ComPAS'2013 (RenPar/ SympA/ CFSE)
Grenoble, France, du 16 au 18 janvier 2013

SLA et qualité de service pour le Cloud Computing

Yousri Kouki*, Damián Serrano†, Thomas Ledoux*, Pierre Sens‡, Sara Bouchenak†

* EMN – INRIA, LINA, Nantes
{Yousri.Kouki, Thomas.Ledoux}@inria.fr

† Université de Grenoble I – LIG, Grenoble
{Damian.Serrano, Sara.Bouchenak}@imag.fr

‡ LIP6 – INRIA, Paris
{Pierre.Sens}@lip6.fr

Résumé
Le Cloud computing marque une nouvelle avancée vers l'infrastructure informatique dématérialisée. Le Cloud fournit des ressources informatiques, logicielles ou matérielles, accessible à distance, en tant que service. L'adoption de ce modèle soulève un certain nombre de défis, notamment au sujet de la qualité de service (QoS) des services fournis. En effet, alors qu'on assiste à une multiplication de services similaires sur le Cloud, rien ne nous permet d'évaluer la qualité d'un service rendu. Cet article présente une solution pour intégrer la QoS et le contrat SLA (*Service Level Agreement*) comme éléments à part entière du Cloud. Les contributions de cet article sont les suivantes. Premièrement, nous proposons le modèle de Cloud SLAaaS (*SLA aware as a Service*) qui apporte une dimension supplémentaire au paradigme de Cloud Computing. Deuxièmement, nous présentons le langage Cloud Service Level Agreement (CSLA) qui permet la définition et l'établissement d'un contrat qui supporte l'instabilité de la QoS dans un environnement hautement dynamique comme le Cloud en gérant finement la violation. Enfin, nous développons deux stratégies dirigées par CSLA pour pouvoir contrôler dynamiquement et de manière autonome la taille et la forme du Cloud pour répondre aux contraintes de SLA : i) une stratégie de planification de capacité pour le redimensionnement sur mesure et ii) une stratégie de verrouillage d'exclusion mutuelle pour assurer un accès exclusif à des ressources partagées sur le Cloud. Les expériences montrent l'intérêt de notre solution qui peut s'appliquer à n'importe niveau du Cloud.

Mots-clés : Cloud Computing, qualité de service (QoS), contrat SLA (Service Level Agreement), planification de capacité, exclusion mutuelle

---

1. Introduction

Le Cloud computing (ou informatique dans les nuages) est une métaphore désignant un réseau de ressources informatiques accessibles à distance par le biais des technologies Internet. C'est un modèle informatique selon lequel des ressources informatiques (logicielles et/ou matérielles) sont fournies sous la forme d'un service à la demande [13]. Les XaaS (Anything as a Service) regroupent l'ensemble de ces services et sont régulièrement présentés en trois modèles de service suivant une approche *top-down* : Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS).
De multiples acteurs comme Google ou Amazon offrent des services de Cloud. Certains Clouds présentent des similarités en termes de services fournis, comme par exemple les Clouds SaaS proposant des services de bureautique, ou les Clouds IaaS proposant des services de calcul ou de stockage. Dans ce contexte, il n'est pas simple pour un utilisateur final ou une entreprise voulant faire appel à un service de Cloud de comparer les services existants entre eux avant d'élire le plus approprié. Nous pensons qu'un des éléments différenciateurs entre les offres de Cloud computing sera la qualité de service (QoS)

fournie, la transparence de cette qualité pour l'utilisateur et la capacité du Cloud à être effectivement élastique pour gérer la qualité de service.
Aujourd'hui, la prise en compte de la qualité de service dans le Cloud reste encore incomplète et ad-hoc. Quelques approches [2][8] se sont intéressées à des critères de QoS comme l'indisponibilité du service due à une panne. Cependant, ces solutions ne s'intéressent qu'à un aspect unique de la QoS. Elles ne traitent pas d'autres critères tels que la performance ou la consommation énergétique du Cloud. D'autre part, la vérification du respect de la qualité de service telle que proposée par les solutions existantes est gérée de manière ad-hoc et peut être très fastidieuse pour le client de Cloud. Par exemple, avec EC2 d'Amazon, l'utilisateur doit prouver que son service a été indisponible en capturant lui-même la panne et en la documentant en conséquence, avant d'envoyer la « preuve » à Amazon, le tout dans un délai de 30 jours après l'occurrence de la panne. Ceci va à l'encontre de l'une de principales motivations du Cloud computing qui est de simplifier la fourniture et la gestion de service pour l'utilisateur.
A l'instar du domaine des télécommunications qui a introduit dès les années 80 le concept de *Service Level Agreement* (SLA), nous préconisons l'adoption des SLA dans le monde du Cloud Computing. Ce concept qui a été introduit pour gérer la QoS est un contrat qui définit la QoS prescrite entre un prestataire et un client. Il spécifie un ou plusieurs objectifs de niveau de service SLO (*Service Level Objective*) afin de garantir que la QoS délivrée a satisfait les attentes du consommateur. En cas de violation, des pénalités sont appliquées. La pénalité non seulement pénalise le prestataire de services en réduisant leur profit, mais aussi permet aux utilisateurs de tolérer la défaillance du service.
Dans cet article, nous présentons les premières contributions du projet ANR MyCloud [1]. Nous proposons le modèle de Cloud SLAaaS (SLA *aware as a Service*) qui apporte une dimension supplémentaire au paradigme de Cloud Computing, en enrichissant les services de Cloud de leur qualité de service et en proposant des contrats SLA entre fournisseur et consommateur de service. La fourniture du langage Cloud Service Level Agreement (CSLA) [15] permet la définition et l'établissement d'un contrat qui supporte l'instabilité de la QoS dans un environnement hautement dynamique comme le Cloud en gérant finement la violation. Enfin, nous proposons deux stratégies pour pouvoir contrôler dynamiquement et de manière autonome la taille et la forme du Cloud pour répondre aux contraintes de SLA. La première repose sur une stratégie de redimensionnement basée sur un algorithme de planification de capacité, la seconde repose sur une stratégie de verrouillage pour l'accès à des ressources partagées.
Le reste de l'article est organisé comme suit. La Section 2 propose un certain nombre de définitions préliminaires, nécessaires pour comprendre le reste de l'article. La Section 3 pose les bases de notre modèle SLAaaS. Puis, la Section 4 donne un bref aperçu du langage CSLA. Ensuite, dans la Section 5, deux études de cas différentes mettant en oeuvre les stratégies de contrôle du Cloud permettent d'évaluer notre modèle SLAaaS. La Section 6 présente les travaux connexes alors que la Section 7 conclut l'article et présente les perspectives de ce travail.

