU time to the hypervisor and receive the granted CPU time that the resources allow. We assume a generic memory model, with memory allocation constant over the runtime of a VM. As is common in industry, we allow overcommission of CPU resources [6], but not of memory resources [57].

Infrastructure phenomena: Cloud datacenters are complex hardware and software ecosystems, in which complex phenomena emerge. We consider in this work two well-known operational phenomena, performance variability caused by performance interference between collocated VMs [43, 44, 61] and correlated cluster failures [9, 20, 22].

Live Platform Management (RM&S in Figure 2)**:** We model a workload and resource manager that performs management and control of all clusters and hosts, and is responsible for the lifecycle of submitted VMs, including their placement onto the available resources [3]. The resource manager is configurable and supports various *allocation policies*, defining the distribution of workloads over resources. The devops team monitors the system and responds to incidents that the RM&S cannot self-manage [8].

Capacity Planning: Closely related with infrastructure and live platform management is the activity of *capacity planning*. This activity is conducted periodically and/or at certain events by a capacity planner (or committee). The activity typically consists of first *modeling* the current state of the system (including its workload and infrastructure) [48], *forecasting* future demand [15], deriving a capacity *decision* [67], and finally *calibrating and validating* the decision [41]. The latter is done for QoS, possibly expressed as detailed Service Level Agreements (SLAs) and Service Level Objectives (SLOs).

Which cloud datacenters are relevant for this model? We focus in this work on capacity planning for *mid-tier cloud infrastructures*, characterized by relatively small-scale capacity, temporary overloads being common, and a lack of in-house tools or teams large enough to develop them quickly. In Section 3 we analyze the current state of the capacity planning practice in this context and in Section 7 we discuss existing approaches in related literature.

Which tools support this model? We are not aware of analytical tools that can cope with these complex aspects. Although tools for VM simulation exist [13, 32, 51], few support CPU over-commissioning and none outputs detailed VM-level metrics; the same happens for infrastructure phenomena. From the few industry-grade procurement tools who published details about their operation, none supports the diverse workloads and phenomena considered here.

3 Real-World Experiences with Capacity Planning in Cloud Infrastructures

Real-world practice can deviate significantly from published theories and strategies. In this section, we conduct and analyze interviews with 8 practitioners from a wide range of backgrounds and multiple countries, to assess whether this is the case in the field of capacity planning.

3.1 Method

Our goal is to collect real-world experiences from practitioners systematically and without bias, yet allowing personalized lines of investigation.

3.1.1 Interview type

The choice of interview type is guided by the trade-off between the systematic and flexible requirements. A text survey, while highly suited for a systematic study, generally does not allow for low-barrier individual follow-up questions or even conversations. An in-person interview without pre-defined questions allows full flexibility, but can result in unsystematic results. We use the *general interview guide approach* [59], a semi-structured type of interview that ensures key topics are covered but permits deviations from the script. We conduct in-person interviews with a prepared script of ranked questions, and allow the interviewer the choice of which scripted questions to use and when to ask additional questions.

Table 1
Summary of interviews. (Notation: TTD = Time to Deploy, CP = Cloud Provider, DC = Datacenter, M = Monitoring, m/y = month/year, NIT = National IT Infrastructure Provider, SA = Spreadsheet Analysis.)

Int.	Role(s)	Backgr.	Scale	Scope	Tooling	Workload Comb.	Frequency	TTD
1	Researcher	CP	rack	multi-DC	M	combined	3m, ad-hoc	?
2	Board Member	NIT	iteration	multi-DC	–	combined	4–5y	12–18m
3	Manager, Platform Eng.	CP	rack	multi-DC	M	combined	ad-hoc	4–5m
4	Manager	NIT	iteration	per DC	M	benchmark	6–7y	18m
5	Hardware Eng.	NIT	iteration	per DC	M	benchmark	6y	18m
6	Researcher	NIT	rack	multi-DC	M	separate	6m	12m
7	Manager	NIT	iteration	multi-DC	M, SA	combined	5y	3.5-4y

3.1.2 Data collection

Our data collection process involves three steps. Firstly, we *selected and contacted* a broad set of prospective interviewees representing various kinds of datacenters, with diverse roles in the process of capacity planning, and with diverse responsibility in the decisions.

Secondly, we *conducted and recorded the interviews*. Each interview is conducted in person and digitally recorded with the consent of the interlocutor. Interviews last between 30 and 60 minutes, depending on availability of the interlocutors and complexity of the discussion. To help the interviewer select questions and fit in the time-limits imposed by each interviewee, we rank questions by their importance and group questions broadly into 5 categories: (1) introduction, (2) process, (3) inside factors, (4) outside factors, and (5) summary and followup. The choice between questions is then dynamically adjusted to give precedence to higher-priority questions and to ensure each category is covered at least briefly. The script itself is listed in the extended technical report [4].

Thirdly, the recordings are *manually transcribed* into a full transcript. Because matters discussed in these interviews may reveal sensitive operational details about the organisations of our interviewees, all interview materials are handled *confidentially*. No information that could reveal the identity of the interlocutor or that could be confidential to an organization's operations is shared and all raw records will be destroyed directly after this study.

3.1.3 Analysis of Interviews

Due to the unstructured nature of the chosen interview approach, we combine a question-based aggregated analysis with incidental findings. Our approach is inspired by the Grounded Theory strategy set forth by Coleman et al. [16], and has two steps. First, for each transcript, we *annotate* each statement made based on which questions it is relevant to. This may be a sub-sentence remark or an entire paragraph of text, frequently overlapping between different questions. We augment this systematic analysis with more general findings, including comments on unanticipated topics.

Secondly, we traverse all transcripts for each question and form *aggregate observations for each question* in the transcript. The technical report [4] details the findings. From these, we synthesize Capelin requirements (§4.1).

3.2 Observations from the Interviews

Table 1 summarizes the results of the interviews. In total, we transcribed over 35,000 words,which is a very large amount of raw interview data. We conducted 7 interviews with practitioners from commercial and academic datacenters, with roles ranging from capacity planners, to datacenter engineers, to managers. We summarize our main observations:

O1: A majority of practitioners find that the process involves a *significant amount of guesswork and human interpretation*. Interlocutors managing commercial infrastructures emphasize multi-disciplinary challenges such as lease and support contracts, and personnel considerations.

O2: In all interviews, we notice the *absence of any dedicated tooling* for the capacity planning process. Instead, the surveyed practitioners rely on visual inspection of data, through monitoring dashboards. We observe two main reasons for not using dedicated tooling: (1) tools tend to underrepresent the complexity of the real situation, and (2) have high costs with many additional, unwanted features.

O3: The organizations using these capacity planning approaches provide a *range of digital services*, ranging from general IT services to specialist hardware hosting. They run VM workloads, in both commercial and scientific settings, and batch and HPC workloads, mainly in scientific settings.

O4: A *large variety of factors* are taken into account when planning capacity. The three named in a majority of interviews are (1) the use of historical monitoring data, (2) financial concerns, and (3) the lifetime and aging of hardware.

O5: *Success and failure* in capacity planning are underspecified. Definitions of success differ: two interviewees see the use of new technologies as a success, and one interprets the absence of total failure events as a success. Challenges include chronic underutilization, increasing complexity, and small workloads. Failures include late decisions, misprediction, and new technology having unforeseen consequences.

