Compas'2015 : Parallélisme/ Architecture / Système
Lille, France, du 30 juin au 3 juillet 2015

Ordonnancement contrôlé de migrations à chaud

Vincent Kherbache †, Éric Madelaine †, Fabien Hermenier ‡

† prénom.nom@inria.fr
INRIA Sophia Antipolis
‡ fabien.hermenier@unice.fr
Université Nice Sophia Antipolis, CNRS, I3S, UMR 7271

Résumé
Migrer à chaud une machine virtuelle (VM) est une opération basique dans un centre de données. Tous les jours, des VM sont migrées pour répartir la charge, économiser de l'énergie ou préparer la maintenance de serveurs en production. Bien que les problèmes de placement des VM soient beaucoup étudiés, on observe que la gestion des migrations permettant de transiter vers ces nouveaux placements reste un domaine de second plan. On observe alors des algorithmes de placement de qualité, couplés à des algorithmes d'ordonnancement prenant des décisions peu pertinentes causées par des hypothèses irréalistes.
Nous présentons dans ce papier mVM, un ordonnanceur de migrations reposant sur un modèle précis du réseau et du protocole de migration à chaud. Cet ordonnanceur a été intégré en place de celui du gestionnaire de VM BtrPlace. Nos premières expérimentations montrent que les durées des migrations sont estimées à 1.5 secondes près. Cette précision a permis de calculer de meilleurs ordonnancements, réduisant la durée des migrations par 3.5 comparée à BtrPlace.

Mots-clés : machine virtuelle, migration à chaud, ordonnancement

---

1. Introduction

La migration à chaud (*live-migration*) est une des fonctionnalités les plus significatives introduite par les infrastructures virtualisées. Elle est exploitée quotidiennement par les gestionnaires de machines virtuelles (VM) pour répartir la charge, améliorer les performances, réduire la consommation énergétique [7, 12] ou préparer la maintenance de serveurs en production.
Dans la pratique, un gestionnaire de VM calcule un nouveau placement pour certaines VM puis un ordonnancement indiquant quand lancer chaque migration. Bien que les problèmes de placement des VM soient beaucoup étudiés, on observe que la gestion des migrations permettant de transiter vers ces nouveaux placements reste un domaine de second plan [12]. Cette phase est cependant critique car chaque migration à un coût en terme de CPU, de bande passante et d'énergie. Des algorithmes de décision reposent alors sur des hypothèses irréalistes et calculent des ordonnancements conduisant à des migrations inutilement longues et incontrôlables qui réduisent finalement les bénéfices attendus de la ré-organisation des VM.
Nous présentons dans ce papier mVM, un ordonnanceur de migrations reposant sur un modèle précis du réseau et du protocole de migration à chaud. En fonction de la topologie réseau et de la charge des VM, mVM calcule pour chaque migration la bande passante à allouer et le

moment pour démarrer l'action afin de réduire sa durée au minimum. Ce modèle se substitue à celui de l'ordonnanceur de BtrPlace [6]. L'évaluation de mVM a été réalisé par rapport à la version actuelle de BtrPlace. Par défaut, BtrPlace embarque le modèle de migration de Entropy [7] qui est également similaire aux ordonnanceurs de migrations embarqués dans [2, 4]. Cette comparaison permet d'estimer les bénéfices pratiques de mVM face à un modèle représentatif de l'état de l'art. De premières expérimentations montrent que les durées des migrations sont estimées à 1.5 secondes près tandis que BtrPlace les sous-estimaient de 48 secondes. Cette précision a permis de meilleurs ordonnancements, réduisant la durée des migrations par 3.5. Le contrôle des migrations nous a également permis d'implanter des contraintes de synchronisation, minimisant la consommation énergétique d'une reconfiguration ou sa consommation instantanée.

2. Modélisation du problème d'ordonnancement

Notre modèle d'ordonnancement est d'abord constitué d'un modèle réseau assurant le partage de la bande passante dans le temps. Il est complété par une modélisation du protocole de migration des VM afin de prédire leur durée en fonction de la bande passante et de leur charge.

