Preprints.org

Article

Not peer-reviewed version

VMPlaceS Enables Scalable Evaluation of Virtual Machine Placement Strategies Using a High-Fidelity Simulation Framework

Apeksha Bhuekar *

Posted Date: 23 December 2025

doi: 10.20944/preprints202512.2039.v1

Keywords: virtual machine placement; cloud computing; resource management; simulation framework; SimGrid; scalability evaluation; dynamic workloads; service level agreements (SLA); distributed algorithms; performance evaluation

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

VMPlaceS Enables Scalable Evaluation of Virtual Machine Placement Strategies Using a High-Fidelity Simulation Framework

Apeksha Bhuekar

Campbellsville University, USA; apeksharaj17@gmail.com

Abstract

Efficient placement of Virtual Machines (VMs) is critical for optimizing resource utilization and ensuring service reliability in cloud computing infrastructures. Existing validation methods for VM placement algorithms, such as limited in-vivo experiments and ad hoc simulators, often fail to reflect real-world complexities and provide fair comparisons. This paper introduces VMPlaceS, a simulation framework built on SimGrid, designed to address these shortcomings by enabling the robust evaluation and comparison of VM placement strategies. VMPlaceS facilitates large-scale scenario modeling with customizable parameters to represent dynamic workloads and realistic platform conditions. By simulating centralized, hierarchical, and distributed algorithms, this study highlights the framework's capability to assess scalability, reactivity, and SLA adherence in various deployment scenarios. VMPlaceS emerges as a valuable tool for researchers and practitioners to explore innovative VM placement solutions and advance the field of cloud computing resource management.

Keywords: virtual machine placement; cloud computing; resource management; simulation framework; SimGrid; scalability evaluation; dynamic workloads; service level agreements (SLA); distributed algorithms; performance evaluation

I. Introduction

The quick reception of Distributed computing (CC) has changed the IT business, empowering adaptable, on-request admittance to computational assets. This shift is upheld by powerful administration frameworks like CloudStack, OpenNebula, and OpenStack [1–3], as well as different Foundation as-a-Administration (IaaS) toolkits [4]. These stages frequently depend on rudimentary virtual machine (VM) arrangement strategies, which, while practical, miss the mark regarding amplifying asset usage and sticking to Administration Level Arrangements (SLAs) that characterize VM asset prerequisites.

Customarily, CC stages utilize static group booking for VM portion, wherein VMs are doled out to actual machines in view of client characterized determinations. Be that as it may, this static methodology is less than ideal as clients regularly misjudge their asset needs, prompting underutilization of actual foundation. Besides, asset requests of a VM can vary essentially all through its lifecycle [5], making static distribution a less compelling procedure for dynamic cloud conditions.

The scholarly local area has proposed progressed systems, for example, dynamic solidification, load adjusting, and SLA-authorizing calculations, to address these challenges [6–10]. In any case, the reception of these modern strategies in genuine frameworks stays restricted. One of the key obstructions is the exploratory cycles utilized for their approval. Most VM situation calculations have been assessed utilizing either custom test systems or limited scope *in-vivo* tests, which are frequently inadequate to catch the intricacies of true CC conditions. These limits impede the capacity to guarantee accuracy, versatility, and unwavering quality of these calculations, as well as their reasonableness for production conditions.

Leading assessments on agent testbeds that record for elements like adaptability, dependability, and responsibility inconstancy are pivotal for the turn of events and reception of cutting edge VM

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

situation methodologies. In any case, performing *in-vivo* tests isn't just asset concentrated and tedious however may likewise yield misleading results in the event that the noticed ways of behaving stray from expected outcomes [11].

To address these difficulties, we present VMPlaceS, a committed recreation structure intended to empower top to bottom examination and correlation of VM situation calculations under reasonable circumstances. VMPlaceS upholds the assessment of huge scope situations, including a large number of VMs, each executing dynamic responsibilities. This structure furnishes scientists with the devices to thoroughly break down algorithmic execution, including adaptability and responsiveness to SLA infringement, subsequently overcoming any barrier between theoretical proposition and functional executions.

As a show of VMPlaceS' capacities, we dissect three notable VM position approaches: Entropy [7], Snooze [6], and DVMS [8]. These calculations address particular position standards — concentrated, various leveled, and completely appropriate models, separately. Our assessment features the adaptability and reactivity of these procedures, as well as the effect of reconfiguration and calculation stages on by and large execution.

By empowering thorough examinations of VM position procedures, VMPlaceS enables specialists to refine their calculations and cutoff *in-vivo* trials to those systems with demonstrated potential to fulfill the needs of production-scale CC frameworks.

II. Preliminaries and Problem Formulation

a) Sets and parameters. Let $\mathcal{H}$ be the set of physical hosts, $\mathcal{V}$ the set of VMs. Each host $h \in \mathcal{H}$ has capacities $(C_h^{\text{cpu}}, C_h^{\text{mem}}, C_h^{\text{net}})$. Each VM $v \in \mathcal{V}$ has demands $(d_v^{\text{cpu}}, d_v^{\text{mem}}, d_v^{\text{net}})$ which may vary over time t due to workload dynamics.

b) Decision variables. $x_{v,h}(t) \in \{0,1\}$ indicates whether VM v is placed on host h at time t. $y_v(t) \in \{0,1\}$ indicates whether v is being migrated at time t.

c) Objective (one example used in our experiments).

$$\min \quad \alpha \sum_t \sum_v \text{vio}_v(t) \; + \; \beta \sum_t \sum_v y_v(t) \tag{1}$$

