Vers un déploiement conforme et économique des licences dans une architecture Cloud

Arthur Chevalier

Compas'2018 : Parallélisme/ Architecture / Système
ENSEEIHT - Toulouse, France, du 03 au 06 juillet 2018

Vers un déploiement conforme et économique des licences dans une architecture Cloud

Arthur CHEVALIER*

Orange S.A., Université de Lyon, ENS Lyon, Inria, CNRS, Université Claude-Bernard, LIP
Laboratoire LIP - 46 Allée d'Italie
69364 Lyon - France
arthur.chevalier@inria.fr

Résumé
Aujourd'hui, l'utilisation des logiciels est généralement réglementée par des licences, qu'elles soient gratuites, payantes et avec ou sans accès à leurs sources. L'univers des licences est très vaste et mal connu. Souvent on ne connaît que la version la plus répandue au grand public (un achat de logiciel est égale à une licence). La réalité est bien plus complexe surtout chez les grands éditeurs. Cet article présente l'impact et l'importance de la gestion de ces licences lors de l'utilisation de logiciels dans une architecture Cloud. Nous proposons un cas d'étude pour prouver l'impact de la gestion dynamique des licences. Ce cas d'étude portera sur un logiciel vendu par Oracle, leur base de données. Nous étudierons brièvement la situation existante et le problème qui en découle, puis nous proposons un algorithme de déploiement qui prendra en compte deux métriques spécifiques pour ce logiciel. Enfin, nous comparons nos résultats avec divers algorithmes de déploiement existants.

Mots-clés : Licences, Cloud, Déploiement, Gestion des actifs logiciels

1. Introduction

Récemment un procès a eu lieu entre SAP et Diageo [2], une société qui utilise des produits SAP [2]. En raison d'un manque de compréhension des droits d'utilisation lié au flou juridique entourant les licences, la société a été verbalisée d'une amende de 55 millions de livres sterling (75 millions de dollars américains) pour non-conformité. Ce problème a été clairement souligné dans une étude de Flexera où l'on voit que 85% des entreprises sanctionnées pour contrefaçon étaient inconsciemment non conformes. Comment cela est-il possible ?
De nos jours, les logiciels peuvent être licenciés de plusieurs façons en utilisant ce qu'on appelle une métrique. La métrique définit **la façon de calculer le nombre de licences requises pour un logiciel**. Il peut donc être licencié avec des licences x sous la métrique A ou y licences sous la métrique B. Le prix des licences dépend de la métrique utilisée. Les métriques des licences de logiciels sont définies contractuellement soit dans les Conditions Générales de Vente figurant sur le site de l'éditeur, soit par un contrat entre l'acheteur et l'éditeur. Les conditions générales de vente disponibles en ligne peuvent être modifiées à tout moment, mais une licence achetée

*. Merci à mes directeurs et co-directeurs de thèse : Eddy Caron (ENS Lyon) et Noëlle Baillon-Bachoc (Orange).
2. https ://www.sap.com/