O6: *The frequency of capacity planning processes seems correlated with the duration of core activities using it:* commercial clouds deploy within 4-5 months from the start of capacity planning, whereas scientific clouds take 1–1.5 years.

O7: We found three *financial and technical factors* that play a role in capacity planning: (1) funding concerns, (2) special hardware requests, and (3) the cost of new hardware. In two interviews, interlocutors state that financial considerations prime over the choice of technology (e.g., vendor, model).

O8: The *human aspect* of datacenter operations is emphasized in 5 of the 7 interviews. The datacenter administrators need training, and wrong decisions in capacity planning lead to stress within the operational teams. Users also need training, to leverage heterogeneous or new resources.

O9: We observe a *wide range of requirements and wishes expressed by interlocutors* about custom tools for the process. Fundamentally, the tool should help manage the increasing complexity faced by capacity planners. A key requirement

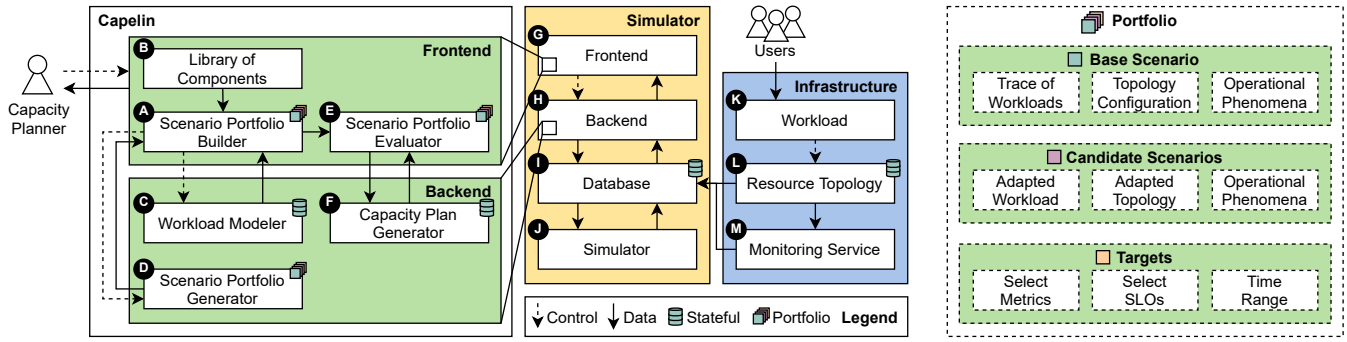

Figure 3. (left) An overview of the architecture of Capelin described in Section 4.2. Capelin is provided information on the current state of the infrastructure and assists the capacity planner in making capacity planning decisions. Labels indicate the order of traversal by the capacity planner (e.g., the first step is to use component A, the scenario portfolio builder). (right) The portfolio is the key data structure of Capelin. Each capacity planning portfolio consists of a base scenario, multiple candidate scenarios, and a set of comparison targets. More details in text, in Section 4.3.

for any tool is interactivity: practitioners want to be able to interact with the metrics they see and ask questions from the tool during capacity planning meetings. The tool should be affordable and usable without needing the entire toolset of the vendor. One interviewee asks for support for infrastructure heterogeneity, to support scientific computing.

O10: Two interviewees detail *"what-if" scenarios* they would like to explore with a tool, using several dimensions: (1) the *topology*, in the form of the computational and memory capacity needed, or new hardware arriving; (2) the *workload*, and especially emerging kinds; and (3) the *operational phenomena*, such as failures and the live management of the platform (e.g., scheduling and fail-over scenarios).

4 Design of Capelin: A Capacity Planning System for Cloud Infrastructure

In this section, we synthesize requirements and design around them a capacity planning approach for cloud infrastructure. We propose Capelin, a scenario-based capacity planning system that helps practitioners understand the impact of alternatives. Underpinning this process, we propose as core abstraction the *portfolio of capacity planning scenarios*.

4.1 Requirements Analysis

In this section, from the results of Section 3, we synthesize the core functional requirements addressed by Capelin. Instead of aiming for full automation–a future objective that is likely far off for the field of capacity planning–the emphasis here is on human-in-the-loop decision support [38, P2].

(FR1) Model a cloud datacenter environment (see O2, O3, O7, and O9): The system should enable the user to model the datacenter topology and virtualized workloads introduced in Section 2.

(FR2) Enable expression of what-if scenarios (see O2, O10): Users can express what-if scenarios with diverse topologies, failures, and workloads. The system should then execute the what-if scenario(s), and produce and justify a set of user-selected QoS metrics.

(FR3) Enable expression of QoS requirements, in the form of SLAs, consisting of several SLOs (see O2, O5, O9). These requirements are formulated as thresholds or ranges of acceptable values for user-selected metrics.

(FR4) Suggest a portfolio of what-if scenarios, based on user-submitted workload traces, given topology, and specified QoS requirements (see O2, O10). This greatly simplifies identifying meaningful scenarios.

(FR5) Provide and explain a capacity plan, optimizing for minimal capacity within acceptable QoS levels, as specified by FR4 (see O2, O9). The system should explain and visualize the data sources it used to make the plan.

4.2 Overview of the Capelin Architecture

Figure 3 depicts an overview of the Capelin architecture. Capelin extends OpenDC, an open-source, discrete event simulator with multiple years of development and operation [37]. We now discuss each main component of the Capelin architecture, taking the perspective of a capacity planner. We outline the abstraction underpinning this architecture, the *capacity planning portfolios*, in §4.3.

4.2.1 The Capelin Process

The frontend and backend of Capelin are embedded in OpenDC. This enables Capelin to leverage the simulator's existing platform for datacenter modeling and allows for inter-operability with other tools as they become part of the simulator's ecosystem. The capacity planner interacts with the frontend of Capelin, starting with the *Scenario Portfolio Builder* (component A in Figure 3), addressing FR2. This component enables the planner to construct scenarios, using pre-built components from the *Library of Components* (B). The library contains workload, topology, and operational building blocks, facilitating fast composition of scenarios. If the (human) planner wants to modify historical workload behavior or anticipate future trends, the *Workload Modeler* (C) can model workloads and synthesize custom loads.

The planner might not always be aware of the full range of possible scenarios. The *Scenario Portfolio Generator* (D) suggests customized scenarios extending the given base-scenario (FR4). The portfolios built in the builder can be explored and evaluated in the *Scenario Portfolio Evaluator* (E). Finally, based on the results from this evaluation, the *Capacity Plan Generator* (F) suggests plans to the planner (FR5).

4.2.2 The Datacenter Simulator

In Figure 3, the *Frontend* (G) acts as a portal, through which infrastructure stakeholders interact with its models and experiments. The *Backend* (H) responds to frontend requests, acting as intermediary and business-logic between frontend,

and database and simulator. The *Database* (❶) manages the state, including topology models, historical data, simulation configurations, and simulation results. It receives inputs from the real-world topology and monitoring services, in the form of workload traces. The *Simulator* (❷) evaluates the configurations stored in the database and reports the simulation results back to the database.

OpenDC [47] is the simulation platform backing Capelin, enabling the capacity planner to model (FR1) and experiment (FR5) with the cloud infrastructure, interactively. The software stack of this platform is composed of a web app frontend, a web server backend, a database, and a discrete-event simulator. This kind of simulator offers a good trade-off between accuracy and performance, even at the scale of mid-tier datacenters and with long-term workloads.