2.1. Modélisation d'un réseau bloquant

Réaliser une migration revient à déplacer une VM d'un serveur vers un autre au travers d'un réseau. Pour des raisons principalement économiques un réseau est rarement non-bloquant. Par exemple, un lien inter-commutateur peut être est trop faible pour assurer un débit satisfaisant la somme des débits maximum des liens entrants. Notre modèle réseau représente le trafic généré par chaque migration en fonction du temps et de la bande passante disponible sur son chemin à travers un ensemble d'éléments réseaux bloquants. Le modèle considère pour chaque migration, une réservation constante de bande passante, un routage statique offrant un chemin unique entre le serveur source et destination, et une latence nulle, donc une propagation instantanée de la migration sur le réseau.

Le modèle réseau considère un ensemble de migrations $\mathcal{M}$ à faire transiter sur un ensemble d'éléments réseaux $\mathcal{N}$ (interfaces, commutateurs). Pour tout élément $n \in \mathcal{N}$, $capa(n)$ indique sa capacité en Mbps. Pour toute migration $m \in \mathcal{M}$, $path(m) \subseteq \mathcal{N}$ indique les éléments traversés (serveur hôte et destination inclus), $bw(m)$ la bande passante allouée en Mbps, $st(m)$ et $ed(m)$ indiquent le début et la fin de l'opération, respectivement. L'équation (1) modélise alors le partage de la capacité d'un élément réseau parmi les migrations qu'il supporte :

$$\forall n \in \mathcal{N}, \forall t \in \mathbb{N}, \sum_{\substack{m \in \mathcal{M},\ n \in path(m),\\ t \in [st(m);\ ed(m)[}} bw(m) \quad < \quad capa(n) \tag{1}$$

2.2. Modélisation du processus de migration

Le modèle de migration repose sur l'algorithme de pre-copy [5] utilisé entre autre dans Xen et KVM. Nous considérons un stockage partagé, qu'il n'est pas nécessaire de migrer. L'algorithme de pre-copy est un processus itératif dont la première phase consiste à envoyer la totalité de la mémoire utilisée par la VM sur le serveur de destination alors que celle-ci est toujours en cours de fonctionnement. Les phases suivantes consistent à envoyer de manière itérative les pages mémoire ré-écrites durant le précédent transfert. Ainsi, le temps de migration dépend du taux de ré-écriture des pages mémoire et de la bande passante allouée à la migration. La migration se termine lorsque la quantité de pages mémoire ré-écrites est suffisamment faible pour être envoyée en un temps inférieur à la période d'indisponibilité (30 ms. par défaut). Une fois cette

condition respectée, la VM est rendue indisponible sur le serveur source, les dernières pages mémoire sont transférés et la VM est réveillée sur le serveur de destination.
D'après la majorité des charges observées [10, 1], le processus de ré-écriture des pages mémoire peut être séparé en deux phases. La première phase correspond à la ré-écriture des *hot-pages*, les pages mémoire qui sont très rapidement ré-écrites. La seconde phase représente l'évolution, supposé linéaire, des *cold-pages* qui correspondent aux pages mémoire générées après la ré-écriture des *hot-pages*. La quantité de *hot-pages* HP_s en MiB et le temps HP_d que prend leur ré-écriture, déterminent $HP_r = \frac{HP_s}{HP_d}$, la bande passante minimale à allouer à une migration pour assurer sa terminaison. Prédire la terminaison d'une migration revient donc à mesurer HP_r sur une durée égale à la période d'indisponibilité D pour s'assurer que sa valeur est inférieure à la bande passante disponible. En effet, les pages mémoire ré-écrites dans cette intervalle seront transmises au dernier tour de l'algorithme de pre-copy, elles doivent donc pouvoir être transmises en un temps inférieur à D. CP_r défini la bande passante requise pour transférer les *cold-pages*. Elle se mesure à partir de $t = HP_d$ et reste dépendante de la charge de la VM.
Étant donné une migration $m \in \mathcal{M}$, avec $mu(m)$ la quantité de mémoire utilisée par la VM en MiB et $bw(m)$ la bande passante allouée en MiB/s, sa durée minimale d_{min} correspond à $d_{min} = \frac{mu(m)}{bw(m)}$.
De manière générale, le temps nécessaire pour transférer une quantité de mémoire X à une bande passante Y et un taux de réécriture des pages mémoires Z, peut s'écrire : $\frac{X}{Y-Z}$ [15]. Ainsi, en considérant notre approche en 2 phases et en supposant que d_{min} est toujours supérieure au temps nécessaire à la ré-écriture des hot-pages, la durée d d'une migration peut alors s'écrire :