avant un changement dans le texte légal doit suivre l'ancienne version de la définition de la métrique, il est donc nécessaire d'avoir un suivi continu de ces métriques et la capacité d'identifier et de récupérer les documents juridiques de chaque licence dans le parc informatique.
L'interprétation des métriques contenues dans les contrats est un véritable casse-tête. Leur absence de clarté génère des incompréhensions et des sources fréquentes de non conformité. Dans le Cloud, il n' y a plus de relation évidente entre le logiciel et le matériel. Sans séparation, chaque instance d'une application peut s'exécuter sur chaque serveur. Pour résoudre ce problème, il est nécessaire de recréer des frontières entre les ressources utilisées par le logiciel et la couche matérielle : un ensemble de ressources, les clusters. Les logiciels doivent fonctionner sur un cluster dédié et ne peuvent pas partager les ressources d'un autre. Lors du déploiement d'une application, il est nécessaire de prendre en compte toutes les métriques utilisées sur le cluster où elle est installée pour vérifier la conformité.
L'intérêt de maîtriser la gestion des métriques est que, dû au fait que le client puisse choisir sa métrique, l'achat de licences avec une métrique spécifique peut être moins cher que l'achat de licences avec une autre pour un même logiciel. Il est donc intéressant de s'interroger sur le choix préférentiel des ces dernières en fonction des usages sur un cluster.
Plus un cluster est grand, plus le besoin d'outils permettant de suivre l'utilisation des logiciels, d'identifier les licences utilisées, de vérifier la conformité et de gérer un placement efficace est grand. Cet outil fait partie d'un processus appelé « Software Asset Management » (SAM) qui doit pouvoir effectuer les actions décrites ci-dessus mais qui permet aussi d'avoir un retour sur investissement et donc de mener une stratégie fournisseur ou même de procéder à des consolidations de budget alloués aux achats de licences.
Dans cet article, nous nous concentrons sur l'impact du choix de la métrique dans le placement d'une application dans une architecture Cloud. Nous nous concentrons les deux métriques de notre cas d'usage : Processor et Named User Plus (NUP). Ces deux métriques sont représentatives des problèmes liés aux licences (coût et compréhension floue du texte juridique) car elles appartiennent à deux familles de licences utilisées par de nombreux éditeurs. Par exemple, IBM a la métrique Processor Value Unit (PVU) tout à fait équivalente à Processor chez Oracle et à celle de SAP (User) pour la métrique NUP d'Oracle.
La définition de ces deux métriques [1] est assez confuse lors de l'achat des licences pour l'entreprise et peut induire des risques de non-conformité avec les licences des applications pré-existantes. Tant que les éditeurs de logiciels continueront à converger vers des modèles de licences complexes, les outils de gestion des actifs logiciels continueront à jouer un rôle majeur.

2. État de l'art

Le sujet du Software Asset Management dans une architecture Cloud est une problématique nouvelle encore très peu étudiée. Les racines du Software Asset Management lui même remontent en 1999 lorsque Holsing et Yen [6] ont proposé une étude conduisant aux premières considérations sur le modèle de l'identification des logiciels. En 2004, Ben-Menachem et Marliss [3] ont souligné la nécessité d'investir et de créer des outils pour ces processus afin d'assurer la gestion à long terme des actifs logiciels. En 2011, McCarthy et Herger [7] ont proposé une solution pour combiner les Technologies de l'Information (TI), les processus et la gestion des ressources logicielles : ceci nécessite la capacité de scanner toute l'infrastructure, de faire un inventaire des licences, d'implémenter la gestion des contrats et de produire des rapports sur l'état de préparation pour la conformité et la vérification. En 2014, Gocek, Kania et Malecki [5] ont décrit le SAM comme des outils de découverte et de collecte d'informations sur les logiciels utilisés dans des environnements surveillés. Récemment, en 2017, Vion, Baillon-Bachoc, Boyer

et De Palma ont passé en revue les outils SAM existants, leurs avantages et ont proposé un modèle SAM pour une architecture Cloud [8].
En ce qui concerne l'algorithme de déploiement, certains auteurs comme Dong et Herbert (2013) [4] ont exprimé leur intérêt pour un déploiement écoénergétique de machines virtuelles dans une architecture Cloud avec un service d'analyse de données. Ils ont réussi à proposer un algorithme capable de réduire significativement la consommation d'énergie et le nombre de migrations VM. Cependant, aucun auteur ne s'est encore penché sur la question du placement des applications dans le Cloud en tenant compte de l'aspect SAM : minimisation de l'utilisation des logiciels, facilité d'inventaire et conformité aux licences utilisées. C'est ce que nous proposerons dans cet article.

