Architectures du cloud

Jinesh Varia
Evangéliste technologique
Amazon Web Services
(jvaria@amazon.com)

Introduction

Ce document illustre le style de création des applications à l'aide des services disponibles dans le cloud Internet.

Les architectures du cloud sont des modèles d'applications logicielles qui utilisent des services à la demande accessibles sur Internet. Les applications élaborées sur des architectures du cloud sont telles que l'infrastructure de calcul sous-jacente est utilisée uniquement lorsqu'elle est nécessaire (par exemple, pour traiter une requête utilisateur), exploite les ressources requises à la demande (comme les serveurs de calcul ou le stockage), effectue une tâche spécifique, puis abandonne les ressources inutiles et souvent s'élimine d'elle-même une fois la tâche effectuée. En cours de fonctionnement, l'application s'adapte de manière élastique en fonction des besoins en ressources.

Ce document est divisé en deux sections. Dans la première section, nous donnons un exemple d'application actuellement en production et utilisant l'infrastructure à la demande fournie par Amazon Web Services. Cette application permet à un développeur de mettre des modèles en correspondance sur des millions de documents web. L'application active des centaines de serveurs virtuels à la demande, exécute un calcul parallèle sur ces serveurs en utilisant une infrastructure de traitement distribuée en Open source appelée *Hadoop*, puis ferme tous les serveurs virtuels en libérant toutes les ressources sur le cloud. Et tout cela s'effectue sans gros effort de programmation, à un coût très faible pour le mandataire.

Dans la seconde section, nous détaillons quelques bonnes pratiques d'utilisation de chaque service AWS, Amazon S3, Amazon SQS, Amazon SimpleDB et Amazon EC2, pour créer une application évolutive de capacité industrielle.

Mots clés

Amazon Web Services, Amazon S3, Amazon EC2, Amazon SimpleDB, Amazon SQS, Hadoop, MapReduce, Cloud Computing

Pourquoi les architectures du cloud ?

Les architectures du cloud gèrent les difficultés clés inhérentes au traitement de données à grande échelle. Tout d'abord, avec le traitement traditionnel des données, il est difficile d'obtenir le nombre de machines nécessaire à une application. Il est par ailleurs tout aussi complexe d'obtenir une machine en cas de besoin. Ensuite, il est compliqué de distribuer et de coordonner une tâche à grande échelle sur différentes machines, d'exécuter des processus sur ces machines et d'affecter une autre machine à cette tâche en cas de défaillance d'une machine. Il est également ardu d'augmenter et de diminuer automatiquement la capacité en fonction des charges de travail dynamiques. Enfin, il est malaisé de se débarrasser de toutes ces machines une fois le travail terminé. Les architectures du cloud résolvent ce type de difficultés.

Les applications élaborées sur des architectures du cloud s'exécutent dans le cloud, où l'emplacement physique de l'infrastructure est déterminé par le fournisseur. Ils s'appuient sur de simples API de services accessibles via Internet qui évoluent à la demande, qui offrent des capacités de type industriel, dont la fiabilité et la logique d'évolutivité complexes des services sous-jacents restent implémentées et masquées à l'intérieur du cloud. Le recours aux ressources des architectures du cloud se fait en fonction des besoins, de façon parfois éphémère ou saisonnière, et offre ainsi le taux d'utilisation le plus élevé et un rendement optimal.

Avantages commerciaux des architectures du cloud

Les avantages commerciaux de l'élaboration d'applications sur les architectures du cloud sont clairs. En voici quelques-uns :

1. *Absence quasi totale d'investissements initiaux en infrastructure* : si vous devez créer un système à grande échelle, vous risquez de dépenser une fortune en immobilier, en matériel (racks, machines, routeurs, sauvegarde et alimentation électrique), en gestion du matériel (gestion de l'alimentation, climatisation) et en personnel d'exploitation. A cause de ces coûts initiaux, plusieurs séries d'approbations de la direction s'avèrent généralement nécessaires avant que le projet puisse même débuter. Désormais, grâce à l'informatique à l'usage, vous ne subissez aucun coût fixe ni frais de démarrage.
2. *Infrastructure juste-à-temps* : auparavant, si vous deveniez célèbre et que vos systèmes ou votre infrastructure n'évoluaient pas au même rythme que vous, vous deveniez victime de votre succès. Inversement, si vous aviez lourdement investi sans atteindre la célébrité, vous deveniez victime de votre échec. En déployant les applications dans le cloud avec un logiciel dynamique de gestion de la capacité, les architectes n'ont pas à se préoccuper d'acquérir, au préalable, la capacité nécessaire aux systèmes à grande échelle. Les solutions présentent un faible risque car vous n'évoluez qu'en fonction de votre *croissance*. Les architectures du cloud peuvent se débarrasser de l'infrastructure aussi vite que vous l'avez obtenue au tout début (en quelques minutes).
3. *Utilisation plus efficace des ressources :* les administrateurs du système s'inquiètent généralement de l'achat du matériel (lorsqu'ils manquent de capacité) et de l'optimisation de l'utilisation de l'infrastructure (lorsque leur capacité est excessive et inactive). Avec les architectures du cloud, ils peuvent gérer les ressources plus efficacement et judicieusement en faisant en sorte que les applications demandent et abandonnent les ressources en fonction de leurs besoins (à la demande).
4. *Tarification à l'usage* : la tarification à l'usage permet de facturer au client uniquement l'infrastructure utilisée. Le client n'est pas responsable de l'intégralité de l'infrastructure éventuellement mise en place. C'est là que réside la subtile différence entre les applications de bureau et les applications web. Une application de bureau ou une application client-serveur traditionnelle s'exécute sur l'infrastructure (PC ou serveur) du client, alors que, dans une application à architecture du cloud, le client utilise une infrastructure tierce et ne doit payer que la partie de l'infrastructure qu'il a utilisée.
5. *Possibilité de diminuer le temps de traitement* : la parallélisation est l'une formidables façons d'accélérer le traitement. Si une tâche requérant énormément de calculs ou de données qui peut être exécutée en parallèle nécessite 500 heures de traitement sur une machine, avec les architectures du cloud, il est possible de générer et de lancer 500 instances et de traiter la même tâche en 1 heure. Le fait de disposer d'une infrastructure élastique offre à l'application la possibilité d'exploiter la parallélisation de manière rentable, en diminuant le temps total de traitement.

Exemples d'architectures du cloud

Il existe de nombreux exemples d'applications qui peuvent utiliser la puissance des architectures du cloud. Cela peut aller des systèmes de traitement back office en bloc aux applications web. En voici quelques-uns :

- Pipelines de traitement
  - Pipelines de traitement de documents – convertissent des centaines de milliers de documents de Microsoft Word en PDF, des millions de pages/images OCR en texte brut pouvant faire l'objet de recherches
  - Pipelines de traitement d'images – créent des vignettes ou des variantes à basse résolution d'une image, redimensionnent des millions d'images
  - Pipelines de transcodage de vidéos – transcodent les films AVI en MPEG
  - Indexation – crée un index des données web
  - Exploitation des données – effectue des recherches dans des millions d'enregistrements
- Systèmes de traitement par lots
  - Applications de back office (dans les secteurs de la finance, des assurances ou de la vente au détail)
  - Analyse des journaux – analyse et génère des rapports quotidiens/hebdomadaires
  - Générations de nuit – génèrent automatiquement, chaque nuit, les référentiels du code source en parallèle

- Tests d'unité et de déploiement automatisés – Testent, déploient et effectuent des tests automatiques des unités (fonction, charge, qualité) sur différentes configurations de déploiement, chaque nuit
- Sites Web
  - Sites web en « veille » la nuit qui évoluent automatiquement au cours de la journée
  - Sites web instantanés – sites web destinés aux conférences ou aux événements (Super Bowl, tournois sportifs)
  - Sites web promotionnels
  - « Sites web saisonniers » - sites web actifs uniquement au cours de la saison fiscale ou des fêtes (période des soldes ou Noël)

Dans ce document, nous détaillerons un exemple d'application, un code appelé « GrepTheWeb ».

Exemple d'architecture du cloud : GrepTheWeb

Le service web Alexa Web Search permet aux développeurs de créer des moteurs de recherche personnalisés face à la quantité massive de données qu'Alexa génère chaque nuit. Une des fonctions de ce service web permet aux utilisateurs de lancer des requêtes dans l'index de recherche Alexa, afin d'obtenir des résultats MSR (Million Search Results), autrement dit par millions. Les développeurs peuvent lancer des requêtes renvoyant jusqu'à 10 millions de résultats.