$$d = d_{min} + \underbrace{\left(\frac{HP_s + (d_{min} - HP_d) \cdot CP_r}{bw(m) - CP_r}\right)}_{\text{Envoi itératif des } \textit{cold-pages}} + \underbrace{\left(\frac{HP_s}{bw(m)} \cdot \frac{HP_r}{bw(m) - HP_r}\right)}_{\text{Envoi itératif des } \textit{hot-pages}} + D$$

3. Implantation

mVM à été développé à partir de BtrPlace [6], une version améliorée du gestionnaire de VM Entropy [7] qui permet d'enrichir le système de décision par le biais d'extensions. BtrPlace utilise la programmation par contraintes et repose principalement sur la librairie Java Choco [1]. Le cœur du problème d'ordonnancement est principalement modélisé par des contraintes *cumulatives*. Une contrainte cumulative ordonne des tâches sur une ressource bornée. Chaque tâche a une hauteur, une durée et une date de démarrage variable. La contrainte assure alors qu'à tout moment, la hauteur cumulée des tâches en cours d'exécution ne dépasse pas la hauteur de la ressource. Nous utilisons une contrainte cumulative par élément réseau où chaque tâche représente une migration et la hauteur la bande passante disponible.
Le développement du modèle de migration et du modèle réseau représente environ 1200 lignes de Java et se substitue au modèle de BtrPlace. La topologie réseau et les routes peuvent être importées depuis le format de description de `SimGrid` [2]. Les métriques nécessaires à la prédiction des temps de migration sont récupérées depuis `libvirt` par une commande `Qemu` développée pour l'occasion qui capture rapidement et de manière non intrusive l'état mémoire des VM.
En s'appuyant sur notre nouveau modèle et les capacités de composabilité de BtrPlace, nous avons également développé des contraintes contrôlant les migrations pour des besoins particuliers. Par exemple, `sync` s'inspire de Comma [16] pour synchroniser la fin de certaines

1. `http://choco-solver.org`
2. `http://simgrid.gforge.inria.fr`

migrations et ainsi réduire le temps où des VM fortement communicantes et à migrer se retrouvent sur des serveurs différents. Nous avons également implanté des contraintes forçant la séquentialité de migrations ou établissant des règles de précédences.
En reprenant le modèle énergétique d'une migration énoncé dans [10], nous avons finalement implanté une contrainte limitant la puissance instantanée consommée par une infrastructure durant la reconfiguration. Cette contrainte permet par exemple d'aligner la consommation énergétique à la disponibilité d'énergies renouvelables. mVM peut alors limiter le nombre de migrations à exécuter en parallèle, les reporter à des moments plus opportuns, ou retarder au possible l'allumage des serveurs les accueillant.

4. Évaluation

Dans cette section, nous évaluons d'abord la précision de notre modèle et le gain apporté par rapport à BtrPlace sur une infrastructure réelle. Nous validons ensuite l'intérêt pratique de mVM à intégrer des contraintes énergétiques, et nous évaluons ces performances sur différentes tailles de problèmes. Nous avons utilisés exclusivement BtrPlace comme socle pour les comparaisons. Premièrement, car son modèle de migration est le même que dans d'autres solution de gestion de VMs [7, 2, 4] représentatives. Leur prises de décisions seront donc les mêmes. Deuxièmement, car il permet une comparaison précise avec mVM étant donné que la seule différence entre ces deux logicielles est l'ordonnanceur de migration qui est la contribution de ce papier.