3. Le problème du placement d'applications selon leur licences

Notre objectif est de déployer des applications en minimisant le coût total des licences associées et en garantissant le respect des droits acquis par contrat. La plupart des algorithmes de déploiement n'utilisent pas de modèle de métriques pour réduire les coûts. Leur but est d'optimiser le placement des applications par rapport aux ressources physiques, cette stratégie cache le problème de conformité lorsque, par exemple, certaines ressources changent.
Notre cas d'étude est une société qui offre à ses clients l'accès à des bases de données Oracle hébergées sur ses clusters. Chaque nouvelle base de données installée est accédée par des utilisateurs différents.
Dans la FIGURE 1, nous prenons l'exemple de la métrique Processor. Nous devons déployer une base de données sur un cluster avec 4 cœurs ; un algorithme générique calculera le prix de 4 licences avec la métrique Processor et l'ajoutera au prix total avant de placer la base de données sur le cluster. Le problème est que la réservation de nouvelles machines pour les ajouter au cluster entrainera l'ajout de cœurs, il faudra alors acheter 2 nouvelles licences pour rester conforme. Des outils SAM seront nécessaires pour identifier les besoins d'achat de licences à chaque ajout de nouvelles ressources car, dans un cas concret, nous parlons de centaines de métriques potentielles dans un même cluster. Actuellement, à notre connaissance, aucun algorithme n'est capable de gérer le cas d'ajout de ressources.

4. Déploiement de licences

Notre proposition est un algorithme divisé en trois étapes (FIGURE 2). La première étape consiste à déployer le logiciel en cherchant le cluster le plus approprié pour minimiser les coûts de ce logiciel tout en restant conforme. La deuxième étape est un vérificateur de conformité, il vérifie qu'avec toutes les ressources ajoutées au système (nouvelles machines, utilisateurs,