2. Définitions préliminaires

2.1. Vocabulaire de base

La qualité de service QoS (*Quality of Service*) désigne la capacité d'un service à répondre par ses caractéristiques aux différentes exigences de ses utilisateurs en termes par exemple de disponibilité (e.g., taux de rejet), fiabilité (e.g., temps moyen entre deux pannes), performance (e.g., temps de réponse) et coût (e.g., économique, énergétique). Les principales composantes de la qualité de service seront fournies par des métriques caractérisées par un type, une unité et une fonction de calcul. Par exemple, concernant le critère QoS de fiabilité, la métrique MTBF (temps moyen entre pannes) peut s'exprimer en heure et se calculer de la façon suivante :

$$\mathrm{MTBF} = \frac{\sum(\text{temps de fonctionnement} - \text{temps de panne})}{\text{nombre de panne}} \quad (1)$$

Le *Service Level Agreement* (SLA) est un document qui définit la qualité de service requise entre un prestataire de service et un client. Le Service Level Agreement, que l'on pourrait traduire en français par accord de niveau de service ou contrat de niveau de service, est donc un contrat dans lequel on formalise la qualité du service en question. Un SLA est composé essentiellement des parties impliquées dans

1. http ://mycloud.inrialpes.fr

le contrat (et leur rôle respectif), la définition des services (leur signature), les objectifs de niveau de service SLO (*Service Level Objective*), une période de validité du contrat et les pénalités quand se produit une violation.

2.2. Protagonistes SLA dans le Cloud

Le Cloud Computing est un modèle qui permet l'accès à la demande et à distance à un ensemble de ressources (applications, réseaux, serveurs, machines virtuelles, supports de stockage) en tant que service. Les XaaS (Anything as a Service) regroupent l'ensemble de ces services et sont régulièrement présentés en trois modèles de service suivant une approche *top-down* : Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS).
Dans le Cloud, les SLAs peuvent être exprimés entre différentes couches (cf. Figure 1), contractualisant de fait des dépendances de ressources entre les différentes couches XaaS et faisant émerger les différents protagonistes des contrats. Ainsi, le client final (*end-user*) est souvent considéré à tord comme le seul consommateur de service, mais les SLAs permettent également l'établissement de contrat inter-couches XaaS, comme par exemple entre le SaaS (le consommateur) et le IaaS (le fournisseur).

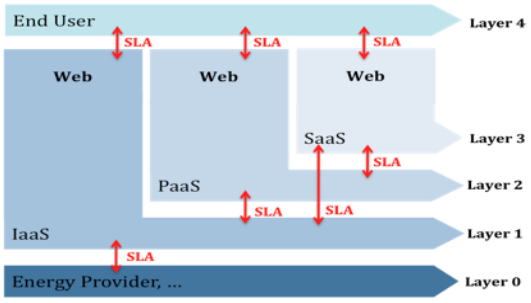


FIGURE 1 – SLA et couches XaaS

3. Modèle de Cloud de SLAaaS

Cette section décrit les grandes lignes d'un Cloud intégrant qualité de service (QoS) et contrat SLA comme éléments à part entière du Cloud.
Le modèle SLAaaS (SLA *aware as a Service*) propose un service orthogonal aux autres modèles de Cloud (IaaS, PaaS et SaaS) et intègre de manière native les concepts de qualité de service et de contrats SLA au Cloud. Ainsi, les services IaaS, PaaS et SaaS hébergés par les Clouds SLAaaS exhibent leur niveau de qualité de service et le modèle SLAaaS propose d'administrer automatiquement des contrats SLA pour assurer la QoS attendue. Un utilisateur voulant faire appel à un service de Cloud SLAaaS comparera les différentes offres existantes de service sur le Cloud, pas uniquement en terme de service rendu mais également en terme de niveau de qualité de service et de garantie de celle-ci. L'utilisateur optera ainsi pour une offre de Cloud en toute connaissance de cause.