L'ensemble obtenu, qui représente un petit sous-ensemble de tous les documents disponibles sur le web, peut ensuite être traité à l'aide d'une langue d'expression régulière. Les développeurs peuvent ainsi filtrer les résultats de leur recherche à l'aide de critères qui ne sont *pas* indexés par Alexa (Alexa indexe les documents en fonction de cinquante attributs de document différents), offrant alors la possibilité aux développeurs de réaliser des recherches plus élaborées. Les développeurs peuvent exécuter des expressions régulières sur les documents réels, même lorsqu'il y en a des millions, afin de rechercher des schémas et de récupérer le sous-ensemble de documents qui correspond à cette expression régulière.

Cette application est actuellement en production chez Amazon.com et son nom de code est *GrepTheWeb* car elle peut effectuer un « grep » (ligne de commande Unix connue qui permet de rechercher des schémas) des documents présents sur le web. GrepTheWeb permet aux développeurs de réaliser des recherches assez spécialisées, comme la sélection de documents renfermant une balise HTML ou META particulière, ou la recherche de documents avec des signes de ponctuation spécifiques (« Hé ! », dit-il. « Pourquoi attendre ? ») ou même la recherche d'équations mathématiques (« f(x) = ∑x + W »), de code source, d'adresses e-mail ou d'autres schémas comme « (dés)intégration de la vie ».

Même si cette fonctionnalité est impressionnante, la façon dont elle a été conçue nous stupéfie encore plus. Dans la section suivante, nous allons examiner les différents niveaux de l'architecture de GrepTheWeb.

La figure 1 présente une illustration détaillée de l'architecture. Le résultat obtenu grâce au service Million Search Results, qui correspond à une liste triée de liens et gzippés (compressés à l'aide de l'utilitaire gzip d'Unix) dans un fichier unique, est fourni à GrepTheWeb en entrée. Il prend une expression régulière comme seconde entrée. Il renvoie alors un sous-ensemble filtré de liens de documents triés et gzippés dans un seul fichier. Etant donné que le processus global est asynchrone, les développeurs peuvent obtenir le statut de leurs tâches en appelant GetStatus() afin de voir si l'exécution est terminée.

L'application d'une expression régulière à des millions de documents n'est pas courante. Différents facteurs peuvent se combiner et prolonger la durée de traitement :

- Les expressions régulières peuvent être complexes.
- L'ensemble de données peut être volumineux, voire contenir des centaines de téraoctets.
- Les schémas de demande peuvent être inconnus, comme lorsqu'un certain nombre de personnes peuvent accéder à l'application à n'importe quel moment.

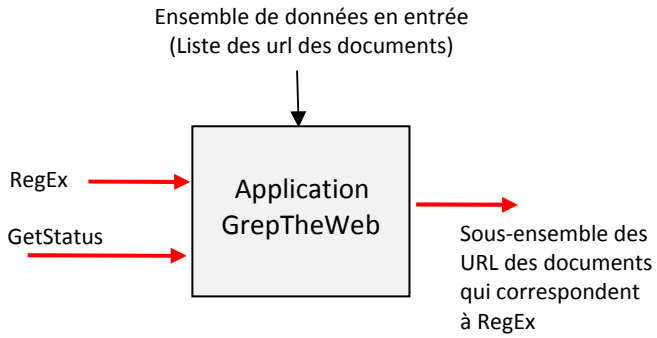


Figure 1 : Architecture GrepTheWeb - Zoom de niveau 1

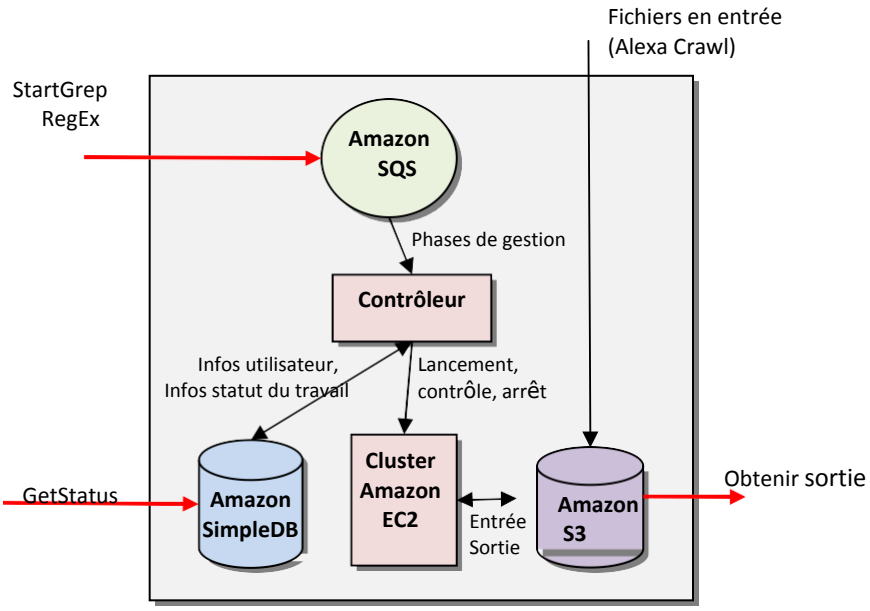


Figure 2 : Architecture GrepTheWeb - Zoom de niveau 2

Par conséquent, GrepTheWeb a notamment été conçu pour évoluer dans toutes les dimensions (des langues de correspondance de schémas plus puissantes, davantage d'utilisateurs simultanés sur les ensembles de données communs, des ensembles de données plus importants, une meilleure qualité des résultats), tout en maintenant au plus bas les coûts du traitement.

L'approche a consisté à créer une application qui non seulement évoluait en fonction de la demande, mais aussi ne nécessitait aucun investissement initial lourd et aucun frais d'entretien de machines inactives (« downbottom »). Pour obtenir une réponse relativement vite, il était important de répartir le travail en plusieurs tâches et de réaliser une opération Distributed Grep qui exécute ces tâches sur plusieurs nœuds en parallèle.

En détaillant encore plus, l'architecture GrepTheWeb ressemble à l'illustration de la figure 2 (ci-dessus). Elle utilise les composants AWS suivants :

- **Amazon S3** pour la récupération des ensembles de données en entrée et pour le stockage de l'ensemble de données en sortie
- **Amazon SQS** pour la mise en tampon durable des demandes agissant comme une « colle » entre les contrôleurs
- **Amazon SimpleDB** pour le stockage du statut intermédiaire, le journal et pour les données utilisateur sur les tâches
- **Amazon EC2** pour l'exécution d'un vaste cluster de traitement Hadoop distribué à la demande
- **Hadoop** pour le traitement distribué, la parallélisation automatique et la programmation du travail

Flux de travail

GrepTheWeb est modulaire. Cette architecture effectue son traitement en quatre phases, comme l'indique la figure 3. La phase de lancement se charge de la validation et du lancement du traitement d'une requête GrepTheWeb, de la génération des instances Amazon EC2, du lancement du cluster Hadoop sur ces instances et du démarrage de tous les processus de travail. La phase de surveillance se charge de la supervision du cluster EC2, mappe, réduit et vérifie le bon déroulement ou l'échec du processus. La phase d'arrêt se charge de la facturation et de l'arrêt de tous les processus Hadoop et des instances Amazon EC2. Enfin, la phase de nettoyage supprime les données temporaires d'Amazon SimpleDB.