Explanation. $\text{vio}_v(t)$ counts SLA violations for VM v at time t (any resource shortfall); The first term penalizes violations (service quality), while the second penalizes migrations (operational overhead). Weights $\alpha \gg \beta$ reflects that avoiding SLA violations is primary.

d) Placement and capacity constraints.

$$\sum_{h \in \mathcal{H}} x_{v,h}(t) = 1 \quad \forall v, t \qquad \text{(each VM on exactly one host)} \tag{2}$$

$$\sum_{v \in \mathcal{V}} d_v^{\text{cpu}}(t)\, x_{v,h}(t) \le C_h^{\text{cpu}} \quad \forall h, t \tag{3}$$

$$\sum_{v \in \mathcal{V}} d_v^{\text{mem}}(t)\, x_{v,h}(t) \le C_h^{\text{mem}} \quad \forall h, t \tag{4}$$

$$\sum_{v \in \mathcal{V}} d_v^{\text{net}}(t)\, x_{v,h}(t) \le C_h^{\text{net}} \quad \forall h, t \tag{5}$$

Explanation. The constraints enforce per-host CPU/memory/network limits at each t.

e) Migration cost coupling. We account for live-migration time T_v^{mig} and traffic B_v^{mig} via SimGrid's live-migration model; T_v^{mig} depends on `mig_speed` and memory update speed `mem_speed`. We set $y_v(t) = 1$ during $[t,\ t + T_v^{\text{mig}})$ to reflect transient overhead captured by the simulator.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

A) Glossary of Simulation Terms

Table 1. Simulator terms used throughout the paper.

Term	Meaning in VMPlaceS/SimGrid
`nb_cpus`	Number of vCPUs allocated to a VM
`ramsize`	Memory allocated to a VM (MB/GB)
`net_bw`	Network bandwidth available to the VM
`mig_speed`	Effective bandwidth used during live migration
`mem_speed`	VM memory dirtying rate during migration

B) Worked Micro-Example

Consider 3 hosts with $(C^{\text{cpu}}) = (16, 16, 16)$ and four VMs with $d^{\text{cpu}} = (8, 6, 4, 4)$. A feasible initial placement is $\{(v_1, h_1), (v_2, h_1), (v_3, h_2), (v_4, h_3)\}$. If v_2 spikes from $6 \rightarrow 10$ cores, h_1 becomes overloaded ($18 \not\leqslant 16$): Entropy proposes migrating v_3 to h_1 is infeasible; moving v_2 to h_2 restores feasibility. This toy example mirrors the reconfiguration logic measured.

III. Related Work

VM placement and consolidation. Early consolidation managers such as Entropy formulate placement as a constraint optimization problem and solve it centrally to reduce overloads and migrations [12,13]. Snooze introduces a hierarchical control plane to scale monitoring and reconfiguration decisions [14]. DVMS adopts a cooperative, fully distributed scheduling model to localize decisions and improve reactivity at scale [15].

Simulation at scale. SimGrid has become a widely used platform for accurate, scalable simulation of distributed systems, and recent work adds VM abstractions and a live-migration model suitable for IaaS studies [16,17]. VMPlaceS leverages these capabilities to compare centralized, hierarchical, and distributed placement strategies under identical, reproducible conditions.

Positioning. Unlike ad hoc simulators or limited in-vivo tests (which hinder fair comparison), VMPlaceS provides a common testbed that (i) reproduces live-migration costs, (ii) models dynamic workloads, and (iii) records traces for post-hoc analysis. Our evaluation thus complements prior work by providing a head-to-head comparison of Entropy, Snooze, and DVMS under controlled scale-out scenarios.

IV. SimGrid: A Toolbox for Building Simulators

SimGrid is a flexible and robust toolbox for simulating complex algorithms at large scale [18]. It has been widely involved in the examination of local areas for reproducing assorted circulated modeling situations, ranging from lattice registering to cloud and HPC (High-Performance Computing) conditions. The tool stash gives an extensive structure to demonstrating and assessing calculations in controlled, reproducible conditions, in this way offering specialists the ability to evaluate the performance, versatility, and productivity of their answers under realistic circumstances.

At the center of SimGrid is its capacity to perform reenactments in view of three major parts: the *program*, the *platform*, and the *deployment* records. The *program* addresses the calculation or interaction to be recreated and is created utilizing SimGrid's Message Passing Programming interface (MSG). This Programming interface gives reflections to demonstrating substances like cycles, undertakings, virtual machines (VMs), and network interchanges. These reflections empower scientists to plan reenactments that intently reflect genuine situations.

The *platform* record portrays the actual framework fundamental to the reproduction. It gives a detailed description of the assets in question, for example, computational hubs, network connections, and capacity gadgets. This degree of granularity permits specialists to display unpredictable structures and asset designs, working with exact execution assessments.

The *deployment* document determines how the reenacted processes from the *program* are planned onto the hubs portrayed in the *platform*. It arranges the execution of the reproduction by characterizing

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

the underlying condition of the framework, including the designation of cycles and assets. By isolating the program rationale from the stage and organization arrangements, SimGrid guarantees adaptability and seclusion in reenactment plan.

Reenactments are executed by the SimGrid motor, which utilizes an inward limitation solver to deal with the distribution and booking of computer chip and organization assets through the reproduction. This approach guarantees that the elements of asset dispute and sharing are precisely demonstrated, empowering exact assessments of algorithmic execution. For a definite outline of SimGrid's engineering and capacities, we allude readers to [18].

