Étude et modélisation des mécanismes de surallocation de mémoire pour la consolidation de machines virtuelles

Simon Lambert [1,3,4], Eddy Caron[2,4], Laurent Lefèvre[3,4], Rémi Grivel[1]
ComPAS
05 juillet 2023

simon.lambert@ens-lyon.fr

[1] Ciril GROUP
[2] ENS de Lyon
[3] Inria
[4] LIP (UMR CNRS - ENS Lyon - UCB Lyon 1 - Inria 5668)

Global context

Observation : Environmental impact of Information and Communication Technologies is no longer discussed.

Datacenters :

- 220-320 TWh of electricity used in 2021 [1]
- ~1% of Greenhouse Gas global emissions in 2021[1]

+ Virtualization
+ Infrastructures (cooling, etc.)

- Increasing demands
- Bad resource management

→ How to improve computing resource management in a datacenter ?

[1] IEA (2022), Data Centres and Data Transmission Networks, IEA, Paris https://www.iea.org/reports/data-centres-and-data-transmission-networks, License: CC BY 4.0

Global context

→ How to improve computing resource management in a datacenter ?

Work on **consolidation**

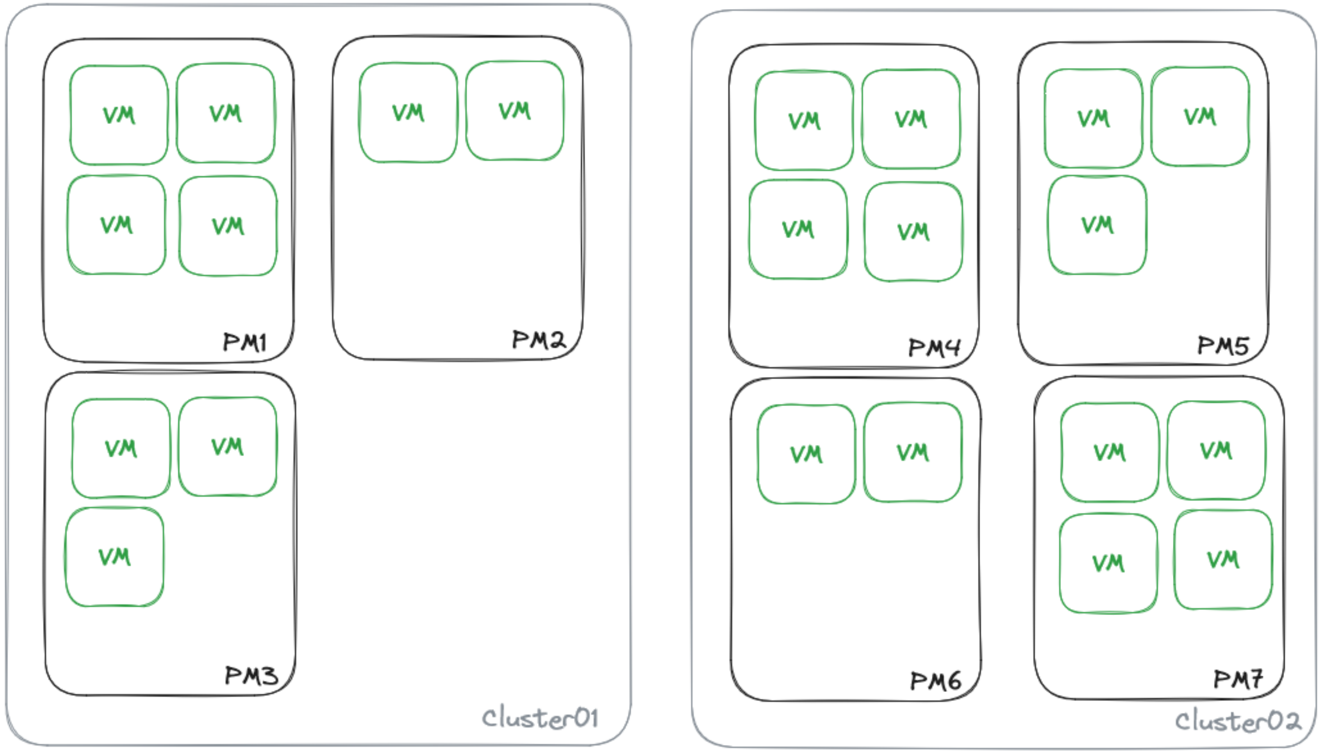
- Limit a resource usage
- Reduce costs
- Improve Quality of Service

In our context :

- Full control from software to hardware
- Identified levers of reduction
- Estimated **56% reduction** in electricity consumption

Lexicon :
- Physical Machine : PM
- Virtual Machine : VM
- Resources : CPU, RAM, disk usage rate, network usage

