Département d'Informatique

TP - ASR7 Programmation Concurrente

—

Première utilisation de threads

Matthieu Moy, Grégoire Pichon, Sylvain Brandel, Guillaume Damiand,
Laurent Lefevre, Frédéric Suter

Printemps 2021

I Préliminaire : accès distant

Pour commencer, vu que cette année les TP se feront majoritairement en distant, voici quelques informations pour vous connecter sur les machines de l'université.

Vous pouvez travailler au choix :

— Sur votre machine personnelle pour les TP sur les threads, du moment que vous avez un compilateur C++.

— Sur les PC de l'université, en y accédant à distance via SSH. Vous ne pourrez pas passer administrateur (root), donc les TP d'ordonnancement et d'administration système ne seront pas faisables sur ces machines.

— Sur les machines virtuelles (VM) de l'université. Vous avez une VM par étudiant, sur laquelle vous pouvez passer administrateur. Ces VM sont prévues pour faire les TP d'ordonnancement et d'administration système, mais vous pouvez y accéder dès aujourd'hui. Attention, ces VM n'ont qu'un cœur, donc l'intérêt est limité pour les TP sur les threads, et **les VM peuvent être détruites sans préavis** donc ne laissez aucune donnée importante dessus sans en avoir fait une sauvegarde.

Dans tous les cas, lisez et expérimentez avec ce qui suit dès maintenant.

Pour l'exemple, nous allons essayer de nous connecter au PC `b710l0201.univ-lyon1.fr` qui se trouve physiquement dans une salle machine (sous réserve que ce PC soit allumé). S'il ne fonctionne pas, essayez de deviner l'adresse d'un autre PC du nautibus, les noms sont toujours du type `b710` (fut un temps ou le Nautibus s'appelait « bâtiment 710 »...), la lettre « l » (pas le chiffre 1), puis 2 chiffres pour la salle (par exemple 02 pour la salle Nautibus TP2), et deux chiffres pour le numéro de poste (par exemple 01 pour la première machine de la salle). Les machines des salles TP2 et TP14 sont allumées sous Linux pendant la crise coronavirus, donc vous pouvez utiliser par exemple `b710l0201.univ-lyon1.fr`, `b710l0202.univ-lyon1.fr`, `b710l0203.univ-lyon1.fr`, ..., `b710l1401.univ-lyon1.fr`, `b710l1402.univ-lyon1.fr`, ... pour vos tests.

Dans les explications qui suivent, on suppose que vous utilisez une machine de type Unix (Linux, le sous-système Linux de Windows, ou Mac OS). Si vous êtes sous Windows, des explications pour utiliser MobaXTerm pour faire les manipulations équivalentes sont disponibles ici : `https://nlouvet.gitlabpages.inria.fr/lifasr5/connec.html`.

Dans un monde sans firewall, on pourrait s'y connecter simplement avec la commande :

```
ssh votre-login-lyon1@b710l0201.univ-lyon1.fr
```

En pratique, ça ne marchera pas de l'extérieur car l'accès est bloqué pour des raisons de sécurité. En master, les étudiants ont accès au VPN qui permet de « faire comme si » la machine faisait partie du réseau local, mais ce n'est pas le cas en Licence. Voici quelques solutions :

I.1 Rebond SSH à la main sur `linuxetu`

Vous avez accès via SSH, même de l'extérieur, à la machine `linuxetu.univ-lyon1.fr`, donc vous pouvez vous connecter d'abord a linuxetu, puis dans le shell ouvert sur linuxetu lancer une connection SSH vers votre machine virtuelle. Par exemple, si votre login Lyon 1 est p1234567, vous pouvez le faire comme ceci :

```
votre-ordinateur$ ssh p1234567@linuxetu.univ-lyon1.fr
p1234567@linuxetu.univ-lyon1.fr's password: <votre-mot-de-passe-Lyon1>
[...]
==============
Ceci est une passerelle SSH.
Votre 'home universitaire' est dans le dossier HOMELYON1.
Si vous utilisez une clé SSH, vous pouvez exécuter la commande mhome pour le monter.
==============

p1234567@linuxetu:~$ ssh p1234567@b710l0202.univ-lyon1.fr
The authenticity of host b710l0202.univ-lyon1.fr can't be established.
ECDSA key fingerprint is SHA256:/VAtIv4LXOEiKzQRNYsRPcxENUvTrsuAvK31gKp8e+s.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.75.165' (ECDSA) to the list of known hosts.
chaprot@192.168.75.165's password: <mot-de-passe-de-la-vm>
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-22-generic x86_64)
[...]
chaprot@vm-26:~$
```

