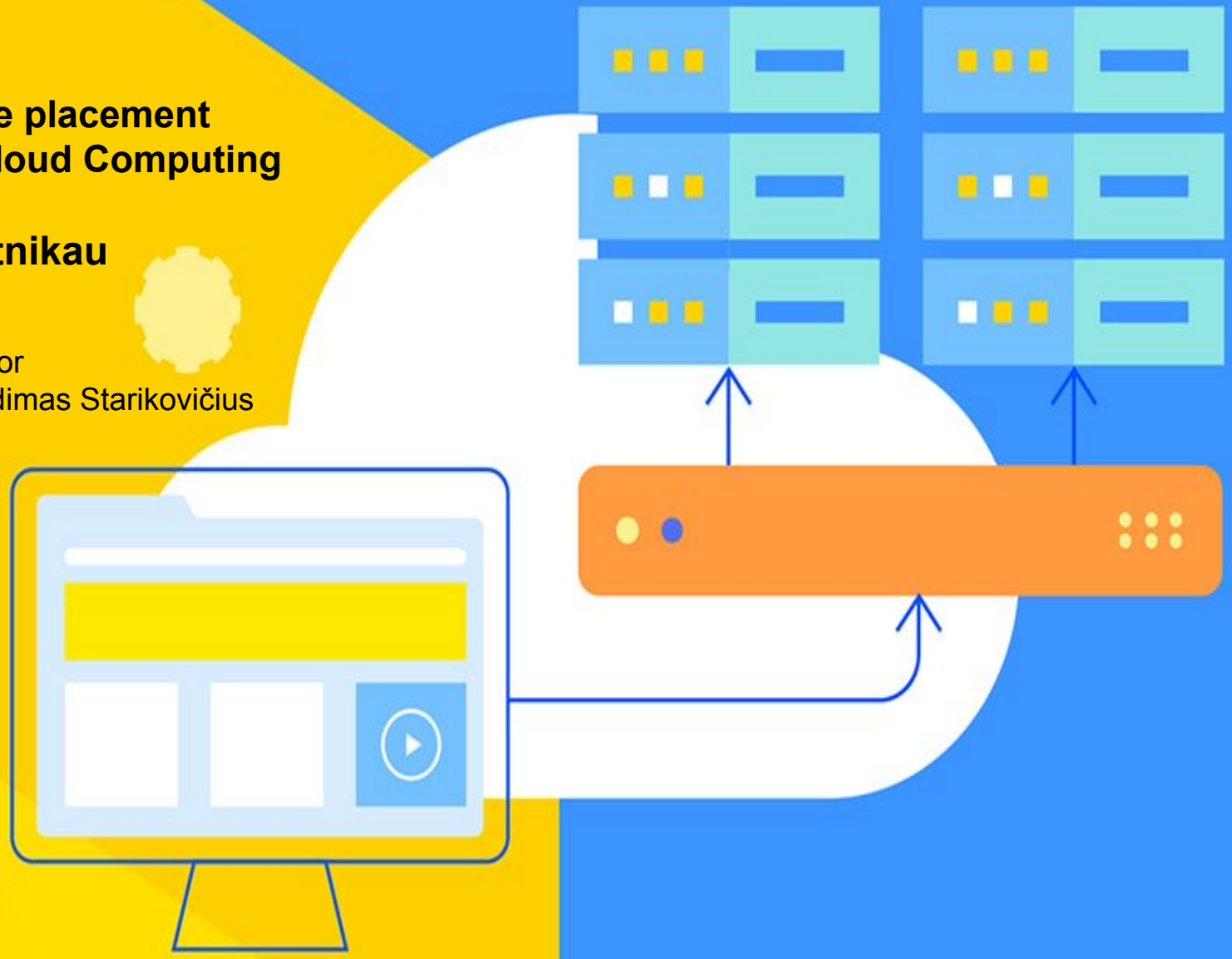
Virtual Machine placement balancing in Cloud Computing