Background information

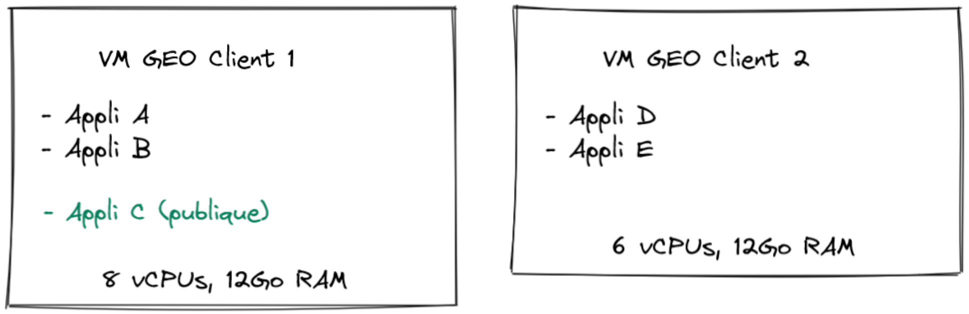


Virtualization infrastructure
segmentation in clusters

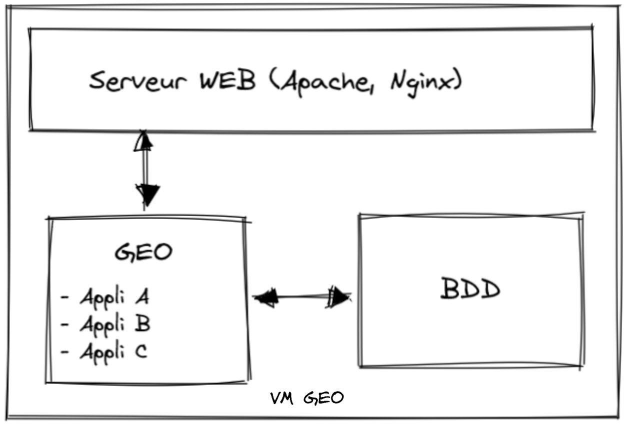
VMWare vSphere 7

- Allows segmentation (licence, security)
- Consolidation handled at this scale

Context – VMs usage



2 GEO VMs and their applications



Within a GEO VM

Inability to shut down VMs to retrieve resources

Context – VMs usage

Web Application

Usage evaluation

2 key aspects :

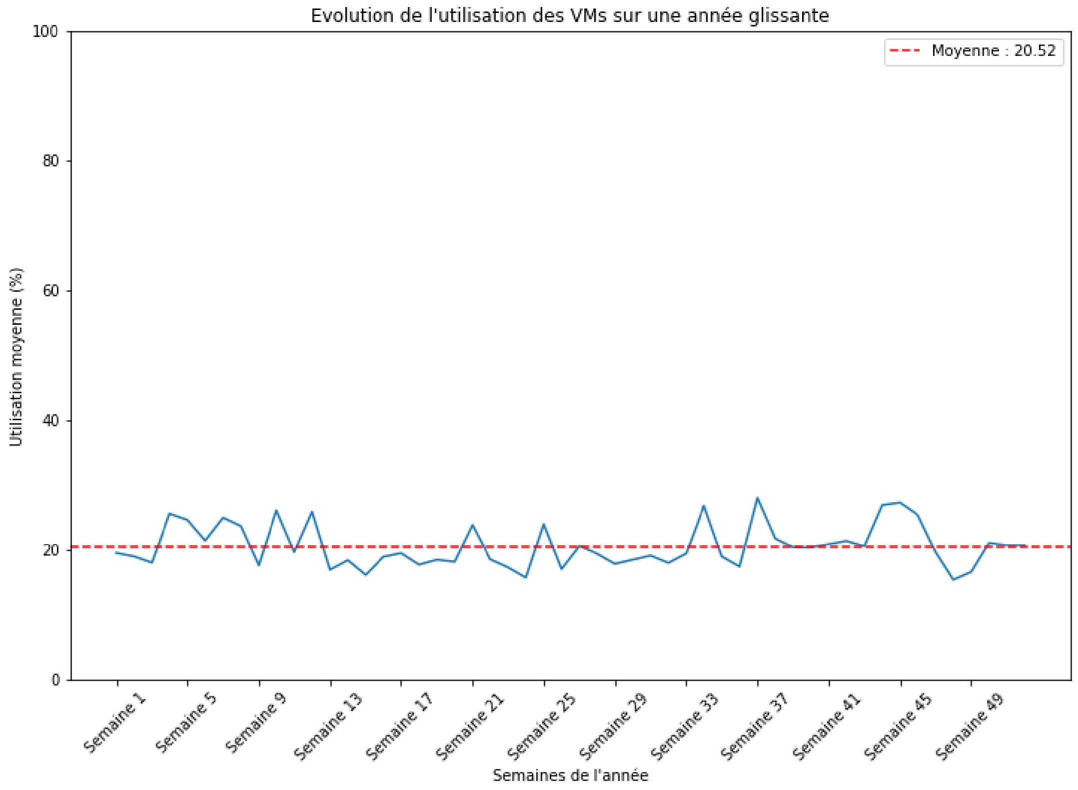
- Application consultation
- Background tasks

