Portfolio Scheduling for Managing Operational and Disaster-Recovery Risks in Virtualized Datacenters Hosting Business-Critical Workloads

Vincent van Beek
Delft University of Technology
vincent.vanbeek@solvinity.com

Giorgos Oikonomou
Delft University of Technology

Alexandru Iosup
VU Amsterdam
a.iosup@vu.nl

***Abstract*—Cloud datacenters are increasingly hosting business workloads. Such long-running, on-demand workloads raise important challenges in datacenter operation, requiring efficient online scheduling of workloads with unprecedented characteristics under strict service level agreements (SLAs). In this work, we propose an approach to manage the risk of not meeting SLAs. Our approach is based on portfolio scheduling, which is an online scheduling technique that dynamically selects a scheduling algorithm from a set (portfolio), subject to a possibly changing utility function. Ours is the first datacenter-scheduling approach to consider operational and disaster-recovery risks. Using trace-based simulation with traces collected from a commercial multi-datacenter environment, we give evidence that portfolio scheduling is able to mitigate risks significantly better than its constituent scheduling algorithms and better than datacenter engineers.**

***Index Terms*—Portfolio Scheduling, Datacenter Resource Management, Risk Management, Risk Tolerance, Operational Risk, Disaster Recoverability Risk.**

I. Introduction

Attracted by the dual promise of infrastructure efficiency [1] and widespread uptake [2], large organizations are increasingly using public- and/or private-cloud resources to run their large-scale *business-critical workloads* (BCWs). Although the promises are enticing, hosting BCWs is relatively new, raising many resource management and scheduling challenges. Particularly challenging is enforcing for such workloads *risk-aware* service-level agreements (SLAs). Traditional datacenters focus on best-effort execution of workloads [3], [4], which may not be suitable for risk-aware SLAs and thus could lead to high financial penalties for cloud operators if SLAs are violated. In contrast, in this work we address the research question of *How to manage the risk of not meeting SLAs for datacenters hosting BCWs?*

Managing BCWs means adapting to new workload characteristics. According to Shen et al. [5], BCWs are significantly different from the workloads previously addressed by the datacenter research community, and in particular from the computationally intensive parallel and grid workloads, and from the analytics workloads common at large web-scale companies (e.g., Facebook and Google). One major difference is that BCWs are expressed as *requests for (non-transparent) VMs*, instead of explicit queries or requests to execute a specific application, mainly because for datacenter customers details about their software are too sensitive to reveal.

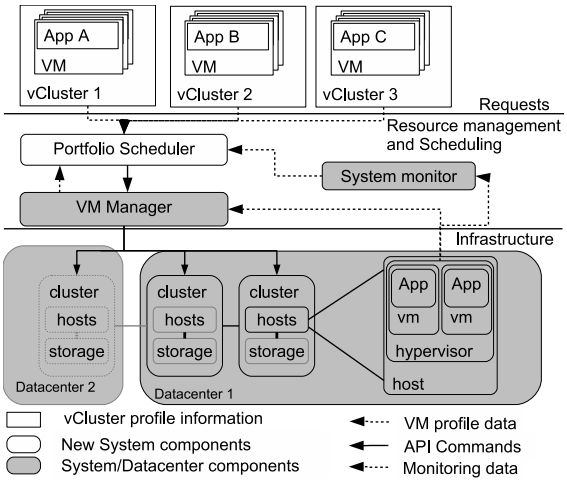


Fig. 1: Virtualized datacenters hosting BCWs.

The operational model emerging in datacenters *hosting* BCWs focuses on *long-running virtual machines (VMs)*—instead of single parallel jobs or many short-lived user-requests to the cloud operator. This means a shift to high reliability and availability of resources and high resource utilization, from the traditional focus on short job runtimes. We discuss this operational model in Section II.

There are many types of risks in hosting BCWs in datacenters: *operational risks* of requested resources not performing as promised, *reliability risks* of resources failing, etc. These risks affect many *datacenter stakeholders*: DevOps engineers, reliability engineers, infrastructure engineers, legal representatives, capacity planners and business managers, etc.

Meeting complex SLAs in datacenters requires monitoring and reasoning about diverse operational metrics, to an extent that exceeds since at least the early 2010s the capabilities of unassisted engineers [6]. Although many automated scheduling approaches exist, over the past two decades numerous studies [3], [4] have shown that datacenter schedulers are brittle—different schedulers will exhibit different periods of poor performance that in turn lead to unexpected performance issues, unnecessary resource overload, and even to (cascading) failures. Instead of trying to develop scheduling policies capable to address all possible workloads, which is error-prone

and ephemeral [7], it may be possible to dynamically select active scheduling policies from a pool of available schedulers.

Selecting the active scheduler automatically, while meeting the SLAs of BCWs and in particular addressing risk, is the focus of this work. Current approaches in practice leverage human expertise [4], [6], but lack online automation. Dynamic selection of schedulers can be done by a meta-scheduler, such as a portfolio scheduler [8], [7], but currently risk is not supported by published meta-schedulers. Addressing the research question, in this work we extend the state-of-the-art in portfolio scheduling for datacenters with explicit support for risk management. Our main contribution is three-fold:

1) We extend with support for two types of risk the state-of-the-art in datacenter portfolio scheduling (Section III).
2) We design two heuristics-based risk-aware scheduling policies (Section IV). The two new policies address risk-aware utility functions related to operational and disaster-recovery risks.
3) Through trace-based simulation, we provide new insight into the operation of portfolio scheduling in datacenters (Section VI). The experimental setup uses traces collected from real datacenters (Section V), enabling the comparison of the portfolio scheduler with both its constituent policies and with the performance of the real-world engineers in a commercial datacenter operator.

II. System Model for Virtualized Datacenters Hosting Business-Critical Workloads

We use in this work the system model for virtualized datacenters hosting BCWs introduced by van Beek et al. [7]. We describe the datacenter and the workload models, in turn.

