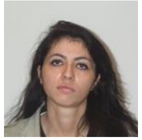
Self-Economy in Cloud Data Centers: Statistical Assignment and Migration of Virtual Machines

Carlo Mastroianni
mastroianni@icar.cnr.it

Giuseppe Papuzzo
papuzzo@icar.cnr.it

ICAR-CNR
Institute for High Performance
Computing and Networks
Cosenza, Italy

Michela Meo
michela.meo@polito.it



Politecnico di Torino
Torino, Italy

Cloud Computing and Data Centers

* Problem: excessive energy consumption!
* 2% of worldwide electrical energy is consumed by IT infrastructures (computers, cooling systems, UPS…)
  * Annual consumption: 120 billions KWh
  * Cost: 10 billions $

Causes of excessive consumption

* Two kinds of inefficiencies:

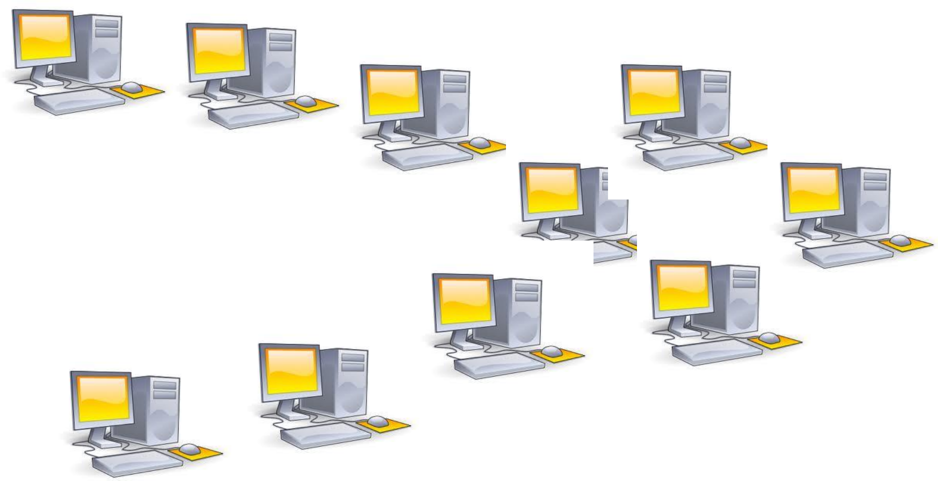
Servers that are switched but under-utilized consume about 70% of the energy they consume when fully utilized

On average only 30% of server capacity is exploited

* Consequences:
  * Increase of costs for companies and therefore for clients
  * Increase of CO_2 emissions

Solution: consolidation of Virtual Machines

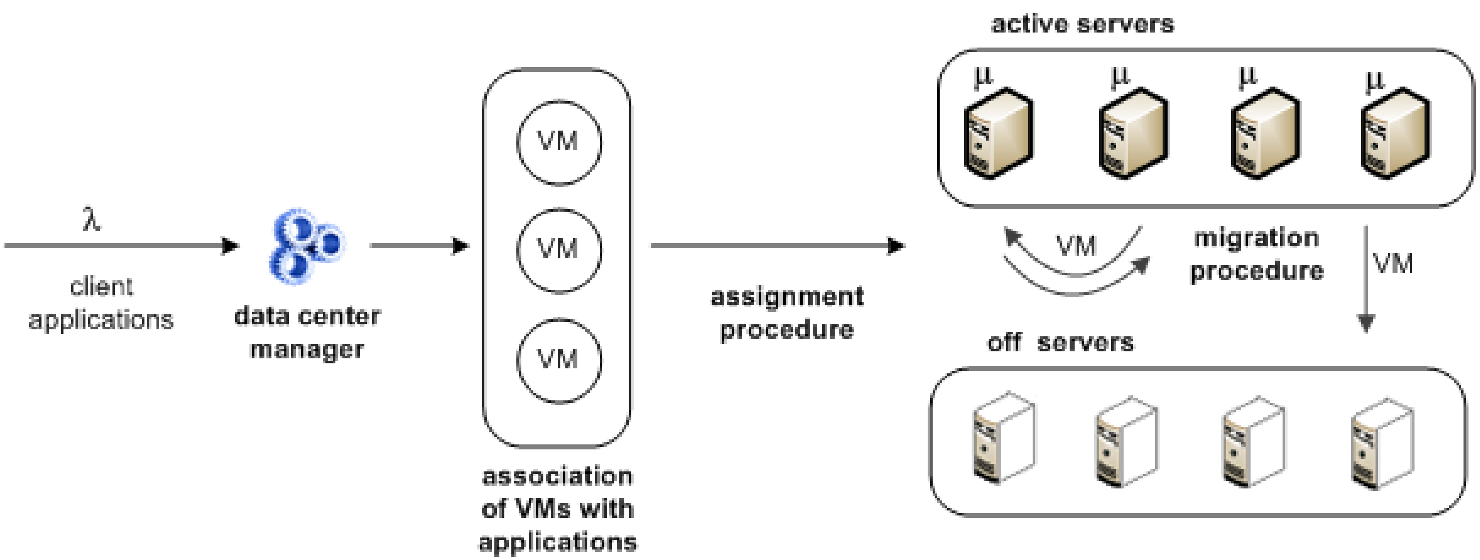
* Consolidate Virtual Machines on the smallest number of servers, so as to switch off the remaining servers
* Solutions available today are centralized, complex, not scalable