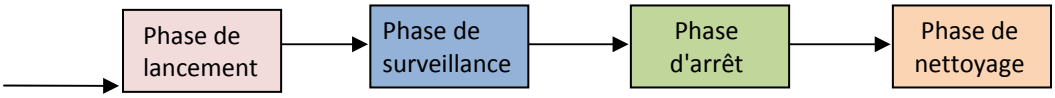


Figure 3 : Phases de l'architecture GrepTheWeb

Flux de travail détaillée pour la figure 4 :

1. Au démarrage de l'application, des files d'attente sont créées, si ce n'est pas encore le cas, et tous les threads de contrôleur sont démarrés. Chaque thread de contrôleur commence à interroger ses files d'attente respectives pour détecter les messages éventuels.
2. Lorsqu'une demande d'utilisateur StartGrep est reçue, un message de lancement est placé en file d'attente dans la file d'attente de lancement.
3. *Phase de lancement* : le thread du contrôleur de lancement récupère le message de lancement et exécute la tâche de lancement, met à jour le statut et les horodatages dans le domaine Amazon SimpleDB, met en file d'attente un nouveau message dans la file de surveillance et supprime le message de la file d'attente de lancement après le traitement.
    a. La tâche de lancement démarre les instances Amazon EC2 en utilisant une AMI JRE pré-installée, déploie les bibliothèques Hadoop requises et démarre un travail Hadoop (exécute les tâches mapper/réduire).
    b. Hadoop exécute les tâches de mappage sur les nœuds subordonnés Amazon EC2 en parallèle. Chaque tâche de mappage prend des fichiers (en multithread en arrière-plan) dans Amazon S3, exécute une expression régulière (attribut du message de file d'attente) sur le fichier issu d'Amazon S3 et inscrit les résultats de la correspondance avec une description contenant jusqu'à 5 correspondances localement, puis la tâche combiner/réduire combine et trie les résultats et consolide la sortie.
    c. Les résultats finaux sont stockés sur Amazon S3, dans le compartiment de sortie.
4. *Phase de surveillance* : le thread du contrôleur de surveillance récupère ce message, valide le statut/l'erreur dans Amazon SimpleDB et exécute la tâche de surveillance, met à jour le statut dans le domaine Amazon SimpleDB, met en file d'attente un nouveau message dans la file d'arrêt et celle de facturation, puis supprime le message de la file d'attente de surveillance après le traitement.
    a. La tâche de surveillance vérifie le statut Hadoop (réussite/échec de JobTracker) à intervalles réguliers, met à jour les éléments SimpleDB avec le statut/l'erreur et le fichier de sortie Amazon S3.
5. *Phase d'arrêt* : le thread du contrôleur d'arrêt récupère ce message dans la file d'attente d'arrêt et exécute la tâche d'arrêt, met à jour le statut et les horodatages dans le domaine Amazon SimpleDB, supprime le message de la file d'attente d'arrêt après le traitement.
    a. La tâche d'arrêt interrompt les processus Hadoop, résilie les instances EC2 après l'obtention des informations de topologie EC2 auprès d'Amazon SimpleDB et élimine l'infrastructure.
    b. La tâche de facturation obtient les informations de topologie EC2, de ressources consommées de SimpleDB, le fichier Amazon S3 et l'entrée de la requête, calcule la facturation et la transmet au service de facturation.
6. *Phase de nettoyage* : archive les données SimpleDB avec les infos utilisateur.
7. Les utilisateurs peuvent exécuter l'attribut GetStatus sur le point de terminaison du service afin d'obtenir le statut du système global (tous les contrôleurs et Hadoop) et de télécharger les résultats filtrés depuis Amazon S3 une fois l'opération terminée.

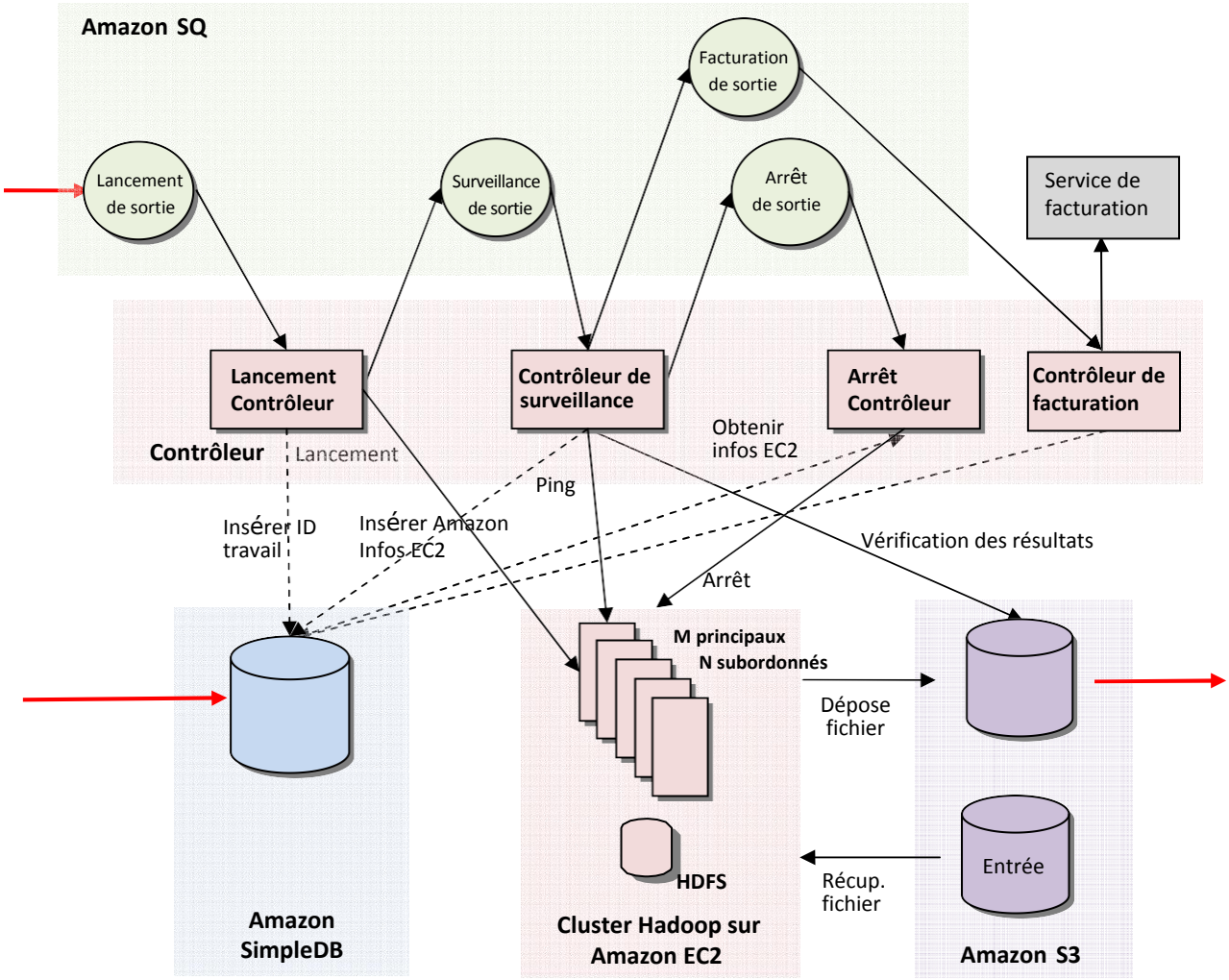


Figure 4 : Architecture GrepTheWeb - Zoom de niveau 3

Utilisation d'Amazon Web Services

Dans les quatre sous-sections suivantes, nous présentons les motifs d'utilisation et expliquons comment GrepTheWeb utilise les services AWS.

Mode d'utilisation d'Amazon S3

Dans GrepTheWeb, Amazon S3 agit comme une entrée mais aussi comme un datastore de résultat. L'entrée dans GrepTheWeb est le web lui-même (au format compressé d'Alexa Web Crawl), stocké sur Amazon S3 sous forme d'objets et mis à jour fréquemment. Etant donné que l'ensemble de données Web Crawl peut être énorme (de l'ordre des téraoctets) et continue de croître, il était nécessaire de disposer d'un stockage distribué, illimité et permanent. Amazon S3 s'est avéré constituer la solution idéale.

Mode d'utilisation d'Amazon SQS

Amazon SQS était utilisé comme mécanisme de transmission de message entre les composants. Il agit comme une « colle » qui relie les différents composants fonctionnels ensemble. Ce mécanisme a non seulement permis de réaliser un couplage faible des différents composants, mais aussi d'élaborer un système global plus résistant aux pannes.

Tampon

Si un composant reçoit et traite les requêtes plus vite que les autres composants (situation de consommateur-producteur déséquilibrée), la mise en tampon permet de rendre le système global plus résistant aux pics de trafic (ou charges). Amazon SQS agit comme un tampon temporaire entre deux composants (contrôleurs) du système GrepTheWeb. Si un message est envoyé directement à un composant, le destinataire aura besoin de le consommer à un débit dicté par l'expéditeur. Par exemple, si le système de facturation était lent ou si le temps de lancement du cluster Hadoop était supérieur à ce qui était prévu, le système global ralentit car il doit attendre. Avec les files d'attente de messages, l'expéditeur et le destinataire sont découplés et le service de file d'attente lisse tout pic du trafic des messages.