A. Virtualized Datacenters Architecture

The typical operation of this architecture is depicted in Figure 1. In the top row (the section labeled "Requests"), workloads arrive in the system as requests for hosting, one or several VMs, grouped as *vClusters*. Datacenter customers define their own vClusters, including detailed resource specifications (e.g., memory in GB, amount of CPU cores, storage space in GB), affinities between VMs (e.g., that one VM is not allowed to be in the same datacenter as another).

Depicted in the bottom row, the physical "Infrastructure" consists of a collection of inter-connected *datacenters* that use commercial off-the-shelf virtualization technology, such as hypervisors on each node, to host the VMs of BCWs. Each datacenter consists of several *clusters* of *physical machines* that can host VMs, and a cluster-wide parallel storage system. A high-speed network operates inside each datacenter; we use in this work an all-to-all Infiniband model, but a Clos topology could also be used in our model without changes.

In the middle row, core resource management and scheduling components respond to requests. The system-wide *VM manager* orchestrates the placement of VMs on the physical machines. The *System monitor* collects the status of each customer request and system component, at run-time. A non-traditional component is *the system meta-scheduler—the new Portfolio Scheduler component*.

B. Business-Critical Workloads

We focus in this work on large-scale BCWs, which have the following reported characteristics [5]:

1) Requests arrive in the system not as specifications of jobs, but as specifications of VMs that will run jobs (the vClusters introduced in Section II-A).
2) Long run-times—VMs often run for months or years. In contrast, scientific workloads consist mainly out of shorter-lived jobs that run for only hours or days.
3) Consequence of point 2, reducing VM-runtime is not a performance goal here, unlike traditional workloads.
4) Most VMs are small relative to parallel workload jobs—60% of the VMs use less than 4 cores and 8 GB.
5) Most VMs have very low resource utilization compared to scientific workloads, that is, over 50% of the VMs have an average utilization of under 10%.

BCWs have additional important properties:

P1. VMs of BCWs can be *consolidated* on the same physical resource, which is a consequence of points 4 and 5 that contrasts with traditional parallel and grid operation. We define *consolidation* as the number of VMs sharing the resources of a the same set of physical machines. Datacenter operators have to limit consolidation, such that VMs can perform according to SLAs. For example, rarely more than 10 VMs can be consolidated on the same physical machine.

P2. The operational risk tolerance is low. Even a small fraction of under-performing VMs quickly leads to an escalated request for engineering time, until the crisis is resolved. This is unscalable and costly, and can cause bad publicity.

P3. The risk tolerance for (near-)full-datacenter outages is very low. Disaster-recovery is critical: an hour of downtime cascading across many VMs will incur high financial penalties and damage datacenter reputation. Instead, the cloud operator should recover from disasters quickly, by having other datacenters absorb the workload of the failed datacenter. (The risk tolerance for single-VM failures remains high.)

III. Portfolio Scheduling for Managing Operational and Disaster-Recovery Risks

To reduce operational and disaster-recover risks in datacenters, in this section we design a *risk-aware* portfolio scheduler that selects dynamically the schedulers guiding the placement of VMs in a (multi-)datacenter environment. We describe the requirements, our risk-aware portfolio scheduling architecture, and three risk-aware utility functions, in turn.

Portfolio scheduling is the result of the intuition that it is very difficult or even impossible [3] to create a single scheduler that will always perform well in the modern, dynamic datacenter. Instead, we posit it is simpler to use a set of schedulers that work well on a subset of workloads and to select the right scheduler dynamically. We extend the portfolio scheduling model described in [8], [7].

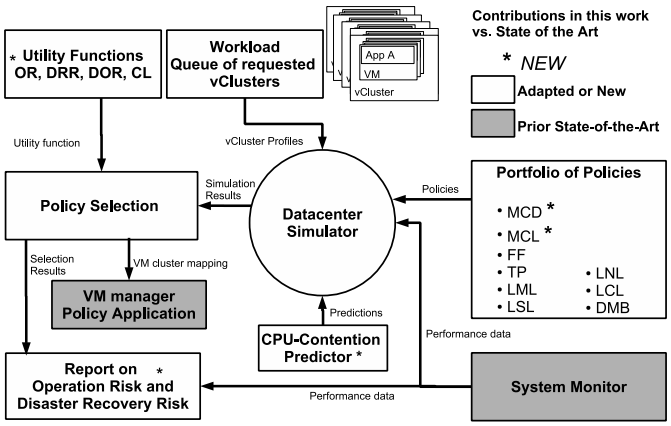


Fig. 2: Architecture for *risk-aware* portfolio scheduling.

A. Requirements When Hosting Business-Critical Workloads

(R1) Mitigate Operational Risk: The placement of VMs must take into account SLAs regarding service perfomance. The *Operational Risk* is the risk of not achieving service-perfomance SLAs. In this work, we use the VM CPU-performance as the service-performance indicator (see Section III-C).

(R2) Guarantee Disaster Recovery: The placement of VMs must ensure that workload from a failed datacenter can be absorbed by the surviving datacenters. The *Disaster-Recovery Risk* is the risk of not achieving this type of SLA (see Section III-C).

(R3) Balance Multiple Optimization Objectives: the scheduler must be able to balance multiple SLAs with cloud operator's cost-related goals. In this work, we consider the cost-related goal of high utilization of resources through consolidation, balanced with the SLA objectives (R1 and R2).

(R4) Adapt Dynamically to New Workloads: The scheduler must adapt dynamically to new or changing workloads, which are common in datacenters that host BCWs. The scheduler should additionally support changing requirements by the datacenter operators (e.g., new optimization goals).

(R5) Explain decisions to human operators: The scheduler must *explain* scheduling decisions to datacenter operators, so that they can understand *why* specific policies were selected. Without this requirement, the portfolio scheduler proposed in this work will face the lack of adoption encountered by many other scheduling approaches [4].

B. Risk-Aware Portfolio-Scheduling Architecture

We propose in this section an architecture for risk-aware portfolio scheduling. The architecture extends the state-of-the-art in portfolio scheduling for datacenters [8], [7] by adding support for risk-awareness in each of the traditional stages of portfolio scheduling, that is, configuration, selection, application, and reflection. Specifically, our new architecture uses new risk-aware utility functions, new risk-aware scheduling policies (Section IV), and a new method to simulate the datacenter that considers CPU-contention accurately.