Notre modèle SLAaaS (cf. Figure 2) a pour objectifs de :
- Proposer un SLA pour chaque service de Cloud de manière systématique entre l'utilisateur du service et son fournisseur quelque soit le niveau XaaS du Cloud ;
- Administrer le contrat SLA, côté fournisseur, en permettant de configurer, dimensionner, déployer le service de Cloud de telle sorte que le SLA reste garanti, et ceci en prenant en compte l'élasticité et le dynamisme du Cloud ;
- Simplifier la gouvernance de SLA, côté client, en permettant, par exemple, de détecter automatiquement toute violation SLA, allégeant ainsi le client de cette tâche fastidieuse. Le client se voit malheureusement aujourd'hui contraint d'effectuer ceci lui-même, de manière manuelle, ad-hoc.

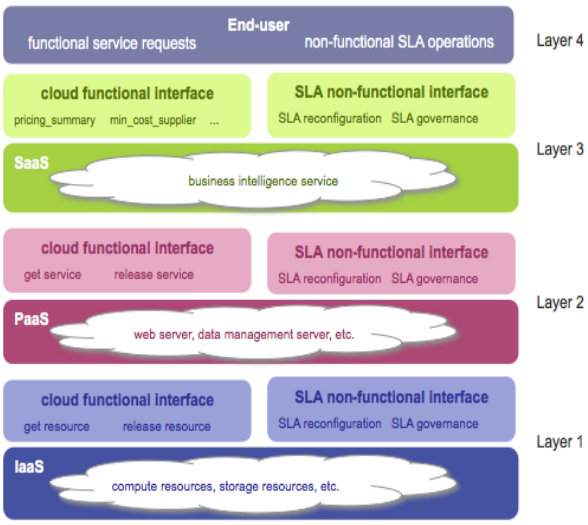


FIGURE 2 – Modèle SLAaaS

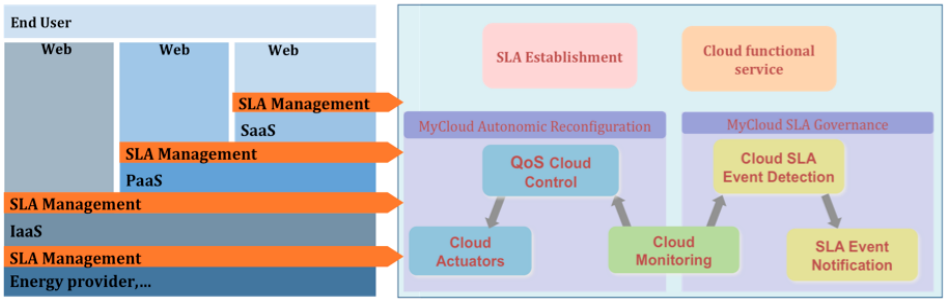


FIGURE 3 – Architecture générale

L'architecture générale est décrite dans la Figure 3 et suit le modèle SLAaaS. Elle repose, tout d'abord, sur la définition et l'association d'un SLA à un service de Cloud (*SLA Establishment*). Le langage pour exprimer un SLA sera présenté dans la Section 4. L'architecture propose, d'une part côté fournisseur de service, des stratégies de (re)configuration autonome du Cloud pour la gestion du SLA et, d'autre part à l'attention du client de service, un service de gouvernance du SLA du Cloud.

La (re)configuration autonome du Cloud a pour objectif de contrôler la QoS du Cloud pour que celle-ci respecte le SLA (*QoS Cloud Control*). Selon le SLA établi, différents critères de QoS peuvent guider ce contrôle, tels que des contraintes de performance, de disponibilité ou d'énergie consommée par le Cloud. Ce contrôle repose, tout d'abord, sur une observation de l'état du Cloud et de la QoS rendu (*Cloud Monitoring*) permettant de guider la (re)configuration du Cloud. Cette dernière fait appel à différentes possibilités comme l'élasticité du Cloud pour permettre un (re)dimensionnement puis un (re)déploiement dynamique de service, ou encore à des stratégies de contrôle basées sur les SLA tels que le contrôle d'accès ou l'ordonnancement des requêtes (*Cloud Actuators*). Les modèles et algorithmes proposés pour la (re)configuration doivent garantir l'objectif premier du fournisseur de Cloud à savoir maximiser la vente de services en minimisant les frais d'infrastructure tout en évitant les pénalités SLA.

L'architecture propose également une gouvernance de SLA de Cloud. Celle-ci permet, par exemple, de détecter l'état du Cloud (comme son empreinte énergétique courante) ou toute violation du SLA établi (*Cloud SLA Event Detection*), et si une telle violation survient, de notifier les différentes parties impli-

quées (fournisseur de Cloud, utilisateur) dans le SLA (*SLA Event Notification*). La détection de violation de SLA repose également sur l'observation de la QoS rendu (*Cloud Monitoring*).