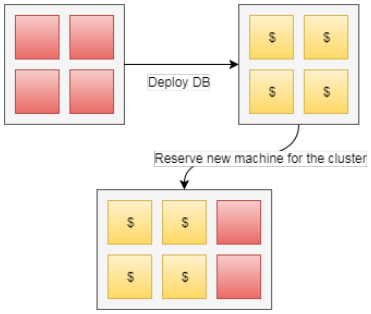

FIGURE 1 – Exemple de la métrique Processor

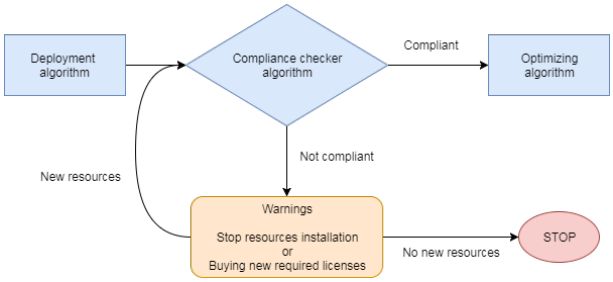

FIGURE 2 – Algorithme de déploiement

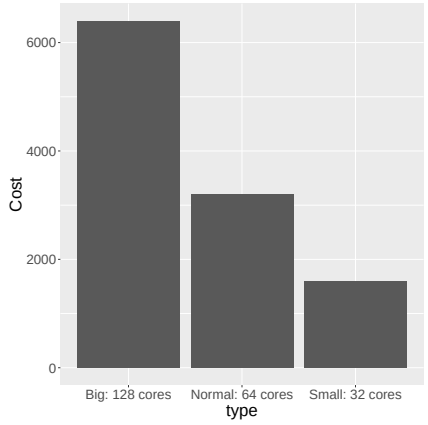

FIGURE 3 – Prix de chaque cluster avec la métrique Processor

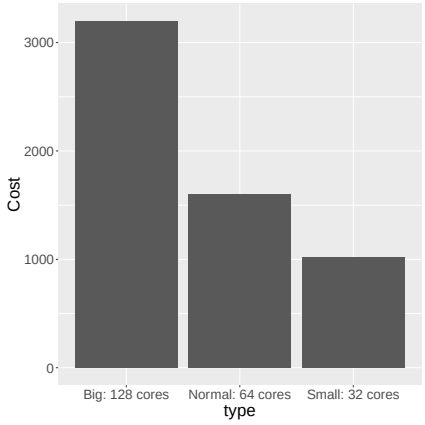

FIGURE 4 – Prix de chaque cluster avec la métrique NUP

nouveaux logiciels, etc.) nous sommes toujours conformes. Dans le cas contraire, cette étape bloque l'installation de toute nouvelle ressource qui briserait la conformité de notre système. La troisième et dernière étape est l'optimiseur, il optimise le placement global de toutes les applications afin de réduire au maximum les coûts, toujours dans le respect des licences.

4.1. Modèles de métriques

Si nous définissons C le nombre de cœurs à licencier par une application et F le core factor alors le nombre de licences L_p nécessaires pour la métrique Processor est : $L_p = \lceil C \times F \rceil$
Comme nous nous concentrons uniquement sur l'édition Entreprise de la base de données Oracle pour la métrique NUP. Pour C le nombre de cœurs et U le nombre d'utilisateurs, N le nombre de licences NUP minimum par licence Processor (comme mentionné N est une constante, nous la fixerons dans notre étude égale à 25), L_n le nombre de licences requis avec la métrique NUP est : $L_n = max(N \times L_p, U)$

4.2. Déploiement standard sur un seul cluster

Pour cette expérience, nous faisons varier le nombre de cœurs de 0 à 64 et le nombre d'utilisateurs à 1024. Le core facteur est fixé à 1 car il s'agit de la valeur standard et le prix d'une licence Processor est cinquante fois supérieur à celui d'une seule licence NUP, de sorte que tous les coûts soient normalisés par rapport au coût de la métrique la moins chère (qui est NUP dans notre cas). Deux points intéressants apparaissent : tout d'abord, nous pouvons voir l'impact du nombre de cœurs sur la fonction de coût de la métrique NUP. En effet, le coût est constant lorsque le nombre d'utilisateurs est plus important que le nombre de cœurs dans la formule. Avec un nombre plus élevé de cœurs, le coût augmentera. Deuxièmement, les deux fonctions se croisent et l'intersection détermine le nombre exact de cœurs où une métrique devient moins chère que l'autre. Ces informations nous indiquent quand choisir une métrique au lieu de l'autre au moment du déploiement.

4.3. Une seule application sur plusieurs clusters du Cloud

Nous voulons comparer les coûts des deux métriques sur plusieurs clusters. Avec trois ensembles de machine virtuellement séparés (clusters) de 128, 64 et 32 cœurs respectivement et 1024 utilisateurs (FIGURE 3 et FIGURE 4)
Nous pouvons voir avec ces deux graphiques que le nombre de cœurs fait varier considérablement le coût de déploiement. En effet, pour ces deux métriques, les coûts sont moindres sur

un cluster avec peu de cœurs et beaucoup plus importants sur un cluster avec plus de cœurs, quelle que soit la métrique utilisée.

4.4. Déploiement de plusieurs applications sur un seul cluster

Pour voir l'impact du nombre de déploiements de base de données nous voulons maintenant comparer le déploiement de plusieurs bases de données sur un seul cluster. Comme l'expérience précédente, nous utilisons un cluster avec 64 cœurs et 1024 utilisateurs pour calculer les coûts. Ensuite nous essayons de déployer le nombre maximum de bases de données, chaque base fonctionnant sur 4 cœurs.

Grâce aux équations de chaque métrique, nous savons que le prix de la métrique NUP augmente assez rapidement alors que le prix de la métrique Processor est constant. Le déploiement de ces bases de données montre que la métrique NUP est bonne pour un petit nombre de bases de données tandis que la métrique Processor permet de déployer autant de bases de données que nous voulons sur un seul et même cluster. Notre algorithme doit évaluer le point où une métrique est plus intéressante que l'autre et ensuite utiliser toutes ces données pour déployer efficacement une application.