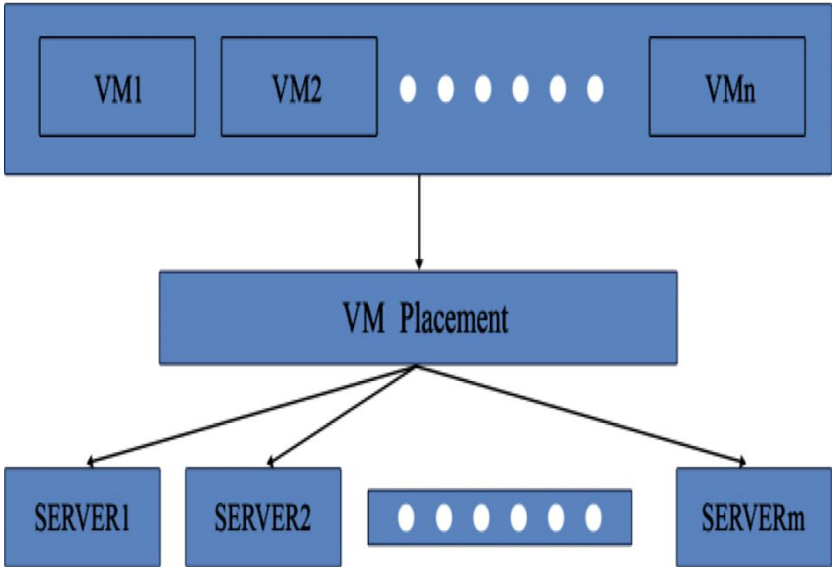
Mikita Syravatnikau

Academic supervisor
Assoc. Prof. Dr Vadimas Starikovičius



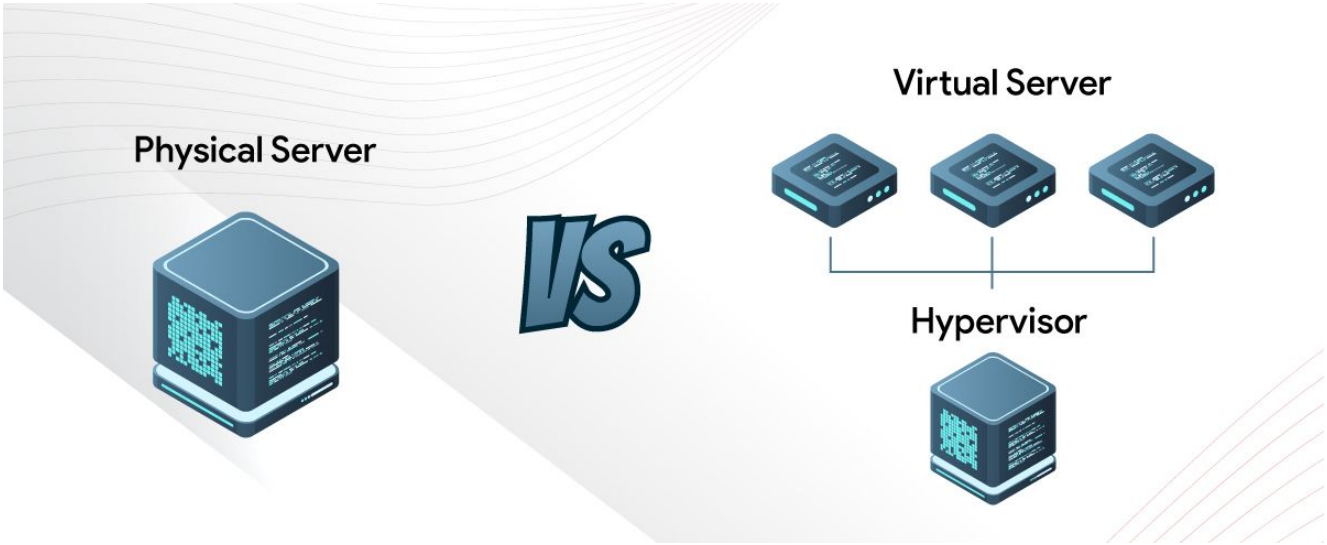
Introduction

Virtual Machine (VM) placement balancing in cloud computing involves optimizing the distribution of VMs across physical servers to enhance resource utilization, reduce energy consumption, and improve overall system performance.



Virtual Machine

A **Virtual Machine** is a software-based emulation of a physical computer that runs an operating system and applications just like a real computer, but it operates within a host system. It allows multiple operating systems to run simultaneously on a single physical machine.



Cloud Computing

VILNIUS TECH

Cloud computing has become a rapidly growing force in recent years. It has allowed quickly adapt and cater to the ever-changing needs of businesses and their employees. The reasons why cloud technology is becoming popular are:

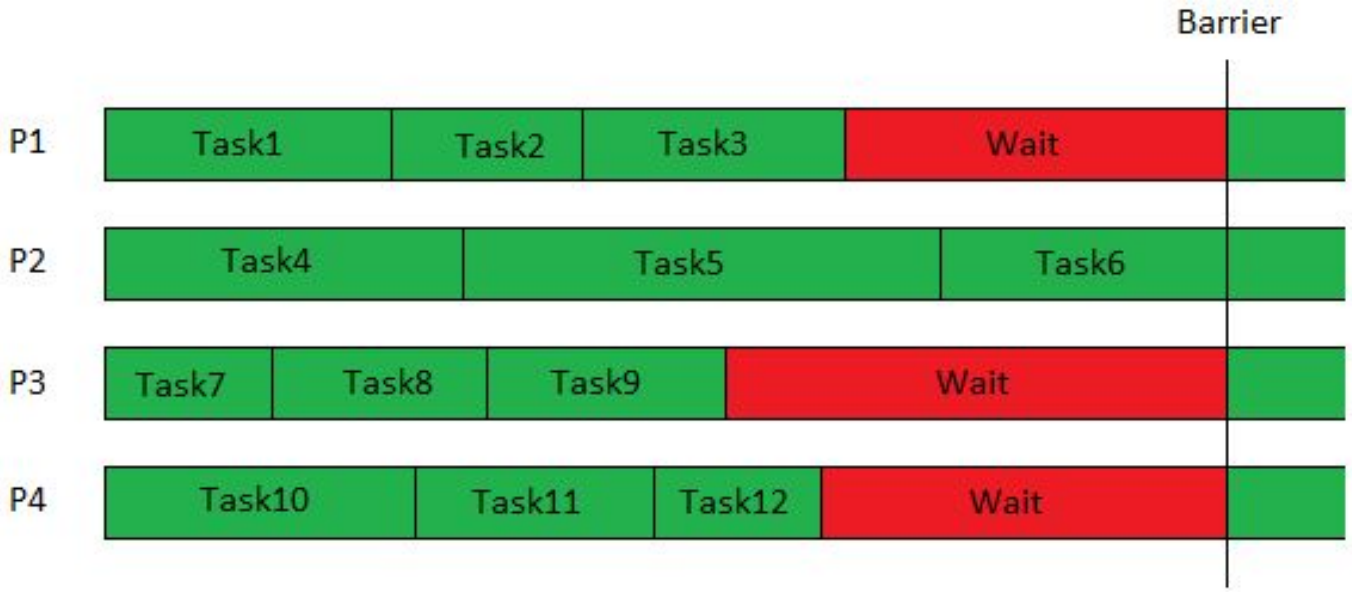
- Flexibility and Scalability
- Mobility
- Cost
- Reliability
- Collaboration
- Convenience

A huge number of IT giants, such as Microsoft, Amazon, Google, DigitalOcean, VMWare receive a huge portion of their income from this market.

Load balancing General Description

VILNIUS TECH

Load balancing is the process of distributing a set of tasks over a set of resources (ex. computing units), with the aim of making their overall processing more efficient. Load balancing can optimize response time and avoid unevenly overloading of some compute nodes while other compute nodes are left idle.