4.2.3 Infrastructure

The infrastructure follows the system model described in Section 2. We consider three components for the cloud infrastructure, that is, the datacenter to be managed and planned: the *workload* (❿), the (logical or physical) *resource topology* (❶), and a *monitoring service* (Ⓜ).

4.3 A Portfolio Abstraction for Capacity Planning

In this work we propose a new abstraction, which organizes multiple scenarios into a *portfolio* (see the right-hand side of Figure 3). Each portfolio includes a base scenario, a set of candidate scenarios given by the user and/or suggested by Capelin, and a set of targets used to compare scenarios and select the best. Our portfolio reflects the multi-disciplinary and multi-dimensional nature of capacity planning by including *multiple* scenarios and a *set* of targets, which we describe in the following, in turn. In contrast, most capacity planning approaches in published literature are tailored towards a *single* scenario—a single potential hardware expansion, a single workload type, one type of service-quality metrics; this approach does not cover the complexities that capacity planners are currently facing (see Section 3.2).

4.3.1 Scenarios

A scenario represents a point in the capacity planning (datacenter design) space to explore. It consists of a combination of workload, topology, and a set of *operational phenomena*. Phenomena can include correlated failures, performance variability, security breaches, etc., allowing the scenarios to accurately capture real-world operations. Such phenomena are often hard to predict intuitively during capacity planning, due to emergent behavior that can arise at scale.

The baseline for comparison in a portfolio is the *base scenario*. It represents the status quo of the infrastructure or, when planning infrastructure from scratch, it consists of very simple base workloads and topologies.

The other scenarios in a portfolio, called *candidate scenarios*, represent changes to the configuration that the capacity planner could be interested in. Changes can be effected in one of the following four dimensions: (1) *Variety:* qualitative changes to the workload or topology (e.g., different arrival patterns, or resources with more capacity); (2) *Volume:* quantitative changes to the workload or topology (e.g., more workloads or more resources); (3) *Velocity:* speed-related changes to workload or topology (e.g., faster resources); and (4) *Vicissitude* combines (1)-(3) over time.

This approach to derive candidate scenarios is systematic, and although abstract it allows approaching many of the practical problems discussed by capacity planners. For example, should we *scale horizontally (scale-out) or vertically (scale-up)?* [55]. Scaling out, which is done by adding cheaper clusters of commodity machines, contrasts to scaling up, which is done by acquiring more expensive, "beefy" machines. Scaling out can be cheaper for the same performance, and offers a more failure-targets (machines). Yet, scaling up could lower operational costs, due to fewer per-machine licenses, fewer switch-ports for networking, and smaller floor-space due to fewer racks. Experiment 5.2 explores this dichotomy.

4.3.2 Targets

A portfolio also has a set of targets that prescribe on what grounds the different scenarios should be compared. Targets include the metrics that the practitioner is interested in and their desired granularity, along with relevant SLOs (FR3). Following the taxonomy defined by the performance organization SPEC [30], we support both *system-provider metrics* (such as operational risk and resource utilization) and *organization metrics* (such as SLO violation rates and performance variability). Targets also include a time range over which metrics should be recorded and compared.

5 Experiments with Capelin

In this section, we explore how Capelin can be used to answer capacity planning questions. We conduct extensive experiments using Capelin and data from operational traces collected long-term from private and public datacenters.

5.1 Experiment Setup

We implement a prototype of Capelin (§5.1.1), and verify the reproducibility of its results and that it can be run within the expected duration of a capacity planning session (§5.1.2). All experiments use *long-term, real-world traces as input*.

Our experiment design, which Table 2 summarizes, is comprehensive and addresses key questions such as: Which input workload (§5.1.3)? Which datacenter topologies to consider (§5.1.4)? Which operational phenomena (§5.1.6)? Which allocation policy (§5.1.5)? Which user- and operator-level performance metrics to use, to compare the scenarios proposed by the capacity planner (§5.1.7)?

The most important decision for our experiments is which scenarios to explore. Each experiment takes in a capacity planning portfolio (see Section 4.3), starts from a base scenario, and aims to extend the portfolio with new candidate scenarios and its results. *The baseline is given by expert datacenter engineers, and has been validated with hardware vendor teams.* Capelin creates new candidates by modifying the base scenario along dimensions such as variety, volume, and velocity of any of the scenario-components. In the following, we experiment systematically with each of these.

5.1.1 Software prototype

We extend the open-source OpenDC simulation platform [37] with capabilities for modeling and simulating the *virtualized workloads* prevalent in modern clouds. We

Table 2
Experiment configurations. The Velocity experiment only appears in the technical report. A legend of topology dimensions is provided below. (Notation: PI = Performance Interference, pub = public cloud trace, pri = private cloud trace.)

		Candidate Topologies				Workloads		Op. Phenomena		
Sec.	Focus	Mode	Quality	Direction	Variance	Trace	Loads	Failures	PI	Alloc. Policy
§5.2	Hor. vs. Ver.	⇄ ⇑	⇲	↔ ↕	▦ ▭	pri	sampled	✓	✓	active-servers
[4]	Velocity	⇄ ⇑	◔	↕	▦ ▭	pri	sampled	✓	✓	active-servers
§5.3	Op. Phen.	–	–	–	–	pri	original	✗ / ✓	✗ / ✓	all
§5.4	Workloads	⇑	⇲ ◔	↔ ↕	▦	pri / pub	sampled	✓	✗	active-servers

Mode	Direction	Quality	Variance
⇄ replace ⇑ expand	↔ horizontal ↕ vertical	⇲ volume ◔ velocity	▦ homogeneous ▭ heterogeneous

model the CPU and memory usage of each VM along with hypervisors deployed on each managed node. Each hypervisor implements a fair-share scheduling model for VMs, granting each VM at least a fair share of the available CPU capacity, but also allowing them to claim idle capacity of other VMs. The scheduler permits *overprovisioning of CPU resources, but not of memory resources*, as is common in industry practice. We also model a workload and resource manager that controls the deployed hypervisors and decides based on configurable allocation policies (described in §5.1.5) to which hypervisor to allocate a submitted VM. Our experiments and workload samples are orchestrated by Capelin, which is written in Kotlin (a modern JVM-based language), and processed and analyzed by a suite of tools based on Python and Apache Spark.

We release our extensions of the open-source OpenDC codebase and the analysis software artifacts on GitHub[1], as part of release 2.0 [47]. We conduct thorough validation and tests of both the core OpenDC and our additions, as detailed in the technical report [4]. Our validation also includes a "replay" experiment, showing how the differing placement of workloads between simulation and original setup has only a minor impact on results.

5.1.2 Execution and Evaluation

Our results are fully reproducible, regardless of the physical host running them. All setups are repeated 32 times. The results, in files amounting to hundreds of GB in size due to the large workload traces involved, are evaluated statistically and verified independently. Factors of randomness (e.g., random sampling, policy decision making if applicable, and performance interference modeling) are seeded with the current repetition to ensure deterministic outcomes, and for fairness are kept consistent across scenarios.

Capelin could be used during capacity planning meetings. A single evaluation takes 1–2 minutes to complete, enabled by many technical optimizations we implemented. The full set of experiments is conveniently parallel, taking around 1 hour and 45 minutes to complete on a "beefy" but standard machine with 64 cores and 128GB RAM; parallelization across machines would reduce this to minutes.