SimGrid was picked as the establishment for VMPlaceS because of a few convincing reasons. In the first place, SimGrid has gained notoriety for its presentation and legitimacy inside the examination community [19]. Its thorough displaying capacities and strong reproduction motor pursue it an optimal decision for concentrating for enormous distributed frameworks. Second, SimGrid has been as of late improved to incorporate VM reflections and a live movement model [20]. These elements are especially important for our work, as they permit scientists to control VMs in a way steady with certifiable tasks. This incorporates making, annihilating, beginning, closing down, suspending, continuing, and moving VMs.

The live relocation model gave by SimGrid is a basic component that separates it from other reenactment systems. It precisely gauges the time and organization traffic related to VM relocations by representing asset dispute and memory invigorate rates during movement occasions. This degree of accuracy is fundamental for assessing the exhibition of VM position calculations, as it mirrors the intricacies of certifiable situations, including asset sharing and dynamic responsibilities.

By utilizing these abilities, we fabricated VMPlaceS to empower the recreation and investigation of VM position methodologies at scale. SimGrid's help for demonstrating VM conduct and live movement was fundamental in accomplishing this objective. With VMPlaceS, scientists can act top to bottom examinations of position calculations under practical circumstances, for example, fluctuating responsibility designs, asset dispute, and enormous scope foundation setups.

The reception of SimGrid as the basic structure for VMPlaceS guarantees that our reenactment device isn't just vigorous and exact yet in addition extensible and versatile to advancing examination needs in the field of distributed computing.

V. VMPlaceS Simulator

The VMPLACES simulator is designed with two primary goals:

1) To liberate scientists from the complex subtleties of dealing with VM creation and dynamic responsibility varieties while assessing novel VM arrangement calculations.
2) To work with a hearty examination of these calculations under reliable and reproducible circumstances.

A. Overview

VMPlaceS is carried out in **Java**, using the **SimGrid (SG) MSG API** to construct its center functionalities. In spite of the fact that Java presents a slight above over SimGrid's presentation, its decision is vital. Java fundamentally works on the improvement of cutting-edge planning techniques by giving an adaptable and engineer-friendly programming climate. For example, complex recommendations, for example, *Snooze*, were carried out in Java, while the *DVMS* approach was executed utilizing both *Scala* and Java. This exhibits the flexibility and simplicity of Java for executing complex calculations.

VMPlaceS works through an organized three-stage reenactment process, as outlined in Figure 1:

1) **Initialization Phase:** This includes setting up the reenactment climate, making VMs, and creating the occasion line.
2) **Injection Phase:** This stage powerfully oversees responsibility changes and occasions during the recreation.
3) **Trace Examination Phase:** In this last step, reenactment information is broke down, and execution measurements are pictured for extensive experiences.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

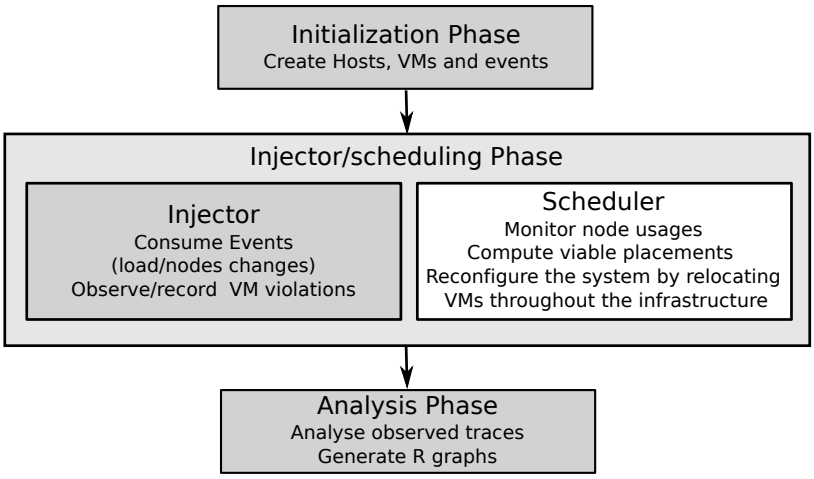


Figure 1. VMPlaceS's Workflow.

The test system utilizes no less than two SimGrid cycles to play out its activities:

1) The **Injector Process**, which handles occasion infusion during the reproduction.
2) The interaction running the **scheduling algorithm** under assessment.

Scientists can carry out their booking calculations utilizing the SimGrid MSG Programming interface, close to a range of unique points of interaction gave by VMPlaceS. These connection points incorporate key classes, for example,

- **XHost:** An expansion of SimGrid's `Host`, addressing the mimicked has.
- **XVM:** An expansion of SimGrid's `VM`, epitomizing VM-explicit traits.
- **SimulatorManager:** A class that oversees connections between test system parts, empowering clients to screen the foundation's state, including measurements, for example, have load, the number of facilitated VMs, and overburden conditions.

B. Initialization Phase

The **Initialization Phase** starts by making *n VMs*, which are dispersed across the first *p hosts* recorded in the stage setup document utilizing a cooperative designation. The stage record characterizes the group geography, indicating *h* facilitating hubs and *s* administration hubs. These boundaries -- *n*, *h*, and *s*– - are input by clients by means of a setup document.

The default stage arrangement reenacts a group, yet clients can characterize more complex geographies, for example, unified server farms, to suit their exploration situations.

Each VM is launched in light of pre-characterized formats called **VM classes**, which determine fundamental attributes:

- **Number of Cores:** `nb_cpus`
- **Memory Size:** `ramsize`
- **Network Bandwidth:** `net_bw`
- **Migration Bandwidth:** `mig_speed`
- **Memory Update Speed:** `mem_speed`

These traits oversee the VM's way of behaving, especially during movements. For instance, the memory update speed (`mem_speed`) straightforwardly impacts movement time and the volume of moved information. Clients can tweak these layouts to reproduce different responsibilities, including memory-escalated undertakings.