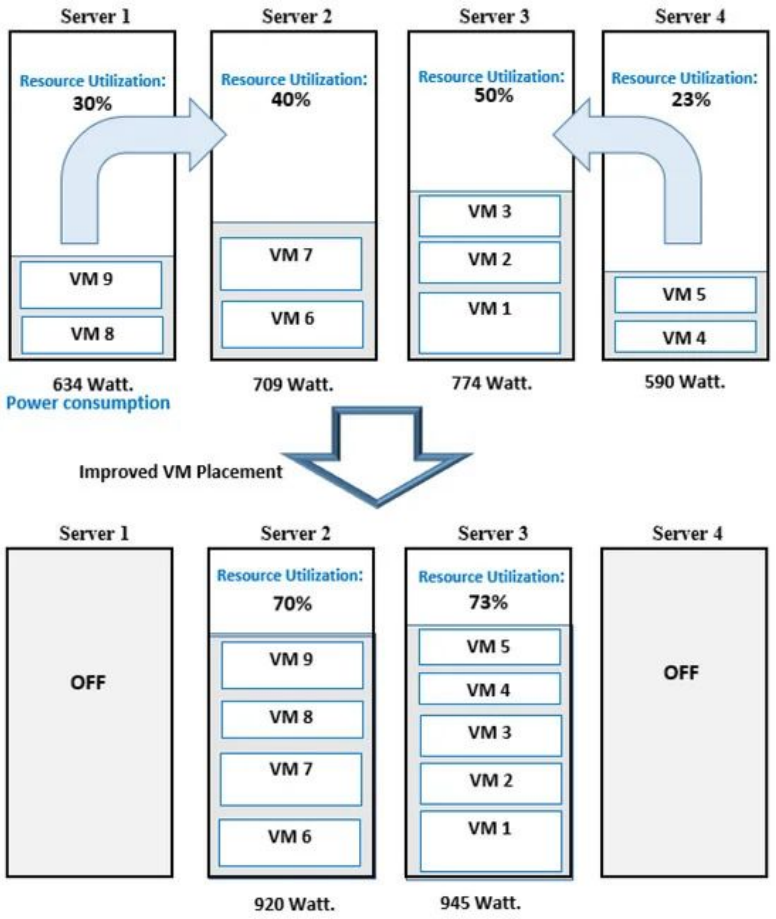
Importance of Load Balancing in Cloud Computing

VILNIUS TECH

Load balancer, is a **key** component in cloud computing.

Correct allocating workload amongst a set of cooperating hosts could significantly improve performance of distributed system.

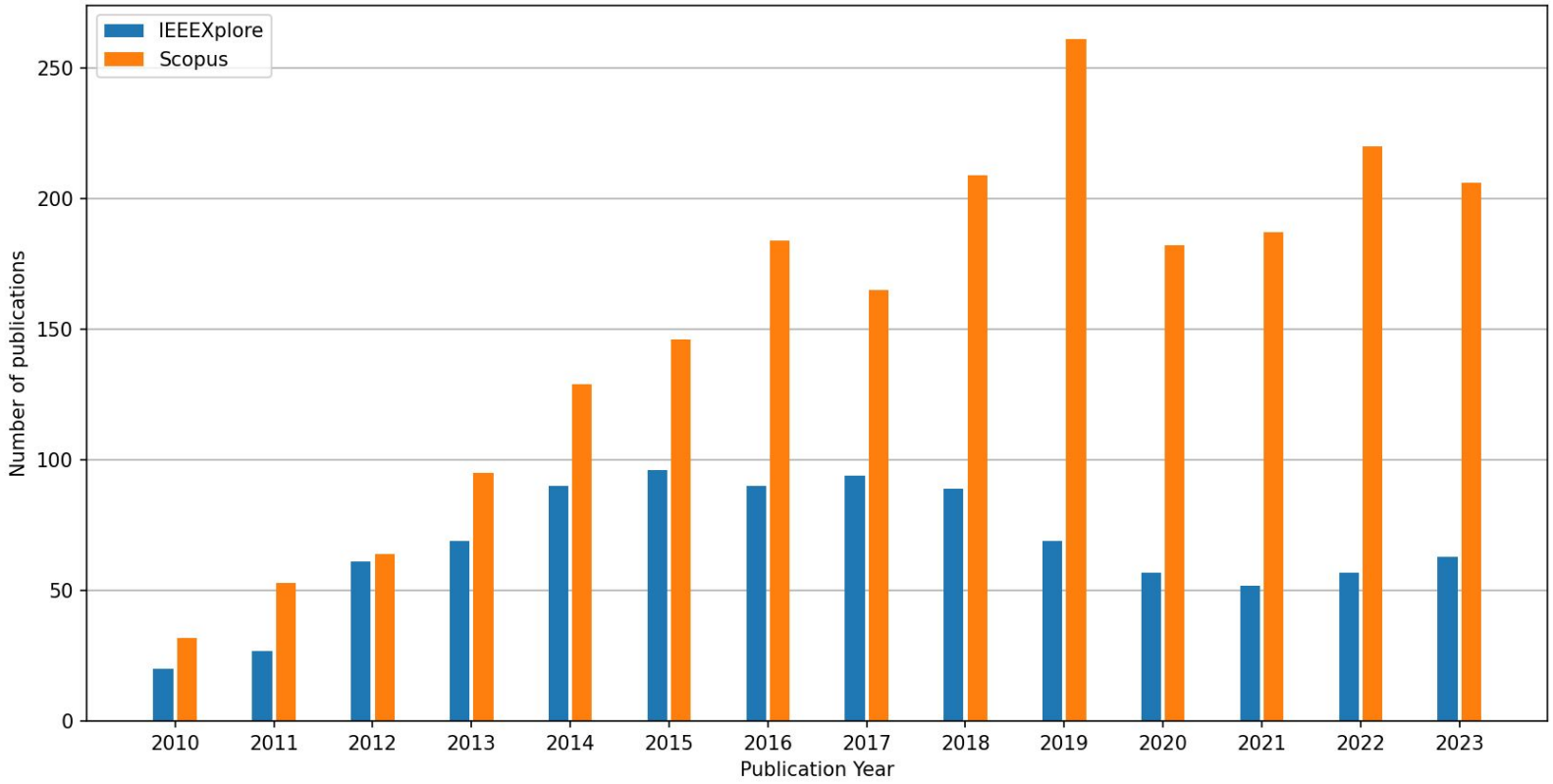
In a real cloud environment balancing of Host Machines is complicated due to frequently occurred addition and removal of Virtual Machines.



Rjeib, H. D., & Kecskemeti, G. (2024). VMP-ER: an efficient virtual machine placement algorithm for energy and resources optimization in cloud data center. Algorithms, 17(7), 295. https://doi.org/10.3390/a17070295

Citations in scientific literature

VILNIUS TECH



Number of research articles returned for “Virtual Machine AND Balancing AND Cloud" query by year in Scopus and IEEEXplore bibliometric databases.

