Compas'2023 : Parallélisme/ Architecture/ Système
LISTIC/USMB - Annecy France, 4-7 juillet 2023

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]

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)

Résumé
La bonne gestion des ressources représente un défi pour les fournisseurs de service Cloud. Ces derniers doivent répondre à une demande croissante en limitant le surdimensionnement de leurs infrastructures, qui peuvent engendrer des coûts et un impact environnemental évitables. Des mécanismes dérivés de la virtualisation sont utilisés pour optimiser le dimensionnement des infrastructures comme l'élasticité et la consolidation, mais des contraintes économiques ou physiques peuvent être un frein à leur adoption. Dans ce papier, nous proposons une méthode de consolidation basée sur l'observation de la consommation de ressources des machines virtuelles (VMs) de l'infrastructure d'un hébergeur Cloud privé. Avec un algorithme d'évaluation propice et en utilisant des mécanismes de surallocation mémoire, nous sommes en mesure de générer des ensembles de VMs positionnés de telle sorte à limiter le nombre de machines physiques utilisées, tout en assurant la qualité de service requise par les utilisateurs.

Mots-clés : Virtualisation, Surallocation, Consolidation, Efficacité énergétique, Green IT

---

1. Introduction

Depuis 2010, nous assistons à une croissance importante des services Cloud. Cela a conduit à une augmentation du nombre de datacenters (DCs) de différents types dans le monde, des datacenters de petites entreprises jusqu'aux datacenters Cloud hyperscales. Face à cet essor et soucieux de l'empreinte carbone de ces infrastructures, différentes optimisations ont permis de limiter l'augmentation de la demande totale d'énergie des datacenters, en hausse de 10 à 60% entre 2015 et 2021. En 2021, les datacenters ont consommé 220 à 320 TWh d'électricité, ce qui représente 1% de l'utilisation mondiale [1, 18]. Cette limite à la consommation d'électricité des datacenters peut être attribuée à des optimisations matérielles notamment sur les serveurs [15] ou logicielles grâce à la démocratisation de la virtualisation [27]. La virtualisation a aidé à contenir cette consommation d'électricité en permettant d'exécuter plusieurs systèmes d'exploitation (OS), applications ou charges de travail sur des ressources physiques mutualisées. La virtualisation facilite également l'accès aux ressources que ce soit en qualité ou en quantité. Pour répondre aux demandes des utilisateurs, les fournisseurs de services Cloud tendent à surdimensionner leurs infrastructures. Il en résulte un gaspillage des ressources [12] qui se traduit par une surconsommation d'électricité. Pour limiter ce gaspillage, des travaux ont été menés afin d'implémenter de nouvelles techniques comme l'élasticité [10, 13, 23, 24], permettant d'améliorer le dimensionnement des infrastructures ou la consolidation [7, 16, 21, 22] pour mieux répartir la charge de travail. L'utilisation de ces mécanismes peut cependant être limitée en raison de contraintes économiques, de la complexité de leurs maintenances ou des dégradations de performances qu'ils peuvent induire.
Les métriques présentées ici proviennent de l'infrastructure d'un fournisseur logiciel et hébergeur de données Cloud privé : Ciril GROUP. Une attention particulière est portée à une solution développée par Ciril GROUP nommée GEO. GEO est un outil de construction de WebSIG permettant la création de

d'application Web carto-centrée, dédiée à la gestion de patrimoine geolocalisé. Les VMs sont hébergées dans la division hébergement de Ciril GROUP nommée SynAApS.
Dans ce papier, nous proposons une approche de consolidation se basant sur l'utilisation de ressources de machines virtuelles (VMs) afin de déterminer l'utilisation ou non des applications qu'elles hébergent. Les VMs *idles* sont placées de telle sorte à limiter le nombre de serveurs physiques (PMs) utilisé, et les VMs utilisées sont consolidées pour assurer un service à la hauteur des Service Level Agreements (SLAs) fixés. L'utilisation du CPU, de la mémoire et les échanges réseau sont utilisés pour évaluer la charge de travail d'une VM et de ses applications. La suite du papier est organisée comme suit. La section 2 définit la méthode et le système conçu. La section 3 présente les résultats théoriques. La section 4 traite de l'état de l'art et la section 5 conclut le document et présente les travaux futurs.