We propose a decentralized approach, based on statistical techniques

Statistical consolidation

* Assignment of VMs to servers is driven by statistical trials.
* Decision is **autonomous** on each server and depends on **local** load:
  - Lightly loaded servers tend to reject new VMs, so as to completely unload and switch themselves off
  - Highly loaded servers equally tend to reject new VMs
  - Servers with intermediate load tend to accept new VMs



Advantages

- Energy saving (from 20% to 60%), because under-utilized servers are switched off
- Higher quality of the service offered to users, because overloaded servers are unloaded
- Main decisions are taken on single servers only using information available locally -> scalability
- The consolidation process is self-organizing -> adaptability, fault-tolerance

VM Assignment procedure

1. A client submits an application to the data center
2. The data center manager associates the application with a compatible VM
3. The manager broadcasts an invitation to the servers
4. Each server evaluates the assignment probability function based on the local CPU utilization
5. The server performs a Bernoulli trial to decide whether or not to be available to accept the VM: if available, the server sends a positive ack to the manager
6. The data center manager collects the positive replies of servers and randomly selects the server that will execute the VM

Assignment probability function (1/2)

- The assignment probability is a function of the **CPU utilization u** (with values between **0** and **1**) and of the **threshold Ta**, defined as the maximum allowed utilization (in this paper, **Ta = 0.9**)

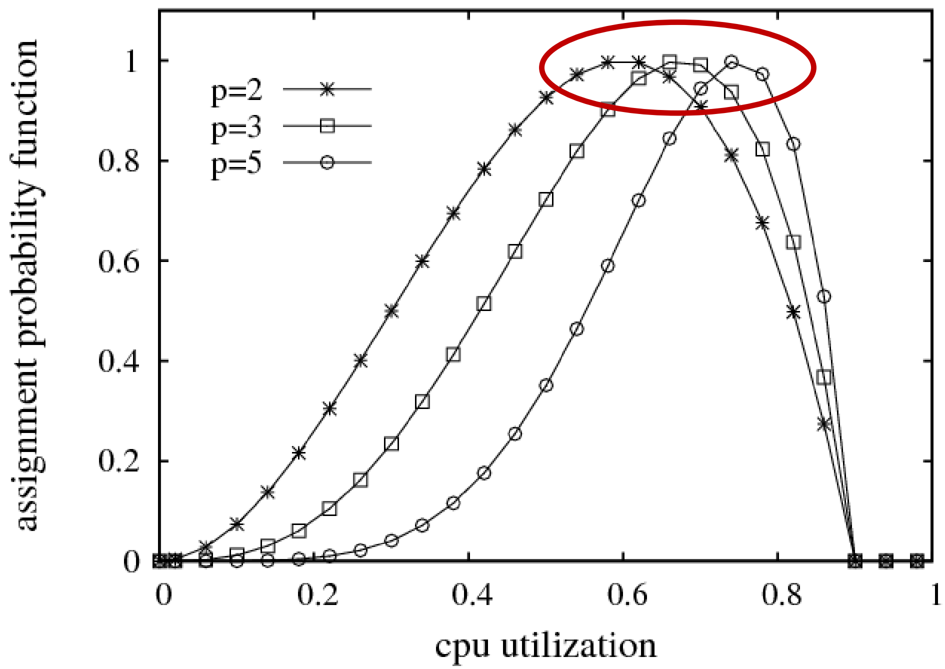
$$f_{assign}(u) = 1/M_p \cdot u^p \cdot (T_a - u)$$

$$M_p = \frac{p^p}{(p+1)^{(p+1)}} \cdot T_a^{(p+1)}$$

- The function assumes a value between **0** and **1**, which is the success probability of the Bernoulli trial
- The parameter **p** is used to tune the function shape (next slide)
- The factor **1/Mp** is used to normalize the function