VM Placement balancing general definition.

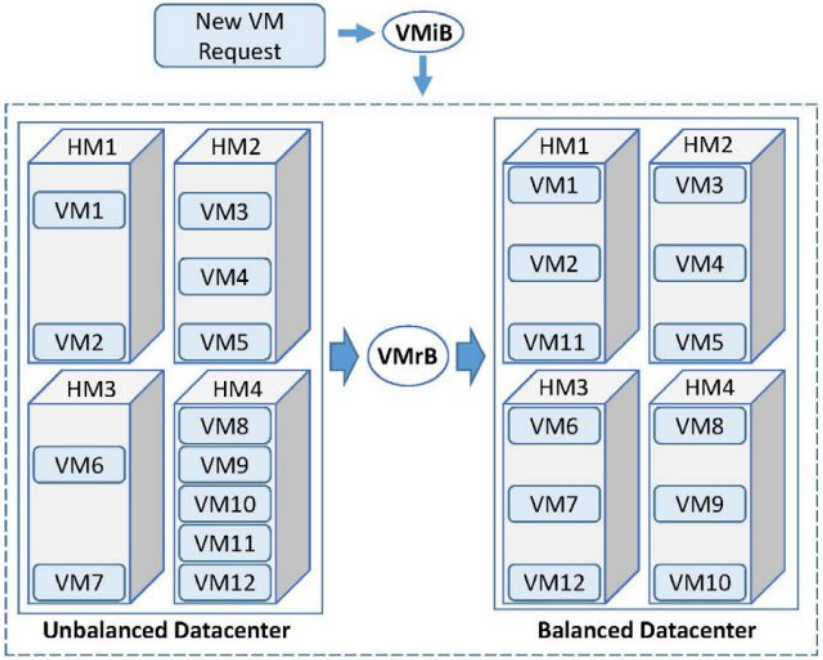
Data Center (DC) has a number of Host Machines(HMs).

These HMs is used to run VM. Single HM can run multiple VM.

To prevent overloading or underloading two types of load balancing exists:

- Incremental VM load Balancing (VMiB)
- VM load re-Balancing (VMrB)

VMrB is possible with use of technique called Virtual Machine Live Migration.



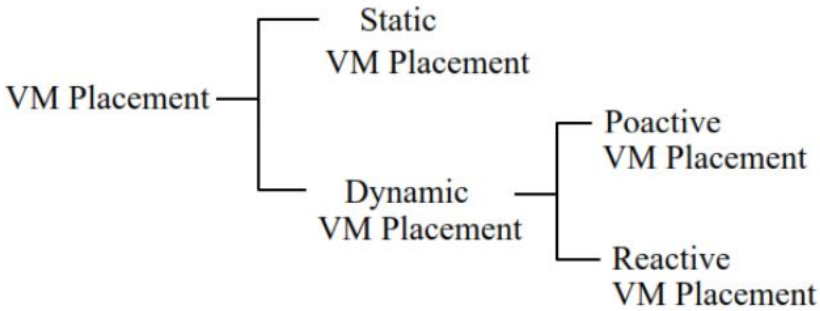
Li, R., Zheng, Q., Li, X., & Yan, Z. (2017a). Multi-objective optimization for rebalancing virtual machine placement. Future Generation Computer Systems, 105, 824–842. https://doi.org/10.1016/j.future.2017.08.027

Classification of VM placement schemes

VILNIUS TECH

VM placement schemes can be classified as dynamic and static:

- **Static VM placement**: in which the mapping of the VMs is fixed throughout the lifetime of the VM and it will not recomputed for a long period of time.
- **Dynamic VM placement**: in which the initial placement is allowed to change due to some changes in the system load or etc.

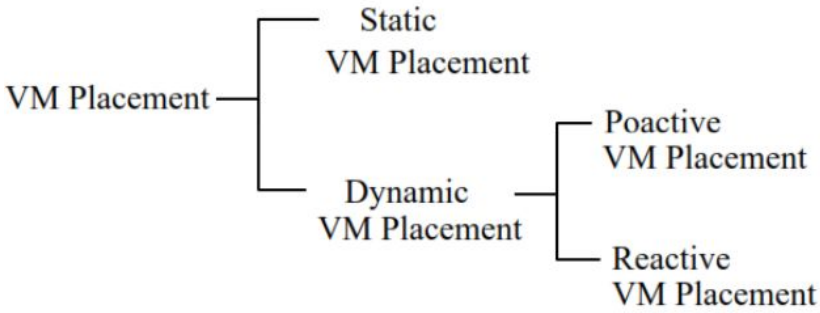


Masdari, M., Nabavi, S. S., & Ahmadi, V. (2016). An overview of virtual machine placement schemes in cloud computing. Journal of Network and Computer Applications, 66, 106–127. https://doi.org/10.1016/j.jnca.2016.01.011

Classification of dynamic VM placement algorithms

The dynamic VM placement algorithms can be categorized as reactive and proactive:

- **Reactive VM placement**: which makes change to an initial placement after the system reaches a certain undesired state.
- **Proactive VM placement**: which changes the VM's Physical Machine (PM) of an initial placement before the system reaches a certain condition.



Masdari, M., Nabavi, S. S., & Ahmadi, V. (2016). An overview of virtual machine placement schemes in cloud computing. Journal of Network and Computer Applications, 66, 106–127. https://doi.org/10.1016/j.jnca.2016.01.011

Mathematical definition of VM placement Balancing problem

VILNIUS TECH

Table 1: Basic notations

Symbols	Description
M	Number of physical machines (PM)
N	Number of virtual machines (VM)
l_j^{res}	Load of j-th VM in resource, like CPU (l_j^{cpu}), Memory (l_j^{mem}), Network load (l_j^{net})
r_j^{res}	Required load j-th VM in resource res
$\hat{l}_j^{res}$	Limit of load of j-th VM in resource res specified during placemnt.
L_i^{res}	Load of j-th PM in resource res
$\hat{L}_i^{res}$	Limit of i-th PM in resource res
$P_{m,n} = (p_{i,j})$	$(p_{i,j})_{1 \leq i \leq M, 1 \leq j \leq N}$ - placement matrix. If virtual machine j is running on physical host i, then $p_{i,j} = 1$, otherwise $p_{i,j} = 0$