Figure 2 gives an overview of the architecture for risk-aware portfolio scheduling. The architecture operates the portfolio scheduler as a *multi-stage cycle*. As input, the architecture receives dynamically client-requests for vClusters (see Section II-A), which are queued for execution (box labeled "Workload" at the top of the figure). We describe in the following the components in the figure that form the multi-stage cycle of the portfolio scheduler.

Utility Functions represent the optimization requirements given by datacenter stakeholders (see Section I). Our architecture is the first to support *risk-aware* utility functions, such as operational risk (R1) and disaster-recovery risk (R2). To this end, Section III-C introduces three risk-aware utility functions ("OR", "DRR", and "DOR" in Figure 2).

Similarly to the creation of utility functions, new in this work we allow the **Portfolio of Policies** to consider not only traditional but also *risk-aware* scheduling policies. We introduce in Section IV two new scheduling policies that respond to requirement (R3).

Addressing (R4), the **Datacenter Simulator**, the **CPU-Contention Predictor**, and the **Policy Selection** components all focus on selecting policies from the portfolio. The Datacenter Simulator uses simulation to estimate, independently for each policy in the portfolio, the performance and risk score of each policy according to the utility function in use. Each simulation receives as input live datacenter-performance data, from the **Performance Monitor** component.

A key issue is the ability of the simulator to cope with virtualized environments, in particular to estimate performance degradation due to consolidation; we address this issue with a CPU-Contention predictor [9]. Finally, Policy Selection includes mechanisms to select a policy and the metrics used by the mechanisms, based on utility functions and on the output of the simulation process; for example, this component could be configured by the datacenter operator to select the policy that minimizes the balanced DOR risk-aware utility function. Additionally, this component also provides data used later to explain the decisions (R5).

The **VM manager [for] Policy Application** applies the policy selected in the previous stage, and integrates our scheduler with the existing management tooling of the datacenter infrastructure. The **Report** component is used by datacenter stakeholders, in particular engineers, for observing the operation of the portfolio scheduler, understanding it (R5), and possibly also tuning it. Section VI demonstrates using Reporting over time, for real-world workloads.

Portfolio schedulers can also include a **self-reflection** stage, which we do not extend or adapt in this work, and thus do not include in Figure 2. This stage uses monitoring data and possibly also guidelines from datacenter stakeholders, to tune the other stages dynamically. For example, in the work of Deng et al. [8], this stage dynamically filters the policies included in the portfolio, to keep the number of policies sufficiently

low so that the selection step can complete within a matter of minutes even for tens of policies, thousands of queued jobs, and thousands of resources. An exploration of this stage falls outside the focus on risk of this article.

C. Risk-aware Utility Functions

To address requirements R1, R2, and R3, we use three risk-aware utility functions designed by the authors of this work in collaboration with the SPEC RG Cloud Group [10]. The Operational Risk utility function addresses the risk of resource contention due to resource oversubscription. The Disaster-Recovery Risk utility function expresses the risk of not being able to recover from a complete datacenter failure. The SLAs with regard to availability dictate that the non-failed datacenters should be able to absorb the workloads from the failed datacenter. Finally, responding to the requirement for managing multiple risks at the same time, we use DOR, an utility function that combines the Operational Risk and Disaster Recovery Risk. We describe each utility function in the following, in turn.

Operational Risk (OR): Responding to R1, we define OR as the risk that VMs will not receive the requested amount of resources. In this work, we consider explicitly *CPU* resources; more resources can also be considered [11]. In Equation 1, we define OR (r_o) as the proportion of resource demands (D_t) that can be met by the used resources (U_t), over a period time (T). The values give an intuition of the severity of performance degradation that VMs may experience, with lower values interpreted as better.

$$r_o = \frac{1}{T} \int_T \frac{D_t - U_t}{D_t} dt \in (0, 1] \tag{1}$$

Disaster Recovery Risk (DRR): Requirement R2 describes the need for guaranteeing the ability to absorb full-datacenter failures when multiple (n) datacenters are present. DRR presents the risk for not meeting this requirement, by expressing to what extend the remaining datacenters can absorb the workload of the failed datacenter. In Equation 2, we define DRR of a datacenter in a multi-datacenter infrastructure as r_{d_i}, where the workload in datacenter i is W_i, and $E_i^{\complement}$ is the complement empty-space (that is, the empty space in all datacenters excluding datacenter i). In Equation 3, we use the geometric mean to combine the *normalized* score of individual datacenters into a single value of *DRR for an entire multi-datacenter system* (r_d).

$$r_{d_i} = \begin{cases} \frac{W_i - E_i^{\complement}}{E_i^{\complement}} & , \quad W_i \leq E_i^{\complement} \\ \frac{W_i - E_i^{\complement}}{W_i} & , \quad W_i > E_i^{\complement} \end{cases} \quad \in [-1, 1] \qquad i \in [1, \ldots, n] \tag{2}$$

$$r_d = \sqrt[n]{\prod_i \frac{r_{d_i} + 1}{2}} \quad \in [0, 1] \qquad i \in [1, \ldots, n] \tag{3}$$

In this work, we consider *memory* as the dominant resource when computing W_i and E_i. When hosting BCWs, it is common to guarantee the reservation of memory resources, which means that this resource is not over-provisioned and instead is guaranteed per-VM. Thus, new memory demands cannot be satisfied if the remaining memory-space is insufficient.

Disaster–Operational Risk (DOR) Combined: Requirement R3 emphasizes the need for complex utility functions, that can be used to construct practical SLAs. We posit that a combined metric is both necessary and useful. Datacenter operators need to continuously solve the complex problem of reducing multiple risks while meeting other datacenter optimization goals, such as operational cost. Although using different utility functions to assess the status of each risk and optimization goal provides detailed information for solving the complex problem, the sheer volume of this information makes it impractical for human consumption. Instead, to facilitate discussing system status and performance, datacenter stakeholders often require that only a few or even a single value represents the current risk level of the datacenters. Even for experts such as DevOps engineers and site reliability engineers, a single value simplifies an already complex monitoring process, especially useful when immediate action is required to improve risk levels.

