Interception système pour l'émulation d'applications

Domaine : Informatique distribuée, mise au point d'applications, évaluation de performances.
Laboratoire : LORIA, Laboratoire lorrain de recherche en informatique et ses applications
Encadrants : Lucas Nussbaum et Martin Quinson (équipe AlGorille)
Contact : Lucas.Nussbaum@loria.fr et Martin.Quinson@loria.fr

1 Contexte scientifique : évaluation d'applications distribuées

1.1 Expériences *in situ* et simulation

Les expérimentations jouent un rôle fondamental en science. En informatique, elles sont particulièrement importante dans le domaine du calcul parallèle et distribué car les modèles mathématiques adaptés à ce sujet sont soit trop complexes pour être vraiment utiles, soit basés sur des suppositions simplistes qui remettent en cause leur pertinence.

Deux grandes approches méthodologiques sont possibles pour réaliser ces expériences [2]. Il est naturellement possible de les mener *in situ*, c'est-à-dire de **tester les applications en conditions réelles sur les plates-formes cibles**. Mais cette approche pose de nombreux défis méthodologiques. Tout d'abord, de telles expériences peuvent être très lourdes à mettre en œuvre. Il faut écrire l'application complète, construire entièrement la plate-forme, et exécuter entièrement l'application sur la plate-forme. Cette approche est donc difficile dans une phase exploratoire, visant par exemple à étudier les avantages comparés d'algorithmes différents, ou à dimensionner au mieux la plate-forme pour répondre aux besoins. De plus, il est difficile de reproduire une expérience *in situ* dans des conditions comparables car les plates-formes sont partagées avec d'autres utilisateurs, qui influent directement sur les conditions expérimentales. Enfin, il est très difficile de mener des campagnes de test sur un ensemble représentatifs de plates-formes car l'expérimenteur n'a typiquement accès qu'à une ou deux ressources de calculs bien particulières.

L'autre approche méthodologique pour des expérimentations dans le domaine de l'informatique distribuée est basée sur la **simulation**. Elle offre une réponse à la plupart des problèmes des expérimentations *in situ* : mise en œuvre rapide et facile, reproductibilité parfaite des expériences et possibilité d'explorer de nombreux scénarios expérimentaux en temps raisonnable. Cette approche pose cependant des défis méthodologiques propres. Tout d'abord, le biais expérimental semble plus important que dans les expériences in situ et doit être évalué avec attention. Ensuite, la simulation impose souvent de coder les applications à évaluer dans un formalisme particulier.

Le cycle de travail classique est souvent le suivant : (1) Évaluation de prototypes sur simulateur (validation des algorithmes, dimensionnement de la plate-forme) (2) Réimplémentation effective de l'application (3) Validation de l'application sur des plates-formes de test (4) Utilisation de l'application sur la plate-forme cible.

1.2 Émulation pour l'évaluation d'applications distribuées

Ce projet s'insère dans un axe de recherche visant à permettre l'étude sur simulateur d'applications complètes. Cette **approche hybride d'émulation** consiste à exécuter sur une plate-forme simulée non pas des prototypes de l'application, mais directement l'application elle-même. Cela évite de devoir développer deux versions de l'application, l'une étant adaptée à l'expérimentation sur simulateur tandis que la seconde est utilisée en production. Cette approche offre de multiples avantages : moins de travail, moins de divergences potentielles entre les versions de test et celles de production, possibilité d'étudier le comportement d'applications directement sans devoir développer un prototype au préalable, ...

L'émulation d'applications réparties consiste à (1) intercepter les actions des applications (communication, calcul ou autre) et à (2) reporter ces actions dans le simulateur. Dans le cas d'une émulation *offline*, on sauvegarde toutes ces actions sur disque, puis on les rejoue après coup dans le simulateur. Dans une émulation *online*, on reporte immédiatement ces actions dans le simulateur, puis on retarde l'application du temps calculé par le simulateur. Ainsi, si l'application envoie des données à une autre machine et que le simulateur indique que cette opération prend cinq minutes sur la plate-forme émulée, on bloquera l'application pour cette durée.

Les objectifs du projet sont d'étudier les moyens d'intercepter les actions des applications, et d'implémenter une méthode d'interception en interagissant avec le simulateur SimGrid [1].

2 Approches possibles

Plusieurs approches semblent possibles pour intercepter les actions des applications. La première est d'utiliser l'approche choisie par exemple par le *debugger* `gdb` en traçant l'application grâce à l'appel système `ptrace`. Il permet à un programme de suivre un processus fils, et d'être tenu au courant chaque fois que ce fils réalise un appel système (comme les communications sur le réseau).

L'approche choisie par le détecteur d'anomalie `valgrind` semble également prometteuse. Il s'agit d'instrumenter le binaire au chargement afin d'ajouter des points d'observation autour d'appels d'intérêts. Les outils `valgrind` classiques surveillent les appels de gestion de la mémoire, mais il semble envisageable d'utiliser ce framework pour surveiller tout autre chose, comme l'utilisation du réseau.

L'éditeur de liens dynamiques de linux (`ldd`) offre également la possibilité d'interposer du code dans l'exécution d'un programme. Les bibliothèques listées dans la variable d'environnement `LD_PRELOAD` sont injectées dans le programme. Ce mécanisme, similaire aux *dll injection* souvent utilisées sous windows pour nuire à la sécurité du système, permettent trivialement d'ajouter des points d'observation dans le code.

Enfin, la machine virtuelle Java offre un mécanisme similaire, souvent mis en œuvre par les programmes de debug et de profiling. Grâce à l'argument `-javaagent`, il est possible d'installer un agent dans la JVM capable de modifier toutes les classes à leur chargement.

3 Description du projet et compétences requises

Les étudiants devront **choisir l'une de ces approches logicielles pour l'interception des actions des applications, et implémenter une solution la plus complète possible suivant cette approche**. Les encadrants ont déjà évalué chacune des approches possibles, et pourront bien entendu guider les étudiants pour les débloquer au besoin.

Bien que rien ne soit extrêmement compliqué, le sujet de ce projet reste relativement technique. Il est donc attendu une certaine aisance en programmation de la part des étudiants, ainsi qu'un certain goût pour les systèmes informatiques.

Références

[1] Henri Casanova, Arnaud Legrand, and Martin Quinson. SimGrid : a Generic Framework for Large-Scale Distributed Experiments. In *10th IEEE International Conference on Computer Modeling and Simulation - EUROSIM / UKSIM 2008*, Cambrige United Kingdom, 2008. IEEE.

[2] Jens Gustedt, Emmanuel Jeannot, and Martin Quinson. Experimental Validation in Large-Scale Systems : a Survey of Methodologies. *Parallel Processing Letters*, 19(3) :399–418, 2009.