2. Travaux et implémentation

Pour mieux gérer le problème de surdimensionnement et de gaspillage des ressources, nous utilisons une approche de consolidation. Notre objectif ici est de mieux placer les VMs dans l'infrastructure pour éteindre des serveurs. À long terme, cela crée un double bénéfice avec i) la réduction de la consommation électrique du datacenter et ii) une meilleure gestion du parc de serveurs. Avec un algorithme de consolidation efficace, nous pouvons réduire l'empreinte carbone d'un datacenter sur les scopes 2 (utilisation d'électricité) et 3 (émissions liées aux achats) [8]. Pour s'adapter au mieux à l'utilisation réelle de l'infrastructure, nous souhaitons mettre en place une consolidation prédictive [6]. Pour cela, nous disposons des traces d'utilisation des machines (CPU, RAM, réseau, disque) sur un an, échantillonnées toutes les 20 secondes. À nous de déterminer la quantité de données nécessaires pour prédire l'utilisation des différentes ressources en i) limitant le coût de prédiction ii) trouvant une fenêtre de prédiction cohérente, précise et limitant le nombre de migrations de VMs. L'algorithme devra également s'adapter au comportement aléatoire et imprévisible des utilisateurs. Il devra répondre efficacement à ces événements imprévus afin d'assurer une haute disponibilité et une bonne qualité de service.

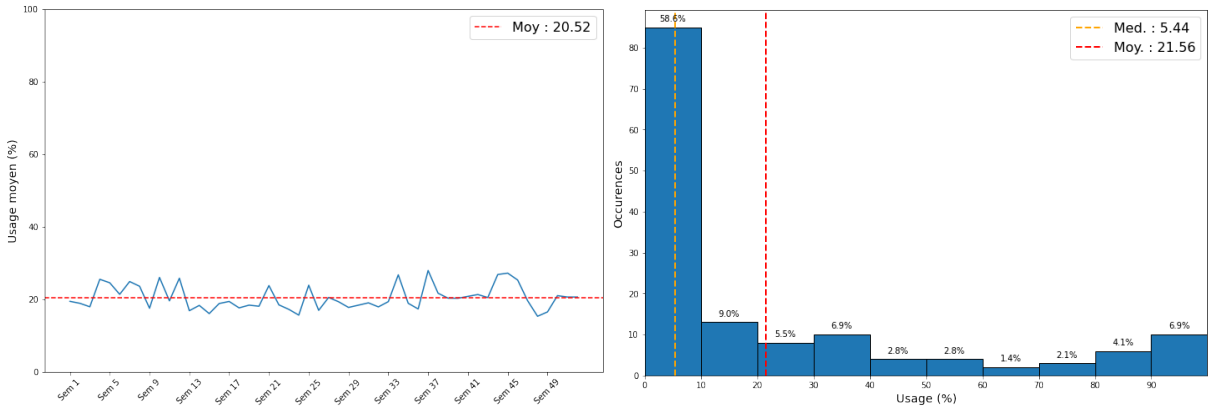
2.1. Étude de l'utilisation des VMs

Pour évaluer l'utilisation ou non d'une machine, nous vérifions 2 paramètres. Premièrement, l'application web ne doit pas être consultée pendant la période d'échantillonnage, et deuxièmement, il ne doit pas y avoir de tâche de fond en cours d'exécution (tâche planifiée, dump de base de données). Nous pouvons superviser ces 2 critères en analysant d'une part les communications aux interfaces réseau de la machine et d'autre part le taux d'utilisation CPU de la VM. Nous proposons l'Algorithme 1 pour évaluer l'utilisation d'une VM.

Algorithme 1 Évaluation de l'utilisation d'une VM

```
1: function IS_USED(vm)
2:     if cpu_usage(vm) ≤ 1% and net_usage(vm) = 0b/s then
3:         return 0
4:     else
5:         return 1
6:     end if
7: end function
```

L'état d'une VM *idle* est 0, et celui d'une machine utilisée est 1. Nous pouvons donc créer une matrice dans laquelle nous avons l'utilisation de chacune de nos VMs durant une période donnée, échantillonnée toutes les 20 secondes. Cela nous permet de calculer la moyenne d'utilisation d'une ou plusieurs VMs sur une période T, dont la distribution est représentée dans la Figure 1b ainsi que la moyenne d'utilisation de toutes les VMs pour un échantillon donné t. La Figure 1a montre l'évolution de la quantité moyenne de VMs utilisées pendant un an.
La Figure 1a nous permet de visualiser un pourcentage de VMs utilisées faible et relativement stable durant l'année. La Figure 1b traduit elle la disparité d'utilisation des VMs, avec 50% de machines utilisées moins de 5.44% du temps.
L'algorithme permet donc de traduire l'utilisation faible, stable et inégalement distribuée des nos VMs. Cette utilisation prend la forme d'une série binaire pour chacune de nos machines. La nature binaire de l'information peut se révéler limitante dans certains cas, mais ce n'est pas un problème ici, car nous n'avons pas la possibilité d'ajuster la quantité de ressources allouée aux VMs en fonction de leur taux d'utilisation.



(a) Évolution de la quantité de VMs utilisées (%)

(b) Répartition de la quantité de VMs utilisées

FIGURE 1 – Répartition et évolution des VMs étudiées selon l'algorithme proposé

2.2. Mécanisme de gestion de surallocation de mémoire

La conception de GEO nous a permis d'identifier 2 contraintes. Étant non distribuée, l'extinction d'une VM engendre l'indisponibilité de l'application qu'elle héberge. Certaines application étant publiques, il est impossible d'éteindre les VMs pour récupérer des ressources.