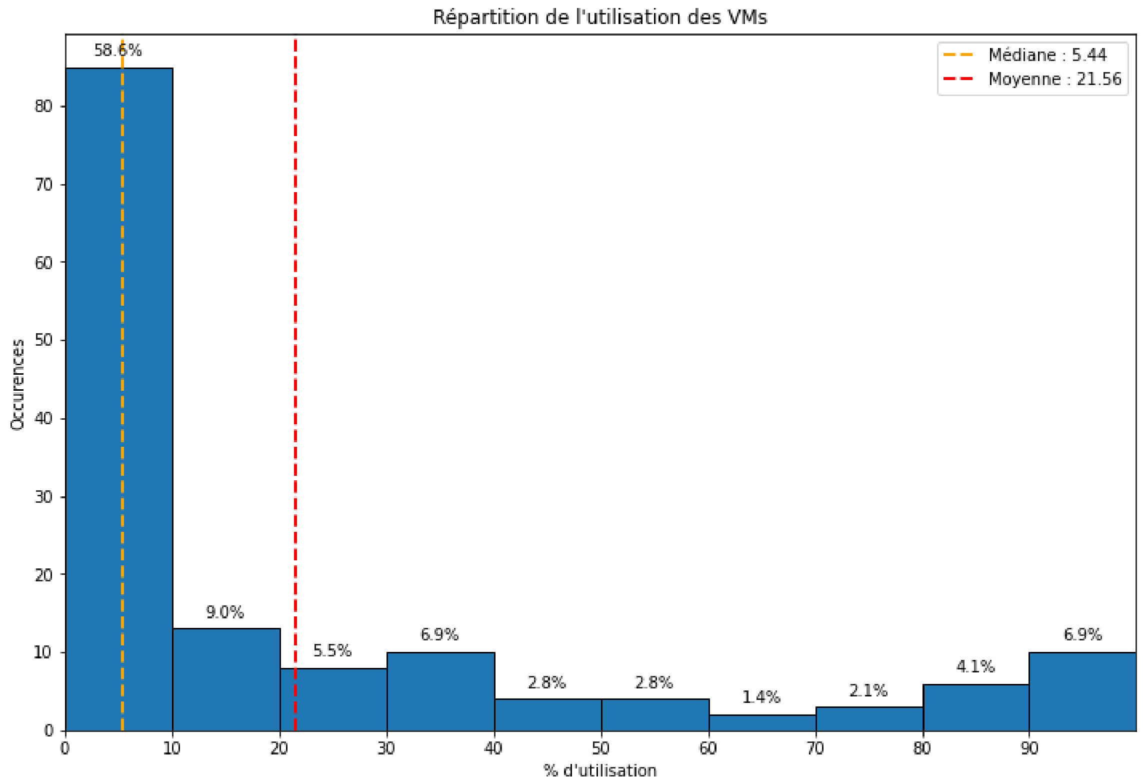
```
VM usage evaluation algorithm :

is_used(vm):
  if cpu_usage(vm) <= 1 and net_usage(vm) == 0:
    return 0
  else:
    return 1


cpu_usage : % CPU usage
net_usage : network traffic at interfaces
```

