MÉMOIRES

ETUDE ET IMPLÉMENTATION D'UNE CACHE L2 POUR MOBICENTS JSLEE

Contexte : Aujourd'hui la plupart des serveurs d'application JEE utilise des niveaux de cache L1-L2 pour permettre un déploiement en cluster plus aisé et ainsi supporter une montée en charge importante. Différents frameworks peuvent être utilisés à cette fin, citons par exemple JBoss cache, INFINISPAN et Terracotta.

Mobicents JSLEE est quant à lui un serveur d'application JAVA open source orienté message et temps réel implémenté par Red Hat dans le cadre de la *Communication Media Platform*. Le but de cette plateforme est de pouvoir créer, déployer, et gérer des services et applications qui intègrent la voix, la vidéo et les données sur les réseaux IP et de communications.

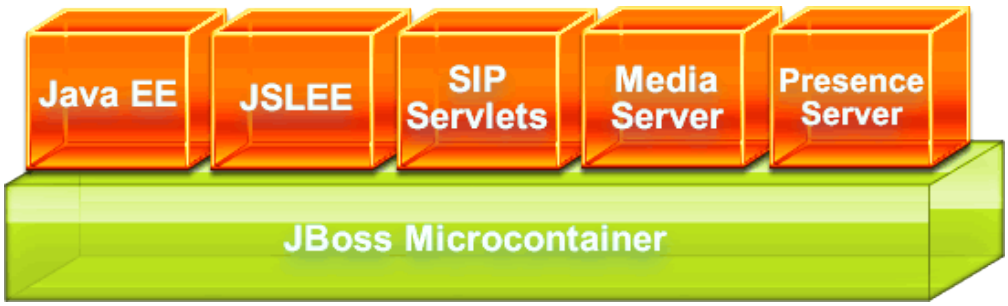

Figure 1 The Mobicents communication platform [1]

La dernière version de Mobicents propose une clusterisation du serveur mais pas encore un niveau de cache pour les accès en base de données, ce qui crée un important goulot d'étranglement lors de la montée en charge.

Contribution : Le but de ce mémoire est d'étudier (1) le container JSLEE de Red Hat et plus particulièrement l'accès aux bases de données à travers les profiles, (2) les systèmes de caches distribués, (3) de proposer une architecture de cache L2 et enfin (4) d'implémenter un prototype.

Organisation: ce mémoire est organisé par l'ULB en collaboration avec Euranova R&D. L'étudiant sera accompagné par l'équipe d'Euranova R&D mais également par les contacts auprès de l'équipe d'ingénierie de Mobicents.

ETUDE ET COMPARAISON DES BASES DE DONNEES ELASTIQUES SUR CLOUD : UN MYTHE ?

Contexte : Aujourd'hui, le cloud computing est certainement un des sujets les plus à la mode. En effet, il promet une évolution élastique de nos serveurs d'applications juste par quelques simples clics sur l'interface de management en y ajoutant dynamiquement de nouvelles machines virtuelles. Mais le cloud est-il vraiment élastique pour tout type d'applications, y compris celles qui possèdent d'importantes quantités de données ?

Amazon propose différentes formules de stockage de données sur EC2 : les Elastic Blocs Storage (EBS) et les fichiers S3. Ces solutions n'offrent pas une grande flexibilité : EBS requiert des backups dont l'interruption de service n'est pas contrôlée (verrous pendant les fenêtres de backup), de plus les EBS ne peuvent être montés que pour un seul nœud, ce qui expose un *single point of faillure*. S3 sont des systèmes de fichiers de stockage qui offrent le stockage de données via des Webservices, ce qui implique un temps de latence important surtout lorsque les écritures sont fréquentes. Depuis quelques mois d'autres vendeurs arrivent sur le cloud d'Amazon, mais sont-ils plus élastiques et sur quelles architectures reposent-ils ?