Isolement

L'interaction entre deux contrôleurs dans GrepTheWeb s'effectue via les messages présents dans la file d'attente et aucun contrôleur n'appelle directement un autre contrôleur. Toutes les communications et interactions ont lieu par stockage des messages dans la file d'attente (mise en file) et par récupération des messages dans cette file d'attente (retrait de file). Le système dans son entier est alors couplé de manière faible et les interfaces sont simples et propres. Amazon SQS a offert un moyen uniforme de transférer les informations entre les différents composants de l'application. Chaque fonction du contrôleur consiste à récupérer le message, à le traiter (exécuter la fonction) et à le stocker dans une autre file d'attente tant qu'il est totalement isolé des autres.

Asynchronisme

Etant donné qu'il était difficile de connaître le temps d'exécution de chaque phase (par ex., la phase de lancement décide de manière dynamique le nombre d'instances qui doivent être démarrées en fonction de la demande et la durée d'exécution est donc inconnue), Amazon SQS a permis d'élaborer des systèmes asynchrones. Désormais, si la phase de lancement prend plus de temps de traitement ou si la phase de surveillance échoue, les autres composants du système ne sont pas affectés et le système global est plus stable et extrêmement disponible.

Mode d'utilisation d'Amazon SimpleDB

Une base de données dans des architectures du cloud peut notamment être utilisée pour assurer le suivi des statuts. Etant donné que les composants du système sont asynchrones, il est nécessaire de pouvoir obtenir le statut du système à tout moment. Par ailleurs, étant donné que tous les composants sont autonomes et discrets, un datastore permettant les requêtes est nécessaire pour capturer l'état du système.

Puisqu'Amazon SimpleDB est dépourvu de schéma, il n'est pas nécessaire de définir la structure d'un enregistrement au préalable. Chaque contrôleur peut définir sa propre structure et ajouter des données à un élément « travail ». Par exemple : pour un travail donné, « exécuter le regex de l'adresse e-mail sur plus de 10 millions de documents », le contrôleur de lancement ajoute/met à jour l'attribut « statut_lancement » avec l'attribut « heure_début_lancement », pendant que le contrôleur de surveillance ajoute/met à jour les attributs « statut_surveillance » et « statut_hadoop » avec les valeurs de numération (exécution, achèvement, erreur, aucun). Un appel GetStatus() lance une requête auprès d'Amazon SimpleDB et renvoie l'état de chaque contrôleur, ainsi que le statut global du système.

Les services des composants peuvent émettre une requête auprès d'Amazon SimpleDB n'importe quand car les contrôleurs stockent de manière indépendante leurs états : une autre façon sympathique de créer des services asynchrones hautement disponibles. Même si une approche simpliste a été utilisée pour l'implémentation de l'utilisation d'Amazon SimpleDB dans GrepTheWeb, une approche plus sophistiquée, avec une surveillance complète en temps quasi-réel, serait également possible. Par exemple, le stockage du statut de Hadoop JobTracker pour indiquer le nombre de mappages réalisé à un moment donné.

Amazon SimpleDB est également utilisé pour stocker les ID de demande actifs à des fins de constitution d'historique et de contrôle/facturation.

Pour résumer, Amazon SimpleDB sert de base de données de statut pour stocker les différents états des composants et de base de données d'historique/de journal pour formuler des demandes sur les données à haute performance.

Mode d'utilisation d'Amazon EC2

Dans GrepTheWeb, l'ensemble du code du contrôleur s'exécute sur les instances Amazon EC2. Le contrôleur de lancement génère les instances principales et subordonnées en utilisant une Amazon Machine Image (AMI) préconfigurée. Etant donné que l'acquisition et la mise hors service dynamiques se produisent à l'aide d'appels simples au service web, GrepTheWeb sait combien d'instances principales et subordonnées doivent être lancées.

Le contrôleur de lancement tente une approximation valable, en se basant sur la logique de réservation, afin de déterminer le nombre d'instances subordonnées nécessaire pour réaliser un travail particulier. La logique de réservation repose sur la complexité de la requête (nombre de prédicats, etc.) et sur la taille de l'ensemble de données en entrée (nombre de documents à rechercher). Cet élément a également été conservé configurable afin que nous puissions diminuer le délai de traitement en indiquant simplement le nombre d'instances à lancer.

Une fois les instances lancées et le cluster Hadoop démarré sur ces instances, Hadoop affecte une instance principale et des subordonnées, gère la négociation, l'établissement de liaison et la distribution des fichiers (clés SSH, certificats) et exécute le travail grep.

Phases de mappage et réduction Hadoop

Hadoop est une infrastructure de traitement distribuée en Open source, qui permet d'effectuer des calculs sur d'importants ensembles de données en divisant l'ensemble de données en parties gérables. Hadoop les répartit ensuite sur toute une flotte de machines et gère le processus global en lançant les travaux, en traitant le travail où que se trouvent physiquement les données et, à la fin, en cumulant le travail obtenu de manière à générer un résultat final.

Cette opération s'effectue généralement en trois phases. Une phase de *mappage* transforme l'entrée en représentation immédiate des paires de valeurs clés, une phase de *combinaison* (gérée par Hadoop) qui combine et trie les clés et une phase de *réduction* qui recombine la représentation intermédiaire en résultat final. Les développeurs implémentent deux interfaces, Mapper et Reducer, pendant que Hadoop gère tout le traitement distribué (parallélisation automatique, planification du travail, supervision du travail et cumul des résultats).

Dans Hadoop, un processus principal s'exécute sur un nœud pour superviser un pool de processus subordonnés (également appelés employés) s'exécutant sur des nœuds distincts. Hadoop divise l'entrée en blocs. Ces blocs sont affectés à des subordonnés. Chaque subordonné effectue la tâche de mappage (logique spécifiée par l'utilisateur) sur chaque paire trouvée dans le bloc et écrit les résultats localement, puis informe le principal du statut terminé. Hadoop combine tous les résultats et trie les résultats en fonction des clés. Le principal attribue ensuite les clés aux réducteurs. Le réducteur extrait les résultats à l'aide d'un itérateur, exécute la tâche de réduction (logique spécifiée par l'utilisateur) et renvoie le résultat « final » au système de fichiers distribué.

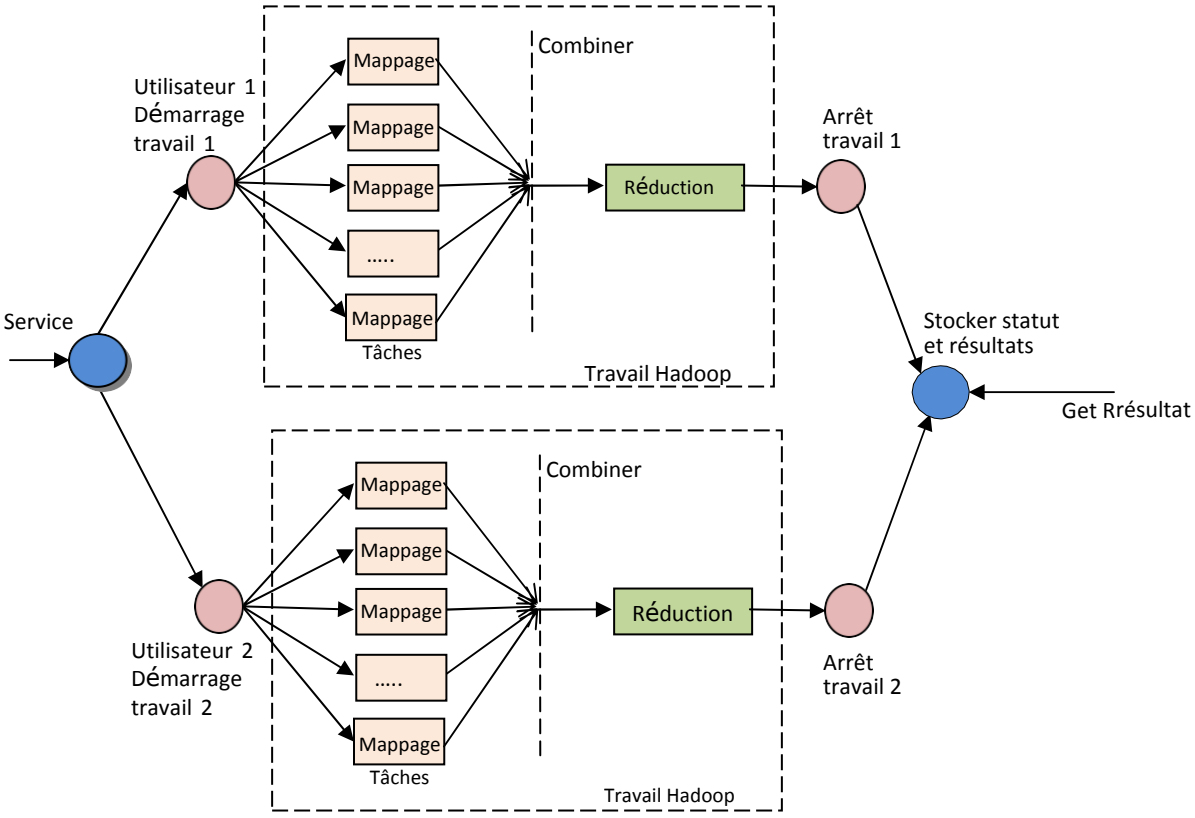


Figure 5: Opération de mappage et réduction (dans GrepTheWeb)