Assignment probability function (2/2)

- The graph shows that servers with medium or moderately high load are **more likely** to accept new VMs
- The parameter **p** can be used to modulate the function shape: the function reaches its maximum value (**=1**) when **u=p/(p+1) . Ta**



Hibernation and activation of servers

The objective is to keep as many servers as possible in a low power state.

Server hibernation

- An active server in which the last VM terminates or migrates **hibernates itself** to save power.

Server activation

- A sleeping server is **switched on** by the data center manager when a new VM arrives and no server is available to execute it

- What if all the servers are already powered on and no server is available?
  *It means that the system is **undersized**: more servers should be acquired!*

Dynamic workload of VMs

- The CPU amount required by VMs may change frequently
  - *for example, the load of a Web server depends on the number and type of client requests*

So, even after an optimal assignment of VMs, dynamic workload and VMs turnover may lead to:

a) Server **under-utilization**, when some VMs terminate or reduce their demand for CPU

b) Server **overload**, when some VMs increase their demand

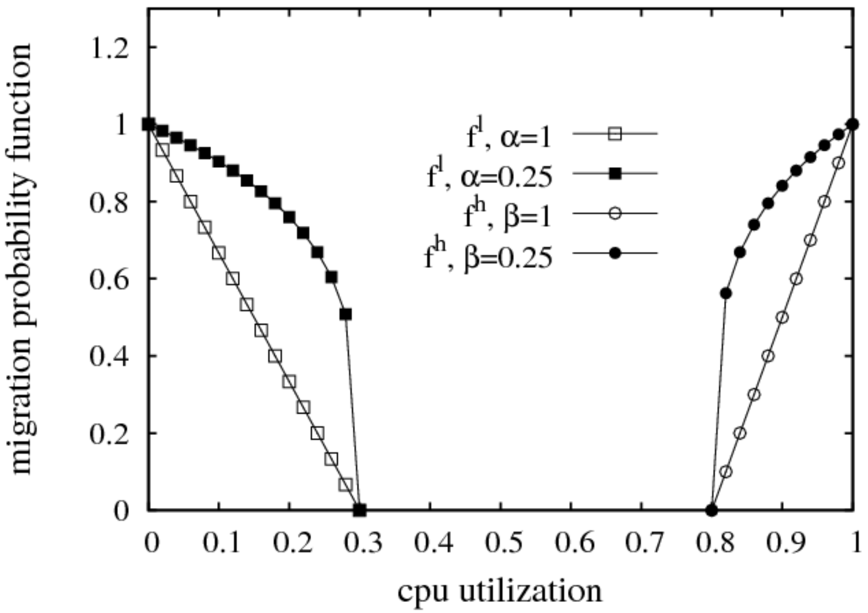
VMs should be migrated both from under-utilized and overloaded servers

VM Migration procedure

1. Periodically each server checks whether its CPU utilization goes below a low threshold **Tl** or above a high threshold **Th**
2. If so, the server evaluates the **migration probability function**
3. A **Bernoulli trial** is performed to decide whether a VM should be migrated
4. If the trial is successful, the server **broadcasts** an invitation to accommodate the VM, or asks the data center manager to do it
5. The rest of the procedure is similar to the assignment procedure, except that the assignment threshold **Ta** is set so as to avoid **ping-pong effects** (*several consecutive migrations of the same VM*)

Migration probability function (1/2)

- The migration probability function is **null** when the server is neither under-utilized or over-loaded (no migration is necessary)
- The function is not null when **u** < **Tl** or when **u** > **Th**
- The function shape can be tuned using parameters **α** and **β**



Migration probability function (2/2)

- If **u** < **Tl** the function is named **f^l** and defined as:

$$f^{l}_{migrate}(u) = (1 - u/T_l)^{\alpha}$$

→ *the Bernoulli trial will trigger a "low migration" procedure*

- If **u** > **Th** the function is named **f^h** and defined as:

$$f^{h}_{migrate}(u) = \left(1 + \frac{u-1}{1-T_h}\right)^{\beta}$$

→ *the Bernoulli trial will trigger a "high migration" procedure*

Analyzed data center

- We implemented an **event-based simulator in Java**
- The evaluated data center has **Ns** = **100** servers. 33 of them have **4 cores**, 34 have **6 cores** and 33 have **8 cores**
- All cores have CPU frequency of **2 GHz.**
- Virtual Machines that host client applications have nominal CPU frequencies of **500 MHz**, **1 GHz** and **2 GHz.**
- **50%** of applications are assigned to **500 MHz** VMs, **25%** to **1 GHz** VMs, and **25%** to **2 GHz** VMs.