4.1. Précision du modèle et bénéfices sur le temps de migration

Pour évaluer la précision de notre modèle et le gain par rapport à BtrPlace nous avons effectué une série d'expérimentations sur la plateforme Grid'5000 [3]. Chaque serveur dispose de 2 processeurs avec quatre cœurs chacun et de 16 GiB de RAM. Chaque serveur est connecté à un commutateur en gigabit ethernet et à un réseau Infiniband 20 Gbps. Le réseau Infiniband partage les images des VM via un serveur NFS dédié tandis que le réseau ethernet supporte les migrations. L'hyperviseur utilisé est Qemu (KVM) 2.2.50, la configuration des VM et les ordres de migration sont réalisés par `libvirt`.
L'expérience consiste à migrer les VM de 4 serveurs source vers 1 serveur destination. Pour émuler un réseau bloquant, les interfaces ethernet des serveurs sources sont bridées à 500 Mbps. Chaque serveur source héberge 2 VM. Chaque VM a un processeur et 2 GiB de RAM. Des niveaux de charge différents sont simulés sur la moitié des VM à l'aide de l'outil `stress` et l'utilisation effective de la mémoire est fixée à 1 ou 1.5 GiB selon la VM.
La Table 1 rapporte la moyenne des durées prédites et observées des migrations en fonction de l'ordonnanceur. On observe d'abord que le modèle de mVM est beaucoup plus précis que le modèle actuel de BtrPlace avec une erreur absolue inférieure à 2 secondes. La faible précision de BtrPlace s'explique par l'ignorance de la topologie du réseau et par une parallélisation exagérée des migrations qui réduit la bande passante disponible par VM et entraîne beaucoup plus de phases de re-transfert des pages mémoire. Nous observons également que les migrations ont été 3.5 fois plus rapides avec mVM. Ce gain s'explique par une parallélisation optimale des migrations (voir Figure 1) : l'ordonnanceur a groupé les migrations selon leurs durées, en les migrant 2 par 2 pour saturer le lien à 1 Gbps tout en séquentialisant les migrations des VM co-localisées (représenté par une même couleur).

	BtrPlace	**mVM**
Prédiction	35.71 s	23.46 s
Observation	83.9 s (σ=1.7)	24.6 s (σ=0.6)

TABLE 1 – Durées moyennes prédites et observées des migrations sur 10 exécutions.

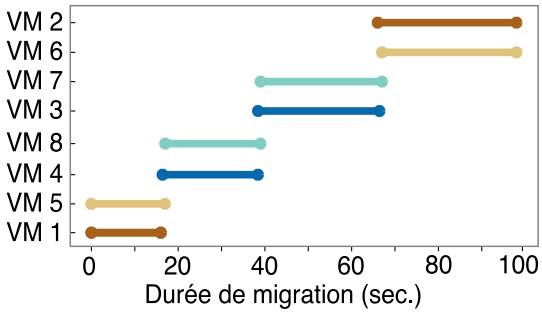

FIGURE 1 – Séquencement des migrations décidé par mVM

4.2. Contrôle des migrations pour l'efficacité énergétique

Cette évaluation simule le décommissionnement de deux groupes (*racks*) de 12 serveurs, hébergeant chacun 4 VM, vers un rack de serveurs plus récents. Chaque rack est connecté à son propre commutateur par des liens 1 Gbps. Les commutateurs sont reliés par un lien 4 Gbps. Les nouveaux serveurs sont initialement éteints et les anciens serveurs seront éteints une fois vidés de leur VM. Le processus de reconfiguration, incluant des migrations, des démarrages et des extinctions de serveurs, est encadré par une contrainte limitant la puissance instantanée de l'infrastructure. Une telle contrainte est nécessaire pour éviter des situations de surchauffe par exemple [13], lorsque le centre de données est alimenté par des énergies renouvelables, ou sous le contrôle d'une autorité de régulation énergétique. Pour calibrer mVM avec des valeurs énergétique réalistes, le modèle énergétique des migrations a été calibré par les valeurs expérimentales de [10], la consommation à vide d'un serveur et la consommation des VM ont été mesurées depuis des serveurs de Grid'5000.