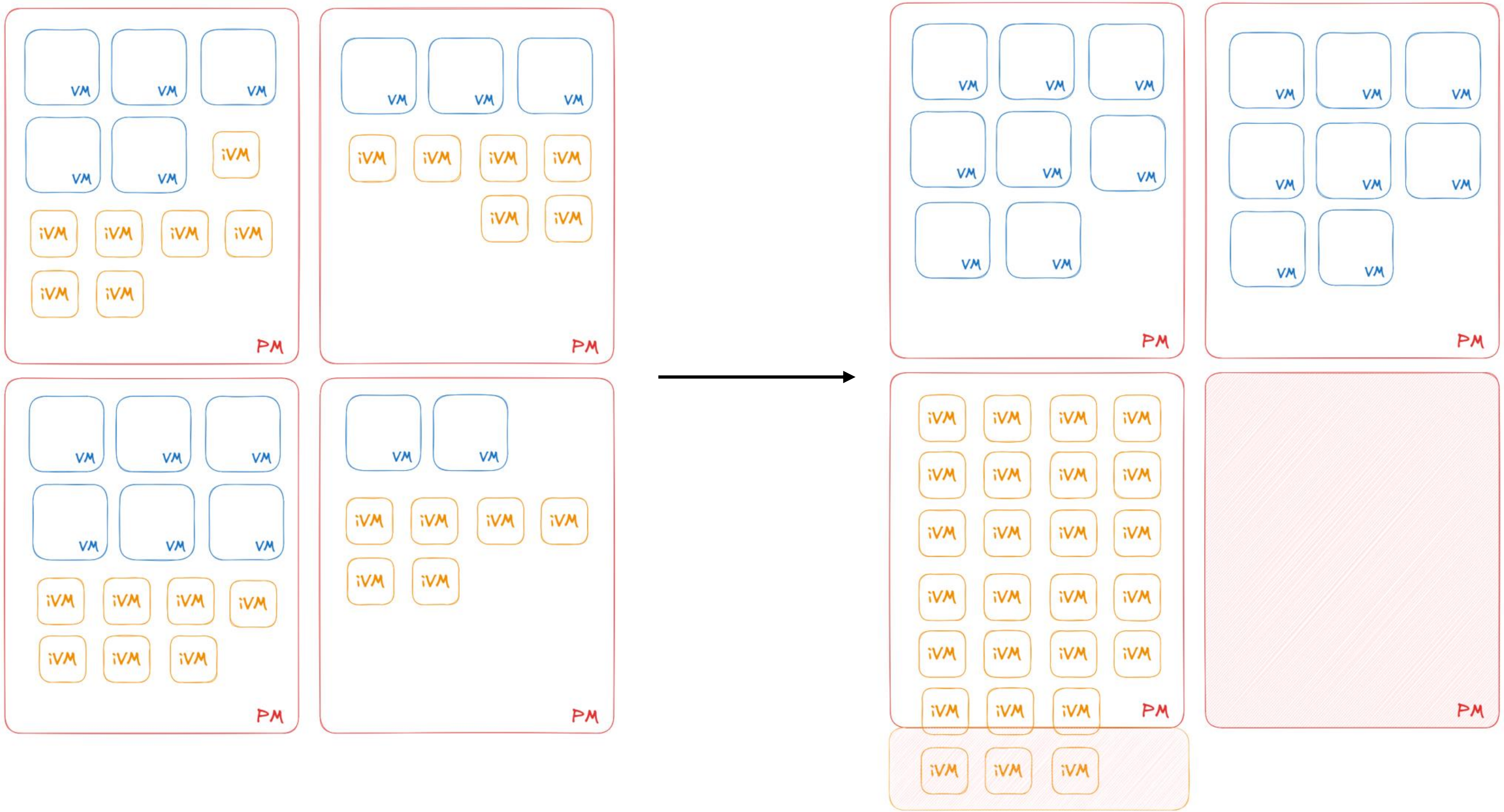
Mean usage of GEO VMs





Low, stable and unevenly distributed GEO VMs usage

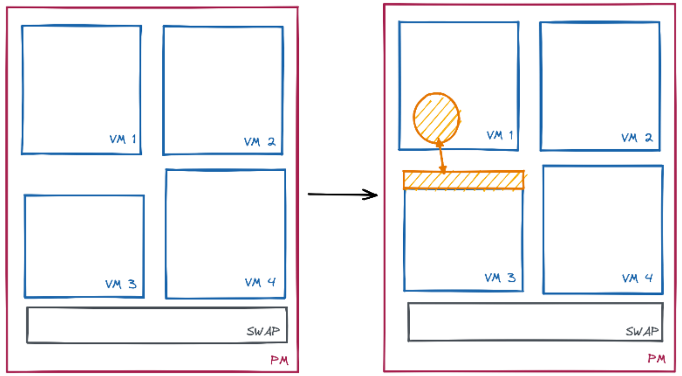
Consolidation approach



Memory overcommitment management mechanisms

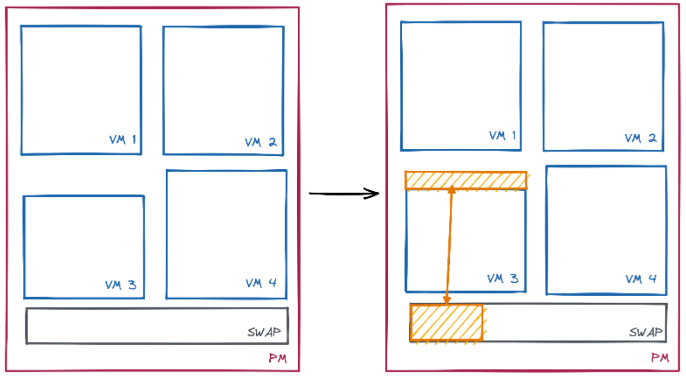
Lab experiment :

- 2 PMs : Intel(R) Xeon(R) CPU E5-2687W @3.00GHz 12 Cores ; 384 GB RAM
- 62 VMs (avg) : 7.3 vCPUs/VM ; 12.6GB RAM/VM
  - Application dependent : GEO (WebGIS Application)



Ballooning

vs.