During instatement, all VMs start with a central processor heap of nothing. Their responsibilities develop progressively all through the recreation, driven by events infused in resulting stages. When the VMs are made and appointed, VMPlaceS generates two basic SimGrid processes:

1) The **Injector Process**, which produces the occasion line.
2) The interaction executing the planning calculation under assessment.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

The occasion line is populated with **CPU load change events**, produced at stretches following an outstanding conveyance with rate boundary λ_t. The heap for every occasion is gotten from a Gaussian distribution characterized by a mean (μ) and standard deviation (σ). Here, λ_t controls how frequently load-change events arrive (higher means more frequent events), while μ and σ set the average load level and its variability for each event, respectively. These boundaries - - t, μ, and σ- - - are client characterized, guaranteeing adaptability in reproduction situations.

To guarantee reproducibility, all irregular cycles in VMPlaceS are instated with a seed determined in the setup record. Specialists can expand the occasion framework by making new occasion classes that carry out the `InjectorEvent` interface and adding the comparing occasion age rationale. For instance, future arrivals of VMPlaceS will incorporate *node expansion/expulsion events* to reproduce situations like accidents or dynamic scaling.

C. Injector Phase

When the introduction is finished and the worldwide occasion line is prepared, the **Injector Phase** starts. The injector interaction iteratively consumes occasions from the line, powerfully changing the VM jobs by making and doling out new SimGrid errands to the VMs. These undertakings adjust the computer chip load, straightforwardly affecting movement times and memory update speeds.

In light of choices made by the scheduler, VMs might go through activities like suspension, resumption, or relocation across accessible hosts. Specialists should carry out both the planning calculation to address the VM situation issue and the rationale to execute reconfiguration plans. These plans, overseen through the `SimulatorManager` class, are basic as reconfiguration costs are integral to the presentation of dynamic arrangement frameworks.

Critically, VMPlaceS doesn't mimic the computational season of the scheduler. All things considered, it conjures the scheduler execution straightforwardly, catching the genuine calculation time. Nonetheless, all responsibility changes and reconfiguration activities (e.g., suspend, continue, relocate) are reproduced utilizing SimGrid.

D. Trace Analysis

The **Trace Investigation Phase** processes the information gathered during the reproduction. VMPlaceS stretches out the SimGrid Follow module to record extensive stage measurements, for example, VM and have loads, the recurrence and span of SLA infringement, the quantity of relocations, and the scheduler's conjuring success rate.

This information is put away in two organizations:

1) **Visualization Files:** Viable with local area apparatuses like **PajeNG** for graphical investigation.
2) **JSON Follow Files:** Utilized for creating point-by-point asset use perceptions in the **R** factual climate.

The Follow module is completely extensible, permitting scientists to characterize and record custom factors well defined for their calculations. For example, extra measurements can be recorded to assess explicit planning ways of behaving or responsibility qualities.

By empowering fine-grained follow examination and representation, VMPlaceS gives specialists important insights into the performance and efficiency of their VM arrangement calculations, preparing for vigorous and informed correlations.

VI. Dynamic Virtual Machine Placement Problem (VMPP) Algorithms

To exhibit the relevance and viability of VMPlaceS in tending to the intricacies of dynamic virtual machine situation, we executed three unmistakable calculations: a unified methodology propelled by the Entropy proposal [7], a progressive arrangement in view of the Nap framework [6], and a completely disseminated procedure utilizing the standards of DVMS [8]. Each approach represents an interesting methodology to determine asset portion infringement emerging from overburden hubs.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

These calculations are intended to keep up with framework execution and guarantee the productive usage of assets in a replicated environment.

The issue of over-burden hubs emerges when the virtual machines (VMs) are facilitated on an actual server, on the server's computational resources, surpassing the server's ability, especially the central processor power. Such situations upset framework balance and require brief reconfiguration. Every one of the three calculations carried out here depends on a goal component that figures out and applies reconfiguration intentions to reestablish framework stability. In particular, we utilized the solver created inside the Entropy framework [21] as the center reconfiguration device for all methodologies. This solver iteratively assesses potential setups inside a predefined break. On the off chance that the break is reached, it applies the best arrangement found, starting live movements to execute the reconfiguration in the recreated climate.

In the segments underneath, we give an exhaustive outline of each methodology, expounding on their extraordinary models, functional strategies, and key benefits.

A. Centralized Approach: Entropy-Based Algorithm

The brought together situation calculation, established in the Entropy structure, works as a solitary, durable element that directs the whole framework from an assigned help hub. This approach includes sending a solitary SimGrid process entrusted with checking asset utilization and evaluating the ongoing design's practicality. At normal stretches, this cycle summons the Entropy VMPP solver to successfully distinguish potential asset portion infringement and resolve it.

Asset checking is performed through direct admittance to the conditions of the hosts and their separate VMs. Measurements, for example, computer processor usage and memory utilization, are utilized to assess whether any hub is working past its ability. On the off chance that the solver distinguishes a requirement for reconfiguration, it decides the ideal movement plan, limiting the quantity of relocations expected while guaranteeing framework dependability.

When a reconfiguration plan is figured out, the framework records a few key measurements, including:

- The quantity of movements performed during the reconfiguration interaction.
- The absolute time taken to execute the reconfiguration.
- Whether the reconfiguration effectively settled all asset designation infringement or presented new ones.