Notes

VILNIUS TECH

Obviously:

$$\forall j \in [1, N], \sum_{i=1}^{i=M} p_{i,j} = 1$$

In real environment determine the load of *i*-th PM with only load of all VM running on this PM impossible. It is also necessary to take into account the load created by the server operating system, the load created by ensuring the life cycle of the VM, and so on. So generally speaking:

$$L_i^{res} \neq \sum_{j=1}^{j=N} p_{i,j} * l_i^{res}$$

Mathematical definition of VMiB

VILNIUS TECH

Given:

- A set of physical machines $H = \{h_1, h_2, ..., h_m\}$, where each physical machine $h_i = \{\{L_i^{res}, \hat{L}_i^{res}\}, res \in \{cpu, mem, ...\}\}$ has set of resources L_i^{res}, such as CPU, memory, and storage and an associated capacity $\hat{L}_i^{res}$, where $L_i^{res} < \hat{L}_i^{res}, \forall res$
- A set of virtual machines $V = \{v_1, v_2, ..., v_n\}$, where each VM $v_j = \{\{r_i^{res}, \hat{l}_i^{res}\}, res \in \{cpu, mem, ...\}\}$ has resource demand r_i^{res} representing the resource requirements for CPU, memory, etc. And resource limit $\hat{l}_i^{res}$.
- An incremental VM arrival sequence, where VMs arrive one by one over time, and each VM needs to be placed on an available host that satisfies its resource requirements.

Mathematical definition of VMiB

VILNIUS TECH

- The balance function $f(H, V, P_{m,n}, ...)$, which measures how well the system is balanced. A typical balance function could penalize resource utilization imbalances across hosts, such as the difference in CPU or memory utilization between hosts.

Mathematical definition of VMiB

VILNIUS TECH

The goal is to place each incoming VM v_j on an appropriate host such that:

- **Resource constraints:** For each VM v_j placed on host h_i, the resource usage on host h_i after placement must not exceed its capacity. That is:

$$L_i^{res} + r_j^{res} \leq \hat{L}_i^{res}$$

- **Balance objectives:** Minimize a balance metric that reflects the distribution of resources across hosts. A common objective is to minimize the maximum resource utilization or the imbalance across hosts. For example:

$$\text{Minimize } \max_i \left(\frac{L_i^{cpu}}{\hat{L}_i^{cpu}}, \frac{L_i^{mem}}{\hat{L}_i^{mem}}, \frac{L_i^{net}}{\hat{L}_i^{net}} \right)$$

This would aim to minimize the worst-case resource utilization across all hosts.

Transition to real world. Overcommitting

VILNIUS TECH

Overcommit in cloud computing refers to the practice of allocating more resources (e.g., CPU, memory, storage) to virtual machines than are physically available on the underlying host hardware. This is based on the assumption that not all VMs or workloads will utilize their full allocated resources at the same time, allowing the cloud provider to run more VMs than would fit if each VM were given exclusive access to its allocated resources.

Resource constraints with overcommitting.

VILNIUS TECH

In almost all IaaS systems overcommit coefficient the only value which control this feature. In mathematical definition overcommit coefficient can be described as $c^{res} \geq 1$. If $c^{res} = 1$ the resource res will not be overcommitted. Otherwise the resource constraints can be described by following equation:

$$\sum_{j=1}^{j=N} p_{i,j} \hat{l}_i^{res} \leq c^{res} \hat{L}_i^{res}$$

In simple words, if $c^{mem} = 2$ and we have host with 2 GB RAM, we allow to deploy 4 virtual machines with 1 GB RAM resource demand on host host.

Balance function

VILNIUS TECH

A review of the literature on the topic revealed a serious problem: There is no clear mathematical definition of the balancing function that all authors agree on. However, many authors agree that the most important aspects of balancing are the following:

- **Resource Utilization**: Ensures that CPU, memory, and storage resources are evenly distributed across physical servers, maximizing the use of available resources while minimizing wastage.
- **Performance Optimization**: By placing VMs based on workload characteristics and resource demands, organizations can enhance application performance and responsiveness.
- **Energy Efficiency**: Effective placement can reduce energy consumption by consolidating workloads onto fewer servers when demand is low, allowing others to be powered down.
- **Fault Tolerance and High Availability**: By strategically placing VMs, the risk of server failure impacting critical applications can be minimized. This approach can enhance disaster recovery strategies as well.
- **Dynamic Placement**: Advanced load balancers or orchestration tools can dynamically place VMs based on real-time monitoring of resource usage, traffic patterns, and performance metrics. This allows for responsive adjustments as demand changes.

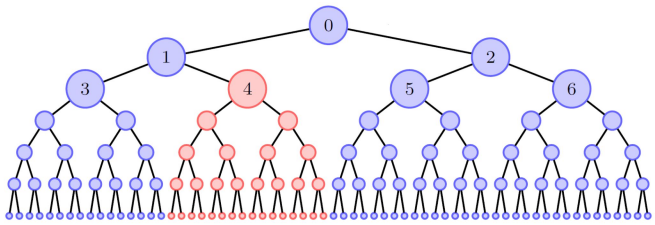
Existing methods for solving problem

VILNIUS TECH

Multi-objective nonlinear programming-based VM placement

Consider the heterogeneous servers and the random requirements of the VMs and model the efficient and cost-aware VM placement as a multi-objective nonlinear programming.

- Zhang, J., et al. SLA aware cost efficient virtual machines placement in cloud computing. in Performance Computing and Communications Conference (IPCCC), 2014 IEEE International. 2014. IEEE.
- Li, R., Zheng, Q., Li, X., & Yan, Z. (2017b). Multi-objective optimization for rebalancing virtual machine placement. Future Generation Computer Systems, 105, 824–842.



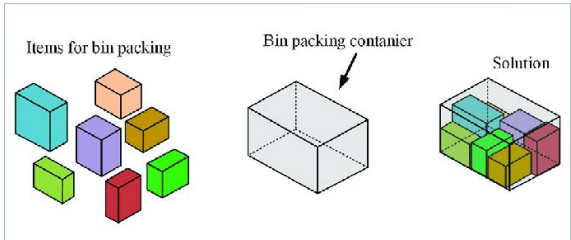
Existing methods for solving problem

VILNIUS TECH

Bin packing-based VM placement

Some VM placement schemes consider the VM placement problem as a bin packing problem which is an extension of the first fit decreasing.