Addressing R3, in this work we design the DOR risk, which *combines* the utility functions for both OR and DRR. In Equation 4, we define the combined utility function, r_{od}, as a weighted average of the two risk metrics, where w_o and w_d are the weights, respectively. We analyze the impact of these weights on the risk exhibited by the system, in Section VI-C.

$$r_{od} = \frac{w_o \cdot r_o + w_d \cdot r_d}{w_o + w_d} \in [0, 1] \tag{4}$$

IV. Portfolio Creation: New Scheduling Policies

The portfolio scheduler requires for its operation the existence of a portfolio (set) of scheduling policies. Inspired by the risk-aware utility functions, in this section we design two new scheduling policies for placing the VMs of vClusters to physical datacenter clusters. As customary for this level of scheduling, we assume that a cluster-level scheduler (e.g., VMware's DRS, Condor, and Mesos) will take the placement request and actually run the VM on the physical machines of the cluster. Besides these policies, the portfolio scheduler can be equipped with many other policies; as described in Section V-B, we include seven more policies in our experiments.

The two policies we design aim to balance the contention on CPU resources, and to maximize the consolidation of VMs on physical hosts, respectively. Reflecting R3, these policies align with the SLAs (e.g., service performance) and business goals (e.g., minimizing the physical footprint and cost) that datacenter operators face when hosting BCWs.

For simplicity, we present in this section single-VM versions of our policies. We have implemented and used in Section VI vCluster (multi-VM) versions, which derive trivially from the single-VM versions presented here.

Algorithm 1 Mean Contention Duration (MCD)

Input: V a new Virtual Machine, C all the Clusters;
1: $P = []$ (contention per cluster)
2: **for** $c \in C$ (all clusters) **do**
3: $\epsilon = 0$ (the contention)
4: **for** $v \in c$ (all VMs in cluster c) **do**
5: $p = (\Sigma D_v)/|D_v|$ (mean contention duration for v)
6: $\epsilon = \epsilon + p$
7: **end for**
8: $P[c] = \epsilon/|c|$ (normalize for the number of VMs)
9: **end for**
10: sort P (in increasing order of mean contention)
11: **for** $c \in P$ **do**
12: Break **if** V fits
13: **end for**
14: if the V fits in no cluster escalate to Engineer
15: **return** Cluster mapping for VM placement

A. Mean Contention Duration (MCD)

The intuition behind this policy is that co-hosted VMs should suffer equally from performance degradation due to contention for resources. This contrasts to having most VMs not suffer at all, while a few VMs suffer greatly. VMs that suffer greatly lead to significant SLA violations and thus to high operational risk (see P2 in Section II-B).

Addressing the intuition, we design the MCD heuristic policy listed in Algorithm 1. MCD aims to balance long-term contention, and thus VM suffering, across all clusters. First, MCD computes the mean duration of the period for which VMs suffer of contention, per cluster (lines 2–9). In the simulator, instead of a real-world measurement, contention (line 5) is estimated (i.e., using the predictor from [9]). Second, MCD sorts the clusters by mean contention, lowest first (line 10). Finally, MCD places each VM in the cluster with the lowest contention that can fit the VM (line 11). Because the datacenter operators have low tolerance for operational risk (P2), engineers are called in if the algorithm cannot find a satisfactory answer (line 14).

B. Maximum Consolidation Load (MCL)

This policy starts from the intuition that increasing consolidation (see P1) leads to higher profit for the datacenter operator, because there can be more VMs than physical machines and physical machines that are not occupied by any VM can be selectively powered off [12]. For operators, achieving high consolidation important for reducing operational costs.

We design the MCL heuristic policy listed in Algorithm 2 to maximize the consolidation of VMs on physical clusters. Per cluster, MCL calculates (lines 2–5) the gap over time between requested and actually used resources. Because VMs for BCWs are often over-provisioned, as indicated in Section II-B, the cumulative gap is often large. Then, MCL sorts the clusters by gap, largest-gap first (line 6). Finally, MCL places a new VM in the cluster with the biggest gap where it fits (lines 7–9).

Algorithm 2 Maximum Consolidation Load (MCL)

Input: V a new Virtual Machine, C all the Clusters;
1: $U = []$ (unused resources per cluster)
2: **for** $c \in C$ **do**
3: u_v (unused resources for VM v)
4: $U[c] = \Sigma u_v$, $v \in c$ (unused resources in cluster c)
5: **end for**
6: sort U (in decreasing order of unused space)
7: **for** $c \in U$ **do**
8: Break **if** V fits
9: **end for**
10: if the V fits in no cluster escalate to Engineer
11: **return** Cluster mapping for VM placement

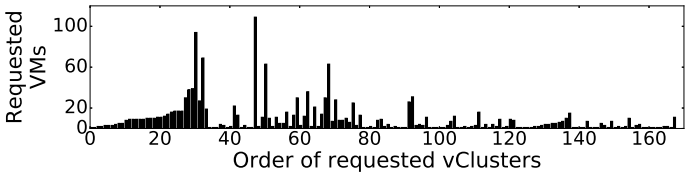


Fig. 3: Sequence of vCluster arrivals in the real multi-datacenter workload, over a period of 3 months.

V. Experiment Setup

A major contribution of this work is the experimental analysis of the portfolio scheduler and the impact of the presented utility functions on the scheduler's decisions. In this section, we present the setup of our experiments.

A. Representative Environment and Workload

We use one representative trace of real-world commercial workloads hosted in a multi-datacenter, multi-cluster infrastructure. The workload trace follows for about 3 months the real operation of the cloud operator, recording both the business-critical workload and the resulting datacenter-wide performance metrics. The latter is consequence to decisions taken by engineers at datacenter-level and by VMware technology at cluster-level, and is representative for the current datacenter industry. We use the former as **input workload** for our simulations, and the latter as a **baseline for comparison** (curve *REPLAY* in Figures 5b and 6b).