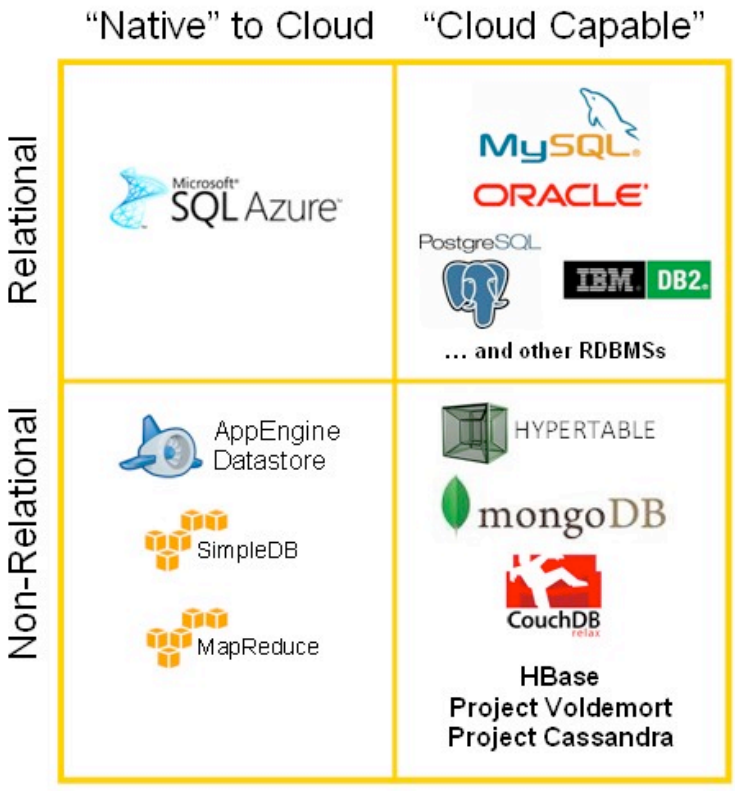

Figure 2 The databases on the cloud [2]

Que faire lorsque l'ont veut déployer différents serveurs d'applications sur plusieurs nœuds EC2 partageant la même base de données ? Quels types d'architecture devrons-nous mettre en œuvre ?

Contribution : L'objectif du mémoire est (1) d'étudier et de décrire les différentes solutions de stockage offertes par Amazon EC2, Oracle et Mysql d'un point de vue architecturale, (2) de déterminer leurs nature élastiques, (3) de les replacer dans un contexte d'un cluster de serveurs d'application, et enfin (4) de proposer une architecture d'exposition de données sur un cluster de nœuds virtuels.

Organisation: ce mémoire est organisé par l'ULB en collaboration avec Euranova R&D. L'étudiant sera accompagné par l'équipe d'Euranova R&D.

MEDIA PLATFORM ON EUCALYPTUS CLOUD

Contexte : La plupart des vendeurs de serveurs d'applications propose aujourd'hui une image Amazon afin d'utiliser leurs produits sur un cloud. D'autres serveurs d'applications open source, pour la plupart, n'ont pas encore été portés sur ce type d'infrastructure. C'est le cas de Mobicents JSLEE, un serveur d'application JAVA open source orienté message et temps réel implémenté par Red Hat dans le cadre de sa *Communication Media Platform*. Le but de cette plateforme est de pouvoir créer, déployer, et gérer des services et applications qui intègrent la voix, vidéo et données sur les réseaux IP et de communications.

Eucalyptus est l'une des implémentations de cloud open source les plus connues. Eucalyptus compte déjà comme utilisateurs, la NASA, Lilly Pharma et il est à la base de l'Entreprise cloud fourni par la distribution Ubuntu. Le caractère open source du framework nous permet, non seulement d'étudier l'architecture cloud, mais aussi d'expérimenter la migration sur le cloud de certains serveurs d'applications depuis l'intérieur.