This incorporated methodology is especially appropriate for limited scope conditions where the above of a solitary observing and dynamic substance doesn't essentially influence execution. Its effortlessness and certainty make it an astounding benchmark for contrasting other, more perplexing systems.

B. Hierarchical Approach: Snooze-Based Algorithm

The Rest framework [6,22] utilizes a progressive plan to deal with the situation and burden adjusting of VMs effectively. The engineering is coordinated into three particular levels, each with explicit obligations:

1) **Group Pioneers (GLs):** At the highest level, GLs total observing information from the moderate layer and keep a worldwide perspective on the framework state. This incorporated point of view empowers them to come to informed conclusions about the asset portion.
2) **Group Chiefs (GMs):** Filling in as delegates, GMs oversee groups of hubs and transfer data between the GLs and Neighborhood Regulators. They are liable for organizing the exercises of hubs within their space.
3) **Local Regulators (LCs):** Arranged at the most reduced level, LCs handle individual hubs and deal with the VMs alloted to them. They additionally screen the hubs' asset utilization and report it to the higher layers.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

Correspondence between these layers is organized and occasional. Observing information streams up from LCs to GMs and at last to GLs, guaranteeing that higher layers get opportune updates about framework load and different boundaries. Alternatively, reconfiguration orders and pulses are spread descending, empowering proficient order dispersal.

In our execution, Rest's center usefulness was demonstrated utilizing Simgrid's occasion dealing with natives. Multicast correspondence, a basic component for spreading updates across the network, was carried out as a die-hard commitment. This assistance guarantees simultaneous state engendering to different collectors, keeping up with framework responsiveness considerably under high loads.

Nap's progressive plan offers a harmony among versatility and reasonability, making it an optimal decision for medium-to huge scope conditions. By dispersing dynamic obligations across various layers, the engineering decreases the weight on any single part while holding a degree of focal oversight.

C. Distributed Approach: DVMS-Based Algorithm

The Dispersed Virtual Machine Scheduler (DVMS) [8] addresses a completely decentralized way to deal with VM position, underscoring participation among hubs to accomplish versatile and shortcoming open minded asset the executives. Dissimilar to concentrated and progressive frameworks, DVMS works without a solitary reason behind control, depending rather on an organization of specialists sent across hubs.

Every hub in the framework has a DVMS specialist liable for observing its asset utilization, managing its VMs, and working together with adjoining specialists. Correspondence between specialists is managed through an overlay organization, which characterizes the connections among hubs and empowers proficient data sharing.

At the point when a hub encounters asset imperatives, it starts a *Iterative Planning Method (ISP)*:

1) The hub holds itself as the critical thinking pioneer and recognizes its closest neighbor in the overlay organization.
2) Assuming that the neighbor is now important for another segment, the calculation moves to the following accessible neighbor. In any case, the neighbor joins the segment and turns into the new pioneer.
3) The pioneer assembles asset use information from all segment individuals and endeavors to form a reconfiguration plan. Assuming no arrangement is found inside the ongoing parcel, the calculation extends to iteratively incorporate extra hubs.

This iterative, parcel based approach guarantees that reconfiguration undertakings are conveyed and parallelized, decreasing and overall goal time. By building little, free parcels, DVMS accomplishes high adaptability and responsiveness, even in enormous and dynamic conditions.

The DVMS execution in VMPlaceS was created utilizing SCALA, utilizing Simgrid's Java natives for between communication. The overlay network was displayed as an unstructured geography, permitting adaptable neighbor connections and quick transformation to changing framework conditions.

This decentralized technique is especially successful in enormous scope, heterogeneous conditions where unified arrangements might struggle to adapt to the scale and variety of asset requests.

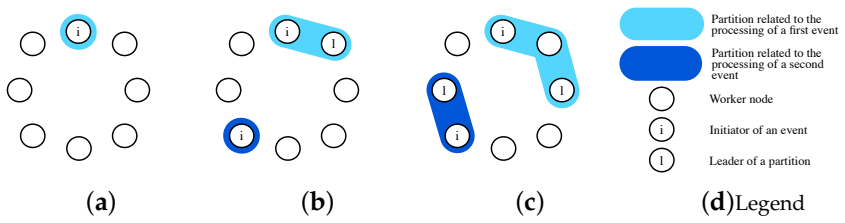


Figure 2. Processing two events simultaneously.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

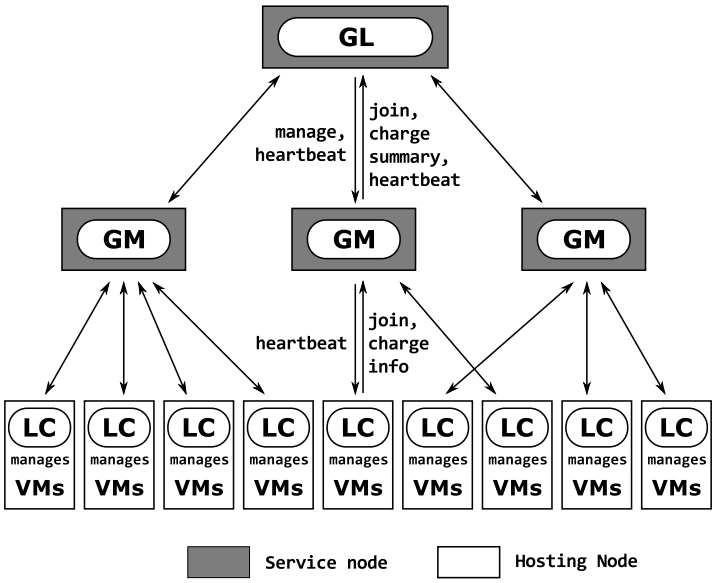


Figure 3. Snooze Architecture.