The trace includes a diverse stream of over 160 $vClusters$, depicted in Figure 3, to be placed in 12 clusters having about 200 physical hosts in total. Typically for BCWs, the input workload combines HPC, web, and many other types of applications; this leads to various anti-/affinities expressed in the request, e.g., HPC VMs have a strict requirement to run on HPC compute nodes in the multi-datacenter environment, a vCluster running a replicated database requesting to run its VMs on separate clusters or datacenters, etc. This emphasizes the need for a dynamic meta-scheduler, as no scheduler has been shown to support such a wide variety of application types and restrictions.

B. Portfolio Setup: 2 New + 7 Common Scheduling Policies

For our experiments, we equip the portfolio scheduler with the 2 new policies we design in this work (Section IV), plus 7 policies commonly used in datacenters. This allows us to see how both the new policies and the complete portfolio scheduler perform; if policies are not useful, they will simply not be selected by the (automated) portfolio scheduler. The 7 common policies are:

1. First-Fit (FF) is a commonly-used policy, simple but applicable in many domains of workload scheduling and VM placement. To place a request, FF chooses the first available cluster in which the request fits.

2. Type Priority (TP) extends FF to also consider request types (e.g., HPC), which are then matched against cluster capabilities. This enables specialized clusters to execute efficiently demanding workloads, such as HPC. TP may be a competitive policy for matching complex SLAs. In contrast to FF, TP will try to not place non HPC workloads on HPC clusters.

3. Memory Datacenter Balance (MDB) is a policy that uses memory utilization to evenly distribute the VMs over datacenters. MDB is a possible competitor to our new policies in lowering the DRR in the system.

4-7. Lowest Resource Load (L*) is a family of policies that attempt to balance over all clusters the loads of a specific resource, e.g., memory. We consider in this work all four common resource types, with the specific L* policies: Lowest CPU Load (**LCL**), Lowest Memory Load (**LML**), Lowest Storage Load (**LSL**), and Lowest Network Load (**LNL**).

VI. Experiment Results

In this section, we analyze how well can the risk-aware portfolio scheduling architecture meet SLAs for datacenters hosting BCWs. We conduct three separate experiments: (1) to analyze whether each policy is useful, that is, each policy is selected by the portfolio scheduler (in section VI-A); (2) to analyze the performance of the portfolio scheduler, and compare the results with the real-world baseline and with the policies in the portfolio (in Section VI-B); and (3) to investigate the performance of the portfolio scheduler when used to balance multiple objectives, which is the realistic situation in datacenters (in Section VI-C).

We use the setup detailed in Section VI: through trace-based simulations based on real-world multi-datacenter workloads, we evaluate the portfolio scheduler when equipped with 9 scheduling policies, and compare it with individual policies (the state-of-the-art) and with a relevant real-world baseline (REPLAY in Section V-A).

A. Is each Policy Useful?

The expectation is that the portfolio scheduler will select different scheduling policies dynamically, responding to changing workloads or datacenter optimization goals (utility functions). Our main findings are:

F-A1 All policies are selected for all utility functions.

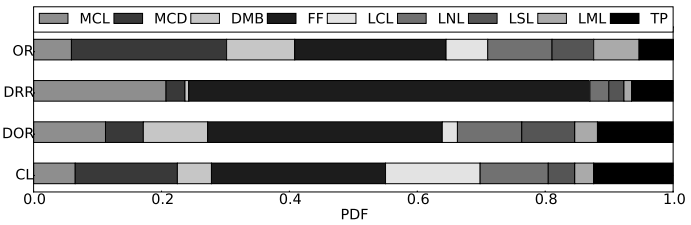


Fig. 4: Distribution of selected scheduling policies.

F-A2 Using different utility functions results in widely different distributions of selected policies.

F-A3 MCL and MCD are selected 16-30% of the time.

Figure 4 depicts the fraction of selected policies, for our new utility functions (OR, DRR, and DOR), and for the utility function Cluster Load (CL). (CL, which is used the datacenters from which we have collected the input traces, expresses the goal of balancing load across clusters and is included to allow us to compare this work with the state-of-the-art on portfolio scheduling [7].) The figure breaks down policy-selection as horizontal bars, one bar for each utility function (setup) that the portfolio used to select policies. Each horizontal bar contains a stacked relative histogram, with each individual box in the bar corresponding to an individual scheduling policy, the *width* representing the fraction of times the policy was selected from the total set of decisions taken by the portfolio scheduler. For each horizontal bar, *high fragmentation* indicates that every policy is important; conversely, a box wider than 50% of the bar indicates that a policy is predominantly the best.

In Figure 4, across all utility functions, we see a wide diversity of distributions. FF, LSL, LNL, TP, and our two policies MCL and MCD all account for over 10% of the selections for at least one utility metric. We conclude that each policy is useful, and the portfolio scheduler or any other method to select among them is highly desirable.

Figure 4 has an interesting outlier, when DRR is the active utility function. Then, the portfolio scheduler selects the FF policy about 60% of the time. This explains why, despite not being designed to address complex performance and risk metrics, FF is still much used by datacenter engineers. However, in Section VI-B results indicate that solely selecting FF will result in much higher DRR risk than our portfolio-scheduling approach can achieve.

B. Does Portfolio Scheduling Improve System Utility?

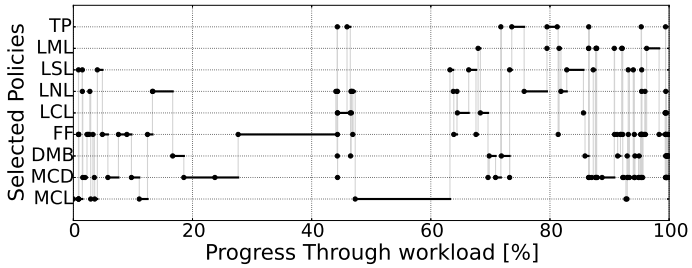
Is the portfolio scheduler improving the utility provided by the system? The main findings are that the portfolio scheduler:

F-B1 Decreases OR by up to 2x vs. individual policies.