Pour surmonter la première contrainte, nous avons étudié l'ajout et retrait à chaud des ressources (CPU, RAM) sur les VMs. L'hyperviseur utilisé (VMWare vSphere) ne permet pas de retirer des ressources sur les VMs sans extinction et donc sans interruption de service. Ainsi, nous ne pouvons pas libérer des ressources en éteignant les VMs ou en utilisant l'élasticité et la reconfiguration à chaud.

Nous utilisons donc une approche de consolidation pour réduire l'utilisation de PMs en se concentrant sur les mécanismes de gestion de surallocation de mémoire intégrés dans l'hyperviseur, notamment le ballooning et le swapping.

Quand la mémoire disponible sur un serveur physique devient trop faible pour répondre à l'utilisation des machines qu'il héberge, il peut en récupérer via différents mécanismes dont le ballooning et le swapping. Le ballooning permet à une PM de récupérer de la RAM en demandant aux VMs de libérer de la mémoire allouée mais inutilisée ou inactive via un composant logiciel appelé *memory controller*. Le swapping consiste lui à écrire les pages mémoires nouvellement allouées aux VMs sur un espace de stockage externe (disques durs, baies de stockage), généralement moins rapide que la mémoire vive. Ces 2 mécanismes permettent de répondre aux saturations de mémoire sur les serveurs physiques, mais provoquent des dégradations de performance sur les VMs.

Pour mesurer la quantité de mémoire récupérable grâce à ces mécanismes, nous avons positionné manuellement le plus de VMs possibles sur une des PMs. Les métriques de consommation ainsi que les configurations des VMs sont présentées dans la Table 1.

	RAM Physique PM (Go)	Utilisation RAM PM (Go)	Utilisation RAM PM (%)	Somme des mémoires configurées sur les VMS (Go)	Somme des vCPUs configurés sur les VMs
Total	384	322.79	84.1	1168	**382**

TABLE 1 – Observation des métriques CPUs et mémoire après le placement manuel des VMs

En plaçant environ 50 VMs, nous pouvons identifier une limite imputable à l'utilisation de vSphere [1]. Le nombre de vCPU par coeur physique sur le serveur étant limité à 32, nous avons une limite de 384 vCPUs ici. Cela nous permet de déduire un taux de surallocation mémoire O où $O = \frac{1168}{384} = 3.04$. Bien que le ratio vCPU/pCPU soit très élevé ici, cela ne représente pas un danger car l'ensemble de nos machines utilisent moins de 1% de la capacité CPU, conformément à notre algorithme.

1. https://tinyurl.com/vSphereSL

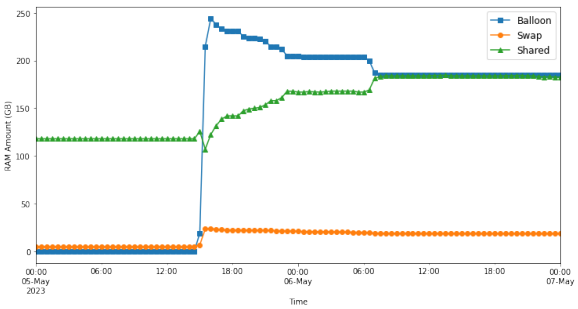


FIGURE 2 – Mémoire balloon et Swap mesurées sur la PM

La Figure 2 montre la quantité de RAM récupérée grâce au ballooning et au swapping suite à la surallocation de mémoire engendrée par notre placement de VMs. Avec le temps, ces valeurs diminuent grâce à l'utilisation d'un autre mécanisme : le Transparent Page Sharing (TPS). Le TPS permet à une PM de scanner la mémoire des VMs et d'identifier les pages mémoires identiques. Pour chaque page mémoire identique sur une VM donnée, le TPS crée une copie unique de la page et la partage avec les doublons identifiés. Pour répondre à des besoins de sécurité, le TPS est désactivé par défaut entre différentes machines virtuelles. Nous avons donc pu identifier ici une limitation physique de nos PMs pour le placement de VMs, et avons mesuré une valeur théorique de surallocation mémoire compte tenu de la configuration moyenne des VMs étudiées, ici égale à 3.04.

2.3. Modélisation du système

Afin d'étudier différents scénarios d'utilisation et d'être en mesure d'estimer les gains apportés par notre solution, nous avons modélisé notre système en identifiant les variables et différentes équations qui le régissent.

