Pre-rapport du PSAR

API générique pour le développement d'applications réparties

Tarik Atlaoui

Nicolas Peugnet

Kimmeng Ly

Max Eliet

09 Mars 2020

1 Introduction

Notre projet a pour but d'implémenter un intergiciel permettant l'execution d'applications réparties, aussi bien sur une infrastructure réelle que sur un simulateur, sans avoir à en modifier le code, afin de simplifier le développement de futures applications réparties.

Dans notre cas, nous avons choisi d'implémenter cet intergiciel pour MPI (Message Passing Interface) en tant qu'API d'infrastructure réelle, et PeerSim en tant que simulateur à événements discrets.

2 Cahier des charges

2.1 Choix d'une interface générique

Nous avons commencé par définir une API générique : *Infrastructure* regroupant les fonctionnalités qu'offriront à la fois PeerSim et MPI aux applications réparties, à commencer par une primitive d'envoi de message, que nous enrichirons au fur et à mesure.

De plus, nous avons encapsulé les fonctions de chaque noeud du système dans une classe *NodeProcess*, par exemple le traitement à effectuer lors du démarrage du noeud ou de la récepion d'un message.

2.2 Implantation de l'interface pour PeerSim et MPI

Nous avons ensuite implanté les premières fonctionnalités de notre interface nécessaires au fonctionnement d'un programme de test basique faisant circuler d'un messsage à travers un anneau de noeuds du système.

2.3 Création d'une classe *Message*

Nous avions ensuite besoin d'une classe représentant l'enveloppe d'un message, qui sera étendue par l'utilisateur pour ses propres messages.

2.4 Unification du lancement de l'application

Ensuite, nous avons souhaité unifier et simplifier le lancement de l'application par le biais d'une interface /textitRunner avec une implantation pour MPI et PeerSim, afin que le lancement d'une application se résume à choisir entre celles-ci.

2.5 Factorisation de l'aiguillage des messages

La fonctionnalité suivante que nous avons ajouter est celle de l'aiguillage automatique d'un message vers le traitement qui lui est adéquat, à l'aide d'annotations.

2.6 Description d'un scenario

La dernière fonctionnalité que nous avons ajouté jusqu'à présent nous permet de décrire un scénario, correspondant à l'execution d'une application, sous la forme d'un fichier JSON.

2.7 Fonctions de wait et notify

La prochaine fonctionnalité sur laquelle nous travaillons est celle de l'attente d'un processus sur une condition ou un message particulier.

2.8 Autres fonctionnalités

Nous pourrons ajouter d'autres fonctionnalités qui nous viendraient à l'esprit, si le temps le permet.