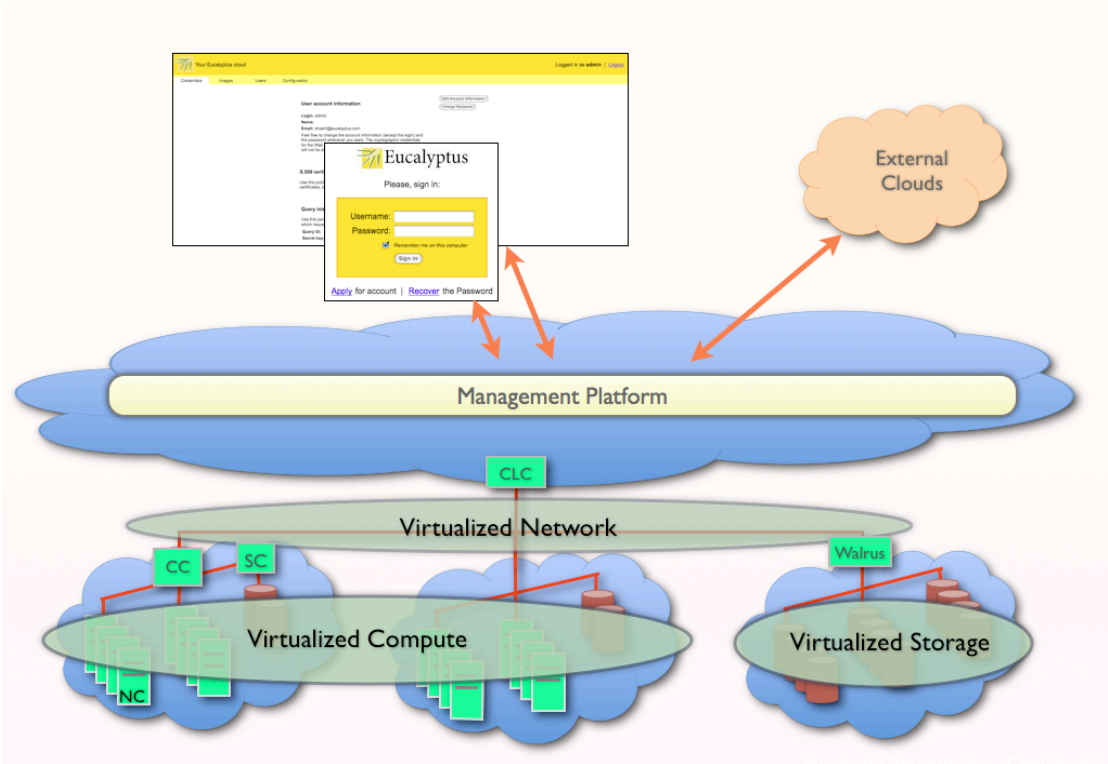

Figure 3 The Eucalyptus cloud and cluster architecture.

Contribution : L'objectif du mémoire est (1) d'étudier et de décrire en détail l'architecture d'Eucalyptus basée sur son grid, (2) de proposer l'architecture nécessaire à la migration de la média platform JSLEE sur Eucalyptus, en tenant comptes des aspects de clusterisation et enfin (3) d'implémenter un prototype d'un serveur JSLEE sur Eucalyptus.

Organisation: ce mémoire est organisé par l'ULB en collaboration avec Euranova R&D. L'étudiant sera accompagné par l'équipe d'Euranova R&D.

ETUDE ET IMPLEMENTATION D'UN ORDONNANCEUR DE JOBS SUR GRID

Contexte : Les grilles de calculs (grids) sont aujourd'hui utilisées dans différents domaines (bancaire, pharmaceutique, biologie moléculaire, physique appliquée, etc.). Pour suivre cette tendance, Red Hat propose un framework MRG (Messaging, Real-Time, Grid), dont la partie grid est implémentée par le projet Condor, un projet open source développé par l'université du Wisconsin [4].

Dans le cadre d'implémentations de projets spécifiques nous avons besoins de localiser certaines données sur des nœuds particuliers du grid. C'est typiquement le cas lorsque de grandes quantités de données doivent être utilisées par les jobs. Les données sont alors distribuées sur des nœuds particuliers de la grille. Dès lors nous devons étendre la politique de distribution de jobs en tenant compte de la localisation de ces données, ce que ne fait pas l'ordonnanceur de Condor.