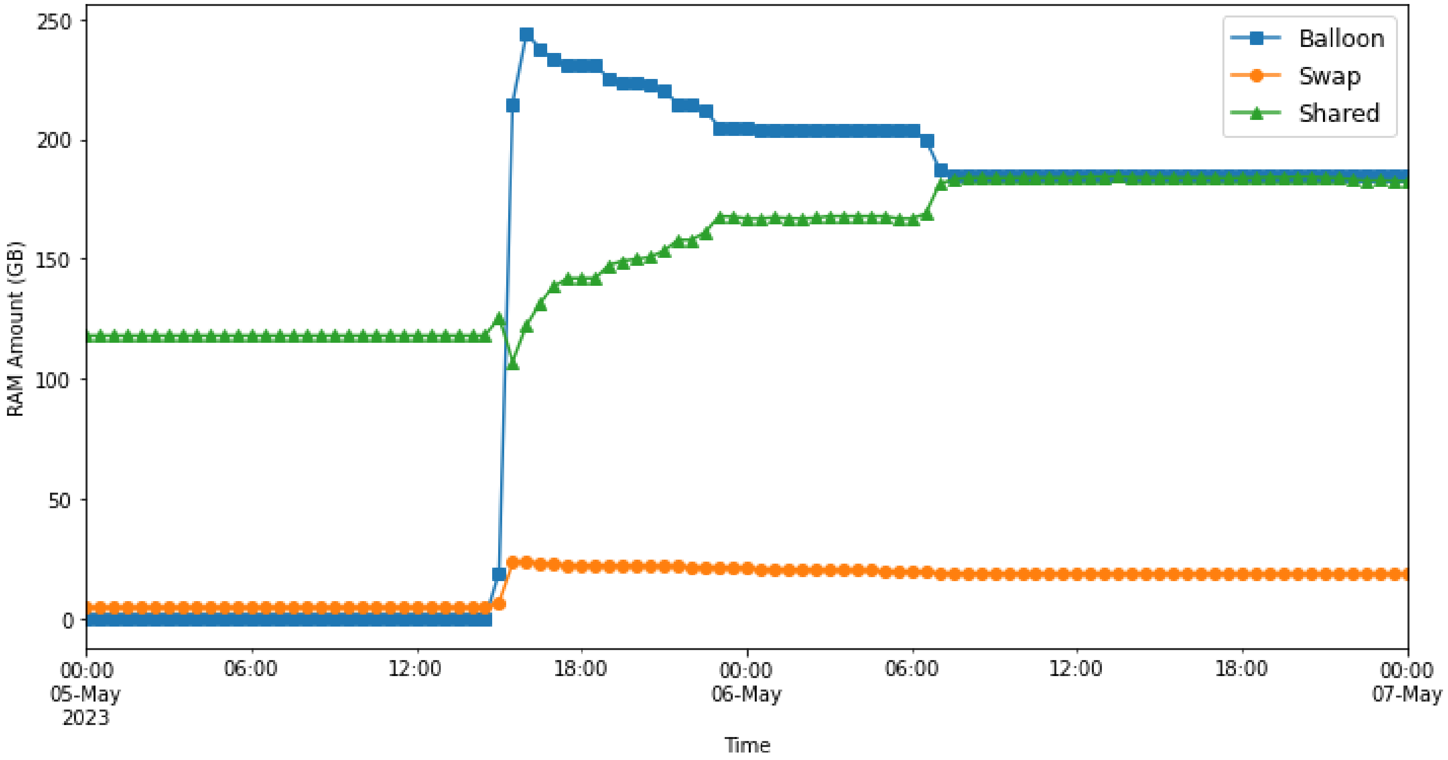
Swapping

If a PM is full, it can:

- **Balloon** : retrieve inactive memory on a VM to provide it to another through a driver
- **Swap** : Write memory pages on an external storage device (i.e. external storage array, disks)
- **Page Sharing** : Share identical memory pages within a single VM

Manual consolidation of VMs

	VM Memory Usage (GB)	RAM Usage (%)	Total VMs Memory configuration (GB)	vCPUs
Total	322,79	84,1	1168	382



- #vCPUs (< 32 per PM cores)
- Overcommitment rate : 3
- Ballooning and swapping as a reaction mechanism
- Transparent Page Sharing (TPS) on the long term

Cluster modeling

Variable	Description
P	Ensemble des serveurs physiques
V	Ensemble des machines virtuelles
M_{p_i}	Nombre de VM sur une PM p_i
p^c	Quantité de CPUs sur une PM
v^c	Quantité de vCPUs sur une VM
p_i^m	Mémoire disponible sur une PM p_i
v_j^m	Mémoire configurée sur une VM v_j
O	Surallocation mémoire
O_{p_i}	Surallocation mémoire sur une PM p_i

Cluster variables

$$O(t) = \frac{\sum_{j=0}^{|V|} v_j^m(t)}{\sum_{i=0}^{|P|} p_i^m(t)}, O_{p_i}(t) = \frac{\sum_{j=0}^{M_{p_i}} v_j^m(t)}{p_i^m(t)}, \text{avec } v_0^m = 0$$

Memory overcommitment

$$|P| = \frac{\sum_{j=0}^{|V|} v_j^m(t)}{O \times p^m}$$

Number of needed PMs (memory)

$$|min(P)| = \frac{\sum_{j=0}^{|V|} v_j^{c'}(t)}{p^c \times Q^*}$$

Number of needed PMs (CPU)

* vCPU per cores limit on a PM. Q = 32

Problem solving

Problem : Increase O(t)