Table 2. Number of Failed Reconfigurations Across Different Infrastructure Sizes.

Infrastructure Size	2 LCs	4 LCs	8 LCs	32 LCs
128	19	0	0	0
256	29	0	0	0
512	83	1	0	0
1024	173	7	0	0

Table 3. Computation duration (mean $\pm$ standard deviation) across infrastructure sizes (aggregated over 30 simulation runs per setting).

Infrastructure Size	2 LCs	4 LCs	8 LCs	32 LCs
128	0.16 ± 1.23	0.34 ± 1.81	0.58 ± 2.40	2.53 ± 4.62
256	0.18 ± 1.31	0.42 ± 1.99	0.66 ± 2.50	2.65 ± 4.69
512	0.15 ± 1.20	0.33 ± 1.78	0.67 ± 2.54	2.83 ± 4.98
1024	0.19 ± 1.37	0.42 ± 2.02	0.89 ± 2.90	2.69 ± 4.91

VII. Experiments

We directed two classes of analyses to approve the adequacy of VMPlaceS: one zeroing in on precision and the other exhibiting its application in breaking down the Entropy, Nap, and DVMS systems.

A. Accuracy Evaluation

To evaluate the precision of VMPlaceS, tests were performed utilizing both certifiable circumstances on the Graphene group and simulations on the Grid'5000 testbed. The investigations included the Entropy technique executed at regular intervals of a one-hour span, looking at reproduced and *in-vivo* results.

The Graphene group comprised hubs with Intel Xeon X3440 central processors, 16 GB memory, and GbE NICs, facilitating 6 VMs per hub. VM arrangements differed across 8 predefined classes, with memory update speeds going from 0% to 80% of relocation transmission capacity. Load profiles followed remarkable and Gaussian appropriations, refreshed at regular intervals. Reproductions duplicated these circumstances, demonstrating network boundaries to mirror Graphene's presentation.

Figure 4 shows the time expected for the Entropy calculation stages under the two conditions. Recreations firmly paired true outcomes, with reconfiguration times contrasting by 6%-18% (middle 12%). Disparities originated from data transmission vacillations during synchronous movements,

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

featuring possible upgrades in reproduction models. In any case, the exactness of reenactments was considered adequate to break down execution patterns.

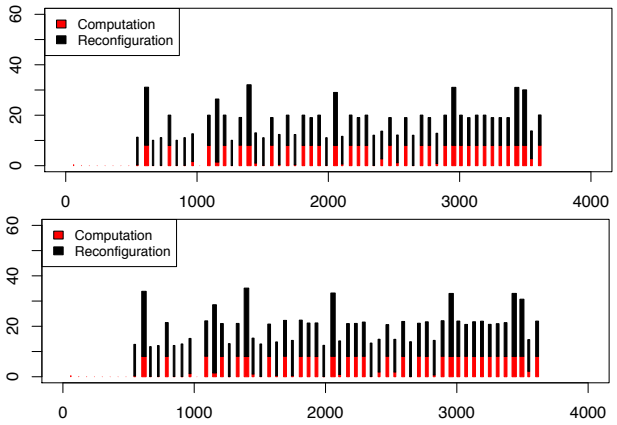
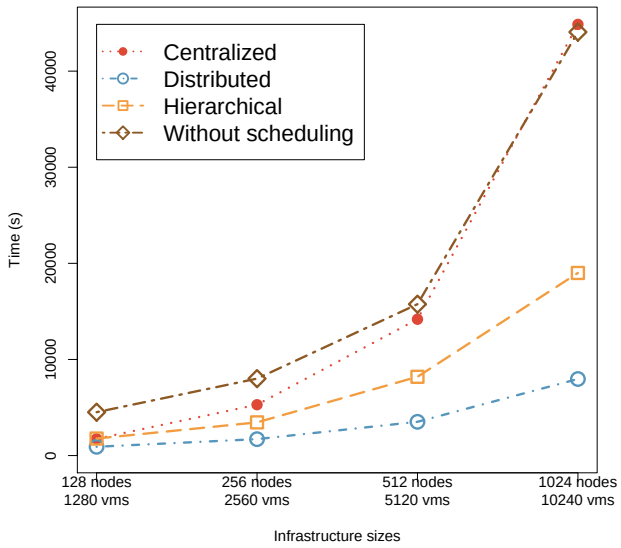


Figure 4. Comparison between simulated (top) and in-vivo (bottom) executions. The Y-axis shows the duration of each Entropy invocation, divided into the computation phase (searching for a configuration) and the reconfiguration phase (relocating VMs). Both axes are in seconds.



Infrastructure size	Duration of **violations** ($\mu \pm \sigma$)		
	Centralized	Hierarchical	Distributed
128 nodes	21.26 ± 13.55	21.07 ± 12.32	9.55 ± 2.57
256 nodes	40.09 ± 24.15	21.45 ± 12.10	9.58 ± 2.51
512 nodes	55.63 ± 42.26	24.54 ± 16.95	9.57 ± 2.67
1024 nodes	81.57 ± 86.59	29.01 ± 38.14	9.61 ± 2.54

Infrastructure size	Duration of **computations** ($\mu \pm \sigma$)		
	Centralized	Hierarchical	Distributed
128 nodes	3.76 ± 7.43	2.52 ± 4.63	0.29 ± 0.03
256 nodes	7.97 ± 15.03	2.65 ± 4.69	0.25 ± 0.02
512 nodes	15.71 ± 29.14	2.83 ± 4.98	0.21 ± 0.01
1024 nodes	26.41 ± 50.35	2.69 ± 4.92	0.14 ± 0.01

Infrastructure size	Duration of **reconfigurations** ($\mu \pm \sigma$)		
	Centralized	Hierarchical	Distributed
128 nodes	10.34 ± 1.70	10.02 ± 0.14	10.01 ± 0.11
256 nodes	10.26 ± 1.45	10.11 ± 0.83	10.01 ± 0.08
512 nodes	11.11 ± 3.23	10.28 ± 1.50	10.08 ± 0.82
1024 nodes	18.90 ± 7.57	10.30 ± 1.60	10.04 ± 0.63

Figure 5. Scalability/reactivity analysis of Entropy, Snooze, and DVMS across infrastructure sizes.