Implémentation Hadoop dans GrepTheWeb

Hadoop est parfaitement adapté à l'application GrepTheWeb. Etant donné que chaque tâche grep peut être exécutée en parallèle indépendamment des autres tâches grep en utilisant l'approche parallèle d'Hadoop, cette application est idéale.

Pour GrepTheWeb, les documents réels (le web) sont parcourus à l'avance et stockés sur Amazon S3. Chaque utilisateur entame un travail grep en appelant la fonction StartGrep au niveau du point de terminaison du service. Une fois la fonction déclenchée, les nœuds principaux et subordonnés (cluster Hadoop) sont démarrés sur les instances Amazon EC2. Hadoop divise l'entrée (le document avec les pointeurs dirigés vers les objets Amazon S3) en plusieurs blocs gérables de 100 lignes chacun et attribue le bloc à un nœud subordonné pour exécuter la tâche de mappage. La tâche de mappage lit ces lignes et est chargée de récupérer les fichiers dans Amazon S3, en exécutant l'expression régulière dessus et en écrivant les résultats localement. En l'absence de correspondance, il n'y a aucun résultat. La tâche de mappage transmet ensuite les résultats à la phase de réduction, qui correspond à une fonction d'identité (passage) afin de cumuler tous les résultats. Le résultat « final » est réinscrit sur Amazon S3.

Exemple
Expression régulière « A(.*)zon »
Format de la ligne dans le jeu de données en entrée [URL] [Titre] [jeu de caractères] [taille] [clé de l'objet S3 du fichier .gz] [décalage] http://www.amazon.com/gp/browse.html?node=3435361 Amazon Web us-ascii 3509 /2008/01/08/51/1/51_1_20080108072442_crawl100.arc.gz
Implémentation de Mapper **1.** Clé = numéro de ligne et valeur = ligne dans le jeu de données en entrée **2.** Créez une URL signée (en utilisant les informations d'identification d'Amazon AWS) à l'aide du contenu de la valeur clé **3.** Lisez (récupérez) l'objet Amazon S3 (fichier) dans un tampon **4.** Exécutez l'expression régulière sur ce tampon **5.** En cas de correspondance, collectez le résultat dans un nouvel ensemble de paires de valeurs clés (clé = ligne, valeur = jusqu'à 5 correspondances)
Implémentation de Reducer - Transmet (fonction d'identité intégrée) et réinscrit les résultats sur S3.

Conseils de conception d'une application sur une architecture du cloud

1. Assurez-vous que votre application est évolutive en concevant chaque **composant** de sorte qu'il soit lui-même évolutif. Si chaque composant implémente une interface de service, **responsable de sa propre évolutivité** dans toutes les dimensions concernées, le système global aura alors une base évolutive.
2. Afin d'optimiser la capacité de gestion et la disponibilité, assurez-vous que vos composants sont **faiblement couplés**. L'essentiel est de concevoir des composants sans créer de dépendances strictes entre ces différents composants. Ainsi, si l'un des composants venait à s'éteindre (dysfonctionner), se mettre en veille (ne pas répondre) ou rester occupé (être lent à répondre) pour quelque raison que ce soit, les autres composants du système sont conçus de telle manière qu'ils peuvent continuer à fonctionner, sans interruption
3. Implémentez la **parallélisation** afin d'optimiser l'utilisation de l'infrastructure et les performances. La distribution des tâches sur plusieurs machines, la répartition sur plusieurs threads de vos demandes et le cumul effectif des résultats obtenus en parallèle sont quelques-unes des techniques qui permettent d'exploiter l'infrastructure.
4. Une fois la fonctionnalité de base conçue, posez la question : « Que se passe-t-il en cas d'échec ? ». Utilisez les techniques et les approches qui garantiront la **résilience** de la solution. En cas de dysfonctionnement d'un composant (ce qui se produit couramment), le système doit automatiquement émettre une alerte, assurer un basculement et se resynchroniser sur le « dernier état connu », comme si aucune interruption n'avait eu lieu.
5. Ne négligez pas le **facteur coût**. La clé de l'élaboration d'une application rentable réside dans l'utilisation de ressources à la demande. Il est inutile de payer pour une infrastructure dont on ne se sert pas.

Chacun de ces points est détaillé dans le contexte de GrepTheWeb.

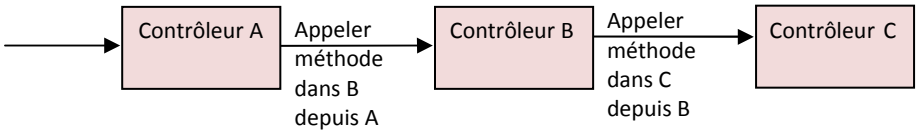
Utilisation d'ingrédients évolutifs

L'application GrepTheWeb utilise des composants hautement évolutifs de l'infrastructure des services AWS qui évoluent et sont facturés à la demande.

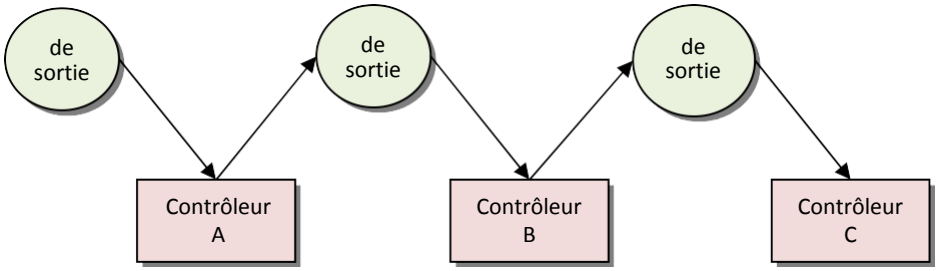
Tous les composants de GrepTheWeb exposent une interface de service qui définit les fonctions et peut être appelée à laide des demandes HTTP et peut générer des réponses XML. Pour des questions de programmation, les petites bibliothèques de client englobent et réduisent le code spécifique du service

Chaque composant est indépendant des autres et s'adapte dans toutes les dimensions. Par exemple, si des milliers de demandes sont envoyées à Amazon SimpleDB, il peut gérer la demande car il est conçu pour gérer des demandes parallèles massives

De même, les infrastructures de traitement distribuées comme Hadoop sont conçues pour évoluer. Hadoop distribue automatiquement les travaux, reprend les travaux en échec et s'exécute sur plusieurs nœuds pour traiter des téraoctets de données



Couplage fort (programmation de procédure)



Couplage faible (phases indépendantes utilisant des files d'attente)

Figure 6 : Couplage faible – Phases indépendantes

Des systèmes faiblement couplés

L'équipe GrepTheWeb a créé un système à couplage faible utilisant les *files d'attente de messagerie*. Si une file d'attente/un tampon est utilisé(e) pour « relier » deux composants l'un à l'autre, le système peut gérer la simultanéité, la haute disponibilité et les pics de charge. Ainsi, le système global offre toujours d'excellentes performances, même si des parties de composants deviennent indisponibles. Si l'un des composants s'éteint ou devient temporairement indisponible, le système met les messages en tampon et les fait traiter lorsque le composant redevient opérationnel

Dans GrepTheWeb, par exemple, si de nombreuses demandes atteignent subitement le serveur (une situation de surcharge provoquée par Internet) ou si le traitement des expressions régulières est plus long que la moyenne (débit de réponse lent d'un composant), les files d'attente d'Amazon SQS mettent les demandes durablement en tampon afin que ces retards n'affectent pas les autres composants.