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

(a) Variables globales

	P_u	$P_{\bar{u}}$
Surallocation des PMs	O_u	$O_{\bar{u}}$
VMs hébergées	V_u	$V_{\bar{u}}$
Somme de mémoire sur les VMs	V_u^m	$V_{\bar{u}}^m$
Nombre de vCPUs sur les VMs	$V_u^{c'}$	$V_{\bar{u}}^{c'}$

(b) Définition des sous ensembles P_u et $P_{\bar{u}}$

TABLE 2 – Variables du systèmes

De nombreuses équations modélisent la consommation mémoire des VMs et des PMs, pouvant avoir un impact sur les performances de l'application. Nous nous concentrons ici uniquement sur la surallocation (à l'échelle de l'infrastructure et d'une PM) :

$$O(t) = \frac{\sum_{j=0}^{|V|} v_j^m(t)}{\sum_{i=0}^{|P|} p_i^m(t)}, O_{p_i}(t) = \frac{\sum_{j=0}^{M_{p_i}} v_j^m(t)}{p_i^m(t)}, avec\ v_0^m = 0 \quad (1)$$

La surallocation se traduit ici par le rapport entre la quantité de mémoire configurée sur les machines d'un hôte/cluster et la quantité de mémoire physique sur le même hôte/cluster. Elle est par définition supérieure à 1. Pour des raisons de simplicité, nous utilisons le terme surallocation qu'elle soit inférieure ou supérieure à 1. Un cluster est un ensemble logique de serveurs hébergeant un groupe de VMs. Notre système défini dans la Table 2a permet de modéliser un cluster.

Par définition, nous voulons augmenter la valeur de O. Cela peut être fait en diminuant le nombre de PMs ($|P|$) tout en gardant un nombre de VMs ($|V|$) constant, ou inversement en augmentant $|V|$ avec $|P|$ constant. Ces solutions ne permettent cependant pas de répondre à notre problématique si l'on considère les effets de bord. L'augmentation de O via ces mécanismes provoquera des dégradations de performance sur l'ensemble du cluster. Nous utilisons donc l'algorithme 2 pour séparer P en 2 sous ensemble P_u et $P_{\bar{u}}$, décrits dans la Table 2b.

Si nous considérons une surallocation initiale $O_s(t)$, nous souhaitons créer 2 sous ensembles P_u et $P_{\bar{u}}$ avec des valeurs $|P_u|$ et $|P_{\bar{u}}|$ tels que $O_u(t) \leq O_s(t)$ et $O_{\bar{u}}(t) > O_s(t)$.

2.4. Algorithme de calcul du nombre de PM

Pour réaliser nos simulations et mesurer le nombre de PMs nécessaires pour héberger nos VMs, nous proposons l'Algorithme 2. La quantité de serveurs requise pour chacun des groupes V_u et $V_{\bar{u}}$ est calculée en prenant compte de la mémoire et de l'overcommitment avec la fonction *CalcPmRam*. Nous veillons aussi à ne pas dépasser la limite vCPU imposée par l'hyperviseur avec la fonction *CalcPmCpu*.

Algorithme 2 Algorithme de calcul du nombre de PMs

```
1: function CalcNumPm(V_u, V_ū, P)
2:     Trier P en ordre décroissant de valeurs p_i^m et p_i^c
3:     NP_u^m = CalcPmRam(V_u^m, O_u, P)
4:     NP_u^c = CalcPmCpu(V_u^c', P)
5:     |P_u| = max(NP_u^m, NP_u^c)
6:     Enlever les premiers |P_u| serveurs de P
7:     NP_ū^m = CalcPmRam(V_u^m, O_u, P)
8:     NP_ū^c = CalcPmCpu(V_u^c', P)
9:     |P_ū| = max(NP_ū^m,NP_ū^c)
10:    return |P_u|, |P_ū|
11: end function

12: function CalcPmRam(ram, oc, PMS)
13:     numPms = 0
14:     remainingMem = ram/oc
15:     for each p in PMS do
16:         if remainingMem ≤ 0 then
17:             break
18:         end if
19:         numPms = numPms + 1
20:         remainingMem = remainingMem − p^m
21:     end for
22:     numPms = min(numPms, |PMS|)
23:     return numPms
24: end function

25: function CalcPmCpu(cpus, PMS)
26:     numPms = 0
27:     remainingCpus = cpus
28:     for each p in PMS do
29:         if remainingCpus ≤ 0 then
30:             break
31:         end if
32:         numPms = numPms +1
33:         remainingCpus = remainingCpus − 32 × p^c
34:     end for
35:     numPms = min(numPms, |PMS|)
36:     return numPms
37: end function
```

La quantité de PM nécessaire pour un groupe de VMs correspond au maximum entre les valeurs retournées par les fonctions *CalcPmRam* et *CalcPmCpu*. L'algorithme retourne en sortie les valeurs $|P_u|$ et $|P_{\bar{u}}|$.

3. Résultats

3.1. Instanciations manuelles

Pour évaluer les gains théoriques, nous avons instancié notre système et fixant certaines valeurs conformément aux observations précédemment faites. Les résolutions nous ont permis d'obtenir les résultats présentés dans la Table 3a.