5.1.3 Workload

We experiment with a business-critical workload trace from Solvinity, a *private cloud provider*. The anonymized version of this trace has been published in a public trace archive [36]. We were provided with the full, deanonymized data artifacts of this trace, which consists of more than 1,500 VMs along with information on which physical resources where used to run the trace and which VMs were allocated to which resources. We cannot release these full traces due to confidentiality, but release the summarized results.

The *full trace* includes a range of VM resource-usage measurements, aggregated over 5-minute-intervals over three months. It consumes 3,063 PFLOPs (*exascale*), with the mean CPU utilization on this topology of 5.6%. This low utilization is in line with industry, where utilization levels below 15% are common [62] to reduce the risk of not meeting SLAs.

For all experiments, we consider the full trace, and further generate three other kinds of workloads as samples (fractions) of the original workload. These workloads are sampled from the trace, resulting, in turn, to 306 PFLOPs (0.1 of the full trace), 766 (0.25), and 1,532 (0.5). To *sample*, Capelin randomly takes VMs from the full trace and adds their total load, until the desired load is reached.

For the §5.4 experiment, we further experiment with a public cloud trace from Azure [17]. We use the most recent release of the trace. The formats of the Azure and the Solvinity traces are very similar, indicating a de facto standard has emerged across the private and public cloud communities. One difference in the level of anonymity of the trace requires an additional assumption. Whereas the Solvinity trace expresses CPU load as a frequency (MHz), the Azure trace expresses it as a utilization metric ranging from 0 to the number of cores of that VM. Thus, for the Azure trace, in line with Azure VM types on offer we assume a maximum frequency of 3 GHz and scale each utilization measurement by this value. The Azure trace is also shorter than Solvinity's full trace, so we shorten the latter to Azure's length of 1 month.

We combine for the §5.4 experiment the two traces and investigate possible phenomena arising from their interaction. We disable here performance interference, because we can only derive it for the Solvinity trace (see §5.1.6). To combine the two traces, we first take a random sample of 1% from the (very large) Azure trace, consisting of 26,901 VMs running for one month. We then further sample this 1%-sample, using the same method as for Solvinity's trace.

5.1.4 Datacenter topology

As explained at the start of §5.1, for all experiments we set the topology that ran Solvinity's original workload (the full

1. https://github.com/atlarge-research/opendc

Table 3
Aggregate statistics for both workloads used in this study.

Characterization		Solvinity	Azure
VM submissions per hour	Mean ($\times 10^{-3}$)	31.836	4.547
	CoV	134.605	17.188
VM duration [days]	Mean	20.204	2.495
	CoV	0.378	3.072
CPU load [TFLOPs]	Mean ($\times 10^{2}$)	9.826	64.046
	CoV	2.992	4.654

trace in §5.1.3) as the base scenario's topology. This topology is very common for industry practice. It is a subset of the complete topology of the Solvinity when the full trace was collected, but we cannot release the exact topology or the entire workload of Solvinity due to confidentiality.

From the base scenario, Capelin derives candidate scenarios as follows. First, it creates a temporary topology by choosing half of the clusters in the topology, consisting of average-sized clusters and machines, compared to the overall topology. Second, it varies the temporary topology, in four dimensions: (1) the *mode of operation*: replacement (removing the original half and replacing it with the modified version) and expansion (adding the modified half to the topology and keeping the original version intact); (2) the *modified quality*: volume (number of machines/cores) and velocity (clock speed of the cores); (3) the *direction of modification*: horizontal (more machines with fewer cores each) and vertical (fewer machines with more cores each); and (4) the *kind of variance*: homogeneous (all clusters in the topology-half modified in the same way) and heterogeneous (two thirds in the topology-half being modified in the designated way, the remaining third in the opposite way, on the dimension being investigated in the experiment).

Each dimension is varied to ensure cores and machine counts multiply to (at least) the same total core count as before the change, in the modified part of the topology. For volume changes, we differentiate between a horizontal mode, where machines are given 28 cores (a standard size for machines in current deployments), and vertical modes, where machines are given 128 cores (the largest CPU models we see being commonly deployed in industry). For velocity changes, we differentiate between the clock speed of the base topology and a clock speed that is roughly 25% higher. Because we do not investigate memory-related effects, the total memory capacity is preserved.

Last, due to confidentiality, we can describe the base and derived topologies only in relative terms.

5.1.5 Allocation policies

We consider several policies for the placement of VMs on hypervisors: (1) prioritizing by available memory (`mem`), (2) by available memory per CPU core (`core-mem`), (3) by number of active VMs (`active-servers`), (4) mimicking the original placement data (`replay`), and (5) randomly placing VMs on hosts (`random`). Policies 1-3 are actively used in production datacenters [60].

For each policy we use two variants, following the Worst-Fit strategy (selecting the resource with the *most* available resource of that policy) and the Best-Fit strategy (the inverse, selecting the *least* available, labeled as `-inv` in §5.3).

Table 4
Parameters for the lognormal failure model we use in experiments. We use the normal logarithm of each value.

Parameter [Unit]	Scale	Shape
Inter-arrival time [hour]	24×7	2.801
Duration [minute]	60	60×8
Group size [machine-count]	2	1

5.1.6 Operational phenomena

Each capacity planning scenario can include operational phenomena. In these experiments, we consider two such phenomena, (1) performance variability caused by interference between collocated VMs, and (2) correlated cluster failures. Both are enabled, unless otherwise mentioned.

We assume a common model [43, 61] of performance interference, with a *score* from 0 to 1 for a given set of collocated workloads, with 0 indicating full interference between VMs contending for the same CPU, and 1 indicating non-interfering VMs. We derive the value from the *CPU Ready* fraction of a VM time-slice: the fraction of time a VM is ready to use the CPU but is not able to, due to other VMs occupying it. We mine the placement data of all VMs running on the base topology and collect the set of collocated workloads along with their mean score, defined as the mean CPU ready time fraction subtracted from 1, conditioned by the total host CPU load at that time, rounded to one decimal. At simulation time, this score is then activated if a VMs is collocated with at least one of the others in the recorded set and the total load level on the system is at least the recorded load. The score is then applied to each collocated VMs with probability $1/N$, where N is the number of collocated VMs, by multiplying its requested CPU cycles with the score and granting it this (potentially lower) amount of CPU time.

The second phenomenon we model are cluster failures, which are based on a common model for space-correlated failures [22] where a failure may trigger more failures within a short time span; these failures form a *group*. We consider in this work only hardware failures that crash machines (full-stop failures), with subsequent recovery after some duration. We use a lognormal model with parameters for failure inter-arrival time, group size, and duration, as listed in Table 4. The failure duration is further restricted by a minimum of 15 minutes, since faster recoveries and reboots at the physical level are rare. The choice of parameter values is inspired by GRID'5000 [22] (public trace also available [39]) and Microsoft Philly [40], scaled to Solvinity's topology.

5.1.7 Metrics

In our article, we use the following metrics: (1) the total overcommitted CPU cycles (in MFLOP) of all VMs, defined as the sum of CPU cycles that were requested but not granted, (2) the total power consumption (in Wh) of all machines, using a linear model based on machine load [10], with an idle baseline of 200 W and a maximum power draw of 350 W, and (3) the number of time slices a VM is in a failed state, summed across all VMs. While no SLO metrics are explicitly present in this list, we have two proxies (both Service Level Indicators (SLIs)) from which we can infer violations of such higher-level metrics. Incidents such as overcommitted CPU cycles and failed VM slices directly affect the availability of the service. High prevalence of these

incidents in a short time-span even cause unavailability of services, propagating into SLO metrics that could be formulated here. We analyze a total of 14 metrics and list full results in all metrics in the technical report [4].