Comme dans un système à locataires multiples il est important d'obtenir les statuts des messages/demandes, GrepTheWeb gère ce type de système. Pour cela, il stocke et met à jour le statut de chacune de vos demandes dans un datastore distinct, permettant les requêtes. Il y parvient grâce à Amazon SimpleDB. Cette combinaison d'Amazon SQS pour la mise en file d'attente et d'Amazon SimpleDB pour la gestion des états permet de bénéficier d'une excellente résilience avec un couplage faible.

Une parallélisation réfléchie

A « l'ère du téra » et des processeurs multi-cœurs, lors de la programmation, nous devons penser en termes de processus à *threads multiples*.

Chaque fois que c'est possible, dans GrepTheWeb, les processus ont été sécurisés au niveau des threads grâce à une philosophie d'absence de partage et ont été appliqués sur plusieurs threads pour améliorer les performances. Par exemple, les objets sont récupérés depuis Amazon S3 par des threads multiples simultanés car ce type d'accès est plus rapide que la récupération d'objets séquentielle, un par un.

Si les threads multiples sont insuffisants, adoptez les *nœuds* multiples. Jusqu'à présent, le calcul parallèle sur d'importants clusters de machines était non seulement coûteux mais aussi difficile à obtenir. Tout d'abord, il était difficile d'obtenir le financement nécessaire à l'acquisition d'un important cluster de machines qui, une fois acquis, s'avérait délicat à gérer et à entretenir. Ensuite, une fois acquis et géré, ce cluster présentait des problèmes techniques. Il était complexe d'exécuter des tâches distribuées massivement sur les machines, de stocker de volumineux ensembles de données et d'y accéder. La parallélisation n'était pas une opération facile et la planification des travaux était sujette à erreur. Par ailleurs, en cas d'échec des nœuds, il était difficile de les détecter et coûteux de les récupérer. Le suivi des travaux et des statuts était souvent négligé car il devenait rapidement compliqué au fil de l'augmentation du nombre de machines en cluster.

Désormais, le calcul a changé. Avec l'avènement d'Amazon EC2, la mise en service d'un grand nombre d'instances de calcul est facile. Un cluster d'instances de calcul peut être mis en service en quelques minutes, avec seulement quelques appels API, puis être retiré tout aussi facilement. Avec l'arrivée des infrastructures de traitement distribuées comme Hadoop, il n'est pas nécessaire de recourir aux consultants en calcul parallèle de haut calibre pour déployer une application parallèle. Les développeurs sans expérience en calcul parallèle peuvent implémenter quelques interfaces en quelques lignes de code et mettre le travail en parallèle sans se préoccuper de la planification du travail, de la supervision et du cumul des résultats.

Réquisition et abandon à la demande

Dans GrepTheWeb, chaque composant de bloc de création est accessible via Internet à l'aide des services web, hébergé en toute sécurité dans les centres de données d'Amazon et disponible à la demande. L'application peut donc demander davantage de ressources (serveurs, stockage, bases de données, files d'attente) ou les abandonner en fonction des besoins.

GrepTheWeb offre un atout de taille : c'est une application quasiment dépourvue d'infrastructure avant et après son exécution. L'intégralité de l'infrastructure est instanciée dans le cloud, déclenchée par une demande de travail (grep), puis renvoyée dans le cloud, une fois le travail effectué. Par ailleurs, au cours de l'exécution, elle évolue à la demande ; autrement dit, l'application s'adapte de manière élastique en fonction du nombre de messages et de la taille de l'ensemble de données en entrée, de la complexité de l'expression régulière, etc.

Pour GrepTheWeb, une logique de réservation décide du nombre d'instances subordonnées de Hadoop à lancer en se basant sur la complexité du regex et de l'ensemble de données en entrée. Par exemple, si l'expression régulière n'a pas beaucoup de prédicats ou si l'ensemble de données en entrée comporte juste 500 documents, elle ne génère que 2 instances. En revanche, si l'ensemble de données en entrée comporte 10 millions de documents, elle génère jusqu'à 100 instances.

Utilisation de modèles résistants aux redémarrages et reprises des lancements

Respectez la règle empirique : soyez pessimiste lorsque vous utilisez les architectures du cloud ; partez du principe que tout peut dysfonctionner. En d'autres mots, concevez, implémentez et déployez toujours de manière à assurer une récupération automatique en cas de défaillance.

Et surtout, présumez que votre matériel *va* dysfonctionner. Partez du principe que des coupures de courant *vont* se produire. Présumez qu'une catastrophe quelconque *va* frapper votre application. Supposez que vous *allez* être submergé par davantage de demandes par seconde un jour ou l'autre. En étant pessimiste, vous finissez, au cours de la conception, par envisager toutes les stratégies de récupération, ce qui permettra d'optimiser la conception du système global. Par exemple, les stratégies suivantes peuvent aider en cas de souci :

1. Adoptez une stratégie cohérente de sauvegarde et de restauration de vos données.
2. Elaborez des threads de processus qui reprennent au redémarrage.
3. Laissez l'état du système se resynchroniser en rechargeant les messages issus des files d'attente.
4. Conservez les images virtuelles préconfigurées et pré-optimisées à des fins de support (2) et (3) au lancement/démarrage.

Les architectures du cloud adaptées doivent être insensibles aux redémarrages et aux reprises de lancement. Dans GrepTheWeb, en utilisant une combinaison d'Amazon SQS et Amazon SimpleDB, l'architecture globale du contrôleur est plus résistante. Par exemple, si l'instance sur laquelle le thread du contrôleur s'exécutait s'éteint, elle peut être rappelée et reprendre l'état antérieur, comme si rien ne s'était passé. Pour parvenir à cela, il a suffi de créer une IMA (Amazon Machine Image) qui, lorsqu'elle est lancée, retire de la file d'attente tous les messages présents dans la file d'attente d'Amazon SQS et leurs états dans l'élément de domaine Amazon SimpleDB au redémarrage.

Si un nœud (subordonné) de suivi de tâche s'éteint en raison d'un dysfonctionnement matériel, Hadoop reprogramme automatiquement la tâche sur un autre nœud. Cette tolérance au panne permet à Hadoop de s'exécuter sur de grands clusters de serveurs de commodité afin de surmonter les pannes matérielles.

Résultats et coûts

Nous avons effectué plusieurs tests. L'expression régulière d'adresse e-mail a été exécutée sur 10 millions de documents. Alors que 48 instances simultanées ont besoin de 21 minutes pour le traitement, il suffit de 6 minutes de traitement à 92 instances simultanées. Ce délai inclut le délai de lancement et le délai d'arrêt du cluster Hadoop. Le coût total pour 48 instances était d'environ 5 $ et celui de 92 instances était inférieur à 10 $.

Conclusion

Au lieu de créer vos applications sur des applications fixes et rigides, les architectures du cloud vous offrent une nouvelle façon d'élaborer les applications sur des infrastructures à la demande.

GrepTheWeb montre comment ce type d'application peut être créé.

Sans investissement initial, nous avons réussi à exécuter un travail massivement distribué sur de multiples nœuds en parallèle et nous avons pu faire évoluer l'infrastructure par paliers, en fonction de la demande (utilisateurs, taille de l'ensemble de données en entrée). Sans temps mort, l'infrastructure de l'application n'a jamais été sous-exploitée.

Dans la section suivante, nous verrons comment chaque service de l'infrastructure Amazon (Amazon EC2, Amazon S3, Amazon SimpleDB et Amazon SQS) a été utilisé. Vous pourrez également découvrir quelques leçons tirées de nos expériences et quelques bonnes pratiques.

Bonnes pratiques tirées de l'expérience

Dans cette section, nous mettons en avant quelques bonnes pratiques issues des leçons apprises dans le cadre de l'implémentation de GrepTheWeb.

Bonnes pratiques d'Amazon S3

Téléchargement de fichiers volumineux, récupération de petits décalages

Les débits des données de transfert de bout en bout dans Amazon S3 sont optimaux lorsque de gros fichiers sont stockés au lieu de tout petits fichiers (avec des tailles de l'ordre des Ko). Aussi, au lieu de stocker des fichiers individuels sur Amazon S3, plusieurs fichiers ont été regroupés et compressés (gzip) dans un blob, puis stockés sur Amazon S3 sous forme d'objets. Les fichiers individuels ont été récupérés à l'aide d'une demande standard HTTP GET, en fournissant une URL (compartiment et clé), un décalage (plages d'octets) et une taille (longueur en octets). Au final, le coût global du stockage a été réduit grâce à la diminution de la taille globale de l'ensemble de données (en raison de la compression) et du nombre inférieur alors nécessaire de demandes PUT.