En considérant un système avec 150 VMs, nous cherchons le nombre de PMs nécessaires pour héberger nos VMs. Les VMs de notre système ont 6 vCPUs (v^c) et 12 Go de RAM (v^m). Les PMs ont elles 12 coeurs (p^c) et 384 Go de RAM physique (p^m). Prenons une valeur de surallocation $O_s(t) = 0.735$, et $O_{\bar{u}}(t) = 3$. Selon notre algorithme, nous fixons l'utilisation moyenne des VMs à 20%, ce qui nous permet de générer nos 2 sous ensembles de VMs V_u et $V_{\bar{u}}$, contenant respectivement $|V_u| = 0.2 \times 150 = 30$ et $|V_{\bar{u}}| = 0.8 \times 150 = 120$. Grâce à nos observations, nous pouvons calculer :

$$|P_u| = \left\lceil \frac{|V_u| \times v^m}{O_u(t) \times p^m} \right\rceil = \left\lceil \frac{30 \times 12}{0.735 \times 384} \right\rceil = \lceil 1.27 \rceil = 2\text{PMs} \; ; |P_{\bar{u}}| = \left\lceil \frac{|V_{\bar{u}}| \times v^m}{O_{\bar{u}}(t) \times p^m} \right\rceil = \left\lceil \frac{120 \times 12}{3 \times 384} \right\rceil = \lceil 1.25 \rceil = 2\text{PMs}$$

Pour que le système soit fonctionnel, nous devons aussi calculer le nombre de serveur théorique requis pour ne pas dépasser la limite de vCPUs par PM.

$$min(|P_{\bar{u}}|) = \left\lceil \frac{|V_{\bar{u}}| \times v^c}{p^c \times 32} \right\rceil = \left\lceil \frac{120 \times 6}{12 \times 32} \right\rceil = \lceil 1.875 \rceil = 2\text{PMs}$$

Avec notre méthode et les données instanciées, nous avons besoin de 4 PMs pour positionner les VMs. Dans le contexte actuel en conservant la valeur $O_s(t)$, nous pouvons calculer le nombre de serveur théorique nécessaire N_t et les gains G :

$$N_t = \left\lceil \frac{|V| \times v^m}{O_s(t) \times p^m} \right\rceil = \left\lceil \frac{150 \times 12}{0.735 \times 384} \right\rceil = \lceil 6.37 \rceil = 7\text{PMs} \;\; ; \;\; G = \frac{(N_t - (|P_u| + |P_{\bar{u}}|))}{N_t}$$

Nous instancions ensuite notre système avec différentes variations sur les configurations des VMs et des PMs. Dans chaque cas, les configurations sont distribuées équitablement. Lorsque nous disposons de plusieurs configurations de PMs, nous priorisons toujours les serveurs avec la plus grande capacité.

À noter, le nombre théorique de serveur N_t dans ce cas de figure sera toujours multiple de 2, compte tenu que nous avons 2 types de serveurs (serveurs *small - S* et *large - L*). Nous considérons que nous avons la même quantité de serveurs S et L. Si les serveurs S ont X Go de RAM, les serveurs L ont Y Go de RAM, nous avons $N_t = 2 \times \left\lceil \frac{\sum_{j=0}^{|V|} v_j^m(t)}{O_u \times (X+Y)} \right\rceil$

	Instance 1	Instance 2	Instance 3
Nombre de VMs	150	150	150
Configurations VMs (vCPUs, RAM)	[(6,12)]	[(4,8), (8,12)]	[(4,8), (8,12)]
Configurations PMs (vCPUs, RAM)	[(12,384)]	[(12,384)]	[(12,384), (24,768)]
Taux d'utilisation	20%	20%	20%
$O_u/O_{\bar{u}}$	0.735/3	0.735/3	0.735/3
$\lvert P_u\rvert/\lvert P_{\bar{u}}\rvert$	2/2	2/2	1/1
N_t	7	6	4
Gains	42.86%	33.33%	50%

(a) Résultats théoriques avec instanciations manuelles

	Valeurs
Nombre de VMs	[1000, 3000]
Configurations VMs (vCPUs, RAM)	[(4,8), (6,12), (8,16)]
Configurations PMs (vCPUs, RAM)	[(12,384), (24,768)]
Taux d'utilisation	20%
$O_u/O_{\bar{u}}$	0.735/3

