UNIVERSITÉ TOULOUSE III PAUL SABATIER

Millian Poquet
Maître de conférences
Univ. Toulouse III, IRIT, équipe Sepia
IRIT2-469, 120 cours Rose Dieng-Kuntz, 31400 Toulouse
millian.poquet@univ-tlse3.fr

Batprotocol et algorithmes d'ordonnancement en Rust

Contexte

Batsim est un simulateur d'infrastructures de calcul qui permet d'étudier des politiques de gestion de ressources. Une personne utilisant Batsim doit décrire comment chaque application doit être exécutée en indiquant quel *modèle d'exécution* l'application doit utiliser. Batsim est ensuite chargé de la *bonne* exécution de ces modèles, typiquement en configurant et en appelant SimGrid de la manière adéquate.

Toutes les décisions lors de l'exécution de la simulation sont découplées de Batsim : un code externe de prise de décision externe est appelé à chaque pas de temps qui nécessite une décision. Ce code prend soit la forme d'une bibliothèque chargée dynamiquement, soit sous la forme d'un processus respectant une API d'appel de procédure distante.

Un protocole de communication existe pour permettre l'échange de messages entre Batsim et le code de prise de décision. L'implémentation actuelle de Batsim utilise FlatBuffers comme technologie de sérialisation et de désérialisation de messages, ce qui permet au code de prise de décision d'être écrit dans n'importe quel langage de. Une bibliothèque C++ nommée batprotocol permet de facilement manipuler les messages du protocole ; cette bibliothèque est utilisée par Batsim et par différents algorithmes de gestion de ressources.

Objectifs du projet

Le but du projet est d'implémenter la version Rust de la bibliothèque batprotocol. FlatBuffers sachant lui-même générer le code bas niveau de dé/sérialisation des messages dans différents langages de programmation dont Rust, votre rôle sera de définir une API ergonomique et efficace que les algorithmes de gestion de ressources vont utiliser pour lire les évènements provenants de Batsim et pour prendre leurs décisions. Vous devrez ensuite implémenter cette API dans une bibliothèque Rust.

Une base solide de tests de régression est attendue sur la version Rust de la bibliothèque. Cette base devra s'intégrer à celle existante en C++, afin qu'on puisse automatiquement détecter une différence de comportement entre la version C++ ou Rust de la bibliothèque.

Enfin, il vous sera demandé d'implémenter des algorithmes simples de gestion de ressources en Rust ; typiquement FCFS et EASY Backfilling. Cette implémentation permettra tout d'abord de déceler si l'API que vous proposez est agréable à utiliser et si elle correspond aux conventions d'API utilisées en Rust. Cette implémentation permettra également d'évaluer la performance de votre bibliothèque en la comparant à la version C++ sur ces mêmes algorithmes, qui sont déjà implémentés en C++.

Candidature

Contactez-moi par mail avec les informations suivantes.

- Très courte motivation par rapport au sujet (2 phrases)
- Bulletins de notes (master et licence)
- CV court (parcours d'études, expériences professionnelles, compétences)