Tri des clés puis téléchargement de votre ensemble de données

Les dispositifs de rapprochement d'Amazon S3 améliorent les performances si les clés sont triées avant le téléchargement. En exécutant un petit script, les clés (pointeurs URL) ont été triées, puis téléchargées dans leur ordre de tri dans Amazon S3.

Utilisation de la récupération sur plusieurs threads

Au lieu de récupérer les objets un par un sur Amazon S3, plusieurs threads de récupération simultanés ont été démarrés dans chaque tâche de mappage pour récupérer les objets. Il est toutefois déconseillé de générer des centaines de threads car chaque nœud présente des contraintes de bande passante. Idéalement, les utilisateurs doivent tenter d'augmenter progressivement le nombre de threads parallèles simultanés jusqu'à parvenir au point où l'ajout de threads supplémentaires n'améliorera pas davantage la vitesse.

Utilisation d'un retour en arrière exponentiel et d'une nouvelle tentative

Pour n'importe quelle application, il est raisonnable de retenter chaque demande de service web en échec. Le plus difficile reste de choisir la stratégie à utiliser pour déterminer l'intervalle de la nouvelle tentative. Nous recommandons d'adopter une approche consistant à utiliser le retour en arrière exponentiel binaire tronqué. Dans le cadre de cette approche, l'intervalle exact entre chaque nouvelle tentative est déterminé par une combinaison qui double successivement le nombre de secondes représentant potentiellement le délai maximum et par le choix aléatoire d'une valeur sur cette plage.

Nous vous recommandons de créer le retour en arrière exponentiel, l'intervalle et la logique de nouvelle tentative dans le code gestion des erreurs de votre client. Le retour en arrière exponentiel diminue le nombre de demandes formulées auprès d'Amazon S3 et réduit ainsi le coût global, sans surcharger une quelconque partie du système.

Bonnes pratiques d'Amazon SQS

Conservation des informations de référence dans le message

Amazon SQS est idéal pour les messages brefs, à courte durée de vie, dans les flux de travail et les pipelines de traitement. Pour rester dans les limites de taille du message, il est conseillé de stocker les informations de référence comme une partie du message et de stocker le fichier réel en entrée sur Amazon S3.

Dans GrepTheWeb, le message de la file d'attente de lancement contient l'URL du fichier d'entrée (.dat.gz), qui correspond à un petit sous-ensemble d'un ensemble de résultats (résultats d'une recherche sur des millions qui peut comporter jusqu'à 10 millions de liens). De même, le message de la file d'attente d'arrêt contient l'URL du fichier de sortie (.dat.gz), qui représente un ensemble de résultats filtrés contenant les liens qui correspondent à l'expression régulière.

Les tableaux suivants indiquent le format de message de la file d'attente et leurs statuts.

ActionRequestId	f474b439-ee32-4af0-8e0f-a62d1f7de897
Code	En file d'attente
Message	Votre demande a été mise en
ActionName	file d'attente. StartGrep
RegEx	A(.*)zon
UrlEntrée	http://s3.amazonaws.com/com.alexa.msr.prod/msr_f474b439-ee32-4af0-8e0f-a979907de897.dat.gz?Signature=CvD9iHA%3D&Expires=1204840434&AWSAccessKeyId=DDXCXCCDEEDSDFGSDDX

ActionRequestId	f474b439-ee32-4af0-8e0f-a62d1f7de897
Code	Terminé
Message	Les résultats sont désormais disponibles au téléchargement depuis UrlTéléchargement
NomAction	StartGrep
DateDébut	2008-03-05T12:33:05
UrlTéléchargement	http://s3.amazonaws.com/com.alexa.gtw.prod/gtw_f474b439-ee32-4af0-8e0f-a62de897.dat.gz?Signature=CvD9iIGGjUIlkOlAeHA%3D&Expires=1204840434&AWSAccessKeyId=DDXCXCCDEEDSDFGSDDX

Utilisation de messageries orientées sur les processus et sur les documents

Il existe deux approches en matière de messagerie. Elles s'avèrent efficaces dans notre cas : la messagerie orientée sur les processus et celle orientée sur les documents. La première est souvent définie par le processus ou les actions. L'approche typique consiste à supprimer l'ancien message de la file d'attente « de », puis d'ajouter un nouveau message avec de nouveaux attributs à la file d'attente « à ».

La messagerie orientée sur les documents est générée lorsqu'un message par utilisateur/thread de travail passe par l'intégralité du système, avec différents attributs de message. Elle est souvent implémentée à l'aide de XML/JSON car il s'agit d'un modèle extensible. Dans cette solution, les messages peuvent évoluer, sauf que le destinataire n'a besoin de comprendre que les parties importantes pour lui. Ainsi, un seul message peut transiter dans le système et les différents composants n'ont besoin de comprendre que les parties du messages importantes pour eux.

Pour GrepTheWeb, nous avons décidé d'utiliser l'approche orientée sur les processus.

Avantage de la fonction du délai de visibilité

Amazon SQS possède une fonctionnalité spéciale absente de nombreux autres systèmes de messagerie ; lorsqu'un message est lu depuis la file d'attente, il est visible pour les autres lecteurs de la file d'attente, alors qu'il n'est pas automatiquement supprimé de la file d'attente. Le consommateur doit explicitement supprimer ce message de la file d'attente. Si ce n'est pas le cas, au bout d'un certain temps après la lecture du message, le système considère que le consommateur a échoué et le message réapparaîtra dans la file d'attente pour être de nouveau consommé. Pour cela, le délai de visibilité est configuré lors de la création de la file d'attente. Dans GrepTheWeb, le délai de visibilité est très important car certains processus (comme le contrôleur d'arrêt) peuvent échouer et ne pas répondre (par ex., les instances restent actives). Lorsque le délai de visibilité est défini sur un certain nombre de minutes, un autre thread de contrôleur récupère l'ancien message et reprend la tâche (d'arrêt).

Bonnes pratiques d'Amazon SimpleDB

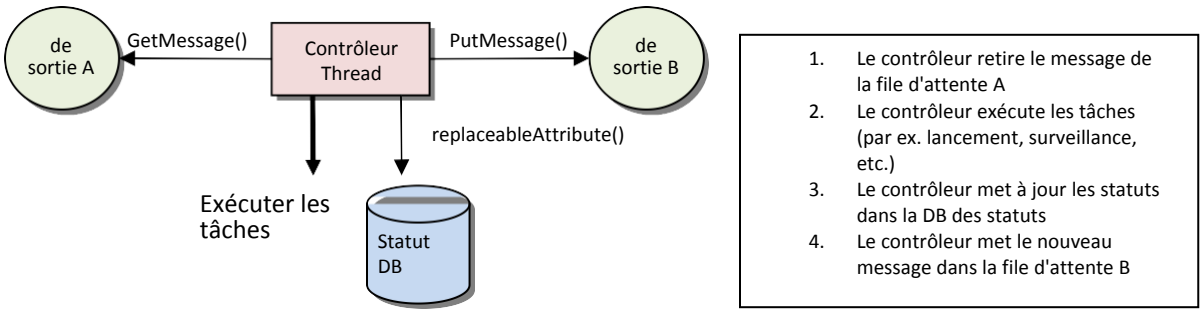
Threads multiples GetAttributes() et PutAttributes()

Dans Amazon SimpleDB, les domaines ont des éléments et les éléments ont des attributs. Une requête formulée sur Amazon SimpleDB renvoie un ensemble d'éléments. Mais, souvent, les valeurs des attributs sont nécessaires pour effectuer une tâche particulière. Dans ce cas, un appel de requête est suivi d'une série d'appels GetAttributes afin d'obtenir les attributs de chaque élément de la liste. Comme vous pouvez le devinez, le temps d'exécution peut être long. Pour remédier à cela, il est fortement recommandé d'appliquer plusieurs threads à vos appels GetAttributs et de les exécuter en parallèle. Les performances globales augmentent considérablement (jusqu'à 50 fois) en cas d'exécution en parallèle. Dans l'application GrepTheWeb pour générer les rapports d'activité mensuels, cette approche permet de créer des rapports plus dynamiques.