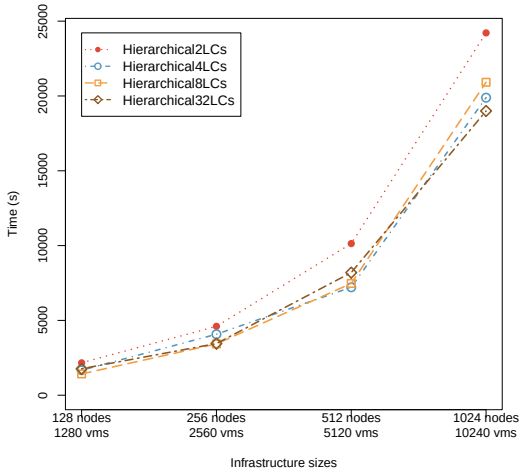


Figure 6. Hierarchical placement with Snooze: influence of varying group sizes (LCs per GM).

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

B. Analysis of Entropy, Snooze, and DVMS

We looked at the three procedures utilizing reproductions over groups going from 128 to 1024 PMs, facilitating 10 to 80 VMs per hub. Reproductions used homogeneous PMs with 8-center computer chips, 32 GB Smash, and 1 Gbps NICs. Entropy and Rest utilized assistance hubs, with intermittent summons like clock hour. The Nap progressive methodology conveyed one Gathering Administrator (GM) per 32 Neighborhood Regulators (LCs).

1) General Comparison The figure shows total infringement times and calculation/reconfiguration terms. Entropy, being concentrated, battled with versatility, showing expansion in larger situations. Rest performed better because of its progressive construction, yet DVMS beat both, profiting from dynamic and confined apportioning. Overall, DVMS exhibits the best scalability: average violation duration and its variance remain low as the infrastructure grows, whereas Entropy degrades markedly with size due to centralized computation overhead. sits in between—its hierarchy reduces central bottlenecks but still incurs coordination overhead at larger scales.

2) Algorithm Variations and Insights We further investigated Nap's presentation with differing LC bunch sizes (2, 4, 8, 32 LCs per GM). More modest gathering sizes brought about higher infringement times because of asset deficiency, while bigger gatherings further developed reconfiguration achievement however expanded calculation times. DVMS powerfully adjusted these compromises by changing parcel sizes in view of the burden.

VMPlaceS empowered far-reaching assessment of calculation variations, measurements like relocation above, and versatility, supporting reproductions of up to 8K PMs and 80K VMs. Future work incorporates refining the Nap convention and further developing reenactment effectiveness as a team with SimGrid engineers.

C. Results and Discussion

a) Centralized (Entropy). As infrastructure size grows, the single decision point becomes a bottleneck: invocation time increases and violation durations widen due to slower reconfiguration start times. This is visible in the upward trend of computation time with scale and the larger spread in violation durations.

b) Hierarchical (Snooze). The hierarchy alleviates the central bottleneck by distributing monitoring and decision-making. We observe substantially lower computation time than centralized control at medium and large scales, while reconfiguration duration remains near-constant because migration cost is dominated by live-migration mechanics rather than control logic. Varying LC-per-GM shows the trade-off: small groups under-provision local resources (more violations), large groups increase coordination cost (slightly higher computation time), with 8–32 LCs/GM balancing both.

c) Distributed (DVMS). DVMS maintains low violation duration and low variance across scales by forming small, local partitions that solve overloads concurrently. Because each partition reasons over a few hosts, computation time per event stays nearly constant, and reconfigurations start sooner after detection.

d) Practical takeaway. For tens of nodes, centralized suffices; at hundreds–thousands of nodes, hierarchical control is a robust default; for highly dynamic or very large clusters, DVMS offers the best reactivity and stability. These observations align with the measured trends.

VIII. Conclusion

In this work, we have introduced *VMPS*, a far-reaching and versatile system intended to offer fundamental help for the turn of events, reenactment, and examination of Virtual Machine (VM) position calculations. The structure offers broad capacities for characterizing nonexclusive VM arrangement procedures, executing huge scope recreations of these systems across different platforms, and performing follow based examinations to assess the adequacy of various calculations. The adaptability of *VMPS* empowers it to take care of an extensive variety of use cases, from scholarly research to genuine applications, including complex virtualized conditions.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

One of the critical commitments of this work is the approval of the system's precision and unwavering quality. To accomplish this, we played out an itemized examination between the consequences of reproduced VM situations and genuine (*in-vivo*) executions utilizing the Entropy position system. The approval was granted on conditions containing up to 32 Actual Machines (PMs) and 192 Virtual Machines (VMs). The outcomes showed that the reenactments given by *VMPS* firmly paired these present reality executions, approving the system's capacity to display VM arrangement ways of behaving precisely. This approval is significant as it guarantees clients of the system's reliability when utilized for additional trial and error and enhancement of VM position calculations.