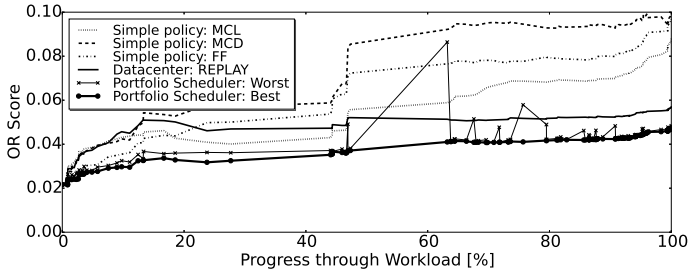
F-B2 Decreases OR by up to 1.5x vs. the real-world baseline of a well-managed commercial datacenter (REPLAY).

F-B3 Improves DRR significantly compared to just using FF up to 35%.

F-B4 Improves DRR significantly, up to 40% over REPLAY.

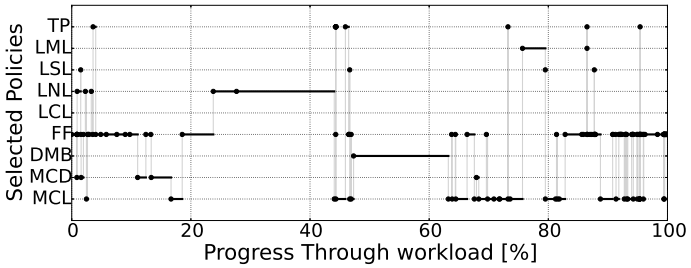


(a) Selected scheduling policies for the OR utility function.

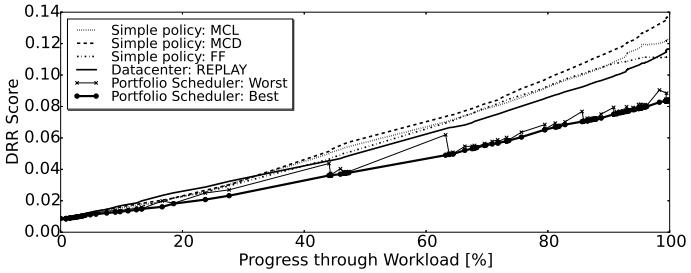


(b) Evolution of OR over the entire workload. Lower scores are better.

Fig. 5: Results for Operational Risk.



(a) Selected scheduling policies for the DRR utility function.



(b) Evolution of DRR over the entire workload. Lower scores are better.

Fig. 6: Results for Disaster Recovery Risk.

1) Operational Risk: Figure 5a depicts how the portfolio scheduler selects different policies, when the OR utility function is active. We create a new type of graph for this depiction, which the portfolio scheduler can also use to *explain its decisions to datacenter stakeholders* (R5 in Section III-A). The vertical axis is divided into parallel rows, each representing the progress (that is, non-/selection) of a different scheduling policy. For example, the new policies MCL and MCD occupy the two bottom-most rows. The horizontal axis depicts progress through the workload: large dots placed on the rows indicate that a policy has been selected, whereas thick horizontal bars emerging from the dots indicate the portion of the workload scheduled by the newly selected policy (wider bars indicate a higher portion). The last element of this graph is the set of vertical gray lines, which aims to simplify reading by marking *transitions* between consecutively selected policies. For each transition, datacenter engineers can understand why it happened, by reading the detailed results comparing the performance of each policy as estimated in simulation.

Figure 5a depicts numerous transitions. Each scheduling policy is selected at least once—each row corresponding to each policy has at least one large dot. In some cases, policies are selected multiple times in succession. This strengthens the findings of the previous section.

Figure 5b depicts the evolution of the OR utility-function value (*score*), for different scenarios. The curves depicted for the portfolio scheduler are for the policy actually selected ("Best Policy") and, to estimate the impact of each selection made by the portfolio, for a worst-case scenario of selecting the worst possible policy in that particular decision moment ("Worst Policy"). The figure also shows comparisons with the use of a single policy (we depict only the best-performing: FF, and our policies MCL and MCD) and with the decisions taken by the real-world datacenter engineers ("REPLAY"). The portfolio scheduler results in much lower Operational Risk compared to selecting individual policies (up to 2x better) and to the REPLAY baseline (up to 1.5x better). More importantly, the results indicate single policies lead to individual errors that seem small, but accumulated can cause significantly larger risk that the continuous selection of best-performing policies. Thus, using a portfolio scheduler leads to much lower risk of violating OR SLAs.

We explain why this happens through a concrete example. In Figure 5b, observe that, at around 50% through the workload, the worst-scoring policy shows a big spike—one of the policies performs significantly worse than the best policy. This shows that portfolio scheduling can effectively alleviate the risk of scheduling policies performing badly sporadically.

2) Disaster Recovery Risk, Operational Risk, and Cluster Load: Similarly to OR, for the DRR utility function we depict the results in Figures 6a and 6b. Overall, we observe that the approach results in lower risks of violating SLAs with regards to Disaster Recovery guarantees. Concretely, figure 6b shows that using the portfolio scheduler can lead to a 40% improvement over the real-world baseline (REPLAY). Importantly, this gives evidence that, for a must-fulfill principle (P3), even when the multi-datacenter is operated by highly trained engineers—the datacenter operator has a policy to hire only top-10% engineers in the market—, an automated, risk-aware portfolio scheduler can offer significant reduction of risk.

Similarly, for both DOR and CL we also investigate the policy selection and utility function scores. The portfolio scheduler results are positive, in-between the results for OR and DRR (for complete results, see technical report [13]).

C. Does Addressing One Risk Influence the Other Risks?

Datacenter operators often combine multiple utility functions (UFs), to optimize for multiple objectives at the same

TABLE I: Maximum utility function (UF) scores for all UFs.

Active UF ⟶ Resulting scores ↓	**OR**	**DRR**	**DOR 1-1**	**DOR 1-3**	**DOR 3-1**	**CL**	**R**
OR score $\times 10^{-2}$	5	10	5	7	5	7	6
DRR score $\times 10^{-2}$	12	8	11	9	11	12	12
DOR score $\times 10^{-3}$	8	9	8	9	6	9	9
CL score $\times 10^{-5}$	4	16	3	13	10	4	5

time. We analyze, in turn, how setting the portfolio scheduler to optimize for each UF affects the maximum scores for all the other UFs; Table I summarizes the results and shows that:

- F-C1 DOR can effectively balance DRR and OR.
- F-C2 When DOR 3-1 is used, both DOR and OR are very close to their (minimum) best.
- F-C3 Activating CL worsens all the other utility functions.