4.5. Équation de la séléction de la métrique en fonction du nombre de base de données à déployer

Le point où la métrique doit changer est facilement déduit avec les deux modèles de métriques. Avec P_p et P_n les prix des métriques de Processor et NUP respectivement les coûts des métriques de Processor et des métriques NUP (C_p et C_n respectivement) sont :

$$\begin{cases} C_p &= L_p \times P_p \\ C_n &= L_n \times P_n \end{cases} \tag{1}$$

Le point où nous voulons changer la métrique avec le déploiement de la base de données X est quand : $C_p < C_n \times X$

En raison du modèle de la métrique NUP, nous décomposons ce problème en deux cas :

$$\begin{cases} N \times L_p > U & \rightarrow & L_p \times P_p < N \times L_p \times P_n \times X \\ U > N \times L_p & \rightarrow & L_p \times P_p < U \times P_n \times X \end{cases} \tag{2}$$

$$\begin{cases} N \times L_p > U & \rightarrow & X > \frac{P_p}{N \times P_n} \\ U > N \times L_p & \rightarrow & X > \frac{L_p \times P_p}{U \times P_n} \end{cases} \tag{3}$$

Nous pouvons décrire les mêmes formules avec la constante $\alpha = \frac{P_p}{P_n} = 50$:

$$\begin{cases} N \times L_p > U & \rightarrow & X > \frac{1}{N}\alpha = 2 \\ U > N \times L_p & \rightarrow & X > \frac{L_p}{U}\alpha \end{cases} \tag{4}$$

Avec cette équation, nous pouvons voir que dans le cas où $N \times L_p > U$, la métrique Processor est plus intéressante que la métrique NUP lorsque nous déployons plus de deux bases de données (parce que dans notre cas N est fixé à 25).

4.6. Configuration expérimentale

Notre configuration expérimentale possède 4 clusters (32, 64, 128 et 256 cœurs) et 1000 utilisateurs par base de données. Comme dans les Conditions Générales de Vente une licence Processor est 50 fois plus cher qu'une licence NUP. Enfin nous déployons 64 base de données dans des VMs (une par VM) et chaque VM se voit assigné de 4 cœurs.

Nous pouvons voir dans la FIGURE 5 les différents coûts associés aux différents algorithmes de déploiement. La courbe aléatoire complet (choix du cluster de déploiement et de la métrique)

passe du meilleur déploiement possible au pire, la courbe réelle n'est qu'une des multiples possibilités. Un autre point est que le déploiement du *load-balance* est le pire déploiement possible (dans ce cas) et le déploiement *smallest first* est bien meilleur. Notre algorithme réduit significativement les coûts car il n'utilise que 10 à 15% du coût de l'algorithme *smallest first*.
Notre deuxième configuration applique les mêmes algorithmes sur toutes les configurations possibles de 4 clusters chacun allant de 32 à 256 cœurs. Une moyenne est ensuite faites sur l'ensemble des résultats. Dans la FIGURE 6 nous pouvons voir que notre algorithme est toujours le meilleur possible en terme de coût de déploiement.

5. Discussions [3]

Comme nous pouvons le constater, la mise en place d'applications sur une architecture Cloud basée sur des modèles de licences permet non seulement de réduire considérablement les coûts, mais aussi de garantir à tout moment la conformité avec les éditeurs. Notre solution de déploiement permet une économie d'environ 70% sur les coûts d'investissement par rapport aux algorithmes de base et nous pouvons donc nous attendre à un gain significatif sur les implémentations en cours dans les entreprises.
Il serait intéressant de voir l'impact des variables sur la fonction de coût final afin de donner des orientations aux services acheteurs, par exemple en fonction de leurs infrastructures, ou de pouvoir négocier certains aspects des licences avec les éditeurs, ce que les gros groupes peuvent se permettre. Enfin, l'ajout de différentes familles de licences à ce modèle améliorerait l'algorithme de placement et conduirait graduellement à un algorithme généraliste qui optimiserait les coûts de licence d'une entreprise tout en restant conforme. Le gain réalisé avec cette étude de seulement deux métriques suggère que le gain potentiel pour toutes les familles de paramètres pourrait être tout aussi important.