- Xiaoli, W. and L. Zhanghui. An energy-aware VMs placement algorithm in Cloud Computing environment. in Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on. 2012. IEEE.
- Fatima, A., Javaid, N., Sultana, T., Hussain, W., Bilal, M., Shabbir, S., Asim, Y., Akbar, M., & Ilahi, M. (2018). Virtual machine placement via bin packing in cloud data centers. Electronics, 7(12), 389.



Existing methods for solving problem

VILNIUS TECH

Heuristic-based VM placement (one of the most popular)

Heuristic-based approaches for virtual machine (VM) placement balancing involve using rules or approximate methods to achieve a good-enough solution for the distribution of VMs across physical hosts in a data center, without necessarily guaranteeing an optimal solution.

- Cho, KM., Tsai, PW., Tsai, CW. et al. A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput & Applic 26, 1297–1309 (2015).
- X. Xu, Q. Liu, L. Qi, Y. Yuan, W. Dou and A. X. Liu, "A Heuristic Virtual Machine Scheduling Method for Load Balancing in Fog-Cloud Computing,"

Existing methods for solving problem

VILNIUS TECH

ML-based VM placement

Popular approaches: Supervised Learning, Reinforcement Learning, Unsupervised Learning.

- Talwani, Suruchi, Alhazmi, Khaled, Singla, Jimmy, Alyamani, Hasan J and Bashir, Ali Kashif. Allocation and migration of virtual machines using machine learning. Computers, Materials and Continua, 70 (2). pp. 3349-3364. ISSN 1546-2218
- Li, X., Pan, L. & Liu, S. A DRL-based online VM scheduler for cost optimization in cloud brokers. World Wide Web 26, 2399–2425 (2023).
- Caviglione, L., Gaggero, M., Paolucci, M. et al. Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters. Soft Comput

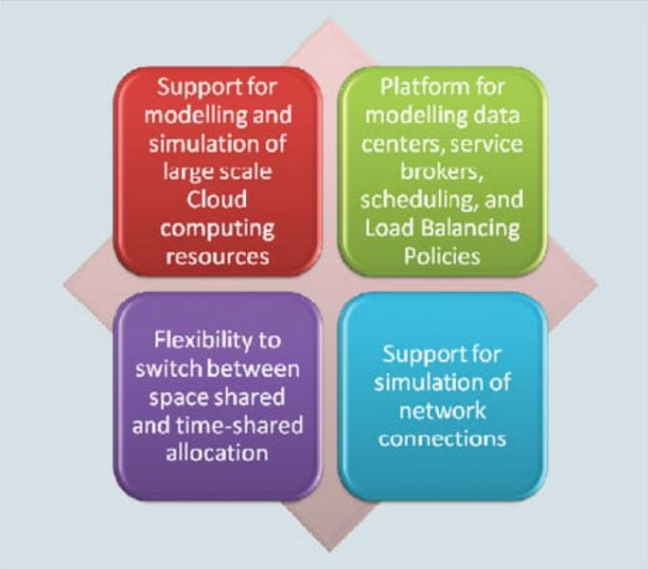
Simulation with CloudSim

VILNIUS TECH

CloudSim is developed in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, at the Computer Science and Software Engineering Department of the University of Melbourne. Available at www.cloudbus.org/cloudsim

CloudSim provides a generalised and extensible simulation framework that enables seamless modelling and simulation of app performance.

- Flexibility of defining configurations
- Ease of use and customisation
- Cost benefits
- Open-Source
- Well-Documented



Simple self-developed simulation framework

VILNIUS TECH

CloudSim great solution, but this framework does not allow you to experiment on your own data without heavily interfering with the source code. Moreover, the main focus of this framework is on simulating the request load.

As part of my work, I implemented a fairly simple framework for simulating and testing Virtual Machine placement balancing algorithms. One of the advantages of this framework is the ability to repeat the experiment on a given dataset.

For example, if there is a dataset containing information about the resource utilization of servers and their virtual machines and their parameters, using the framework, you can simulate the operation of this cluster.

Simple self-developed simulation framework (synthetic load)

VILNIUS TECH

Simple synthetic load VM behaves similarly to Desktop Virtual Machine. It has two states **burst** and **idle**. The key parameters are following:

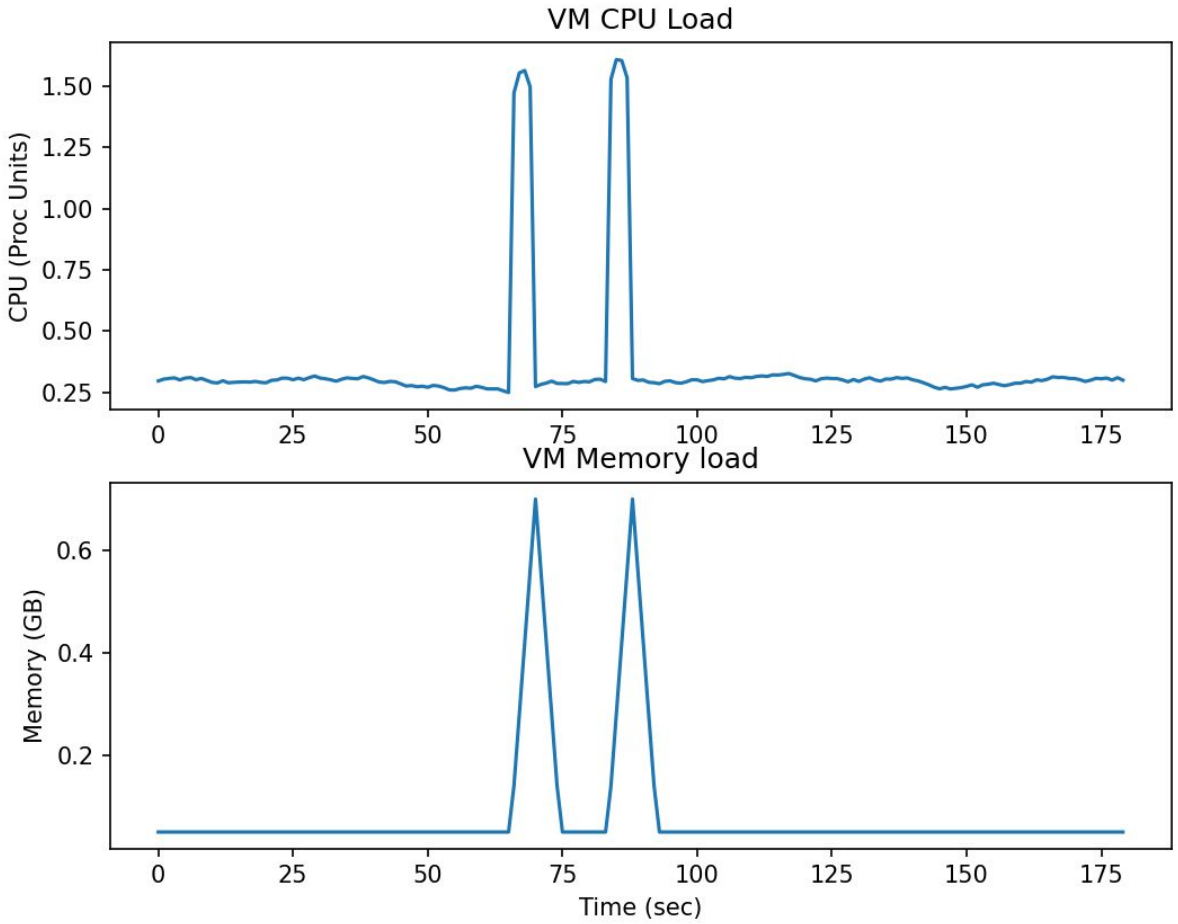
- **burst prob interval** i - interval in which burst occurs with 50% chance
- **load dispersion** $a^{res} \in [0, 1)$ - random value added to **burst** or **idle** load.
  Example: $l^{res} = max(cur\_load + rand(0, a^{res})\hat{l}^{res}, \hat{l}^{res})$
- **burst** res **duration** $b^{res}_{burst} \in (0, 1)$ - duration of burst for resource res
- **burst** res **load** $c^{res}_{burst} \in [0, 1]$ - cur_load during burst state
  Example: $cur\_load = c^{res}_{burst}\hat{l}^{res}$
- **idle** res **load** $c^{res}_{idle} \in [0, 1]$ - cur_load during idle state
  Example: $cur\_load = c^{res}_{idle}\hat{l}^{res}$

For each time t VM Load Simulator desire will it in **burst** or **idle** state

Simple self-developed simulation framework (synthetic load)

VILNIUS TECH

$$d = 0.1$$
$$\hat{l}^{cpu} = 2$$
$$\hat{l}^{mem} = 1$$
$$i = 30$$
$$a^{cpu} = 0.1$$
$$a^{mem} = 0$$
$$c_{burst}^{cpu} = 5$$
$$c_{burst}^{mem} = 10$$
$$c_{burst}^{cpu} = 0.8$$
$$c_{burst}^{mem} = 0.7$$



MAX-BRU

VILNIUS TECH

Nguyen Trung Hieu, Mario Di Francesco proposed Max-BRU (Maximized Balanced Resource Utilization) algorithm that optimizes the resource utilization and balances the usage of resources across multiple dimensions, so that the total number of active servers is minimized.

This algorithm is pretty simple, and operates with static VM resource utilization.

MAX-BRU

VILNIUS TECH

Algorithm 1: Max-BRU(V)

Output: $M,\ RU,\ RB$

Set $P \leftarrow \emptyset$;
Set $M \leftarrow 0$;
while $V \neq \emptyset$ **do**
  Set $status \leftarrow false$;
  if $P \neq \emptyset$ **then**
    $p \leftarrow$ FINDMOSTSUITABLESERVER(P);
    $v \leftarrow$ FINDPOTENTIALVM(p, V);
    if $v \neq \varnothing$ **then**
      Place VM v on physical server p;
      Update RU^d and RB metrics of p;
      $V \leftarrow V \setminus \{v\}$;
      Set $status \leftarrow true$;
  if $status = false$ **then**
    Remove VM request v from V in FIFO order;
    Start-up a new physical server p_{new};
    Place VM v on p_{new};
    Compute RU^d and RB metrics of p_{new};
    $P \leftarrow P \cup \{p_{new}\}$;
    Increase M by 1;
$RU \leftarrow \frac{1}{M} \sum_{i=1}^{M} \min_{d \in \{1,\ldots,|D|\}} RU_i^d$;
$RB \leftarrow \frac{1}{M} \sum_{i=1}^{M} RB_i$;
return $M,\ RU,\ RB$

Resource utilization:

$$RU_i^{res} = \frac{L_i^{res}}{\hat{L}_i^{res}}, i \in \{1, ..., M\}$$

Resource balance:

$$RB_i = \frac{\frac{1}{|res|} \sum_{res} RU_i^{res}}{max_{res} RU_i^{res}}, i \in 1, ..., M$$

Algorithm 2: FINDMOSTSUITABLESERVER(P)

Output: $p \in P$

$$p \leftarrow \begin{cases} \min_{i \in \{1,\ldots,M\}} \max_{d \in \{1,\ldots,|D|\}} RU_i^d; \\ \min_{i \in \{1,\ldots,M\}} RB_i; \end{cases}$$

return p

Algorithm 3: FINDPOTENTIALVM(p, V)

Output: $v = \varnothing$ or $v \in V$

$$v \leftarrow \begin{cases} \max_{j \in \{1,\ldots,N_k\}} r_j^d; \\ r_j^d + u^d(p) + w^d(p) \leq \hat{r}^d(p); d \in \{1, \ldots, |D|\}; \end{cases}$$

return v

Static VM Load

VILNIUS TECH

There are a huge number of scientific publications that consider the problem without taking into account the dynamic load of the virtual machine itself over time. With this approach, following assumptions were taken:

No overcimmitting:

$$c^{res} = 1$$

VM always consumes all required resources:

$$r_j^{res} = \hat{l}_j^{res}$$

The PM resource usage is simply sum of load created by VM deployed on i'th PM and initial load

$$L_i^{res} = \sum_{j=1}^{j=N} p_{i,j} l_i^{res} + I_i^{res},$$

where I_i^{res} is initial load of i'th PM in resource res:

Improved MAX-BRU for dynamic VM Load with overcommitting

VILNIUS TECH

The algorithm operates on the server load, so to apply this algorithm to solve the balancing problem taking into account the dynamic load of the virtual machine, we can simply replace the static load on the server with the instantaneous current load.