4. Langage spécifique CSLA

Un langage SLA est un langage qui permet de spécifier le contrat entre client et fournisseur de service portant sur le niveau de QoS que doit fournir le service. Il n'existe pas de standard reconnus par la communauté Cloud en ce qui concerne la description et l'établissement de propriétés de QoS. Certaines initiatives, comme SLA@SOI [21], proposent des solutions pour la définition, la négociation, l'établissement et le suivi du SLA. Cependant, les solutions proposées ne prennent pas en compte l'instabilité de la QoS dans un environnement hautement dynamique comme le Cloud dans lequel les ressources et le réseau fluctuent énormément. De plus, les variations de charge cliente importantes sont parfois difficile à maitriser pour le fournisseur et peuvent engendrer des violations SLA.
Nous proposons Cloud Service Level Agreement (CSLA) [15], un langage pour améliorer la gestion de SLA dans le Cloud, en particulier la gestion des violations. CSLA est inspiré de SLA@SOI [21], WSLA [17] et WS-Agreement [3].
CSLA propose de nouvelles propriétés directement au niveau langage – "Fuzziness", "Confidence" et "Penalty" – pour adresser l'instabilité du Cloud. "Fuzziness" définit une marge acceptable autour de la valeur de seuil d'un paramètre QoS, alors que la propriété "Confidence" définit un pourcentage de la conformité des clauses SLOs (*Service Level Objective*). En outre, CSLA comprend un modèle de pénalité qui permet des sanctions appliquées en cas de violation en fonction de la durée. La conséquence directe d'un tel langage est qu'il permet au fournisseur de service d'affiner ses stratégies de (re)configuration (par ex, ordonnancement, contrôle d'admission, allocation) pour arbitrer subtilement la gestion de ressources dans un contexte très dynamique. Par exemple, un service d'ordonnancement peut jouer sur une marge d'erreur grâce à la propriété "Fuzziness" pour ordonnancer différemment les tâches à exécuter et peut-être éviter une allocation de ressource inutile (cf. [15]).
Le méta-modèle de CSLA, décrit Figure 4, représente les concepts les plus importants et leurs relations. Un contrat est spécifié sous la forme d'une instance de la classe CloudSLA. Cette instance est composée des parties et des obligations. Les parties représentent le consommateur et le fournisseur de services. La section "Obligations" est basée sur les garanties SLOs. Pour chaque SLO, nous définissons les exigences, la confiance et la pénalité associée. Une garantie est décrite par une expression simple ou une expression composée. Une expression simple est caractérisée par une métrique de QoS, un seuil, un comparateur et une marge d'erreur. Une expression composite est constituée d'autres expressions. La combinaison de garanties est basée sur les opérateurs ensemblistes définis dans la classe dédiée.
Pour la version 1.0 du langage, le format de présentation est le XML. Cependant, CSLA peut être représenté par n'importe quel langage dédié pour n'importe quel service Cloud (SaaS, PaaS ou IaaS).
Plusieurs exemples dans la section suivante illustreront l'utilisation de CSLA.

5. Etudes de cas

Cette section illustre l'application du modèle SLAaaS et du langage CSLA dans deux études de cas. Elles vont mettre en oeuvre des stratégies de contrôle du Cloud issu du modèle SLAaaS (fonction *QoS Cloud Control*) pour respecter le SLA. La partie gestion de gouvernance du modèle SLAaaS (fonction *Cloud SLA Event Detection*) ne sera pas présentée dans ce papier, faute de place.

5.1. SLA et auto-élasticité, niveau SaaS

Nous considérons ici un service de Cloud logiciel SaaS. Le service considéré est un service de librairie accessible en ligne par plusieurs clients. La charge du service, en terme de nombre de clients concurrents accédant au service, peut être plus ou moins importante et peut varier au cours du temps. Un contrat SLA est établi entre les clients du service et le fournisseur de ce service. Un exemple de contrat SLA est décrit dans la Figure 5 avec le langage CSLA. Ce contrat spécifie un ensemble d'objectifs de qualité de service à remplir et qui sont un temps de réponse aux requêtes des clients du service n'excédant pas

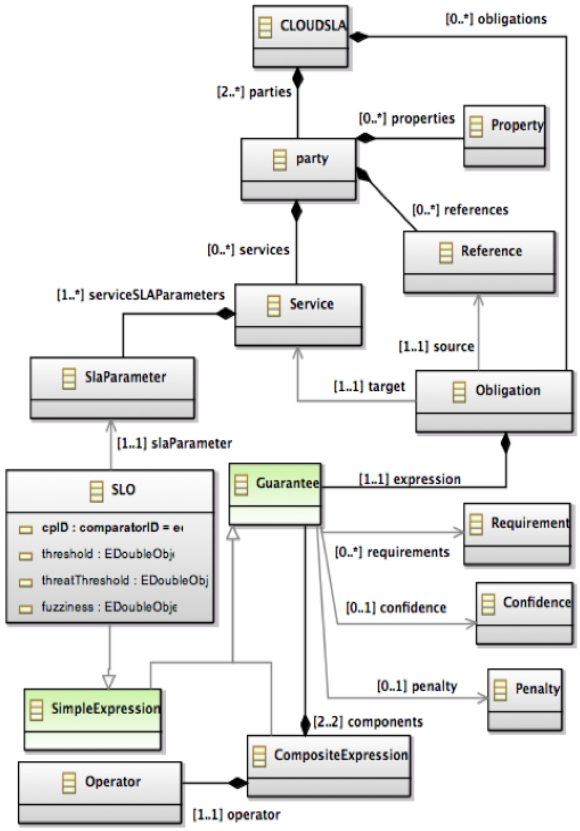


FIGURE 4 – Méta-modèle CSLA

30 secondes, un taux de disponibilité de service d'au moins 90% et un coût financier du service SaaS minimal.

