Simulation à base de réseau de pétri pour le Cloud Computing

Hélène Coullon

18 septembre 2017

Email : helene.coullon@imt-atlantique.fr

Mots Clés : *déploiement/installation d'applications, Cloud Computing, YAML, Réseaux de pétri*

Contexte

L'objectif de ce projet se place dans le contexte du Cloud Computing, ou plus généralement de l'informatique utilitaire. Le Cloud Computing est désormais largement utilisé par le grand public, par les sociétés et par les institutions publiques. Son principe est d'offrir des ressources informatiques (IT) de type stockage, calcul ou réseau à la demande avec une haute disponibilité et une grande flexibilité. Le modèle économique associé est basé sur l'idée de ne payer que ce que l'on consomme ce qui représente une grande économie, pour les sociétés principalement. L'avancée technologique qui a permis de mettre en place le Cloud Computing est la technologie de virtualisation des machines. La virtualisation permet de dissocier la machine physique (hardware) de toutes les couches logicielles supérieures. Ainsi on peut déplacer une machine virtuelle d'une machine physique à une autre sans difficulté. Trois niveaux de services sont offerts par les fournisseurs de Cloud. Le plus bas niveau est plutôt réservé à des experts informatiques et permet de louer de l'Infrastructure à la demande (IaaS), c'est à dire des machines virtuelles avec uniquement un OS. Le niveau supérieur est le PaaS. Il permet de louer des machines virtuelles déjà équipées de toute une plate-forme de développement (bases de données, frameworks etc.). Enfin, le niveau le plus haut, et souvent le plus utilisé par le grand public est de louer des applications à la demande (SaaS). Tout en restant transparent pour l'utilisateur, le SaaS cache la mise en place par le fournisseur de Cloud de machines virtuelles avec des couches logicielles complètes jusqu'à l'application.

L'un des enjeux important du Cloud Computing c'est d'être capable d'installer, mettre en place (on utilise souvent le mot *déploiement*) de façon automatique n'importe quelle application sur une infrastructure de Cloud Computing. Le fait de gérer ce fonctionnement automatique quelque soit l'application est un gain de temps très important. En effet actuellement chaque fois qu'un fournisseur de cloud ou qu'un utilisateur souhaite mettre en place une nouvelle application sur le cloud il doit retravailler sur des scripts permettant sa mise en place automatique. De plus, bien souvent la complexité de déploiement des applications est très grande.

Nous avons proposé dans notre équipe de recherche un outil de déploiement automatique d'applications (et de systèmes) basé sur les réseaux de pétri. Le but de ce projet est de coder une

version simplifiée du simulateur de cet outil de déploiement en java. Il s'agit de coder les fonctionnalités de base d'un réseau de pétri associé à des services et des dépendances supplémentaires, et de simuler le déroulement du déploiement avec une interface graphique.

Travail demandé

La Figure 1 représente l'installation d'une application contenant 3 *services* (ou trois composants logiciels qui constitue l'application). Chaque service est décrit par :

— un nom ;
— un réseau de pétri multi-jetons qui représente les différentes phases du cycle de vie du service (comme par exemple la configuration de l'OS, l'installation de paquets, le fait d'être prêt à l'utilisation etc.) ;
— des dépendances entrantes et sortantes (représentées par les flèches pointillées sur la figure).

Un réseau de pétri multi-jetons est constitué de :

— une liste d'états (un état est représenté par un cercle sur la figure) ;
— une liste de transitions (une transition est représentée par un traits sur la figure).

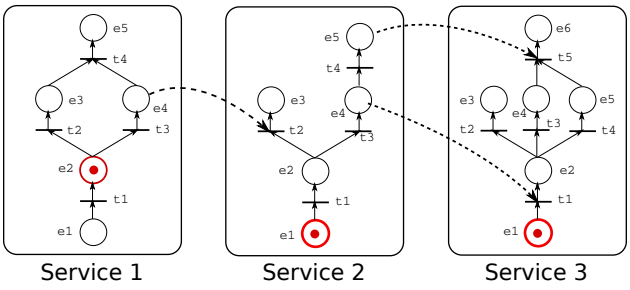

FIGURE 1 – Exemple d'application composée de 3 services, chacun exprimé par un réseau de pétri et des dépendances avec les autres services.

Le fonctionnement du réseau de pétri interne à un service suit les règles suivantes :

— les états représentent les statuts du service ;
— les transitions correspondent à des actions à effectuer pour pouvoir passer à l'état suivant (celui qui se trouve plus haut) ;
— il y a toujours une transition entre deux états d'un réseau de pétri ;
— lorsque plusieurs transitions sortent d'un état cela signifie que les transitions peuvent être effectuées en même temps ;
— lorsque plusieurs transitions sont réunies pour aller vers un seul état cela signifie que les deux transitions doivent être terminées pour arriver à l'état suivant, et c'est donc la transition la plus longue qui débloquera l'état ;

Nous ajoutons aux réseaux de pétri classiques des dépendances qui permettent de simuler des dépendances entre les différents services de l'application :

— une flèche pointillées (une dépendance) va toujours d'un état vers une transition ;
— lorsque qu'une dépendance arrive sur une transition, cela signifie que l'état d'où vient la flèche doit contenir un jeton, où doit avoir déjà été validé, sinon la transition ne peut pas

avoir lieu ;

Enfin, afin de simuler le déroulement d'un scénario, nous ajoutons des temps d'exécution aux transitions.

Le travail demandé est découpé en 2 phases :

Phase 1 - Dans la première phase nous vous demandons d'implémenter les fonctionnalités demandées, sans interface graphique, c'est-à-dire :

— définir le fichier d'entrée qui permet de décrire les services, leur réseau de pétri interne et les relations entre les services ;
— un parseur de ce fichier d'entrée ;
— concevoir les fonctions d'exécution du système en suivant les règles évoquées plus haut ;

Phase 2 - Juste avant la deuxième phase vous allez interchanger vos codes. Le deuxième phase consiste à implémenter une interface graphique qui permet :

1. de concevoir une application comme celle présentée sur la Figure 1, et écrire le fichier correspondant ;
2. le simulateur graphique du déploiement de l'application créée.

C'est pour implémenter le simulateur que le temps d'exécution associé aux transitions est utile. Il faut que l'interface graphique montre en temps réel et dynamiquement le déroulement du déploiement en suivant les règles évoquées plus haut, donc en utilisant les fonctionnalités codées dans la phase 1.

Chacun d'entre vous va utiliser le code de la phase 1 d'un autre pour implémenter la phase 2, donc bien évidemment il faut faire en sorte de donner un code propre, bien structuré et bien commenté à votre camarade de classe.

Outils

La façon d'implémenter les fonctionnalités et l'interface graphique est libre, voici toutefois les règles qui s'imposent à vous pour le développement :

— les entrées du simulateur, c'est à dire la description des services et leur assemblage sera faite en YAML (http://yaml.org/), format de plus en plus utilisé pour les applications et beaucoup moins lourd que du XML, pour cela vous pouvez utiliser une bibliothèque de votre choix (voir le site web) ;
— Il vous est demandé de ne pas utiliser une bibliothèque de réseau de pétri déjà existante mais de coder les fonctionnalités de base d'un réseau de pétri vous même.