Utilisation d'Amazon SimpleDB conjointement aux autres services

Créez des infrastructures, des bibliothèques et des utilitaires qui utilisent la fonctionnalité d'au moins deux services ensemble en un. Pour GrepTheWeb, nous avons créé une petite infrastructure qui utilise Amazon SQS et Amazon SimpleDB ensemble pour externaliser l'état approprié. Par exemple, tous les contrôleurs sont hérités de la classe BaseController. La principale responsabilité de la classe BaseController est de retirer le message de la file d'attente « de », de valider les statuts à partir d'un domaine Amazon SimpleDB particulier, d'exécuter la fonction, de mettre à jour les statuts avec un nouvel horodatage et un nouveau statut et de placer un nouveau message dans la file d'attente « à ». L'avantage d'une telle configuration réside dans le fait que, en cas de dysfonctionnement matériel ou d'extinction de l'instance du contrôleur, un nouveau nœud peut être amené presque immédiatement et reprendre l'opération en récupérant les messages de la file d'attente Amazon SQS et leur statut dans Amazon SimpleDB après un redémarrage, ce qui rend le système global d'autant plus résilient.

Même si elle n'est pas utilisée dans ce modèle, une pratique courante consiste à stocker les fichiers réels sous forme d'objets sur Amazon S3 et à stocker toutes les metadonnées associées à l'objet sur Amazon SimpleDB. De même, l'utilisation d'une clé Amazon S3 sur l'objet en tant que nom de l'élément dans Amazon SimpleDB est une pratique courante.



Résumé public de Abstract BaseController (SQSMessageQueue fromQueue, SQSMessageQueue toQueue, domaine SDBDomain)

Figure 7 : Architecture et flux de travail du contrôleur

Bonnes pratiques d'Amazon EC2

Lancement de plusieurs instances simultanément

Au lieu d'attendre que vos instances EC2 démarrent l'une après l'autre, nous vous recommandons de les démarrer toutes simultanément avec une commande simple d'exécution d'instances qui indique le nombre d'instances de chaque type.

Automatisation maximale

Ceci s'applique à tout ce que nous faisons et nécessite une mention spéciale car l'automatisation d'Amazon EC2 est souvent ignorée. L'une des plus importantes fonctions d'Amazon EC2 vous permet de mettre en service n'importe quel nombre d'instances de calcul en réalisant un simple appel à un service web. L'automatisation permet au développeur d'exécuter un centre de données dynamique programmable qui s'étende et se contracte en fonction des besoins. Par exemple, le fait d'automatiser votre cycle création-test-déploiement sous la forme d'une AMI (Amazon Machine Image), puis de l'exécuter automatiquement sur Amazon EC2 chaque nuit (à l'aide d'un travail CRON) permet de gagner beaucoup de temps. En automatisant le processus de création AMI, vous pouvez gagner beaucoup de temps en configuration et en optimisation.

Ajout des instances de calcul à la volée

Avec Amazon EC2, nous pouvons lancer un nœud en quelques minutes seulement. Hadoop prend en charge l'ajout dynamique de nouveaux nœuds et des nœuds de suivi de tâche à un cluster en cours d'exécution. Il est possible de lancer simplement de nouvelles instances de calcul et de démarrer les processus Hadoop dessus, de les orienter vers l'instance principale et de faire croître (et diminuer) dynamiquement le cluster en temps réel, afin d'accélérer le processus global.

Préservation de vos informations d'identification AWS lors de la création d'un bundle d'une AMI

Si votre AMI exécute des processus qui doivent communiquer avec d'autres services web AWS (pour interroger la file d'attente Amazon SQS ou pour lire les objets depuis Amazon S3), on fait souvent l'erreur d'intégrer les informations d'identification d'AWS dans la conception de l'AMI. Au lieu d'intégrer ces informations d'identification, il convient de les communiquer sous forme d'arguments en utilisant la fonction de lancement paramétrée et chiffrée avant de les envoyer sur le réseau. Procédez comme suit :

1. Générez une nouvelle paire de clés RSA (utilisez les outils OpenSSL).
2. Copiez la clé privée sur l'image, avant de la regrouper (afin qu'elle soit intégrée à l'AMI finale).
3. Envoyez la clé publique avec les détails de l'image, afin que les utilisateurs puissent l'utiliser.
4. Lorsqu'un utilisateur lance l'image, il doit d'abord chiffrer sa clé AWS secrète (ou clé privée si vous vouliez utiliser SOAP) avec la clé publique que vous lui avez donnée à l'étape 3. Ces données chiffrées doivent être injectées via les données utilisateur au lancement (autrement dit, la fonction de lancement paramétrée).
5. Votre image peut ensuite déchiffrer cela au démarrage et l'utiliser pour déchiffrer les données requises pour contacter Amazon S3. Veillez également à supprimer cette clé privée au moment du redémarrage, avant d'installer la clé SSH (autrement dit, avant que les utilisateurs ne se connectent à la machine). Si les utilisateurs n'auront pas d'accès à la racine, vous n'avez pas à supprimer la clé privée : veillez simplement à ce qu'elle ne soit pas lisible par les utilisateurs autres que la racine.

Crédits

Nous remercions tout spécialement Kenji Matsuoka et Tinou Bao – l'équipe indispensable qui a développé l'architecture de GrepTheWeb.

Suggestions de lecture

Livres blancs d'Amazon
SimpleDB Livre blanc
d'Amazon SQS Hadoop Wiki
Site web Hadoop
Exemples de Grep
distribué Livre sur le
mappage et la réduction

Blog : Taking Massive Distributed Computing to the Common man – Hadoop on Amazon EC2/S3 (Le calcul distribué en masse à la portée de tout un chacun – Hadoop sur Amazon EC2/S3)

Annexe 1 : Amazon S3, Amazon SQS, Amazon SimpleDB – Quand utiliser quoi ?

Le tableau explique quel service Amazon utiliser et à quel moment :

	Amazon S3	Amazon SQS	Amazon SimpleDB
Idéal pour	Stocker des types d'objets volumineux, à inscription unique, lus par beaucoup de monde	Formuler de brefs messages temporaires à courte durée de vie	Demander des données d'attributs légères
Exemples	Fichiers de type média, audio, vidéo, images volumineuses	Travaux de flux, messages XML/JSON/TXT	Requête, mappage, balisage, journaux de flux de clics, metadonnées, gestion des états
Déconseillé pour	Formuler des requêtes, la distribution de contenu	Objets volumineux, persistants	Systèmes transactionnels
Exemples	Base de données, systèmes de fichiers	Datastores permanents	OLTP, DW cube rollups

Recommandations

Etant donné que les services AWS sont des services primitifs de création de blocs, ils offrent des avantages optimaux lorsqu'ils sont utilisés conjointement à d'autres services.

- **Utilisez Amazon S3 et Amazon SimpleDB ensemble lorsque vous voulez demander des objets Amazon S3 en utilisant leurs metadonnées**

  Nous vous recommandons de stocker les fichiers volumineux sur Amazon S3 et les metadonnées associées, ainsi que les informations de référence, sur Amazon SimpleDB afin que les développeurs puissent demander les metadonnées. Les metadonnées en lecture seule peuvent également être stockées sur Amazon S3 sous forme de metadonnées sur objet (par ex., auteur, date de création, etc.).

Entités Amazon S3	Entités Amazon SimpleDB
Compartiment	Domaine (privé pour le souscripteur)
Clé/3 URI	Nom de l'élément S
Metadonnées décrivant l'objet S3	Attributs d'un élément

- **Utilisez SimpleDB et Amazon SQS ensemble lorsque vous voulez qu'une application se déroule en phases.**

  Stockez les messages temporaires dans Amazon SQS et les statuts des travaux/messages dans Amazon SimpleDB, afin de pouvoir mettre fréquemment à jour les statuts et d'obtenir le statut de n'importe quelle demande, à tout moment, en lançant simplement une requête sur l'élément. Cette opération est très efficace dans les systèmes asynchrones.

- **Utilisez Amazon S3 et Amazon SQS ensemble lorsque vous voulez créer des pipelines de traitement ou des solutions producteur-consommateur.**

  Stockez les fichiers bruts sur Amazon S3 et insérez un message correspondant dans une file d'attente Amazon SQS, avec la référence et les metadonnées (S3 URI, etc.).