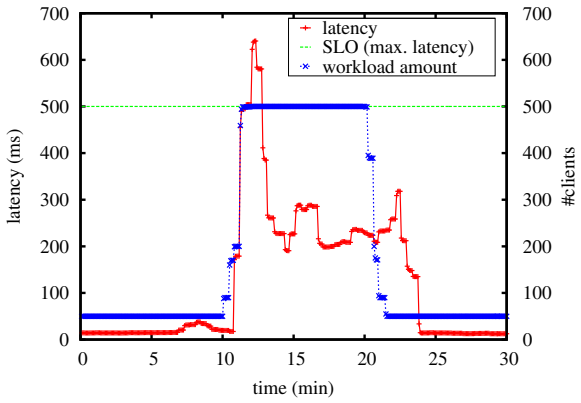
```
<Guarantees>
   <Guarantee guaranteeID="G2" serviceID="S2" >
      <SLO sloID="Rt" Metric="ResponseTime"
           unit="second"
           comparator="le"
           threshold="30" fuzziness="0" />
      <SLO sloID="Av" Metric="Availability"
           unit="% of requests"
           comparator="ge"
           threshold="90" fuzziness="0" />
      <SLO sloID="Fc" Metric="FinancialCost"
           unit="dollars"
           comparator="min" />
      <SLO sloID="PerfSlo" A="Rt" Operator="and"
           B="Av"/>
      <SLO sloID="CompSlo" A="PerfSlo" Operator="and"
           B="Fc" />
   </Guarantee>
</Guarantees>
```

FIGURE 5 – Description du SLA de Cloud SaaS avec CSLA

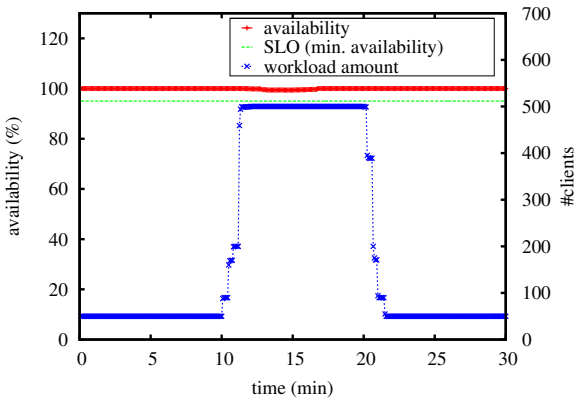
Le service de Cloud considéré est hébergé par des instances de serveurs web et par des instances de serveurs de bases de données. Le nombre de ces instances peut être plus ou moins important, pour répondre à une charge de clients plus ou moins importante. Intuitivement, plus le nombre d'instances est important et meilleures sont la performance et la disponibilité de service rendu, mais plus élevé est le coût du service de Cloud. De plus, une configuration unique ne peut répondre à toutes les variations de charge. Il faut alors que le service SaaS soit auto-élastique de telle sorte à allouer au service le nombre d'instances de serveurs web et de serveurs de bases de données pour garantir le contrat de qualité de service spécifié dans la Figure 5.
Pour cela, le modèle SLAaaS va nous permettre d'administrer le contrat SLA, côté fournisseur, en permettant un redimensionnement sur mesure des instances serveurs. Nous appliquons un algorithme de planification de capacité du service de Cloud [5]. Cet algorithme prend en entrée les objectifs de qualité de service tels que définis dans le SLA et calcule la configuration à appliquer au service de Cloud pour que le SLA soit garanti. La configuration représente, d'une part, le nombre d'instances de serveurs web et le nombre d'instances de serveurs de bases de données à allouer au service de Cloud pour garantir l'objectif de performance et, d'autre part, le degré de parallélisme (*MPL : Multi-Programming Level*) qui permet un contrôle d'admission à chacun des serveurs pour garantir l'objectif de disponibilité de service. La configuration calculée par l'algorithme de planification de capacité est ensuite appliquée au service de Cloud en ligne pour assurer son auto-élasticité face aux variations de charge. L'algorithme de planification de capacité est basé, d'une part, sur une modélisation des serveurs sous-jacents au Cloud par un réseau de files d'attente et, d'autre part, sur une recherche dichotomique efficace dans l'espace des configurations possibles [5].
Nous avons conduit des expériences sur un service de librairie en ligne [1], que nous avons rendu auto-élastique tel que décrit précédemment. Ces expériences ont été conduites sur la plate-forme Grid'5000, dans une grappe de machines 4-core 2-cpu 2.5 GHz Intel Xeon E5420 QC, dotées de 8 Go de RAM et 160 Go de disque dur SATA, connectées via un reseau Ethernet de 1 Gbit. L'environnement logiciel sous-jacent est constitué des éléments suivants : le système d'exploitation Linux v2.6.32, le serveur de Servlets Apache Tomcat v7 et le serveur de bases de données MySQL v.5.5.1.
La Figure 6 présente les résultats de ces expériences, en considérant le SLA décrit dans la Figure 5. Ici, la charge du service varie au cours du temps : elle connait une augmentation de 50 à 500 clients concurrents accédant au service, puis une diminution du nombre de clients durant la fin de l'expérience. Initialement, le service est constitué d'une instance de serveur web et d'une instance de serveur de bases de données et le SLA est valide. Lorsque la charge augmente, le SLA est violé, ce qui provoque une reconfiguration du service de Cloud avec plus d'instances allouées et de nouveaux degrés de parallélisme pour le contrôle d'admission aux serveurs, tel qu'illustré dans les Figures 6(c) et 6(d). Ce qui permet au SLA d'être à nouveau respecté tel qu'illustré dans les Figures 6(a) et 6(b). Lorsque la charge du service baisse, une nouvelle configuration est appliquée au service pour réduire le nombre de ressources qu'il utilise (et donc diminuer son coût) tout en respectant le SLA.