5.2 Horizontal vs. Vertical Resource Scaling

Our main findings from this experiment are:

MF1: Capelin enables the exploration of a complex trade-off portfolio of multiple metrics and capacity dimensions.

MF2: Vertically scaled topologies can improve power consumption (median lower by 1.47x-2.04x) but can lead to significant performance penalties (median higher by 1.53x-2.00x) and increased chance of VM failure (median higher by 2.00x-2.71x, which is a high risk!)

MF3: Capelin reveals how correlated failures impact various topologies. Here, 147k–361k VM-slices fail.

The scale-in vs. scale-out decision has historically been a challenge across the field [55][29, §1.2]. We investigate this decision in a portfolio of scenarios centered around horizontally (symbol ↔) vs. vertically (↕) scaled resources (see §5.1.4). We also vary: (1) the decision mode, by replacing the existing infrastructure (🏚) vs. expanding it (🏛), and (2) the kind of variance, homogeneous resources (▦) vs. heterogeneous (▭). On these three dimensions, Capelin creates candidate topologies by *increasing* the volume (⬚) and compares their performance using four workload intensities, two of which are shown in this analysis (we refer to the technical report for the full analysis [4]). We consider three metrics for each scenario: Figure 4 (top) depicts the overcommitted CPU cycles, Figure 4 (middle) depicts the power consumption, and Figure 4 (bottom) depicts the number of failed VM time slices.

Our key *performance* indicator is overcommitted CPU cycles, that is, the count of CPU cycles requested by VMs but not granted, either due to collocated VMs requesting too many resources at once, or due to performance interference effects taking place. We observe in Figure 4 (top) that vertically scaled topologies (symbol ↕) have significantly higher overcommission (lower performance) than their horizontally scaled counterparts (↔, the other three symbols identical). The median value is higher for vertical than for horizontal scaling, for both replaced (🏚) and expanded (🏛) topologies, by a factor of 1.53x–2.00x (calculated as the ratio between medians of different scenarios at full load). This is a large factor, suggesting that vertically scaled topologies are more susceptible to overcommission, and thus lead to higher risk of performance degradation. Among replaced topologies (all combinations including 🏚), the horizontally scaled, homogeneous topology (🏚⬚↔▦) yields the best performance, and in particular the lowest median overcommitted CPU. We also observe that expanded topologies (🏛) have lower overcommission than the base topology, so adding machines is worthwhile. We observe all these effects strongly for the full trace (3,063 PFLOPs), but less pronounced for the lower workload intensity (1,531 PFLOPs).

But performance is not the only criterion for capacity planning. We turn to *power consumption*, as a proxy for analyzing cost and environmental concerns. We see in Figure 4 (middle) that vertically scaled topologies (↕) drastically improve power consumption over horizontal scaling (↔), for median values by a factor of 1.47x–2.04x.

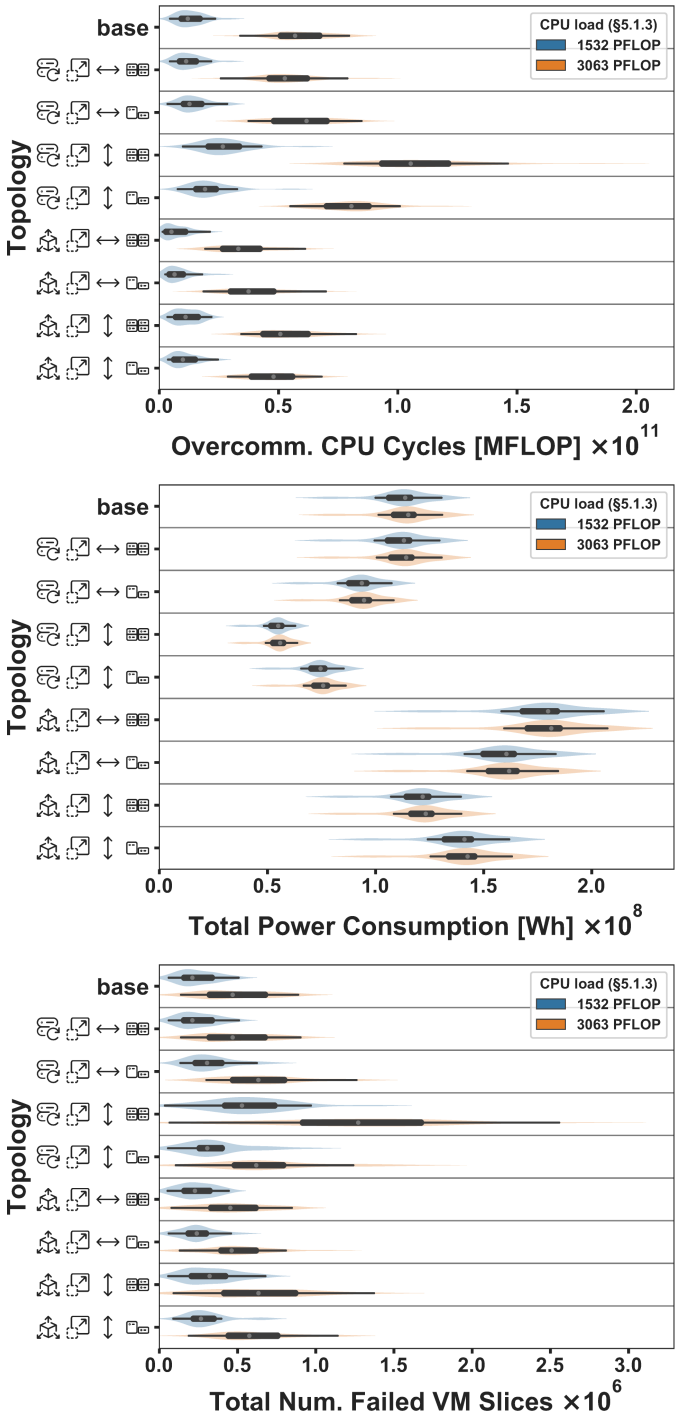

Figure 4. Results for a portfolio of candidate topologies and different workloads(§5.2): (top) overcommitted CPU cycles, (middle) total power consumption, (bottom) total number of time slices in which a VM is in a failed state. Table 2 describes the symbols used to encode the topology.

As expected, all expanded topologies (🏛), which have more machines, incur higher power-consumption than replaced topologies (🏚). Higher workload intensity (i.e., results for the 3,063 PFLOPs over 1,532) incurs higher power consumption, although less pronounced than the aspects considered earlier. The lower magnitude of this effect is consistent with the relatively high power consumption of (close to) idle machines and workload stability (i.e., large bursts are rare).