L'ordinateur vous demande de vérifier l'empreinte de la clé, nous supposerons que tout est bon.

I.2 Rebond SSH automatique sur `linuxetu`

Sur la machine locale, on peut demander à faire passer automatiquement les connections par la machine `linuxetu`. Pour cela, dans le fichier `~/.ssh/config` (à créer s'il n'existe pas), ajoutez les lignes (remplacez votre-login-lyon1 bien sûr) :

```
Host !forge.univ-lyon1.fr !linuxetu.univ-lyon1.fr !*.tpr.univ-lyon1.fr *.univ-lyon1.fr
ProxyCommand ssh -N -W %h:%p votre-login-lyon1@linuxetu.univ-lyon1.fr
```

On peut lire ces lignes comme « Pour toutes les machines, à l'exception de la forge, de linuxetu, et des machines des salles TPR, dont le nom termine par univ-lyon1.fr, ouvrir une

connexion avec linuxetu pour créer la connexion demandée. ». Si vous ne vous êtes pas trompés, vous devriez pouvoir ouvrir une connexion SSH avec simplement :

```
ssh votre-login-lyon1@b710l0202.univ-lyon1.fr
```

Pour pouvoir vous connecter à votre VM à partir de son adresse IP, il faudra ajouter l'adresse IP de la VM à la liste dans la ligne `Host`.

La commande devrait vous demander deux fois votre mot de passe : une fois pour `linuxetu` et une autre pour `b710l0202.univ-lyon1.fr`.

I.3 Tunnel SSH

Mise en place du tunnel ssh (à lancer depuis votre machine de travail, remplacer `IP_VM` par l'adresse IP de votre machine virtuelle et `login` par votre login Lyon 1) :

```
ssh -L 20002:b710l0202.univ-lyon1.fr:22 votre-login-lyon1@linuxetu.univ-lyon1.fr
```

Dans un autre terminal, pour accéder à la machine :

```
ssh -p 20002 login@localhost
```

Modifier le numéro du port (ici 20002) au besoin !

Plus d'informations ici par exemple :
`https://blog.netapsys.fr/creer-des-tunnels-ssh/`.

I.4 Transfert de fichiers

Les explications qui précèdent vous permettent d'exécuter des commandes à distance sur une machine. Pour transférer des fichiers depuis votre compte UCBL vers votre machine personnelle (et l'inverse), une documentation est disponible ici (FileZilla, rsync, accès web, ...) :

`https://forge.univ-lyon1.fr/dpt-info/docs/-/blob/master/fichiers-distants.md`

II Accès aux machines virtuelles

II.1 [Pas cette année] Créer sa VM

Édition spéciale coronavirus : Cette partie est gardée pour mémoire, mais cette année la VM est déjà créée pour vous par votre serviteur. L'adresse IP de votre VM est disponible dans TOMUSS, colonne IP_VM. Dans toutes les commandes qui suivent, IP_VM est à remplacer par l'adresse IP en question.

Pour cela, vous devez vous connecter à l'interface d'administration :
`http://cloud-info.univ-lyon1.fr/` (inaccessible depuis l'extérieur du réseau de l'université) et utiliser le domaine `UNIV-LYON1.fr`, avec vos identifiants habituels Lyon 1. Vous faites partie du projet `OpenStack` ASR7 (si besoin, sélectionnez-le depuis le menu déroulant en haut de l'écran. Vous devez bien faire attention à ne pas créer trop de machines, ni utiliser trop de ressources.