5.2. SLA et verrouillage d'exclusion mutuelle, niveau PaaS

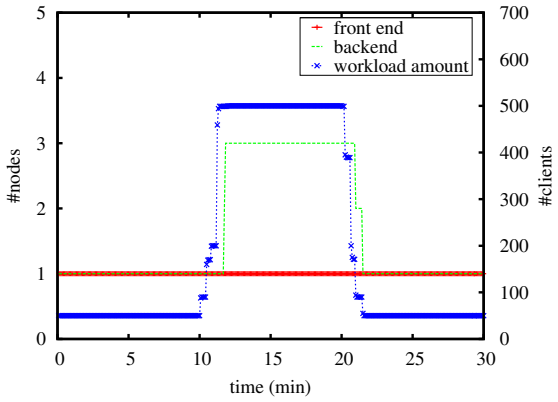
Nous avons réalisé un service de verrouillage d'exclusion mutuelle pour assurer un accès exclusif à des ressources partagées sur le Cloud [16]. Dans les Clouds, la synchronisation et le verrouillage de données ont été identifiés comme étant un problème important et actuellement mal résolu [4]. Par exemple, même si le Cloud de Google utilise l'algorithme optimisé de Chubby, les opérations de verrouillage restent particulièrement coûteuses. Ces protocoles sont non seulement confrontés au problème de passage à l'échelle mais ils doivent également prendre en compte les contraintes applicatives en termes de qualité de service. Basé sur le modèle SLAaaS, nous proposons donc un service réparti qui permet d'administrer les SLA des applications en termes de temps de réponses et de priorités.
Nous avons proposé deux nouveaux algorithmes dans lesquels les demandes d'accès à la section critique sont associées à différentes classes de clients. Pour chaque classe, on définit en CSLA soit des niveaux de priorités, soit des temps de réponses souhaités (*deadline*). La Figure 7 représente un exemple de CSLA établi entre une application (l'utilisateur final ou une application de niveau SaaS) et le service de verrouillage de niveau PaaS.
Nos deux algorithmes reposent sur l'algorithme à jeton de Raymond [20] connu pour ses bonnes performances : l'ensemble des nœuds forme un arbre logique dans lequel la racine est le possesseur du



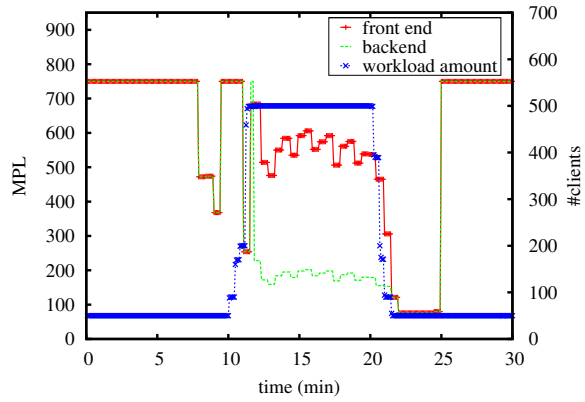
(a) Performance de service



(b) Disponibilité de service



(c) Auto-élasticité du nombre d'instances dans le cloud



(d) Auto-élasticité du degré de parallélisme

FIGURE 6 – Service de Cloud SaaS auto-élastique

```
<Guarantees>
        <Guarantee guaranteeID="G1" serviceID="SL1">
                <SLO sloID="Pri" Metric="Priority"
                        unit="priority level"
                        comparator="max" />
                <SLO sloID="Rt" Metric="ResponseTime"
                        unit="milli second"
                        comparator="le"
                        threshold="2500", fuzziness="0"/>
                <SLO sloID="CompSlo" A="Prio" Operator="and" B = "Rt" />
        </Guarantee>
</Guarantees>
```

FIGURE 7 – Description de SLA du service PaaS de verrouillage avec CSLA

jeton (i.e., le dernier nœud ayant obtenu la section critique). Les requêtes de section critique transitent de nœud en nœud sur l'arbre jusqu'à atteindre la racine. Nous avons avons modifié l'algorithme pour prendre en compte les SLA de la manière suivante : au niveau de chaque nœud, lors de la réception d'une requête un filtrage local est opéré. Dans le cas des priorités, si la requête est moins prioritaire que la requête locale elle n'est pas retransmise ; elle reste dans une file locale et ne sera traitée qu'ultérieurement. Pour les deadlines, les demandes locales pendantes sont réordonnées en fonction de leur deadline.
Les deux algorithmes ont été implémentés et évalués sur une grappe de 20 nœuds avec deux cœurs. Nous avons mesuré différentes métriques dont le nombre de violations de SLA, le nombre moyen de messages générés par section critique et le débit en terme du nombre de section critique par seconde. Le nombre de violations exprime pour l'algorithme à priorité le nombre de requêtes satisfaites alors que des requêtes plus prioritaires sont pendantes, pour les deadlines il mesure le nombre de requêtes dont le deadline n'est pas respecté.
Le Tableau 1 résume les résultats obtenus. L'algorithme à priorité a été comparé celui de Kanrar-Chaki [14]. Cet algorithme à priorité est également basé sur le Raymond, mais souffre d'un problème d'inversion de priorité pour assurer sa vivacité. Pour rapport au Kanrar-Chaki, on observe 10 fois moins de violations et un coût en messages inférieur. Pour les deadlines, nous nous sommes comparés directement à l'algorithme de Raymond en définissant 8 classes d'utilisateurs associés à des deadlines différents. L'algorithme de Raymond génère plus de 50 % de violations tandis alors qu'il n'y a aucune violation avec notre approche. En revanche, le nombre de messages de notre algorithme est deux fois plus important. Cependant, le débit (le nombre de sections critiques par seconde) reste élevé avec notre approche, ce qui démontre un bon recouvrement entre les calculs et les communications.