Dynamic behavior and traffic load

- Client requests arrive at the data center with frequency **λ** ranging between **1.2** and **24** requests per minute
- The VM execution time is extracted with a Gamma distribution, the average **1/μ** is set to 100 minutes (**μ** is the service rate of a single core)
- The fraction of CPU requested by each VM varies between **0%** and **100%** of the VM nominal capacity
- The overall load of the data center **ρ** is computed as $0.5\ \lambda/\mu_T$, where μ_T is the service rate of the whole data center
- In the experiments, the load **ρ** is varied between **0** and **1**, to test the system in different traffic conditions

results are compared to …

1. **Random strategy**: each VM is assigned to any server with sufficient capacity

   (*it is the naïve choice adopted by many data centers today!*).

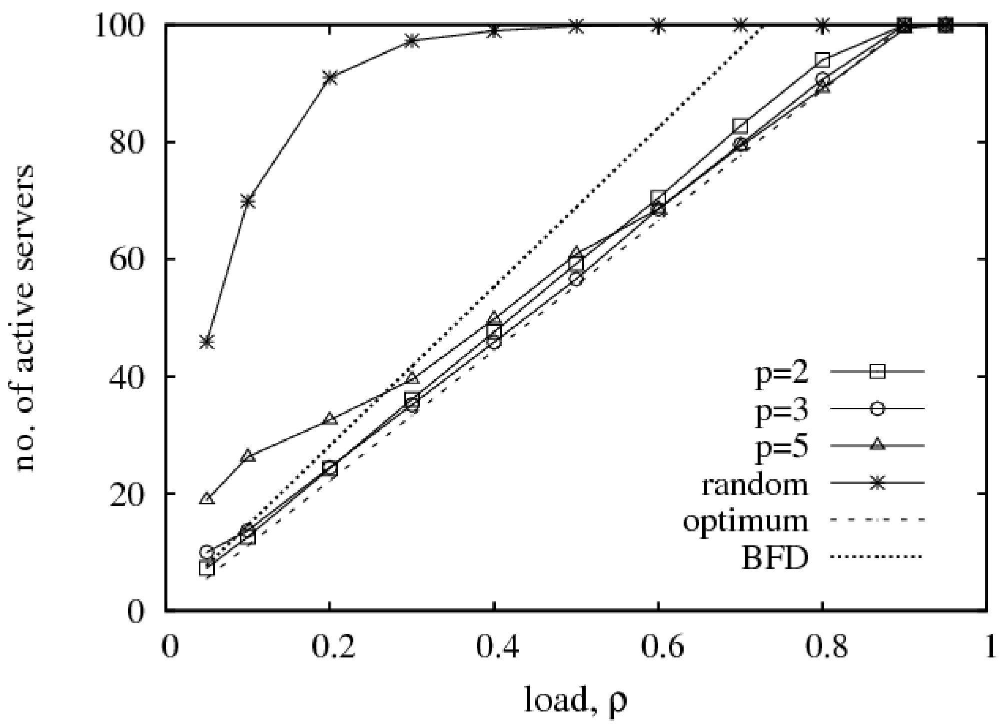
2. **Optimum strategy**: VMs are assigned so as to minimize the number of active servers

   *unfortunately, the assignment problem is **NP-hard**: it corresponds to the well known **bin packing problem***

2. **Best Fit Decreasing**: an algorithm with quadratic complexity that guarantees to use at most **11/9 MIN + 1** servers, where MIN is the theoretical minimum number

Number of active servers

❖ **Avg. number of active servers** vs. load, for 3 values of the parameter **p**