Remarque : L'URL précédente, tout comme les VMs que vous pourrez générer, n'est pas accessible depuis *l'extérieur* de l'Université, sauf en passant par une machine relais, c'est-à-dire un proxy. Des explications pour vous y connecter sont disponibles en section I.3.

Vous devez créer une instance de votre machine grâce à `Projet->Calcul->Instances->Lancer une instance`. Faites attention à :

— donner un nom reconnaissable pour votre instance ;

— utiliser l'image `ASR7` ;

— utiliser le gabarit (ensemble de ressources) `m1.tiny` ;

— ne créer qu'une seule instance ;

— Lancer la création.

La machine virtuelle va être créée (cela demande un peu de temps) et apparaître dans la liste des Instances. Elle obtiendra une **adresse IP** que vous noterez. Nous allons l'utiliser plus loin, sous le nom `VM_IP`. **Attendez** que l'`État de l'alimentation` soit `En fonctionnement` avant de pouvoir vous y connecter.

II.2 Se connecter à sa VM

Édition spéciale coronavirus : vous pouvez reprendre la lecture ici.

On peut noter que la VM n'a qu'un CPU (VCPU, pour Virtual CPU), même si elle tourne en réalité sur un serveur physique qui en a beaucoup plus que ça. Tout se passera donc comme si nous étions sur une machine mono-cœur, ce qui simplifie un peu les choses en terme d'ordonnancement.

II.3 S'y connecter

Q.II.1) - Connectez-vous en utilisant l'utilisateur chaprot. Depuis l'intérieur de l'université, on peut utiliser SSH normalement comme ceci :

```
ssh chaprot@VM_IP
```

Si vous faites le TP de l'extérieur de l'université, alors malheureusement la machine virtuelle n'est pas accessible directement. Sinon, utilisez un rebond SSH sur `linuxetu` avec l'une des méthodes indiquées ci-dessus (en utilisant l'IP de votre VM comme nom de machine au lieu de b710l0202.univ-lyon1.fr, et le login `chaprot` au lieu de votre login Lyon 1). Le mot de passe par défaut vous sera donné par votre enseignant.

Par exemple, la connexion avec rebond SSH manuel ressemble à ceci :

```
moy@moylip:~$$ ssh p1234567@linuxetu.univ-lyon1.fr
p1234567@linuxetu.univ-lyon1.fr's password: <mot-de-passe-Lyon1>
[...]
matthieu.moy@linuxetu:~$ ssh chaprot@IP_VM
chaprot@IP_VM's password: <mot-de-passe-de-chaprot>
[...]
chaprot@vm-26:~$
```

Q.II.2) - La première chose à faire est de **changer le mot de passe**, avec la commande `passwd`. Attention à ne pas l'oublier, vous en aurez besoin au prochain TP. Sauf autre configuration particulière, vous seul aurez accès à votre VM.

Q.II.3) - La commande SSH vous a donné un accès shell, en mode texte. Les outils graphiques dont vous avez probablement l'habitude ne sont donc pas disponibles. La commande SSH permet en général d'utiliser des commandes graphiques si on utilise `ssh -X` pour se connecter au serveur, mais cela ne marche pas sur linuxetu. Profitons-en pour nous familiariser avec des éditeurs de textes en mode texte : `nano` est un éditeur peu avancé mais simple à utiliser. `emacs` et `vim` sont deux éditeurs très avancées mais demandant une phase d'apprentissage relativement longue. Dans les trois cas, vous pouvez ouvrir un fichier simplement en appelant l'éditeur avec le nom du fichier en paramètre sur la ligne de commande, par exemple `nano lancement.cpp`.