Solution : Determine 2 sets P_u et $P_{\bar{u}}$ such as

Variable	Description
P	Ensemble des serveurs physiques
V	Ensemble des machines virtuelles
M_{p_i}	Nombre de VM sur une PM p_i
p^c	Quantité de CPUs sur une PM
v^c	Quantité de vCPUs sur une VM
p_i^m	Mémoire disponible sur une PM p_i
v_j^m	Mémoire configurée sur une VM v_j
O	Surallocation mémoire
O_{p_i}	Surallocation mémoire sur une PM p_i

	P_u	$P_{\bar{u}}$
VMs	Used VMs	Idle VMs
Memory commitment	Low $O_u(t)$ (≤ 1)	High $O_{\bar{u}}(t)$ (≈ 3)

$$|NP_u^m| = \left\lceil \frac{|V_u| \times v^m}{O_u(t) \times p^m} \right\rceil \quad |NP_u^c| = \left\lceil \frac{|V_u| \times v^{c\prime}}{p^m \times Q} \right\rceil$$

$$|P_u| = max(|NP_u^m|, |NP_u^c|)$$

Number of needed PMs for P_u

Increase O(t) keeping $O_u(t) \leq$ observed allocation rate on the infrastructure.

Consolidation Algorithm

Algorithme 2 Algorithme de calcul du nombre de PMs

```
function CalcNumPm(V_u, V_ū, P)
    Trier P en ordre décroissant de valeurs p_i^m et p_i^c
    NP_u^m = CalcPmRam(V_u^m', O_u, P)
    NP_u^c = CalcPmCpu(V_u^c', P)
    |P_u| = max(NP_u^m, NP_u^c)
    Enlever les premiers |P_u| serveurs de P
    NP_ū^m = CalcPmRam(V_u^m', O_u, P)
    NP_ū^c = CalcPmCpu(V_u^c', P)
    |P_ū| = max(NP_ū^m, NP_ū^c)
    return |P_u|, |P_ū|
end function

function CalcPmRam(ram, oc, PMS)
    numPms = 0
    remainingMem = ram / oc
    for each p in PMS do
        if remainingMem ≤ 0 then
            break
        end if
        numPms = numPms + 1
        remainingMem = remainingMem − p^m
    end for
    numPms = min(numPms, |PMS|)
    return numPms
end function

function CalcPmCpu(cpus, PMS)
    numPms = 0
    remainingCpus = cpus
    for each p in PMS do
        if remainingCpus ≤ 0 then
            break
        end if
        numPms = numPms + 1
        remainingCpus = remainingCpus − 32 × p^c
    end for
    numPms = min(numPms, |PMS|)
    return numPms
end function
```

Consolidation Algorithm

Algorithme 2 Algorithme de calcul du nombre de PMs

function CALCNUMPM(V_u, $V_{\bar{u}}$, P)
  Trier P en ordre décroissant de valeurs p_i^m et p_i^c
  NP_u^m = CALCPMRAM(V_u^m, O_u, P)
  NP_u^c = CALCPMCPU($V_u^{c'}$, P)
  $|P_u|$ = max(NP_u^m, NP_u^c)
  Enlever les premiers $|P_u|$ serveurs de P
  $NP_{\bar{u}}^m$ = CALCPMRAM(V_u^m, O_u, P)
  $NP_{\bar{u}}^c$ = CALCPMCPU($V_u^{c'}$, P)
  $|P_{\bar{u}}|$ = max($NP_{\bar{u}}^m$, $NP_{\bar{u}}^c$)
  return $|P_u|$, $|P_{\bar{u}}|$
end function

function CALCPMRAM(ram, oc, PMS)
  numPms = 0
  remainingMem = ram
  ...
    numPms = numPms + 1
    remainingMem = remainingMem $-p^m$
  end for
  numPms = m...
  return numP...
end function

function CALCPMCPU(cpus, PMS)
  numPms = 0
  remainingCpus = cpus
  ...
  ...en
    numPms = numPms + 1
    remainingCpus = remainingCpus
  ...
  ...umPms, |PMS|)
end function

$$|NP_u^m| = \left\lceil \frac{|V_u| \times v^m}{O_u(t) \times p^m} \right\rceil$$

$$|NP_u^c| = \left\lceil \frac{|V_u| \times v^{c\prime}}{p^m \times Q} \right\rceil$$

$$|P_u| = max(|NP_u^m|, |NP_u^c|)$$

Consolidation Algorithm

Algorithme 2 Algorithme de calcul du nombre de PMs

function CALCNUMPM(V_u, $V_{\bar{u}}$, P)
    Trier P en ordre décroissant de valeurs p_i^m et p_i^c
    NP_u^m = CALCPMRAM(V_u^m, O_u, P)
    NP_u^c = CALCPMCPU($V_u^{c'}$, P)
    $|P_u| = \max(NP_u^m, NP_u^c)$
    Enlever les premiers $|P_u|$ serveurs de P
    $NP_{\bar{u}}^m$ = CALCPMRAM(V_u^m, O_u, P)
    $NP_{\bar{u}}^c$ = CALCPMCPU($V_u^{c'}$, P)
    $|P_{\bar{u}}| = \max(NP_{\bar{u}}^m, NP_{\bar{u}}^c)$
    return $|P_u|$, $|P_{\bar{u}}|$
end function