La Figure 2 représente l'évolution de la puissance consommée selon différents seuils. Sans seuil, le pic de consommation est de 5.9 kW. Il est atteint lorsque les premières migrations sont lancées, juste après le démarrage simultané des 12 serveurs de destination. Avec un premier seuil à 5.2 kW, on observe une réduction de la puissance instantanée tout en conservant un temps de reconfiguration équivalent à 5 secondes près. Ce phénomène, induisant un gain énergétique de 5.86%, est justifié par la décision de mVM de synchroniser la fin du démarrage des serveurs avec le début des migrations les remplissant. Le facteur limitant le temps de reconfiguration reste le lien à 4 Gbps, les migrations ne sont pas retardées. Avec un seuil à 5 kW, la reconfiguration prends 2 minutes supplémentaires : le seuil ne peut plus être respecté en retardant uniquement le démarrage des serveurs et mVM a du réduire le parallélisme des migrations.

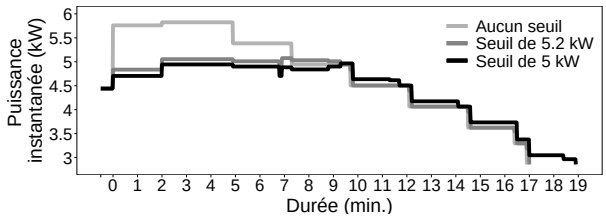

FIGURE 2 – Consommation énergétique de l'infrastructure en fonction d'une contrainte de seuil

5. Performance

Planifier chaque migration en fonction de la bande passante disponible est un problème NP-Complet. En pratique, le temps requis pour calculer le plan de migration dépend de la quantité de VM à migrer, du nombre d'éléments réseau et de leur capacités. Dans cette expérimentation nous évaluons la durée de calcul pour l'ordonnancement d'un scénario de décommissionnement jusqu'à 10 fois plus important que l'expérimentation réalisée en section 4.2. Le nombre de serveurs, de VM, la capacité des commutateurs et le nombre de racks ont ainsi été multiplié par 10 à la plus grande échelle pour atteindre un ensemble de 960 VM à migrer à partir de 20 racks vers 10 nouveaux racks contenant chacun 12 serveurs, et le cœur du réseau atteignant une bande passante de 100 Gbps.

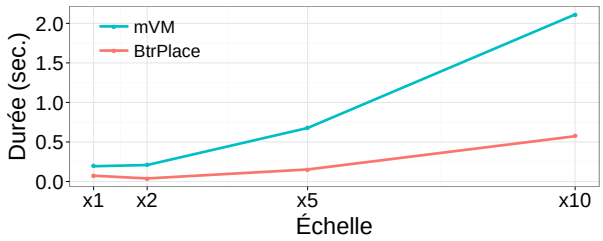

FIGURE 3 – Évaluation des performances de mVM

La figure 3 représente le temps de calcul nécessité par mVM comparée à BtrPlace pour trouver la meilleure solution possible et prouver son optimalité. Comme attendu, nous observons d'abord que la durée de résolution augmente de façon exponentielle étant donné la nature du problème. Toutefois, nous observons de manière générale un faible temps de calcul ainsi qu'un temps supplémentaire acceptable par mVM au regard des bénéfices réalisés en terme de temps de migration et d'efficacité énergétique. Pour trouver le meilleur ordonnancement possible à grande échelle (x10), mVM ne requiert que 1.5 secondes supplémentaires par rapport à BtrPlace. À très grande échelle, le temps de résolution de mVM pourrait devenir significatif. Pour remédier à ce problème, par exemple lors d'un scénario de décommissionnement, une solution raisonnable consiste à répartir l'opération en plusieurs étapes successives. En effet, lorsque la bande passante utilisée pour migrer les VM excède la capacité des commutateurs, le nouveau modèle va choisir de migrer les VM par groupes. Il est ainsi possible de réaliser plusieurs calculs successifs sur de plus petits ensemble de migrations pour obtenir, plus rapidement, le même ordonnancement que pour un unique calcul.