	Nombre de violations de SLA	Nombre moy. de message par section critique	Débit (sections critiques / sec.)
Algorithme à priorité	5 %	5	82
Kanrar-Chaki	60 %	7	82
Algorithme à deadline	0 %	7	28
Raymond	53 %	3,5	29

TABLE 1 – Performances des algorithmes de verrouillage

6. Travaux apparentés

6.1. SLA et le Cloud computing

Le Cloud computing est typiquement déployé dans un modèle ou une composition des trois modèles (SaaS, PaaS et IaaS) allant du plus partiel au plus complet, et s'adressant à des acteurs différents. Les SLAs sont caractérisés à différents niveaux dans cette hiérarchie pour assurer la QoS attendue. Chaque fournisseur XaaS doit gérer la demande multiple de ses consommateurs et les obligations décrites dans le SLA tout en minimisant les ressources louées à son propre fournisseur. En d'autres termes, il doit trouver un équilibre entre la minimisation des coûts et la satisfaction des exigences SLA.
Peu de solutions de Cloud Computing SLA@SOI [21], Contrail [10] et CompatibleOne [9] prennent en compte le niveau de qualité de service (SLA) fourni par le Cloud. Dans la suite, nous nous intéressons plus particulièrement à la description de SLA et la gestion de ressources dans le Cloud, et présentons les travaux apparentés dans ces domaines.

6.2. Description de SLA

Suite à l'étude des travaux précédents dans la littérature, nous avons identifié principalement : WSLA [17], WS-Agreement [3] et WS-Agreement Negotiation [12]. Ces travaux ont contribué de manière signi-

ficative à la standardisation de SLA. Cependant, aucun ne propose une solution qui supporte l'instabilité de la QoS dans un Cloud hautement dynamique comme le supporte CSLA.
De nombreuses solutions industrielles NIST [13], Cloud Standards Customer Council [11] et Rackspace [19] et les projets académiques Optimis [24], SLA@SOI [21], Contrail [10] et CompatibleOne [9] ont été développés pour répondre aux limites proposées par le Cloud mais restent limitées pour plusieurs raisons. En effet, les solutions proposées sont liées à : i) un seul rôle (consommateur, producteur, broker) et ii) un seul niveau XaaS (SaaS, PaaS, IaaS). Elles n'abordent pas non plus le côté hiérarchique, *cross-layer* des SLA.
Un des buts de notre recherche est de définir un langage qui facilite et améliore la gestion de SLA inter-couches et multi-rôles. Nous étudions en particulier les couches SaaS et IaaS ainsi que les rôles consommateur et producteur.

6.3. Gestion de ressources dans le Cloud

Des nombreux travaux portent sur le *provisioning* de ressources. Nous reportons ici les travaux basés sur : i) des heuristiques d'approvisionnement de ressources pour le dimensionnement de service [7] [18] et ii) des modèles mathématiques non linéaires basés sur la théorie des files d'attente [22] [23]. Tous ces travaux ont contribué de manière significative à la configuration du Cloud. Cependant, aucun ne propose une solution qui prend en compte le dynamisme et l'échelle du Cloud et prenne en compte, de manière cohérente et optimale, de divers aspects, parfois antagonistes, de la qualité de service et du SLA dans le cloud, tels que la consommation énergétique, la performance et la disponibilité du Cloud.
Comme tout système réparti, les applications Cloud sont basées sur une mise en commun des ressources. L'accès à ces ressources est considéré comme une section critique ce qui induit le besoin d'un service d'exclusion mutuelle pour contrôler leur accès. Les algorithmes actuels [6] et [4] ne sont pas adaptés aux applications Cloud, car ils ne prennent pas en compte la topologie physique, la mobilité des applications et services et les contraintes de SLA.
Le deuxième but de notre recherche est la reconfiguration autonome du Cloud (côté fournisseur) pour gérer le niveau de service et garantir le SLA. Ceci se fera via la proposition de cloud élastique s'adaptant dynamiquement pour répondre aux variations de l'usage du Cloud via des algorithmes intégrant les propriétés de CSLA (par ex., *fuzziness*). Des critères de QoS telles que la performance, la disponibilité, la consommation énergétique et les coûts économiques du Cloud seront prises en compte.