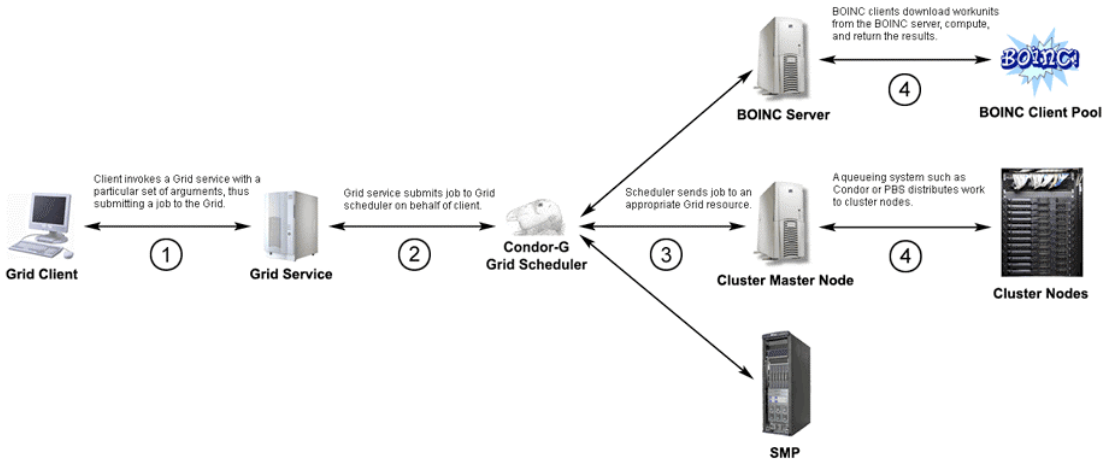

Figure 4 BOINC Project: utilisation de Condor [5].

Contribution : L'objectif du mémoire est (1) d'étudier et de décrire l'architecture du projet Condor, (2) d'étudier les politiques de distribution de jobs et l''implementation de l'ordonnanceur, (3) étendre l'algorithme de distribution en tenant compte de la localisation des données et enfin (4) de proposer un prototype et de le déployer sur une grille condor.

Organisation: ce mémoire est organisé par l'ULB en collaboration avec Euranova R&D. L'étudiant sera accompagné par l'équipe d'Euranova R&D.

[1] http://code.google.com/p/mobicents/

[2] http://www.cloudbzz.com/wp-content/uploads/2009/07/ScreenHunter_0194.jpg

[3] The Eucalyptus web site, http://open.eucalyptus.com

[4] University of Wisconsin, http://www.wisc.edu/

[5] Description du projet BOINC,
http://ditwww.epfl.ch/SIC/SA/SPIP/Publications/spip.php?article928

STAGES:

ETUDE DES « CONTINOUS QUERIES » : FROM ACADEMIC TO BUSINESS

Les architectures de types EDA (Event-Driven Architecture) tentent de fournir aux entreprises une réaction en temps réel de l'infrastructure informatique aux contextes de l'entreprise. Pour cela, ce type d'architecture nécessite la mise en place de moteurs d'inférence capablent de détecter des motifs d'événements. De plus, le support de requêtes temporelles permet une corrélation d'événements en fonction du temps. Plusieurs vendeurs comme Tibco (Tibco Business event et Tibco Event-Driven SOA) ou Oracle (Aqualogic, Oracle CEP) se sont déjà positionné sur ce type de marché. Cependant, un problème demeure : être capable de pouvoir interroger, à chaque nouvel événement une base de données historiques afin d'en inférer une reconnaissance de motif. Cette problématique est plus connue sous le nom de « continous queries » . Pourtant un certain nombre de recherches universitaires aborde ce problème et y apporte des solutions.

Le but de ce stage est de déterminer la distance qui sépare les recherches scientifiques dans ce domaine des implémentations fournies aujourd'hui par les acteurs du marché (Tibco, Orcale, IBM). Le stagiaire devra dresser un état de l'art du *continuous querry* à travers les recherches académiques menées dans ce domaine. Ensuite il devra analyser trois vendeurs (Orcacle/Bea, Tibco, Coral8), en se concentrant sur le *continous query* et plus particulièrement au sein des moteurs d'inférence, afin de déterminer si les implémentations actuelles peuvent être améliorées.

BENCHMARK DES SYSTEMES DE CACHES L2 DISTRIBUES

Avec l'avènement des serveurs d'application JEE, il y a quelques années, la résistance à la montée en charge est devenue un élément critique du moment. La solution mise en avant par les vendeurs est la mise en réseau (cluster) de ces serveurs. Ce qui pose un important problème pour les applications dites Statefull, c'est-à-dire qui maintiennent un état, comme une session http par exemple, mais aussi pour les applications accédant à une base de données. C'est pourquoi les systèmes de cache de nivaux 2 sont apparus.

Le but de ce stage est d'étudier et de comparer deux Frameworks de caches L2 distribués: INFINISPAN de Red Hat et Terracotta. Le stagiaire définira un banc d'essais avec une base de données centralisée contenant d'une part un schéma simple et quelques millions de tuples et d'autre part un cluster d'application sur différentes machines devant accéder en lecture et écriture à ces données.

L'objectif du stage est de décrire les architectures et les mécanismes de ces caches distribués mais aussi d'établir un benchmark de comparaison.