function CALCPMRAM(ram, oc, PMS)
    numPms = 0
    remainingMem = $\frac{ram}{oc}$
    for each p in PMS **do**
        if remainingMem ≤ 0 **then**
            break
        end if
        numPms = numPms + 1
        remainingMem = remainingMem $-p^m$
    end for
    numPms = min(numPms, $|PMS|$)
    return numPms
end function

function CALCPMCPU(cpus, PMS)
    numPms = 0
    remainingCpus = cpus
    for each p in PMS **do**
        if remainingCpus ≤ 0 **then**
            break
        end if
        numPms = numPms + 1
        remainingCpus = remainingCpus $-32 \times p^c$
    end for
    numPms = min(numPms, $|PMS|$)
    return numPms
end function

Consolidation Algorithm

Algorithme 2 Algorithme de calcul du nombre de PMs

function CALCNUMPM(V_u, $V_{\bar{u}}$, P)
    Trier P en ordre décroissant de valeurs p_i^m et p_i^c
    NP_u^m = CALCPMRAM(V_u^m, O_u, P)
    NP_u^c = CALCPMCPU($V_u^{c\prime}$, P)
    $|P_u| = \max(NP_u^m, NP_u^c)$
    Enlever les premiers $|P_u|$ serveurs de P
    $NP_{\bar{u}}^m$ = CALCPMRAM(V_u^m, O_u, P)
    $NP_{\bar{u}}^c$ = CALCPMCPU($V_u^{c\prime}$, P)
    $|P_{\bar{u}}| = \max(NP_{\bar{u}}^m, NP_{\bar{u}}^c)$
    return $|P_u|$, $|P_{\bar{u}}|$
end function

function CALCPMRAM(ram, oc, PMS)
    numPms = 0
    remainingMem $= \frac{ram}{oc}$
    for each p in PMS **do**
        if remainingMem ≤ 0 **then**
            break
        end if
        numPms = numPms + 1
        remainingMem = remainingMem $-p^m$
    end for
    numPms = min(numPms, $|PMS|$)
    return numPms
end function

function CALCPMCPU(cpus, PMS)
    numPms = 0
    remainingCpus = cpus
    for each p in PMS **do**
        if $remainingCpus \leq 0$ **then**
            break
        end if
        numPms = numPms + 1
        remainingCpus = remainingCpus $-32 \times p^c$
    end for
    numPms = min(numPms, $|PMS|$)
    return numPms
end function

Manual instantiation of the problem

	Identical configuration
Number of VMs	150
VMs configurations	[(6,12)]
PMs configurations	[(12,384)]
Usage rate	20%
O_u / $O_{\bar{u}}$	0,735/3
#PM ($\|P_u\|$ / $\|P_{\bar{u}}\|$)	4 (2/2)
Theoretical #PM ($O = O_u = O_{\bar{u}}$)	7
Gains	$\frac{7-4}{7} = 42.86\%$