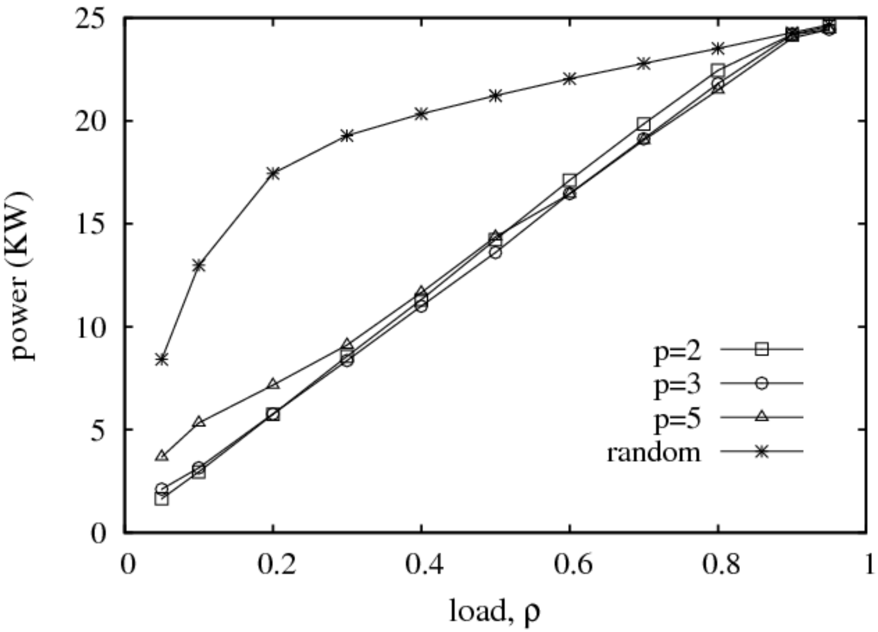


➢ Our statistical algorithm performs much better than the random strategy, better than BFD, and performance is **very close to the optimum**

➢ The parameter **p** (hence, the shape of the assignment function) has **little impact**. Higher values of p are slightly preferable with high load, and vice versa.

Consumed power

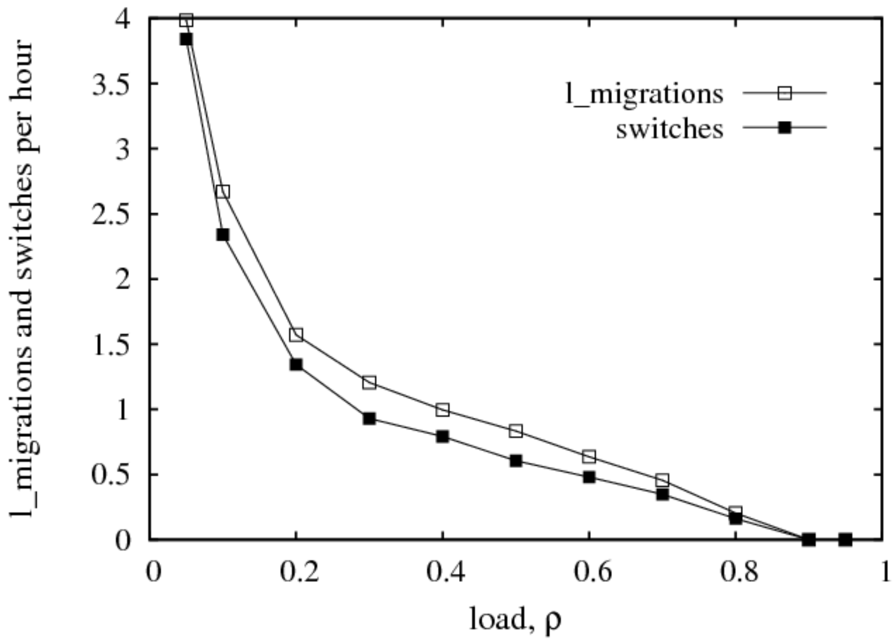
❖ **Consumed power** vs. load



- Consumed power increases linearly with load: it is hint of a **green behavior** (no extra power is wasted)
- Again, the parameter **p** has little impact, except that a low value is not preferable with low load.