We will also try to replace the instantaneous load with the average one. We will call such algorithm **Improved MAX-BRU.**

To satisfy the possibility of overcommit, we will remove resource constraints rule in the auxiliary algorithm *FindPotentialVM.*

Server overload

In the case when when overcommit coefficient $c > 1$ a situation may arise when sum of required VM loads deployed on PM exceed server capacity. This situation can be descried by following equation:

$$\sum_{j=1}^{j=N} p_{i,j} r_i^{res} + I_i^{res} > \hat{L}_i^{res}, i \in \{1, ..., M\}$$

Let assume that $t_i, i \in \{1, ..., M\}$ is set of time intervals where equation above were satisfied.

We denote total overload time as $T_{overload} = \sum_{i=1}^{i=M} \sum t_i$

MAX-BRU

VILNIUS TECH

Let assume we have M Physical Machines, and K Virtual machines, which should be placed on these servers. We denote deployment coefficient S:

$$S^{res} = \frac{\sum_{j \in P} \hat{l}_j^{res}}{\sum_{i=1}^{i=M} \hat{L}_i^{res}},$$

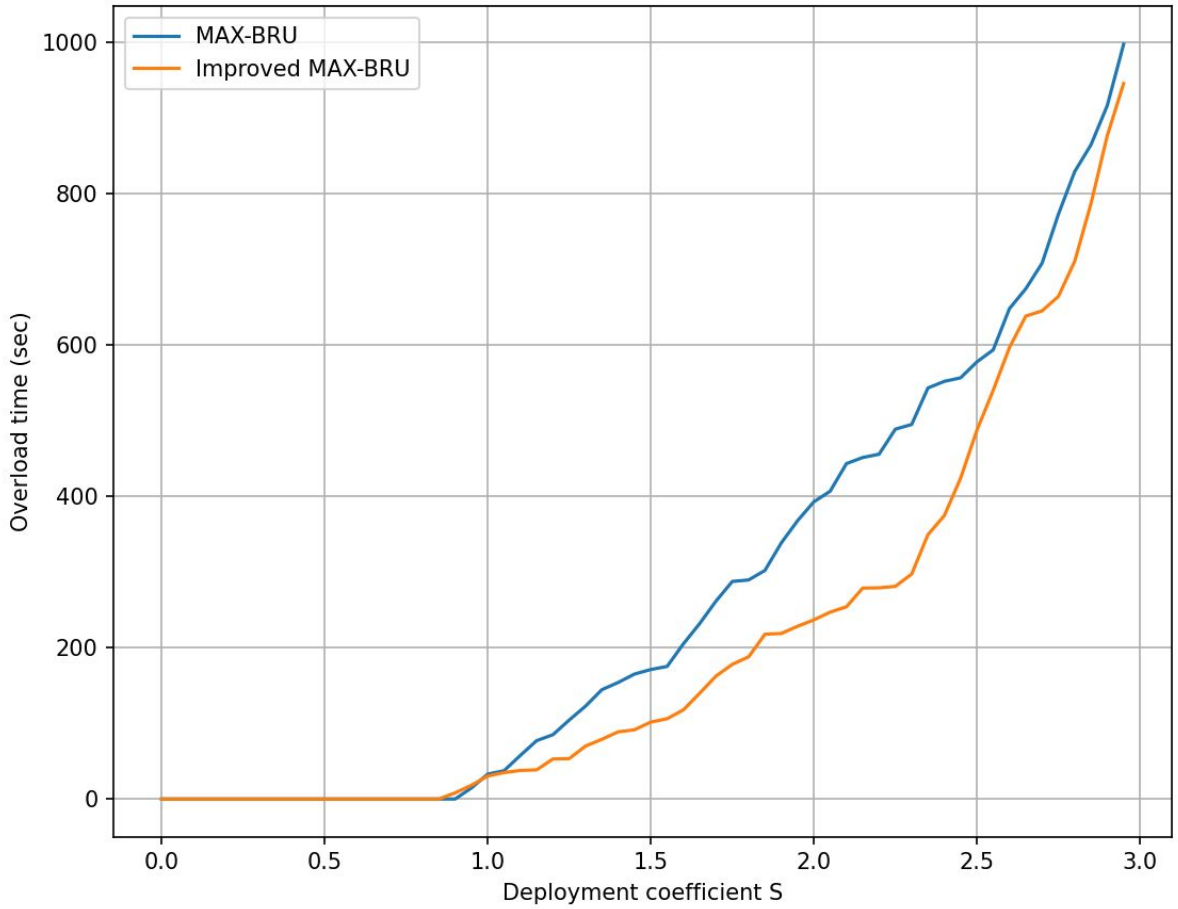
where P is set of virtual machines already deployed.
We perform testing of MAX-BRU algorithm in following conditions:

- Test duration 1 hour
- Overcommit coefficient $c^{cpu} = 3$
- Each VM deploy request arrives in 1 minute
- Total number of VM requests 60

Let's take look at $T_{overload}$ vs S^{cpu} plot.

MAX-BRU vs Improved MAX-BRU

VILNIUS TECH



Issues and challenges

VILNIUS TECH

The conducted analysis of the literature on this topic revealed the following key problems:

- Although the problem is quite clear, it is quite difficult to formalize it. There are many metrics of "good" balancing, but there are no universal ones;
- Comparison of results with other works is complicated by the fact that one of the important elements of the problem is the determining the optimization function itself. However, some basic requirements for all algorithms are the same and many authors agree on some metrics (for example, resource balance);
- There are very few publicly available datasets. Many of these datasets contain only one metric, for example, cpu utilization;

Conclusions

VILNIUS TECH

During this seminar:

- A general description of the load balancing problem was formulated;
- The importance of the load balancing problem was shown;
- A mathematical description of the virtual machine placement balancing problem was formulated;
- A general overview of existing methods for solving this problem was given;
- Some of the existing algorithms are considered in detail;
- A part of original research was shown;
- The problems that researchers face in this topic are considered;

VILNIUS TECH
Vilnius Gediminas Technical University

Thank you!