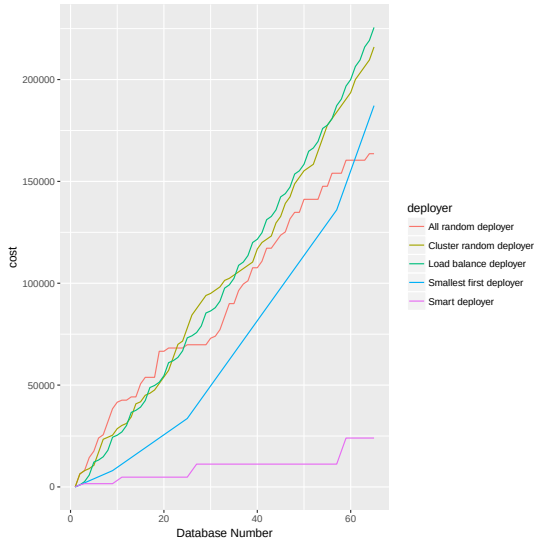

FIGURE 5 – Coût de déploiement de chaque algorithme

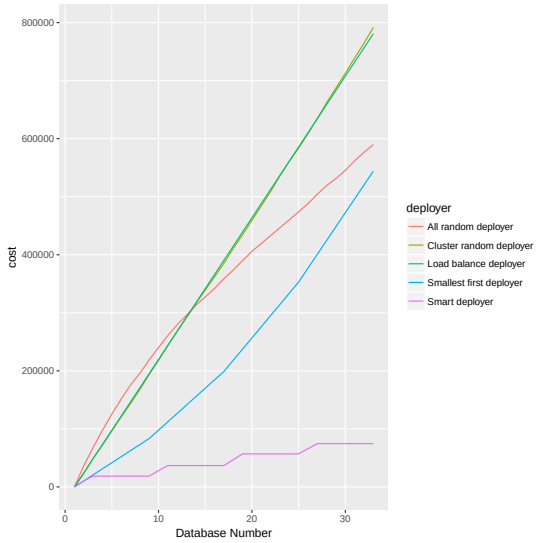

FIGURE 6 – Coûts de déploiement de chaque algorithme (moyenne sur 1000 lancement)

3. Tous les résultats de cet article peuvent être reproduits dans un carnet Jupyter. L'accès protégé est disponible sur simple demande à l'auteur.

Bibliographie

1. Oracle metrics definition, décembre 1 2015.
2. Sap uk ltd v diageo great britain ltd [2017] ewhc 189 (tcc), février 16 2017.
3. Ben-Menachem (M.) et Marliss (G. S.). – Inventorying information technology systems : supporting the "paradigm of change". *IEEE Software*, vol. 21, n5, Sept 2004, pp. 34–43.
4. Dong (D.) et Herbert (J.). – Energy efficient VM placement supported by data analytic service. – In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 648–655, May 2013.
5. Gocek (P.), Kania (P.), Malecki (B.), Paluch (M.) et Stopa (T.). – Obtaining software asset insight by analyzing collected metrics using analytic services, mai 16 2017. US Patent 9,652,812.
6. Holsing (N. F.) et Yen (D. C.). – Software asset management : Analysis, development and implementation. *Information Resources Management Journal*, vol. 12, n3, 1999, p. 14.
7. McCarthy (M. A.) et Herger (L. M.). – Managing software assets in a global enterprise. – In *2011 IEEE International Conference on Services Computing*, pp. 560–567, July 2011.
8. Vion (A.), Baillon-Bachoc (N.), Boyer (F.) et De Palma (N.). – Software license optimization and cloud computing. *CLOUD COMPUTING 2017*, 2017, p. 125.