Low migrations and switches

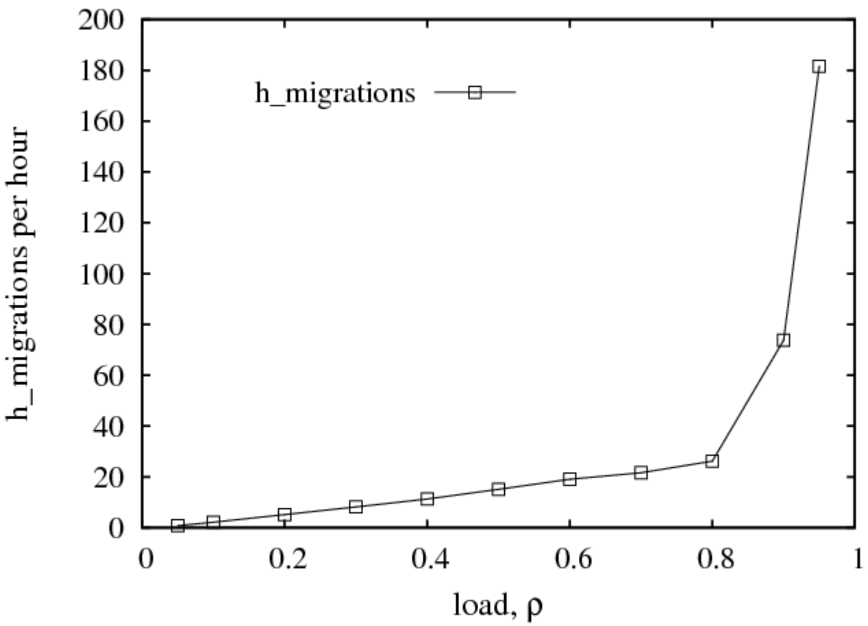
❖ **Frequency of low migrations and switches,** vs. load
(both events cause some performance degradation, so their frequency should be limited)



- **Frequency is higher with low load** (under-utilized servers are unloaded and often they can be switched off)
- Both frequencies are always lower than **4 events per hour,** which is an easily sustainable burden

High migrations

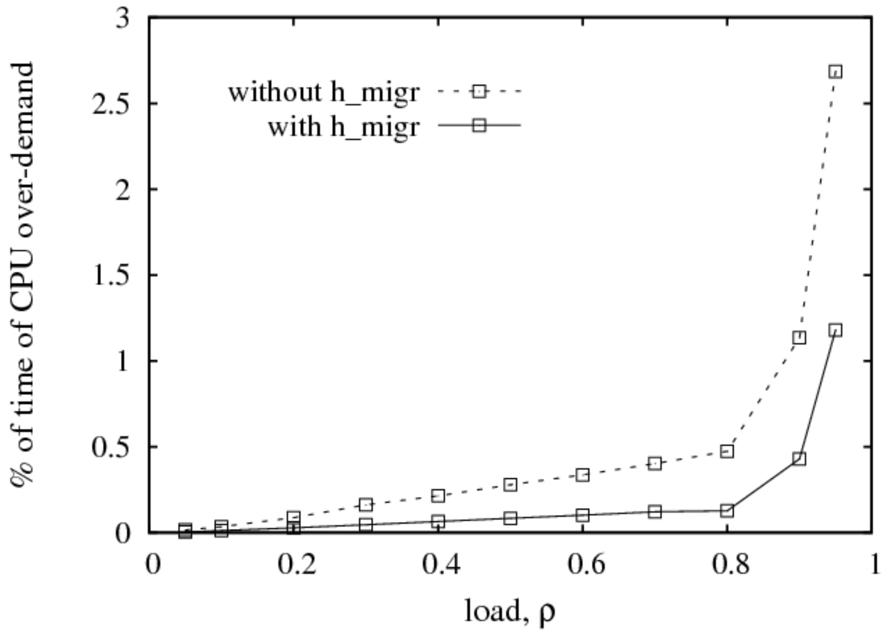
❖ **Frequency of high migrations** (migrations *from overloaded servers*)



- The frequency is proportional to the load
- The trend is nearly **liner**, but becomes **exponential** when load approaches the system capacity (**ρ > o.8**). In these conditions, new servers should be acquired

CPU overload

- **Perc. of time** in which VMs demand more CPU than the server capacity
- *This can lead to Service Level Agreements violations and **low QoS!***



- Violations are very rare when **ρ > 0.8**, then their frequency increases (the system is undersized)
- The index is reported **with and without the use of high migrations**

Conclusions

- An approach to minimize power consumption in data centers
- The approach is statistical, partially derived by **ant-inspired** algorithms and **swarm intelligence** techniques
- We think that our approach has some interesting benefits:

1. No use of complex centralized algorithms: important decisions are **local**
2. The assignment process is **self-organizing** and **adaptive**
3. The process of VM migrations is **smooth**: no concurrent migrations of VMs
4. The approach is **scalable**: further results show that it works even better in larger data centers, with more than 100 servers!

Thank you for your attention!