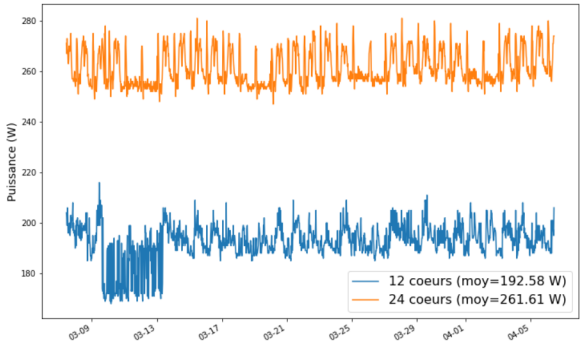
(b) Paramètres de simulation

Scénario	Gains (%)
Utilisation 100%	1.60
Utilisation 20%	55.93
Utilisation 0%	67.75

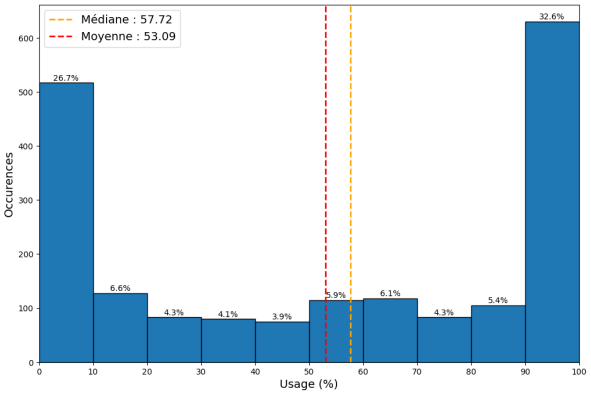
(c) Gains en % avec 2000 simulations

TABLE 3 – Résultats des instanciations ; Paramètres et résultats des simulations

Pour évaluer un passage à l'échelle, nous calculons les résultats avec un nombre de VMs plus élevé et de nouvelles configurations distribuées aléatoirement. Grâce aux paramètres de la Table 3b, nous pouvons calculer la borne inférieure et supérieure du gain, qui correspondent à des taux d'utilisation des VMs de 100% et 0%. Enfin, nous convertissons les gains obtenus en % de puissance électrique économisée, en se basant sur les résultats de la Figure 3a. Les résultats sont présentés dans la Table 3c.
À l'échelle, nous obtenons une réduction de consommation électrique de 56%. Pour évaluer la pertinence de ce résultat appliqué au cluster initialement étudié, nous traçons la répartition de l'utilisation de ses VMs dans la Figure 3b. Nous observons une distribution plus hétérogène de l'utilisation des machines avec un ensemble conséquent de machines faiblement utilisées. Cela traduit un potentiel de réduction de consommation en appliquant notre méthodologie dans cette infrastructure.



(a) Puissance moyenne sur les serveurs 12 & 24 coeurs de l'infrastructure sur 30 jours



(b) Répartition de la quantité de VMs utilisée dans un parc hétérogène sur un an

FIGURE 3 – Puissance des serveurs et distribution de l'utilisation d'un parc hétérogène de VMs

4. État de l'art

Différents travaux se sont concentrés sur les bénéfices de la virtualisation pour contenir la consommation électrique et l'impact environnemental des datacenters [2, 14, 27]. Dans [25], Waldspurger *et al.* proposent de nouveaux mécanismes de réclamation de mémoire pour les serveurs VMWare ESXs, plus tard étudiés et implémentés dans d'autres hyperviseurs [4] au vu des gains qu'ils procurent. Dans [28], Zhang *et al.* cherchent à limiter les dégradations de performance engendrées par le ballooning sur Xen. Liu *et al.* [17] proposent eux un système visant à gérer la surallocation de mémoire via d'un côté le ballooning ou de l'autre l'ajout à chaud de mémoire [20]. Ces travaux ne font cependant pas mention des bénéfices que ces derniers peuvent apporter d'un point de vue consommation électrique. Ici, nous tirons profit de ces mécanismes et proposons une méthode de consolidation visant à limiter la consommation électrique d'un parc de serveurs. Au delà de la mémoire, d'autres ressources doivent être gérées finement par les opérateurs de datacenters. Une réponse à cet enjeu est l'utilisation de l'élasticité [10], qui permet de d'ajouter et retirer de manière dynamique des ressources pour s'adapter à la charge de travail sur un système. Quand Tesfatsion *et al.* [24] utilisent l'élasticité verticale et des mécanismes comme le DVFS [9] pour réduire la consommation de leurs systèmes, d'autres chercheurs comme Jangiti *et al.* [13] avancent l'utilisation de l'élasticité horizontale, qui se rapproche de la consolidation. Les travaux de consolidation suivent généralement une approche similaire pour leur résolution : (1) Détection des hôtes *idles* ou surchargés, (2) Sélection des VMs à migrer, (3) Placement et migration des VMs. Beloglazov *et al.* [5] utilisent cette approche avec différentes variantes (Best Fit Decreasing, Minimum Migration, Random Choice) pour les points (2) et (3). Mashhadi Moghaddam *et al.* [19] souligne l'importance de la prédiction en utilisant des modèles de machine learning afin d'améliorer l'efficacité du placement des VMs. Ces travaux, comme beaucoup d'autres [16, 21, 22], se basent prioritairement sur la consommation CPU des *workloads*, qui concorde avec une utilisation en HPC. Nous nous concentrons nous sur des traces d'un fournisseur de Cloud privé, sujet à des utilisations plus hétérogènes. D'autres approches utilisent des métriques diverses pour résoudre ce problème. Hieu *et al.* [11] proposent une prédiction multi-usage basée sur le CPU, la mémoire et l'utilisation disque des machines. Wood *et al.* [26] utilisent eux une approche en "black-box" se basant sur les consommations CPU, RAM et réseaux des VMs. Bien que proche de notre travail, ils ne traitent pas la question de consommation énergétique dans leur article. Dans une autre mesure, Bacou *et al.* [3] calcule la probabilité qu'une VM soit *idle*, et placent les VMs en fonction de cette probabilité. Cette démarche se rapproche de la notre pour le point (2), sans pour autant utiliser la consommation réseau des VMs. Également, nous proposons une nouvelle approche pour le placement des VMs en utilisant les mécanismes de surallocation mémoire.