6. État de l'art

Les algorithmes de placement embarquent des ordonnanceurs, souvent dédiés uniquement à la satisfaction des dépendances entre les actions. Les algorithmes sont par contre souvent trop simplifié par rapport à la réalité lorsqu'il est nécessaire de migrer les VM rapidement. Entropy [7], BtrPlace [6] et CloudSim [4] par exemple considèrent une charge nulle pour les VM et un réseau non-bloquant. La durée de migration est ainsi calculée en divisant l'utilisation mémoire de la VM par la bande passante réseau allouée [2]. Cependant, notre précédente étude [9] ainsi que notre évaluation ont cependant révélé l'importance de ces paramètres pour réaliser des ordonnancements de qualité.

Memory buddies [14] s'intéresse à l'impact lié aux migrations concurrentes et propose de limiter le parallélisme par une constante à définir. Cette solution améliore la qualité de la planification dans certains cas, mais le nombre de migrations concurrentes devrait être dynamique pour plus d'efficacité puisqu'il dépend principalement du trafic réseau et du chemin emprunté par la migration.
Des modèles de migration ont été développés pour des simulateurs [1, 10]. Ces modèles sont précis, mais dédiés à la simulation et non à la prise de décisions pour l'amélioration de l'ordonnancement. Contrairement à notre modèle, [10] ne supporte pas les migrations concurrentes. Takahiro *et al.* [8] ont implémentés l'algorithme de *pre-copy* dans le simulateur SimGrid. Bien qu'ils reproduisent avec précision le comportement de KVM pour la transmission des pages mémoire modifiées, ils ne considèrent néanmoins que le taux de pages ré-écrite est proportionnel à l'utilisation du CPU. Dans ce papier, nous définissons le taux d'écriture des pages mémoire de manière statique, dans un processus en deux phases, basé sur l'observation réelle de l'utilisation mémoire des VM et donc indépendamment de la charge CPU.
COMMA [16] synchronise en temps réel la terminaison des migrations de VM fortement communicantes pour réduire l'impact du trafic inter-VM sur la durée des migrations. En contrepartie, COMMA considère un lien réseau unique à traverser, typiquement un lien inter-site. Notre approche prédit les temps de migration hors ligne mais considère une topologie réseau complète.
Finalement, Sarker *et al.* [11] proposent un ordonnanceur qui cherche à réduire le temps de reconfiguration en considérant la topologie réseau et le trafic entre les VM. Leur modèle considère un taux de ré-écriture des pages mémoire fixe et l'évaluation est uniquement réalisée sur simulateur. Nous proposons un modèle de ré-écriture des pages en deux phases, déduit de l'observation de plusieurs charges et validé sur une plateforme expérimentale. En contrepartie, le trafic des VM est ignoré car physiquement séparé du réseau dédié à la migration.

7. Conclusion

Avec des décisions de placement de VM toujours plus efficaces et fréquentes, planifier un ensemble de migrations est devenu une tâche incontournable dans la gestion d'infrastructures virtualisées. En pratique, nous avons constaté qu'une connaissance précise des temps de migration était tout de même nécessaire pour prendre des décisions d'ordonnancement optimales.
Nous avons proposé dans ce papier un ordonnanceur de migrations précis, intégrant les particularités du réseau et de la charge des VM. Par rapport à l'ordonnanceur de BtrPlace, nous avons pu réduire la durée moyenne des migrations par un facteur de 3.5. La flexibilité de notre implantation nous a également permis d'enrichir mVM afin de pouvoir contrôler efficacement les migrations pour des besoins de synchronisation, de séquentialisation ou de gestion de l'énergie.
Par la suite, nous envisageons d'inclure la problématique de placement dans notre modèle. En effet, certains ordonnancements, même optimaux, peuvent s'avérer inappropriés pour l'algorithme de placement. Un couplage fort entre le modèle de placement et le modèle d'ordonnancement pourrait alors permettre de trouver un meilleur compromis.

