Systèmes distribués - Projet

Ensimag - 3A

édition 2018-2019

1 Introduction

L'objectif du projet de systèmes distribué vise à développer une version distribuée de *GNU Make* et à la valider expérimentalement sur Grid5000. Les choix algorithmiques, d'implantation et d'expérimentation sont entièrement à votre charge.

Ce travail est à réaliser par groupe de 3 étudiants ou 4 étudiants.

2 FAQ

Est-il possible de rendre le travail en retard ?

Non, il vous faudra présenter vos résultats. Il est donc important de ne pas réaliser les expériences à la dernière minute quand tout le monde aura besoin des salles machines.

Il faut aussi une note avant votre jury.

Est-il possible de ne pas rendre le projet ?

Ce projet est votre note.

Ai-je une raison de réaliser le projet en quadrinome ?

La taille des groupes est prise en compte lors de la notation.

Est-il possible d'utiliser des bibliothèques et outils de programmation parallèle / distribuée ?

Oui. Mais vous devez choisir une bibliothèque différente de celle de vos camarades.

Qui sera l'orateur de la présentation orale ?

L'orateur sera tiré au sort parmi les membres de l'équipe au début de la présentation.

3 Objectif

Le projet de systèmes distribués vise au développement d'une version distribuée de l'utilitaire GNU Make.

L'accent est porté ici sur les performances de la distribution des différentes commandes. De ce fait, un certain nombre de choses ne sont pas demandées comme :

— le parsing de makefiles complexes : on se restreindra ici à un format simple,
— la tolérance aux pannes (autre que celle gérée par votre framework) : on supposera ici qu'en cas de pannes, l'utilisateur relancera l'exécution
— la réalisation d'un ordonnanceur distribué complexe. Certain frameworks font le travail pour vous, d'autres, non. Dans ce cas, nous utilisons ici un processus maitre, contrôlant l'ensemble de l'exécution.

Concrètement, votre programme distribué doit donc être capable de :

— parser un fichier Makefile simplifié en entrée
— exécuter les commandes distantes en prenant en compte les dépendances
— réaliser un équilibrage de charge centralisé
— réaliser les transferts de données nécessaires. Attention, l'utilisation de NFS avec vos différents framework n'est pas forcément correcte, et, dans tous les cas, elle diminuera probablement vos performance.
— détecter la fin des calculs

4 Travail attendu

4 choses différentes sont demandées :

1. le cahier de laboratoire de votre projet (fichier CAHIER_DE_LABORATOIRE.org) dans la racine de votre projet ;
2. le programme en lui même ;
3. l'évaluation aux travers de tests de performance ;
4. l'écriture d'un rapport incluant les tests ;
5. une présentation orale incluant les tests.

L'analyse expérimentale compte autant que le programme en lui-même.

4.1 Programmation

Vous êtes libres d'utiliser le langage de programmation ainsi que les bibliothèques et outils de votre choix.

Il est possible de réaliser une version simple du projet, correspondant à un nombre minimal de fonctionnalités mais la présence de fonctionnalités supplémentaires est un plus.

4.1.1 Fonctionnalités requises

Votre programme doit être capable de parser des fichiers Makefiles simples : sans variables, chaque ligne est soit une déclaration d'une cible avec ses dépendances (de type *test.o : test.c*) ou bien une commande à exécuter directement (les caractères spéciaux du type $< et autres ne sont pas demandés).

Vous pouvez, si votre outil présente un intérêt pour vous l'étendre pour parser plus de fichiers mais cette fonctionnalité n'est pas demandée et ne sera pas prise en compte lors de l'évaluation.

Un ensemble de fichiers de tests est disponible sur `https://gitlab.ensimag.fr/5MMSYSD/2018-2019/ensimag-parmake.git`. Vous clonerez ce projet sur le gitlab de l'Ensimag (dans le sous-groupe 5MMSYSD/2018-2019).

Le déploiement de l'application est réalisée par des outils de votre choix (mpirun / taktuk / ...).

L'équilibrage de charge doit être dynamique et réalisé de manière centralisé, par exemple à l'aide de l'algorithme d'ordonnancement par liste.

Il vous est demandé de détecter la terminaison des calculs et de stopper alors chaque processus.

La gestion des dépendances entre cibles et des communications est à votre charge. En particulier attention aux interactions avec NFS.

4.1.2 Fonctionnalités supplémentaires

De nombreux ajouts peuvent être effectués sur le travail initial :

— version multi-coeurs : est-il possible d'optimiser les transferts de fichiers ?

— version privilégiant la localité : on souhaiterai éviter de générer trop de transferts de fichiers en privilégiant la réutilisation.

Chaque ajout effectué devra être décrit dans le rapport.

4.2 Cahier de laboratoire et reproductibilité

Le but du cahier de laboratoire est de maintenir l'information sur vos expérimentations. En particulier, au moins, pour chaque expérience :

— version du code utilisé (SHA git)

— date de l'expérimentation

— machines utilisées

— fichiers stockant les résultats (idéalement en les versionnant)

Il faudra aussi noter soigneusement, les scripts ou les commandes pour compiler, déployer et exécuter votre code à chaque expérience.

On n'édite jamais le passé du cahier ! Au pire, on ajoute des lignes pour indiquer les erreurs.

4.3 Tests de performance

Une fois le programme réalisé et débogué, il est nécessaire d'effectuer différents tests de performances afin de le valider (Temps d'exécution, accélération, efficacité, avec les intervalles de confiance des mesures).

Il est important que les résultats obtenus puissent être explicables.

Les tests devront être réalisés sur 64 coeurs. Bien entendu, l'utilisation d'un nombre plus élevé de machines est possible pour démontrer le passage à l'échelle.

Les courbes bruitées, ou lissées, sont à éviter, tout comme les présentations de résultats sous forme de tableaux.

4.4 Rapport

Le rapport se décompose en 2 parties : une première partie (1 page max) expliquant le travail réalisé, le langage, les bibliothèques et algorithmes utilisés ; une seconde partie détaillant les tests de performances.

La première partie sert à mettre en valeur vos choix personnels et vos capacités d'analyse : essayez donc de justifier un maximum les choix réalisés. Décrivez également la manière d'installer et déployer votre application.

La seconde partie doit être capable de me convaincre que votre programme fonctionne efficacement et me faire sentir ses limites actuelles. Il est important de présenter clairement les tests réalisés : méthodologie, conditions expérimentales, résultats obtenus, ... Il vous est également demandé de réaliser une analyse de ce que vous obtenez : est-ce conforme à vos attentes ? Est-ce que les choses semblent passer à l'échelle ? Quels sont selon vous les facteurs limitant ? L'utilisation des coeurs a-t'elle un impact ?

4.5 La présentation orale

Un des buts est de faire un retour sur votre projet à vos camarades. La présentation orale durera environ 17 minutes. Elle sera découpée en deux parties.

— 17 minutes de présentation réalisée par un orateur tiré au sort parmi les membres de l'équipe,
— 5 minutes de questions, où tous les membres de l'équipe peuvent répondre. L'assemblée pourra elle-aussi poser des questions.

Un video-projecteur sera disponible. La présentation sera faite avec 7 à 8 diapositives maximum (hors titre, annexes pour les questions, animations, etc.). Ces 7 à 8 diapositives devront comporter

— une introduction didactique à l'intergiciel utilisé et à ses paradigmes ;
— les choix d'implantations ;
— au moins une courbe de performance ;
— un retour critique sur l'intergiciel (facilité d'utilisation, points forts, points faibles, etc.).