5. Conclusion & travaux futurs

La gestion des ressources est un challenge pour de nombreux fournisseurs de service Cloud aujourd'hui du fait qu'elle joue un rôle dans l'impact environnemental des datacenters. L'utilisation croissante du numérique, associée aux effets rebond, génère chez les utilisateurs une perception de ressources infinies et un surdimensionnement des infrastructures Cloud. Pour répondre à ces challenges, les opérateurs de datacenter peuvent utiliser des méthodes comme l'élasticité et la consolidation de VMs, mais des restrictions financières et techniques peuvent limiter leur démocratisation.

Nous proposons ici une méthode de consolidation basée sur des observations en boite noire (depuis l'hyperviseur) de la consommation de ressources des VMs. Couplées à un algorithme propice et des connaissances préalables de l'infrastructure, nous sommes capables d'évaluer l'utilisation d'un ensemble de VMs. Grâce à ces observations, nous séparons nos VMs en 2 sous ensembles, respectivement *Idle* et *Used*. Notre méthode de consolidation utilise les mécanismes de gestion de surallocation mémoire implémentés dans l'hyperviseur. Nous plaçons les VMs *idle* sur des serveurs à forte surallocation et les VMs *used* sur des serveurs à faible surallocation. Ceci nous permet de maintenir un bon niveau de qualité de service en réduisant la consommation théorique du datacenter de 56%. Nos travaux futurs porteront sur la prédiction de l'utilisation des VMs afin de proposer un système dynamique. Grâce à un algorithme de prédiction suffisamment précis, nous pourrons implémenter en conditions réelles notre système et en évaluer les gains. Par la suite, nous travaillerons sur l'évaluation de notre algorithme à plus grande échelle. Étendre notre étude sur un ensemble de VMs plus hétérogènes avec des applications diverses nous permettra de diversifier et d'affiner notre algorithme d'évaluation d'utilisation.