The results in Table I, when considered column-wise, indicate the influence that selecting an UF (e.g., OR) has on the other UFs. When considered row-wise, the results allow comparing for an UF its scores when it is active vs. when other UFs are active. If we optimize for the Disaster-Recovery Risk (column "DRR" in Table I), the OR score will increase by up to 2x compared to the case when OR is active, from 5 to 10 units (risk nearly 2x worse). Conversely, activating OR increases DRR, from 8 to 12 units. Thus, combining these two risks into DOR is important.

DOR provides a risk-management trade-off through its w_o and w_d weights. The results for DOR $w_o - w_d$ set, in turn, to 1-1, 3-1, and 1-3, show that the OR and DRR utility functions can be combined and, depending on the datacenter SLAs and optimization target, can be balanced. For example, for "DOR 3-1", increased emphasis put on OR leads to an OR score that is very close to its achievable minimum.

We also compare with the state-of-the-art UF CL [7], which balances load distribution for one or more resource types (e.g., CPU, Memory, IO). Activating OR actually leads to the best outcome we have observed for CL. This means that our new OR utility function has a better long term effect then directly optimizing for CL. Conversely, activating CL worsens all the other utility function scores, so the state-of-the-art CL is a bad predictor for the optimization goals considered in this work.

VII. Related Work

In this section, we survey related research from two main areas: portfolio scheduling, including applications to computer scheduling, and risk management in clouds.

Portfolio scheduling originates from the field of finance [14]. In computational portfolio design [15], extensive work in the field of artificial intelligence has focused on the selection and application stages of the portfolio [16], [17]. Much work has focused on speeding up the selection step [18], [8], whereas here we are concerned primarily with the quality of the selected policy. Using performance models instead of simulation has also been shown to facilitate fast selection from the portfolio of algorithms [18], but no current model captures accurately datacenters and BCWs.

Closest to our work, portfolio scheduling has been applied to academic [8] and commercial [19], [7] datacenters. In contrast to earlier work [8], [19], our work adds the focus on risk management, which as a new problem greatly extends the scope of portfolio scheduling in datacenters, and on long-running VMs, vs. the much shorter jobs considered by the others. Compared to van Beek et al. [7], our work greatly extends the problem scope to *combined* risks, and proposes many new advancements in scheduling and in experimental analysis: new policies, new utility functions, new experiments, new types of plots leading to new types of insights.

Workload scheduling in datacenters and other cloud computing context: The performance of any portfolio scheduler depends on the set of policies included in the portfolio. Previous work in datacenter and cloud computing scheduling has focused on three classes of policies:resource-related provisioning (including VM placement), workload-related job selection, and job-structure-related VM selection.

Deng et al. [8] survey a large number of policies for VM provisioning. Pires and Baran conduct a comprehensive survey [20] of policies for VM placement. Traditional work underlies both job selection [21] and VM selection (policies often based on bin-packing approaches [22]). Our study complements this body of work with focus on selecting resources across multiple datacenters, for a new problem domain.

Risk management in clouds includes an increasing body of work on SLAs between cloud customers and operators. Following more than a decade of evolution in the context of grids [23], [24], the state-of-the-art focuses on defining various system properties and SLA types [25], on negotiating and brokering SLAs [26], [27], on monitoring for [28] and on assessing [29], [30] SLA-violations, and on selecting clouds to minimize them [31] and other aspects of SLA-lifecycle management [32]. Eyraud-Dubois et al. [33] present the notion of SLAs and consolidation and show that with dynamic bin-packing a theoretical global CPU utilization of 66% can be achieved. Our work also considers resource consolidation and the associated operational risk, but we additionally investigate the reliability risk of datacenter disaster and also combined risks. In previous work on risk management in clouds, we see the introduction of assessing SLA violations [29] by a cloud broker and selecting cloud providers by imposing penalties to their reputation scores if violations occur. Optimis [31] quantifies risks by evaluating historical SLA violations and estimates the reliability of the cloud environment under test. Our work extends the notion of risk management in cloud systems by introducing utility functions addressing SLAs which reflect the complexity of the current cloud ecosystems.

VIII. Conclusion and Future Work

For large enterprises and governments to host their BCWs in virtualized datacenters, risk management needs to become a first-class citizen of datacenter resource management and scheduling. Toward this end, in this work we propose a new *risk-aware* portfolio scheduling architecture, which repeatedly and dynamically selects the scheduler that is estimated to minimize risk, among a set of schedulers considered in the portfolio.

Our is the first work to address scheduling that is aware of the following two types of risks: Operational Risk, which is the risk of not meeting the SLA performance requirements, and Disaster Recovery Risk, which is the risk of not being able to absorb the failure of an entire datacenter in a multi-datacenter operation. We further extend the state-of-the-art in portfolio scheduling through the following contributions: two new risk-aware scheduling policies, and a new method for selecting at runtime the risk-minimizing scheduling policy while taking into account the important but difficult to predict CPU-contention arising from multiple VMs sharing the same physical machine.

We show through trace-based simulations that use traces collected from a commercial multi-datacenter cloud operator (and, in the associated technical report [13], also synthetic traces) that our risk-aware portfolio scheduling compares favorably with all its constituent policies, and also with the real baseline of datacenter operations manually optimized by expert engineers as recorded by the cloud operator.

In future work, we will extend our contention predictor and DRR to more resource types. We have created for this work graphs that explain scheduling decisions in detail, over time. For the future, we plan to conduct an empirical study to see how datacenter engineers use these and similar results, and use the results to improve the explanations given to datacenter engineers–understanding is believing.

REFERENCES

[1] IDC, "Worldwide and regional public it cloud services: 2013-2017 forecast," IDC Tech Report. [Online] Available: www.idc.com/getdoc.jsp?containerId=251730, 2013.