We also consider the *amount of failed VM time-slices*, in Figure 4 (bottom). Each failure here is full-stop (§5.1.6),

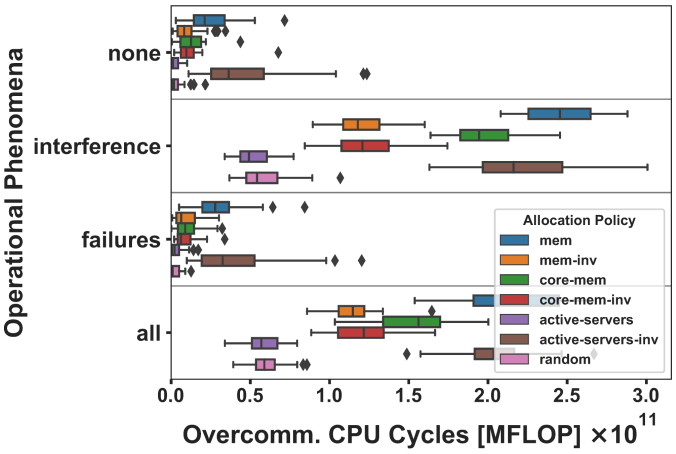

Figure 5. Overcommitted CPU cycles for a portfolio of *operational phenomena* (the "none" through "all" sub-plots), and *allocation policies* (legend), for Experiment 5.3.

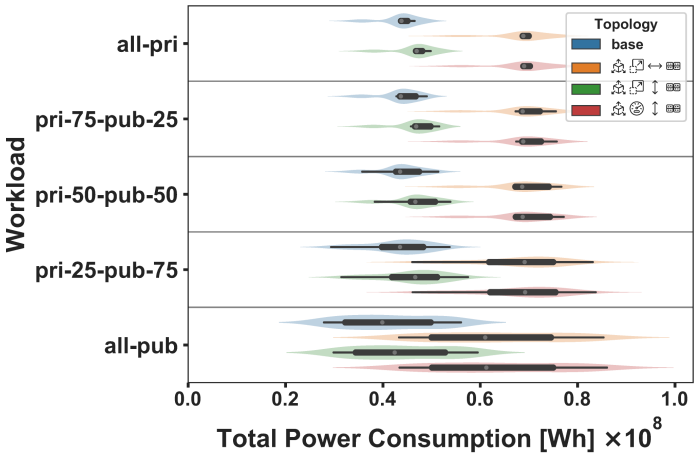

Figure 6. Total power consumption for a portfolio of candidate topologies (legend), subject to *different workloads* (the "all-pri" to "all-pub" sub-plots), for Experiment 5.4.

which typically escalates the alarm to engineers. Thus, this metric should be minimized. We observe significant differences here: the median failure time of a homogeneous vertically scaled topology (↕) is between 2.00x-2.71x higher than the base topology. Qualitatively, this metric shows similarities with the overcommitted CPU cycles. Vertical scaling is correlated with worse performance and with higher failure counts. The effect is less pronounced when making heterogeneous compared to homogeneous procurement. We obtained similar results when scaling velocity (⏲) [4].

Takeaway: Capelin gives practitioners the possibility to *explore a complex trade-off portfolio* of dimensions such as power consumption, performance, failures, workload intensity, etc. Optimization questions surrounding horizontal and vertical scaling can therefore be approached *with a data-driven approach*. We find that decisions including heterogeneous resources can provide meaningful compromises between more generic, homogeneous resources; and lead to different decisions related to personnel training (not shown here). We show significant differences between candidate topologies in all metrics, translating to very different power costs, long-term. We conclude that *Capelin can help test intuitions and support complex decision making*.

5.3 Impact of Operational Phenomena

Our main findings from this experiment are:

MF4: Capelin enables the exploration of diverse allocation policies and operational phenomena, both of which lead to important differences in capacity planning.

MF5: Modeling performance interference can explain 80.6%—94.5% of the overcommitted CPU cycles.

MF6: Different allocation policies lead to different performance interference intensities, and to median overcommitted CPU cycles different by factors between 1.56x and 30.3x compared to the best policy—high risk!

This experiment addresses operational factors in the capacity planning process. We explore the impact of better handling of physical machine failures, the impact of (smarter) scheduler allocation policies, and the impact of (the absence of) performance interference on overall performance. Figure 5 shows the impact of different operational phenomena on performance, for different allocation policies. We observe that performance interference has a strong impact on overcommission, dominating it compared to the "failures" sub-plot, where only failures are considered, or with the "none" sub-plot, where no failures or interference are considered. Depending on the allocation policy, it represents between 80.6% and 94.5% of the overcommission recorded in simulation for the "all" sub-plot, where both failures and interference are considered. We also see the large impact that live resource management (in this case, the allocation policy) can have on Quality of Service. Median ratios vary between 1.56x and 30.3x vs. the best policy, with *active-servers* (see §5.1.5) generally best-performing. While the *random* policy is second-best in terms of performance, its performance is much worse for other metrics, such as power consumption, failures, and maximum number of VMs per machine (see the technical report [4]).

Takeaway: We conclude *Capelin can help model aspects that are important but typically not considered for capacity planning*.

5.4 Impact of a New Workload

Our main findings from this experiment are:

MF7: Capelin enables exploring what-if scenarios that include new workloads as they become available.

MF8: Power consumption can vary significantly more in all-private vs. all-public scenarios, by 4.79x–5.45x.

This experiment explores the impact of adding a new workload type to an existing datacenter. This appears often in practice, e.g., for new customers. We combine here the 1-month Solvinity and Azure traces (see §5.1.3).

Figure 6 shows the power consumption for different combinations of both workloads and different topologies. (The technical report [4] analyzes more metrics.) We observe the unbiased variance of results [18, p. 32] is positively correlated with the fraction of the workload taken from the public cloud (Azure). Depending on topology, the variance increase with this fraction ranges from 4.78x to 5.45x. Expanding the volume horizontally (↔) leads to the lowest increase in variance. The workload statistics listed in Table 3 show that the Azure trace has far fewer VMs, with higher load per VM and shorter duration, thus explaining the increased variance. Last, all candidate topologies have a higher power consumption than the base topology.

Takeaway: We conclude *Capelin can support new workloads as they appear*, even before they are physically deployed.

6 Threats to Validity

We identify three main sources of threats to validity for this work: (1) the interviews, (2) the experiments, and (3) the simulator. We list the most important threats here and provide more details in Sec. 6 and 7 of the technical report [4].

Related to (1), we see two points of concern for the *validity of the interview study*. First, confidentiality limits us from sharing source transcripts, but we include detailed analysis results [4]. Second, we recognize the limited sample size of our study, which follow-up studies could address.

Related to (2), we see three points of concern for the *validity of the experimental study*. First, the diversity of modeled resources is limited, but these other resources would have limited impact on CPU contention. Second, the availability of input data from our experiments is limited due to confidentiality; however, similar (anonymized) datasets exist. Third, the choice of allocation policies may influence the validity of the results; we address this by comparing to industry practice.

Related to (3), we see three points of concern for the *validity of the simulator*. First, the simulator's outputs need to be valid, which we ensure by inspecting a wide variety of metrics. Second, the simulator's outputs need to be sound, which we address by a "replay experiment" mimicking real-world operation and by meeting with experts to discuss our assumptions and results. Third, the simulator's outputs need to be reliable, which we ensure by employing snapshot testing and industry-standard development practices.

7 Related Work

We summarize in this section the most closely related work, which we identified through a survey of the field that yielded over 75 relevant references. For further details, see the technical report [4]. Overall, our work is the first to: (1) conduct community interviews with capacity planning practitioners managing cloud infrastructures, which resulted in unique insights and requirements, (2) design and evaluate a data-driven, comprehensive approach to cloud capacity planning, which models real-world operational phenomena and provides, through simulation, multiple VM-level metrics as support to capacity planning decisions.