Bibliographie

1. Data Centres and Data Transmission Networks – Analysis.
2. Atiewi (S.), Abuhussein (A.) et Saleh (M. A.). – Impact of Virtualization on Cloud Computing Energy Consumption : Empirical Study. – In *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control*, *ISCSIC '18*, ISCSIC '18, pp. 1–7, New York, NY, USA, septembre 2018. Association for Computing Machinery.
3. Bacou (M.), Todeschi (G.), Tchana (A.), Hagimont (D.), Lepers (B.) et Zwaenepoel (W.). – Drowsy-DC : Data center power management system. – p. 825, mai 2019.
4. Barham (P.), Dragovic (B.), Fraser (K.), Hand (S.), Harris (T.), Ho (A.), Neugebauer (R.), Pratt (I.) et Warfield (A.). – Xen and the art of virtualization. – In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, *SOSP '03*, SOSP '03, pp. 164–177, New York, NY, USA, octobre 2003. Association for Computing Machinery.
5. Beloglazov (A.) et Buyya (R.). – Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation : Practice and Experience*, vol. 24, n13, 2012, pp. 1397–1420. – _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.1867.
6. Cortez (E.), Bonde (A.), Muzio (A.), Russinovich (M.), Fontoura (M.) et Bianchini (R.). – Resource Central : Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. – In *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 153–167, Shanghai China, octobre 2017. ACM.
7. Fu (X.) et Zhou (C.). – Virtual machine selection and placement for dynamic consolidation in Cloud computing environment. *Frontiers of Computer Science*, vol. 9, n2, avril 2015, pp. 322–330.
8. Gupta (U.), Kim (Y. G.), Lee (S.), Tse (J.), Lee (H.-H. S.), Wei (G.-Y.), Brooks (D.) et Wu (C.-J.). – Chasing Carbon : The Elusive Environmental Footprint of Computing. *arXiv :2011.02839 [cs]*, octobre 2020. – arXiv : 2011.02839.
9. Herbert (S.) et Marculescu (D.). – Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. – In *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED '07)*, pp. 38–43, août 2007.
10. Herbst (N. R.), Kounev (S.) et Reussner (R.). – Elasticity in Cloud Computing : What It Is, and What It Is Not.
11. Hieu (N. T.), Francesco (M. D.) et Ylä-Jääski (A.). – Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers. *IEEE Transactions on Services Computing*, vol. 13, n1, janvier 2020, pp. 186–199. – Conference Name : IEEE Transactions on Services Computing.
12. Jacquet (P.), Rouvoy (R.) et Ledoux (T.). – La chasse au gaspillage dans le cloud et les data centers, janvier 2023.
13. Jangiti (S.), Sriram (V. S.) et Logesh (R.). – The role of cloud computing infrastructure elasticity in energy efficient management of datacenters. – In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pp. 758–763, septembre 2017.
14. Jin (Y.), Wen (Y.) et Chen (Q.). – Energy efficiency and server virtualization in data centers : An empirical investigation. – In *2012 Proceedings IEEE INFOCOM Workshops*, pp. 133–138, mars 2012.
15. Koomey (J.), Berard (S.), Sanchez (M.) et Wong (H.). – Implications of Historical Trends in the Electrical Efficiency of Computing. *IEEE Annals of the History of Computing*, vol. 33, n3, mars 2011, pp. 46–54. – Conference Name : IEEE Annals of the History of Computing.
16. Li (Z.), Yu (X.), Yu (L.), Guo (S.) et Chang (V.). – Energy-efficient and quality-aware VM consolidation method. *Future Generation Computer Systems*, vol. 102, janvier 2020, pp. 789–809.
17. Liu (H.), Jin (H.), Liao (X.), Deng (W.), He (B.) et Xu (C.-z.). – Hotplug or Ballooning : A Comparative Study on Dynamic Memory Management Techniques for Virtual Machines. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, n5, mai 2015, pp. 1350–1363. – Conference Name : IEEE Transactions on Parallel and Distributed Systems.
18. Masanet (E.), Shehabi (A.), Lei (N.), Smith (S.) et Koomey (J.). – Recalibrating global data center energy-use estimates. *Science*, vol. 367, n6481, février 2020, pp. 984–986.
19. Mashhadi Moghaddam (S.), O'Sullivan (M.), Walker (C.), Fotuhi Piraghaj (S.) et Unsworth (C. P.). – Embedding individualized machine learning prediction models for energy efficient VM consolida-

tion within Cloud data centers. *Future Generation Computer Systems*, vol. 106, mai 2020, pp. 221–233.

20. Pinter (S.), Aridor (Y.), Shultz (S.) et Guenender (S.). – Improving machine virtualization with 'hot-plug memory'. – In *17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'05)*, pp. 168–175, octobre 2005. – ISSN : 1550-6533.
21. Sharma (Y.), Si (W.), Sun (D.) et Javadi (B.). – Failure-aware energy-efficient VM consolidation in cloud computing systems. *Future Generation Computer Systems*, vol. 94, mai 2019, pp. 620–633.
22. Srikantaiah (S.), Kansal (A.) et Zhao (F.). – Energy aware consolidation for cloud computing. – In *Proceedings of the 2008 conference on Power aware computing and systems*, *HotPower'08*, HotPower'08, p. 10, USA, décembre 2008. USENIX Association.
23. Stauffer (J. M.), Megahed (A.) et Sriskandarajah (C.). – Elasticity management for capacity planning in software as a service cloud computing. *IISE Transactions*, vol. 53, n4, avril 2021, pp. 407–424. – Publisher : Taylor & Francis _eprint : https ://doi.org/10.1080/24725854.2020.1810368.
24. Tesfatsion (S. K.), Wadbro (E.) et Tordsson (J.). – A combined frequency scaling and application elasticity approach for energy-efficient cloud computing. *Sustainable Computing : Informatics and Systems*, vol. 4, n4, décembre 2014, pp. 205–214.
25. Waldspurger (C. A.). – Memory resource management in VMware ESX server.
26. Wood (T.), Shenoy (P.), Venkataramani (A.) et Yousif (M.). – Sandpiper : Black-box and gray-box resource management for virtual machines. *Computer Networks*, vol. 53, n17, décembre 2009, pp. 2923–2938.
27. Yamini (B.) et Vetri Selvi (D.). – Cloud virtualization : A potential way to reduce global warming. – In *Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010)*, pp. 55–57, novembre 2010.
28. Zhang (P.), Chu (R.) et Wang (H.). – MemHole : An Efficient Black-Box Approach to Consolidate Memory in Virtualization Platform. – In *2012 41st International Conference on Parallel Processing Workshops*, pp. 608–609, septembre 2012. – ISSN : 2332-5690.