7. Conclusion

Le Cloud computing marque une réelle avancée vers l'infrastructure informatique dématérialisée. Le Cloud fournit des ressources informatiques, logicielles ou matérielles, accessible à distance en tant que service. L'adoption de ce modèle soulève un certain nombre de défis, notamment au sujet de la qualité de service (QoS) des services fournis. L'objectif de cet article est de définir le premier modèle de Cloud SLAaaS (SLA aware Service). Ce modèle permet d'intégrer la qualité de service et le contrat SLA (Service Level Agreement) comme éléments à part entière du Cloud. Ainsi, le paradigme général de Cloud Computing est enrichi avec le nouveau modèle SLAaaS. Ce dernier est orthogonal aux modèles IaaS, PaaS et SaaS et peut s'appliquer à n'importe lequel d'entre eux. La fourniture du langage CSLA permet la définition et l'établissement d'un contrat entre les protagonistes du Cloud (utilisateur final jusqu'à IaaS). Il permet de gérer finement la violation grâce des propriétés inhérentes au langage. Enfin, nous proposons deux stratégies pour pouvoir contrôler dynamiquement la taille et la forme du Cloud pour répondre aux contraintes CSLA. La première repose sur une stratégie de redimensionnement basée sur un algorithme de planification de capacité, la seconde repose sur une stratégie de verrouillage d'exclusion mutuelle pour l'accès à des ressources partagées.

8. Remerciements

Ces travaux ont été financés en partie par l'Agence Nationale de la Recherche via le projet MyCloud (*http ://mycloud.inrialpes.fr/*). Les évaluations expérimentales décrites dans cet article ont été en partie conduites sur l'environnement Grid'5000, projet soutenu par l'action de développement INRIA ALADDIN avec un soutien du CNRS, RENATER et différentes Universités (*http ://www.grid5000.fr/*).

Bibliographie

1. TPC-W : a Transactional Web e-Commerce Benchmark. – http ://www.tpc.org/tpcw/.
2. 3teraInc-Cloud Computing without Compromise. – http ://www.3tera.com/. 2012.
3. Andrieux (A.) et al. – *Web services agreement specification (ws-agreement)*. – OGF, 2007.
4. Armbrust (M.), Fox (A.), Griffith (R.), Joseph (A. D.), Katz (R. H.), Konwinski (A.), Lee (G.), Patterson (D. A.), Rabkin (A.) et Zaharia (M.). – *Above the Clouds : A Berkeley View of Cloud Computing*. – Rapport technique, 2009.
5. Arnaud (J.) et Bouchenak (S.). – *Performance and Dependability in Service Computing*, chap. Performance, Availability and Cost of Self-Adaptive Internet Services. – IGI Global, 2011.
6. Birman (K.), Chockler (G.) et van Renesse (R.). – Toward a cloud computing research agenda. *SIGACT News*, vol. 40, n2, juin 2009, pp. 68–80.
7. Bouchenak (S.), De Palma (N.), Hagimont (D.) et Taton (C.). – Autonomic management of clustered applications. *In : Cluster Computing, 2006 IEEE International Conference on*, pp. 1 –11.
8. cloud (Amazon EC2) (A. E. C.). – http ://aws.amazon.com/ec2/. 2012.
9. Compatibleone. – http ://www.compatibleone.org/. 2012.
10. CONTRAIL. – http ://contrail-project.eu/. 2012.
11. et al. (J. M.). – Practical Guide to Cloud Service Level Agreements Version 1.0. *Cloud Standards Customer Council*, 2012.
12. et al. (O. W.). – WS-Agreement Negotiation Version 1.0. *Open Grid Forum*, 2011.
13. Hogan (M.) et al. – Nist cloud computing standards roadmap, version 1.0. 2011.
14. Kanrar (S.) et Chaki (N.). – Fapp : A new fairness algorithm for priority process mutual exclusion in distributed systems. *JNW*, vol. 5, n1, 2010, pp. 11–18.
15. Kouki (Y.) et Ledoux (T.). – CSLA : a Language for Improving Cloud SLA Management. *In : Proceedings of the International Conference on Cloud Computing and Services Science*. – Porto, Portugal, avril 2012.
16. Lejeune (J.), Arantes (L.), Sopena (J.) et Sens (P.). – Service Level Agreement for Distributed Mutual Exclusion in Cloud Computing. *In : 12th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID'12)*. pp. 180–187. – IEEE Computer Society Press.
17. Ludwig (H.), Keller (A.), Dan (A.), King (R. P.) et Franck (R.). – Web service level agreement (wsla) language specification. 2003.
18. Menascé (D. A.), Barbará (D.) et Dodge (R.). – Preserving qos of e-commerce sites through self-tuning : a performance model approach. *In : Proceedings of the 3rd ACM conference on Electronic Commerce*. pp. 224–234. – New York, NY, USA, 2001.
19. Rackspace. – http ://www.rackspace.com/cloud/. 2012.
20. Raymond (K.). – A tree-based algorithm for distributed mutual exclusion. *ACM Trans. Comput. Syst.*, vol. 7, n1, 1989, pp. 61–77.
21. SLA@SOI. – http ://sla-at-soi.eu/. 2012.
22. Tipper (D.) et Sundareshan (M. K.). – Numerical methods for modeling computer networks under nonstationary conditions. *IEEE J.Sel. A. Commun.*, vol. 8, n9, décembre 1990, pp. 1682–1695.
23. Villela (D.), Pradhan (P.) et Rubenstein (D.). – Provisioning servers in the application tier for e-commerce systems. *ACM Trans. Internet Technol.*, vol. 7, n1, février 2007.
24. Ziegler (W.) et Jiang (M.). – OPTIMIS SLA Framework and Term Languages for SLAs in Cloud Environment. 2011.