[2] European Commission, "Uptake of cloud in europe," Final Report. Digital Agenda for Europe report. Publications Office of the European Union, Luxembourg, 2014.

[3] D. Talby and D. G. Feitelson, "Improving and stabilizing parallel computer performance using adaptive backfilling," in *IPDPS*, 2005.

[4] D. Klusácek and S. Tóth, "On interactions among scheduling policies: Finding efficient queue setup using high-resolution simulations," in *Euro-Par*, 2014.

[5] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *CCGRID*, 2015.

[6] L. A. Barroso, "Warehouse-scale computing: Entering the teenage decade," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, Jun. 2011.

[7] V. van Beek, J. Donkervliet, T. Hegeman, S. Hugtenburg, and A. Iosup, "Self-expressive management of business-critical workloads in virtualized datacenters," *IEEE Computer*, vol. 48, no. 7, pp. 46–54, 2015.

[8] K. Deng, J. Song, K. Ren, and A. Iosup, "Exploring portfolio scheduling for long-term execution of scientific workloads in iaas clouds," in *SC*, 2013.

[9] V. van Beek Giorgos Oikonomou and A. Iosup, "A cpu contention predictor for business-critical workloads in cloud datacenters," in *HotCloudPerf-2019 The Second Workshop on Hot Topics in Cloud Computing Performance*. IEEE, 2019, [Submitted].

[10] N. Herbst, A. Bauer, S. Kounev, G. Oikonomou, E. V. Eyk, G. Kousiouris, A. Evangelinou, R. Krebs, T. Brecht, C. L. Abad, and A. Iosup, "Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges," *TOMPECS*, vol. 3, no. 4, pp. 19:1–19:36, 2018. [Online]. Available: https://doi.org/10.1145/3236332

[11] N. Herbst, R. Krebs, G. Oikonomou, G. Kousiouris, A. Evangelinou, A. Iosup, and S. Kounev, "Ready for rain? a view from spec research on the future of cloud metrics," SPEC Research Group — Cloud Working Group, Standard Performance Evaluation Corporation (SPEC), Tech. Rep. SPEC-RG-2016-01, 2016.

[12] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in *CCGRID*, 2012, pp. 482–489.

[13] V. van Beek, G. Oikonomou, and A. Iosup, "Portfolio scheduling for managing operational and disaster-recovery risks in virtualized datacenters hosting business-critical workloads: an extended technical report," TU Delft, Tech. Rep. DS-2016-002, Apr. 2016, http://www.ds.ewi.tudelft.nl/fileadmin/pds/reports/2016/DS-2016-002.pdf.

[14] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[15] B. A. Huberman, R. M. Lukose, and T. Hogg, "An economics approach to hard computational problems," *Science*, vol. 27, no. 5296, 1997.

[16] M. J. Streeter, D. Golovin, and S. F. Smith, "Combining multiple heuristics online," in *AAAI*, 2007, pp. 1197–1203.

[17] A. Goldman, Y. Ngoko, and D. Trystram, "Malleable resource sharing algorithms for cooperative resolution of problems," in *IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[18] M. Gagliolo and J. Schmidhuber, "Learning dynamic algorithm portfolios," *Ann. Math. Artif. Intell.*, vol. 47, no. 3-4, pp. 295–328, 2006.

[19] O. Shai, E. Shmueli, and D. G. Feitelson, "Heuristics for resource matching in intel's compute farm," in *JSSPP*, 2013.

[20] F. L. Pires and B. Barán, "A virtual machine placement taxonomy," in *CCGRID*, 2015, pp. 159–168.

[21] W. Tang, Z. Lan, N. Desai, and D. Buettner, "Fault-aware, utility-based job scheduling on Blue Gene/P systems," in *CLUSTER*, 2009, pp. 1–10.

[22] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," in *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1997, pp. 46–93.

[23] K. Czajkowski, I. T. Foster, C. Kesselman, V. Sander, and S. Tuecke, "SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems," in *Job Scheduling Strategies for Parallel Processing, 8th International Workshop, JSSPP 2002, Edinburgh, Scotland, UK, July 24, 2002, Revised Papers*, 2002, pp. 153–183.

[24] P. Wieder, J. Seidel, O. Wäldrich, W. Ziegler, and R. Yahyapour, "Using sla for resource management and scheduling-a survey," in *Grid Middleware and Services*, 2008, pp. 335–347.

[25] P. Wieder and R. Yahyapour, "Grid environments: Service level agreements (slas)," in *Encyclopedia of Software Engineering*, P. A. Laplante, Ed. Taylor & Francis, 2010, pp. 347–360.

[26] K. Djemame, J. Padgett, I. Gourlay, and D. Armstrong, "Brokering of risk-aware service level agreements in grids," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 13, pp. 1558–1582, 2011.

[27] F. Jrad, J. Tao, and A. Streit, "SLA based service brokering in intercloud environments," in *CLOSER*, 2012, pp. 76–81.

[28] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, "Establishing and monitoring slas in complex service based systems," in *IEEE International Conference on Web Services, ICWS 2009, Los Angeles, CA, USA, 6-10 July 2009*. IEEE Computer Society, 2009, pp. 783–790. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5175785

[29] K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voss, "Introducing risk management into the grid," in *e-Science*. IEEE, 2006, pp. 28–28.

[30] K. Lu, T. Röblitz, R. Yahyapour, E. Yaqub, and C. Kotsokalis, "Qos-aware sla-based advanced reservation of infrastructure as a service," in *CloudCom*, 2011, pp. 288–295.

[31] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame *et al.*, "Optimis: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[32] K. Lu, R. Yahyapour, P. Wieder, E. Yaqub, M. Abdullah, B. Schloer, and C. Kotsokalis, "Fault-tolerant service level agreement lifecycle management in clouds using actor system," *Future Generation Comp. Syst.*, vol. 54, pp. 247–259, 2016.

[33] L. Eyraud-Dubois and H. Larchevêque, "Optimizing resource allocation while handling SLA violations in cloud computing platforms," in *27th IEEE IPDPS 2013, Cambridge, MA, USA, May 20-24, 2013*, 2013, pp. 79–87.