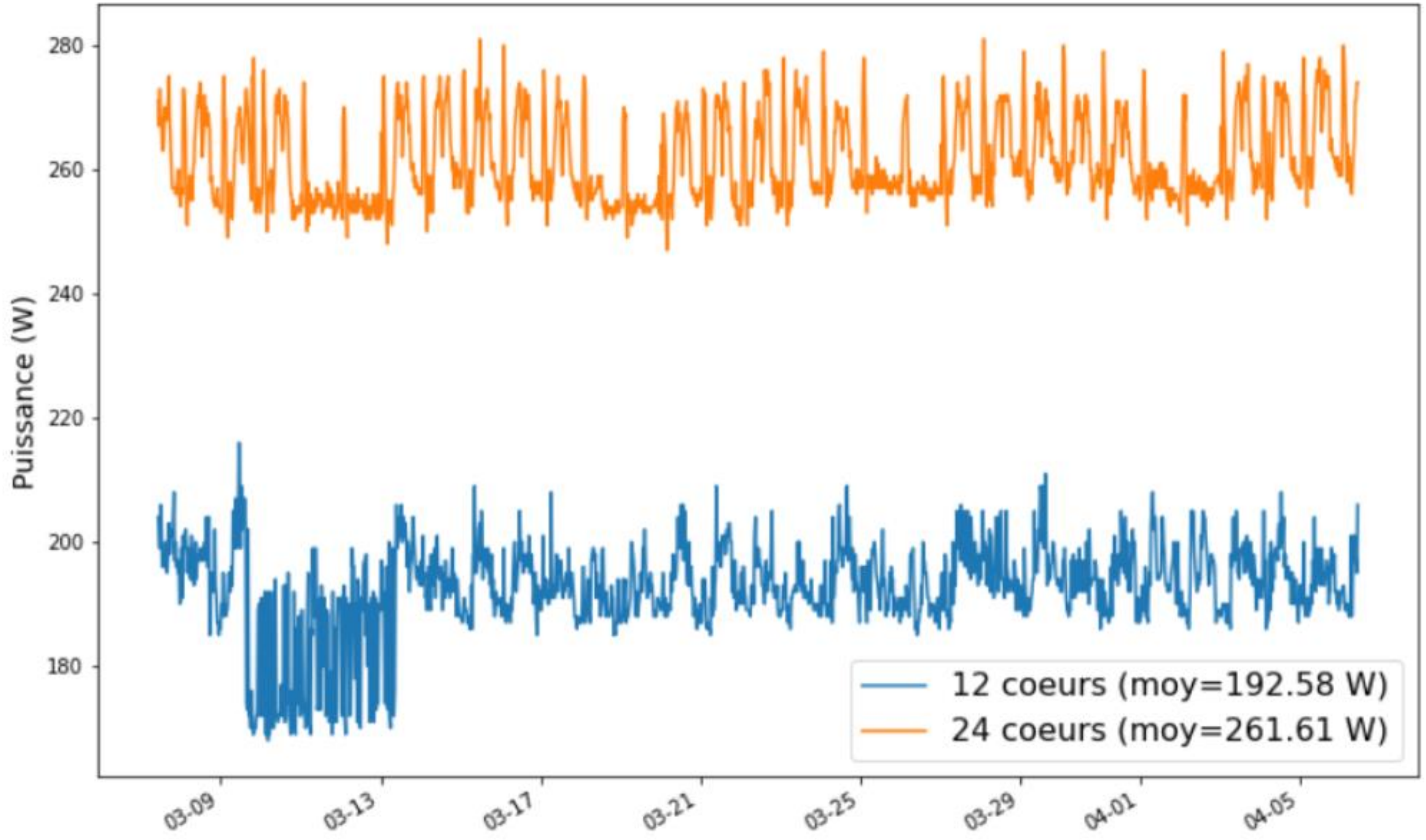
Manual instantiation of the problem

	Identical configuration	Different VMs
Number of VMs	150	150
VMs configurations	[(6,12)]	[(4,8), (8,12)]
PMs configurations	[(12,384)]	[(12,384)]
Usage rate	20%	20%
O_u / $O_{\bar{u}}$	0,735/3	0,735/3
#PM ($\lvert P_u \rvert$ / $\lvert P_{\bar{u}} \rvert$)	4 (2/2)	4 (2/2)
Theoretical #PM ($O = O_u = O_{\bar{u}}$)	7	6
Gains	$\frac{7-4}{7} = 42.86\%$	$\frac{6-4}{6} = 33.33\%$

Manual instantiation of the problem

	Identical configuration	Different VMs	Different VMs and PMs
Number of VMs	150	150	150
VMs configurations	[(6,12)]	[(4,8), (8,12)]	[(4,8), (8,12)]
PMs configurations	[(12,384)]	[(12,384)]	[(12,384), (24,768)]
Usage rate	20%	20%	20%
O_u / $O_{\bar{u}}$	0,735/3	0,735/3	0,735/3
#PM ($\lvert P_u \rvert$ / $\lvert P_{\bar{u}} \rvert$)	4 (2/2)	4 (2/2)	2 (1/1)
Theoretical #PM ($O = O_u = O_{\bar{u}}$)	7	6	4
Gains	$\frac{7-4}{7} = 42.86\%$	$\frac{6-4}{6} = 33.33\%$	$\frac{4-2}{4} = 50\%$

Electricity consumption



- Consumption not proportional to the configuration.
- Allows a reduction estimation in terms of power consumption.

Results

	Values
Number of VMs	Entre 1000 et 3000
VMs configurations	[(4,8), (6,12), (8,16)]
PMs configurations	[(12,384), (24,768)]
O_u / $O_{\bar{u}}$	0,735/3

- Worst case : 100% usage rate
- Usage case : 20% usage rate
- Best case : 0% usage rate

Mean reduction on 2000 simulations :

- Worst case : 1.60% (Lower bound)
- Usage case : 55.93%
- Best case : 67.75% (Upper bound)

$$N_t = 2 \times \left\lceil \frac{\sum_{j=0}^{|V|} v_j^m(t)}{O_u \times \frac{(X+Y)^*}{2}} \right\rceil$$

* X = 384 ; Y = 768

Conclusion

Issue : Complex resource management in datacenters

Approach : Consolidation method using memory overcommitment mechanisms

Works & results :

- VM usage evaluation algorithm (GEO)
- Study of memory overcommitment mechanisms
    - Ballooning, Swapping, TPS
- Modeling and instantiation of the consolidation in a cluster

→ With our approach and observed values, **56% reduction** of electricity consumption.

Thanks

Questions ?

@mail :
simon.lambert@ens-lyon.fr