Related to (1), we see two works as closely related to our *interview study* of practitioners. In the late-1980s, Lam et al. conducted a written questionnaire survey [45] and, mid-2010s, Bauer et al. conducted semi-structured interviews [7]. The target group of these studies differs from ours, however, since both focus on practitioners from different industries planning the resources used by their IT department.

Related to (2), our work extends a three-fold body of related work. First, we survey *process models for capacity planning*. To enable their comparison, we unify the terminology and the stages proposed by these models, and create the super-set of systems-related stages summarized in Table 5. From a systems perspective, Capelin proposes a comprehensive process. Second, we survey systematically the main scientific repositories and collect 57 *works related to long-term capacity planning*. Our scope of long-term planning (procurement) excludes more dynamic, short-term process such as Google's Auxon [33] or the Cloud Capacity Manager [42], which address the live management of capacity *already procured*; explained differently, Capelin (this work) helps decide on long-term capacity procurement, whereas Auxon and others like focus on the different problems of what to do with that capacity, short-term, once it is already there. Other work investigates the dynamic management of physical components, such as CPU frequency scaling [46]. We find that the majority of studies only consider one resource dimension, and four inputs or less for their capacity planning model. Few are simulation-based [1, 14, 49, 52, 53, 54], with the rest using primarily analytical models. The notable Janus [1] focuses only on datacenter networks. Janus operates in a design space adjacent to Capelin: it considers network topologies, network traffic patterns, topology changes, and evaluation metrics, all to derive network topology change plans. Considering this parallel structure of plans, if Janus is open-sourced, it could likely be integrated into Capelin on two levels: (1) to provide feedback on the network dimension of simulated scenarios, and (2) to optimize the network topology, during scenario generation. Third and last, we survey *system-level simulators*, and study 10 of the best-known in the large-scale distributed systems community. Among the simulators that support VMs [13, 32, 51] and could be useful for simulating cloud datacenters, few have been tested with traces at the scale of this study, few support CPU over-commissioning, none support both operational phenomena (§5.1.6), and none can output detailed VM-level metrics used in this work.

Table 5
Comparison of process models for capacity planning. Sources: Lam [45, p. 92], Howard [34] (referenced by Browning [12, p. 7]), Menascé [48, p. 179], Gunther [28, p. 22], and Kejariwal [41, p. 4].

Stage	[45]	[12]	[48]	[28]	[41]	Capelin
Assessing current cap.	✓		✓	✓	✓	✓
Identifying all workloads		✓				✓
Characterize workloads	✓	✓	✓	✓		✓
Aggregate workloads		✓				✓
Validate workload char.			✓			✓
Determine resource req.		✓				✓
Predict workload	✓		✓		✓	✓
Characterize perf.	✓		✓			✓
Validate perf. char.	✓		✓			✓
Predict perf.	✓		✓			✓
Characterize cost			✓			✓
Predict cost			✓			✓
Analyze cost and perf.			✓			✓
Examine what-if scen.	✓					✓
Design system				✓		✓
Iterate and calibrate					✓	✓

8 Conclusion and Future Work

Accurately planning cloud datacenter capacity is critical to meeting the needs of the 2020s society whilst saving costs and ensuring environmental sustainability. However, the current practice has not been analyzed in decades and publicly available tools to support practitioners are scarce. Capelin, a data-driven, scenario-based alternative to current planning approaches, addresses these problems.

In this work, we have designed, implemented, and evaluated Capelin. We have conducted a guided interview with diverse practitioners from a variety of backgrounds, whose results led us to synthesize five functional requirements. We have designed Capelin to meet them, including the ability to model datacenter topologies and virtualized workloads, to

express what-if scenarios and QoS requirements, to suggest scenarios to evaluate, and to evaluate and explain capacity plans. Capelin uses a novel abstraction, the capacity planning portfolio, to represent, explore, and compare scenarios. Experiments based on real-world workload traces collected from private and public clouds demonstrate Capelin's capabilities. Results show that Capelin can support capacity planning processes, exploring changes from a baseline scenario alongside four dimensions. We found that capacity plans common in practice could potentially lead to significant performance degradation. We also gave evidence of the important, but often discounted, impact of operational choices (e.g., the allocation policy) and operational phenomena (e.g., performance interference).

We have released Capelin as FOSS for capacity planners to use. We are investigating the use of AI/ML search techniques to make the Capacity Plan Generator component more capable of exploring the enormous design-space.

ACKNOWLEDGMENTS

Funded by NWO Vidi MagnaData and TOP OffSense.

REFERENCES

[1] Alipourfard et al. Risk based planning of network changes in evolving data centers. In *SOSP*, 2019.
[2] Amvrosiadis et al. On the diversity of cluster workloads and its impact on research results. In *ATC*, 2018.
[3] Andreadis et al. A reference architecture for datacenter scheduling: Design, validation, experiments. *SC*, 2018.
[4] Georgios Andreadis et al. Capelin: Data-Driven Capacity Procurement for Cloud Datacenters using Portfolios of Scenarios – Extended Technical Report, 2021. URL https://arxiv.org/abs/2103.02060.
[5] Barroso et al. *The Datacenter as a Computer: Designing Warehouse-Scale Machines*. Synthesis lectures on comp. arch. Morgan and Claypool, 2018. 3rd ed.
[6] Baset et al. Towards an understanding of oversubscription in cloud. *USENIX HOT-ICE*, 2012.
[7] Bauer et al. Latent effects of cloud computing on IT capacity management structures. *IJCCE*, 6(2), 2017.
[8] Beyer et al. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016.
[9] Birke et al. Failure analysis of virtual and physical machines: Patterns, causes, characteristics. *IFIP*, 2014.
[10] Blackburn et al. Five ways to reduce data center server power consumption. *The Green Grid*, 42:12, 2008.
[11] Bolze et al. Grid'5000: A large scale and highly reconfigurable experimental grid testbed. *IJHPCA*, 20(4).
[12] Tim Browning. *Capacity Planning for Computer Systems*. Academic Press, 1994.
[13] Calheiros et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.*, 41(1), 2011.
[14] Carvalho et al. Capacity planning for IaaS cloud providers offering multiple service classes. *FGCS*, 77.
[15] Mark Chamness. Capacity forecasting in a backup storage environment. *USENIX LISA*, 2011.
[16] Coleman et al. Using grounded theory to understand software process improvement: A study of Irish software product companies. *Inf. and Sw. Tech.*, 49(6), 2007.
[17] Cortez et al. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *SOSP*, 2017.
[18] Devore. *Probability and Statistics for Engineering and the Sciences, 7Ed*. Brooks Cole Cengage Learning, 2009.
[19] Duplyakin et al. The design and operation of CloudLab. *ATC*, 2019.
[20] El-Sayed et al. Learning from failure across multiple clusters. In *ICDCS*, 2017.
[21] Flexera. State of the Cloud Report, Sep 2020.
[22] Gallet et al. A model for space-correlated failures in large-scale distributed systems. In *Euro-Par*, 2010.
[23] Gartner Inc. Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17% in 2020, Sep 2019.
[24] Frank Gens. Worldwide and Regional Public IT Cloud Services 2019–2023 Forecast. Tech. Rep. by IDC, Doc. #US44202119, Aug 2019.
[25] Ghosh et al. Stochastic model driven capacity planning for an infrastructure-as-a-service cloud. *IEEE Transactions on Services Computing*, 7(4), 2014.
[26] James Glanz. Data centers waste vast amounts of energy, belying industry image. *N.Y. Times*, 2012.
[27] Greenberg et al. The cost of a cloud: research problems in data center networks. *ACM CCR*, 39(1), 2009.
[28] Gunther. *Guerrilla Capacity Planning*. Springer, 2007.
[29] Mor Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.
[30] Herbst et al. Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges. *ACM TOMPECS*, 3(4), 2018.
[31] Hewlett-Packard Development Company. HP Capacity Advisor Version 7.4, 2014.
[32] Hirofuchi et al. Simgrid VM: virtual machine support for a simulation framework of distributed systems. *IEEE Trans. Cloud Computing*, 6(1), 2018.
[33] Hixson et al. Capacity Planning. *USENIX ;login*, 40(1), 2015.
[34] Howard. IS Capacity Management Handbook Series–Volume 1–Capacity Planning. *Institute for Computer Capacity Management*, 1992.
[35] International Business Machines Corporation. IBM Z Performance and Capacity Analytics tool, 2019.
[36] Iosup et al. The Grid Workloads Archive. *FGCS*, 24(7).
[37] Iosup et al. The OpenDC vision: Towards collaborative datacenter simulation and exploration for everybody. *ISPDC*, 2017.
[38] Iosup et al. Massivizing computer systems: A vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems. *ICDCS*, 2018.
[39] Javadi et al. The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems. *JPDC*, 73(8), 2013.
[40] Jeon et al. Analysis of large-scale multi-tenant GPU clusters for DNN training workloads. In *ATC*, 2019.
[41] Kejariwal et al. *The Art of Capacity Planning: Scaling Web Resources in the Cloud*. O'Reilly, 2017.
[42] Mukil Kesavan et al. Practical compute capacity management for virtualized datacenters. *IEEE TCC*, 1(1): 1–1, 2013.