Besides, we have featured the flexibility and meaning of *VMPS* by assessing an assortment of VM position calculations that address three significant classes of virtualization conditions: brought together, various leveled, and completely circulated. These position systems are usually utilized across various ventures and examination fields. Our assessment cycle included running tests in conditions with arrangements that included up to 1,000 hubs and 10,000 VMs, giving important insights into the performance of these calculations when increased to large-scale frameworks. The near investigation led as a component of this study is the first of its sort to deliberately assess these different VM situation calculations at such a large scope, offering a complete outline of their assets, limits, and reasonableness for various virtualization situations.

Looking forward, we are effectively teaming up with the center engineers of the *SG* stage to additional improve the capacities of *VMPS*. This coordinated effort means to expand the system's versatility, with a definitive objective of empowering recreations in conditions containing up to 100,000 PMs and 1,000,000 VMs. This scaling exertion will permit scientists and experts to show significantly greater frameworks, working with the investigation of VM position methodologies with regards to large-scale server farms and cloud conditions.

As well as increasing the system, we are additionally dealing with broadening the recreation capacities of *VMPS*. In particular, we intend to consolidate new aspects that record for extra responsibility varieties, for example, fluctuations in network conditions and hard disk drive (HDD) input/output (I/O) execution. By including these variables, we can make more practical recreation situations that better address the intricacy of genuine world virtualized frameworks, where network blockage, I/O bottlenecks, and other execution requirements frequently assume a critical part in the effectiveness of VM positions.

One more interesting area of future work includes the improvement of a committed Programming interface for dynamic VM provisioning and removal during the execution of reproductions. This Programming interface will permit clients to change the reenactment climate continuously, adding or eliminating VMs on a case by case basis, and giving a more significant level of flexibility and authenticity in demonstrating dynamic responsibilities. This element will likewise empower the testing of VM position calculations in additional versatile and reasonable situations, where VM relocation and responsibility rebalancing are common functional requirements.

All in all, *VMPS* addresses a critical headway in the field of VM situation reproduction and examination, offering an adaptable and versatile stage for scientists, designers, and specialists working with virtualized conditions. The system's capacity to precisely reenact and dissect VM arrangement procedures at scale gives important bits of knowledge into the performance and versatility of various calculations. By proceeding to advance *VMPS* and tending to enter difficulties in the reproduction of huge scope foundations, we mean to additional upgrade the comprehension of virtualization elements and improve the effectiveness of future cloud and server farm frameworks. With these continuous turns of events, *VMPS* holds extraordinary commitment as a useful asset for improving VM position calculations in complex and asset escalated virtual conditions.

References

1. "CloudStack, Open Source Cloud Computing," http://cloudstack.apache.org.
2. "Open Source Data Center Virtualization," http://www.opennebula.org.
3. "The Open Source, Open Standards Cloud," http://www.openstack.org.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 December 2025 doi:10.20944/preprints202512.2039.v1

4. R. Moreno-Vozmediano *et al.*, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *Computer Journal*, vol. 45, no. 12, Mar. 2012.
5. R. Birke *et al.*, "Multi-resource characterization and their (in)dependencies in production datacenters," in *IEEE NOMS'14*, May 2014.
6. E. Feller *et al.*, "Snooze: A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds," in *IEEE CCGRID'12*, May 2012.
7. F. Hermenier *et al.*, "Entropy: A Consolidation Manager for Clusters," in *VEE'09: Virtual Execution Environments*. New York, NY, USA: ACM, 2009.
8. F. Quesnel, A. Lebre, and M. Südholt, "Cooperative and Reactive Scheduling in Large-Scale Virtualized Platforms with DVMS," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1643–1655, Aug. 2013.
9. H. N. Van, F. Tran *et al.*, "SLA-aware virtual resource management for cloud infrastructures," in *IEEE CIT'09*, Oct. 2009.
10. M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *IEEE INFOCOM'11*, Apr. 2011.
11. A. Barker *et al.*, "Academic Cloud Computing Research," in *6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 2014)*, Jun. 2014.
12. F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE)*, 2009, pp. 41–50.
13. F. Hermenier, J.-M. Menaud, and G. Muller, "Bin packing with reordering for virtual machine placement," *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS)*, pp. 427–434, 2011.
14. E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 482–489.
15. F. Quesnel, E. Caron, and C. Morin, "Dvms: A scalable, efficient and flexible scheduling approach for virtual machines," in *Proceedings of the IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2013, pp. 131–138.
16. H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms with simgrid," *Journal of Cluster Computing*, vol. 17, no. 4, pp. 803–824, 2014.
17. T. Hirofuchi, G. Pierre, and E. Caron, "Simgrid vm: Virtual machine support for a simulation framework of distributed systems," in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2013, pp. 546–551.
18. H. Casanova *et al.*, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Parallel and Distributed Computing*, vol. 74, no. 10, Jun. 2014.
19. "Simgrid publications," http://simgrid.gforge.inria.fr/Publications.html.
20. T. Hirofuchi, A. Lebre, and L. Pouilloux, "Adding a live migration model into simgrid: One more step toward the simulation of infrastructure-as-a-service concerns," in *CloudCom'13: Cloud Computing Technology and Science*. IEEE, 2013.
21. F. Hermenier, S. Demassey, and X. Lorca, "Bin Repacking Scheduling in Virtualized Datacenters," in *CP'11: Constraint Programming*, ser. LLNCS. Springer, 2011.
22. "Snooze web site," http://snooze.inria.fr,

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.