Remerciements

Ce travail a été financé par le projet européen DC4Cities (FP7-ICT-2013.6.2). Les expériences de ce papier ont été supportées par la plateforme expérimentale Grid'5000 [3] [3], développée par INRIA avec le support du CNRS, de RENATER, d'universités et d'organismes de financement.

Bibliographie

1. Akoush (S.), Sohan (R.), Rice (A.), Moore (A. W.) et Hopper (A.). – Predicting the performance of virtual machine migration. – In *MASCOTS*. IEEE, 2010.
2. Beloglazov (A.) et Buyya (R.). – Energy Efficient Resource Management in Virtualized Cloud Data Centers. – In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, *CCGRID '10*, CCGRID '10, pp. 826–831, Washington, DC, USA, 2010. IEEE Computer Society.
3. Bolze (R.), Cappello (F.), Caron, Daydé (M.) et *al.* – Grid'5000 : A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *Int. Journal of High Performance Computing Applications*, vol. 20, novembre 2006.
4. Calheiros (R. N.), Ranjan (R.), Beloglazov (A.), De Rose (C. A.) et Buyya (R.). – CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software : Practice and Experience*, vol. 41, n1, 2011, pp. 23–50.
5. Clark (C.), Fraser (K.), Hand (S.) et *al.* – Live migration of virtual machines. – In *Proceedings of the 2nd NSDI*. USENIX Association, 2005.
6. Hermenier (F.), Lawall (J.) et Muller (G.). – BtrPlace : A Flexible Consolidation Manager for Highly Available Applications. *IEEE Trans. on Dependable and Secure Computing*, 2013.
7. Hermenier (F.), Lorca (X.), Menaud (J.-M.), Muller (G.) et Lawall (J.). – Entropy : a Consolidation Manager for Clusters. – In *VEE*, NY, USA, 2009. ACM.
8. Hirofuchi (T.), Lèbre (A.) et Pouilloux (L.). – Adding a Live Migration Model into SimGrid : One More Step Toward the Simulation of Infrastructure-as-a-Service Concerns. – In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*volume 1, pp. 96–103. IEEE, 2013.
9. Kherbache (V.), Madelaine (E.) et Hermenier (F.). – Planning Live-Migrations to Prepare Servers for Maintenance. – In *Euro-Par : Parallel Processing Workshops*. Springer, 2014.
10. Liu (H.), Jin (H.), Xu (C.-Z.) et Liao (X.). – Performance and energy modeling for live migration of virtual machines. *Cluster Computing*, vol. 16, n2, 2013.
11. Sarker (T.) et Tang (M.). – Performance-driven live migration of multiple virtual machines in datacenters. – In *IEEE International Conference on Granular Computing*, 2013.
12. Verma (A.), Bagrodia (J.) et Jaiswal (V.). – Virtual Machine Consolidation in the Wild. – In *Middleware'14*, New York, USA, 2014. ACM.
13. Wang (X.) et Wang (Y.). – Coordinating power control and performance management for virtualized server clusters. *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, n2, 2011, pp. 245–259.
14. Wood (T.), Tarasuk-Levin (G.), Shenoy (P.), Desnoyers (P.), Cecchet (E.) et Corner (M. D.). – Memory Buddies : Exploiting Page Sharing for Smart Colocation in Virtualized Data Centers. – In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, *VEE '09*, VEE '09, pp. 31–40, New York, NY, USA, 2009. ACM.

3. https://www.grid5000.fr

15. Zheng (J.), Ng, Sripanidkulchai (K.) et Liu (Z.). – Pacer : A Progress Management System for Live Virtual Machine Migration in Cloud Computing. *IEEE Transactions on Network and Service Management*, vol. 10, n4, décembre 2013.
16. Zheng (J.), Ng (T. S. E.), Sripanidkulchai (K.) et Liu (Z.). – COMMA : Coordinating the Migration of Multi-tier Applications. – In *VEE*, NY, USA, 2014. ACM.