[43] Koh et al. An analysis of performance interference effects in virtual environments. In *ISPASS*, 2007.
[44] Krebs et al. Metrics and techniques for quantifying performance isolation in cloud environments. *Science of Computer Programming*, 2014.
[45] Lam et al. *Computer capacity planning: theory and practice*. Academic Press, 1987.
[46] Drazen Lucanin et al. Performance-based pricing in multi-core geo-distributed cloud computing. *IEEE TCC*, 2016.
[47] Mastenbroek et al. OpenDC 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters. In *CCGRID*, 2021.
[48] Menascé et al. *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall, 2001.
[49] Mylavarapu et al. An optimized capacity planning approach for virtual infrastructure exhibiting stochastic workload. *SAC*, 2010.
[50] Nazareth et al. Capacity management for cloud computing: a system dynamics approach. *AMCIS*, 2017.
[51] Nuñez et al. iCanCloud: A flexible and scalable cloud infrastructure simulator. *J. Grid Comput.*, 10(1), 2012.
[52] Ostberg et al. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. *EuCNC*, 2017.
[53] Patel et al. Knowledge based data center capacity reduction using sensitivity analysis on causal Bayesian belief network. *IKSM*, 12(2), 2013.
[54] Rolia et al. A capacity management service for resource pools. *WOSP*, 2005.
[55] Semih et al. Response to "scale up or scale out for graph processing". *IEEE Internet Comput.*, 22(5), 2018.
[56] Shen et al. CloudScale: Elastic resource scaling for multi-tenant cloud systems. *SoCC*, 2011.
[57] Shen et al. Statistical characterization of business-critical workloads hosted in cloud datacenters. *CCGrid*, 2015.
[58] Tang et al. Joint pricing and capacity planning in the IaaS cloud market. *IEEE TCC*, 5(1), 2017.
[59] Daniel W Turner. Qualitative Interview Design: A Practical Guide for Novice Investigators. *Qualitative Report*, 15(3):7, 2010.
[60] van Beek et al. Self-Expressive Management of Business-Critical Workloads in Virtualized Datacenters. *IEEE Computer*, 48(7):46–54, 2015.
[61] van Beek et al. A CPU contention predictor for business-critical workloads in cloud datacenters. In *HotCloudPerf*, 2019.
[62] Vasan et al. Worth their watts? - an empirical study of datacenter servers. *HPCA*, 2010.
[63] VMware. VMware Capacity Planner, 2009.
[64] Xu et al. Optimal pricing and capacity planning of a new economy cloud computing service class. *ICCAC*, 2015.
[65] Zhai et al. An auditing language for preventing correlated failures in the cloud. *PACMPL*, 2017.
[66] Zhang et al. R-capriccio: A capacity planning and anomaly detection tool for enterprise services with live workloads. *Middleware*, 2007.
[67] Zhang et al. An optimal capacity planning algorithm for provisioning cluster-based failure-resilient composite services. *SCC*, 2009.

Georgios Andreadis received his B.Sc. and M.Sc. degrees in computer science from Delft University of Technology, the Netherlands. For his M.Sc. thesis, he has worked closely with industry leader Solvinity and with a research group at Vrije Universiteit Amsterdam, both the Netherlands. He has received the Young Talent Incentive Award for Informatics during his Honours B.Sc. programme and was named Best Graduate of the university for his research on capacity planning. Currently, he is a Ph.D. candidate at the Radiotherapy department of the Leiden University Medical Center and at the Life Sciences and Health group of the Centrum Wiskunde & Informatica, both the Netherlands. His research interests include cloud computing, evolutionary computation, and medical applications of multi-objective optimization techniques. Contact: info@gandreadis.com.

Fabian Mastenbroek received his B.Sc. degree (2019) in computer science from Delft University of Technology, the Netherlands. As part of his Honours B.Sc. programme, he became involved in the development of OpenDC, an open-source platform for datacenter modeling and simulation. He is currently pursuing a M.Sc. degree computer science at Delft University of Technology and leads the development of OpenDC. Topics of interest include cloud computing, resource management and scheduling, simulation and optimization techniques. Contact: fabianishere@outlook.com.

Vincent van Beek is team lead and senior software engineer at cloud specialist Solvinity, the Netherlands. He received his B.Sc. and M.Sc. degrees in computer science from the Delft University of Technology, the Netherlands, and is currently a Ph.D. candidate at the same institution. His research interests include datacenter management, planning, and scheduling, and their application to customer-facing, business-critical workloads.
Contact: vincent.vanbeek@solvinity.com.

Alexandru Iosup is tenured full Professor and University Research Chair with the Vrije Universiteit Amsterdam, the Netherlands. He is also Chair of the SPEC Research Cloud Group. He received a Ph.D. in computer science from TU Delft, the Netherlands (2009). He was awarded the yearly Netherlands Prize for Research in Computer Science (2016), the yearly Netherlands Teacher of the Year (2015), and several SPECtacular awards (2012-2017). His research interests are in massivizing computer systems, that is, making computer systems combine desirable properties such as elasticity, performance, and availability, yet maintain their ability to operate efficiently in controlled ecosystems. Topics include cloud computing and big data, with applications in big science, big business, online gaming, and (upcoming) massivized education. Contact: a.iosup@vu.nl.