Q.II.4) - Récupérez maintenant le fichier `lancement.cpp` sur votre VM. Une manière de faire est d'utiliser la commande `wget` (qui télécharge un fichier) depuis la VM :

```
wget https://asr-lyon1.gitlabpages.inria.fr/prog-concurrente/tp1/lancement.cpp
```

Une autre est de télécharger le fichier sur votre PC physique, puis de l'envoyer à la VM avec une commande comme :

```
rsync -av lancement.cpp chaprot@VM_IP:
```

(malheureusement, cette commande ne fonctionnera pas directement depuis l'extérieur de l'université sans VPN ou ajout dans le `~/.ssh/config` comme indiqué ci-dessus)

Q.II.5) - Vous pouvez passer root via la commande `sudo su`. Il est conseillé de travailler comme utilisateur normal (`chaprot`), et de ne passer root que quand c'est nécessaire, c'est à dire pour lancer l'exécutable de votre TP pour le cas qui nous intéresse. Par exemple :

```
$ g++ -std=c++11 -pthread lancement.cpp  -o lancement
$ sudo ./lancement
$ nano lancement.cpp
```

Les commandes `g++` et `nano` s'exécuteront avec l'utilisateur `chaprot`, alors que `./lancement` s'exécutera avec l'utilisateur `root`.

Attention : la VM que vous avez créée pourra être supprimée sans avertissement. En fin de TP, récupérez vos données sur votre compte (par exemple avec la commande `rsync`).

III Premiers pas avec les threads

Vous devez paralléliser un petit programme : `lancement.cpp` disponible sur le site de l'UE. Ce dernier répète un certain nombre de fois un calcul qui n'a pas d'importance. Le but de l'exercice est de savoir si vous pouvez paralléliser les appels à la fonction `fct()`, récupérer le résultat (qui est le temps écoulé pendant le calcul d'après la fonction) et comparer ce temps avec le temps réellement écoulé.

Pour cela :

Q.III.1) - Dans la fonction `main()` remplacez la boucle qui appelle `nbthreads` fois la fonction `fct()` par le lancement de `nbthreads` threads qui exécutent la fonction.

Q.III.2) - Si cela ne vous est pas déjà arrivé, faites une erreur d'argument entre le lancement du thread et la fonction afin de vous habituer à lire les messages d'erreur de compilation. Par exemple, essayez un passage par référence en oubliant `std::ref()` au moment de l'appel : le message d'erreur est rarement convivial !

Q.III.3) - Normalement, dès qu'il y a un nombre de threads important, vous pouvez observer que les messages qu'ils affichent se mélangent. Sachant que les fonctions système sont prévues pour fonctionner en environnement multithread, pourquoi y a-t-il ce mélange et comment l'éviter ?

Q.III.4) - À la fin des threads obtenez le résultat de la fonction et affichez le temps que chaque thread pense avoir mis pour s'exécuter ainsi que la somme de ces valeurs.

La commande `time` permet de voir les différents temps utilisés par une commande :

```
1  $ time ./lancement.exxb 20
2  Th principal : lancement de 10 fois la fonction
3  0 lancement de la fonction pour 5 itérations
4  ...
5  Th principal : Le temps total de calcul est 9.87181
6
7  real    0m9.876s
8  user    0m9.863s
9  sys     0m0.002s
```

- *real* est le temps écoulé pendant le calcul.
- *user* est la somme des temps processeur utilisés par les threads en mode utilisateur (pour faire les calculs eux-mêmes).
- *sys* est la somme des temps processeur passés en mode noyau (pour faire les opérations système comme les affichages).

Q.III.5) - Faites plusieurs essais avec 2, 3, 4, 8, 10, 20 threads.

5(a) - Pourquoi le temps utilisateur est-il supérieur au temps réel ?

5(b) - Le somme des temps que vous calculez est-elle liée au temps réel ? Au temps utilisateur ? Au temps système ?

5(c) - Est-elle identique, et s'il y a une différence pourquoi ?

Pour la dernière question il faut comparer cela avec le nombre de processeurs/cores de l'ordinateur (`cat /proc/cpuinfo`).

TP à terminer en dehors des séances si besoin. Des éléments de corrections sont disponibles sur la page du cours pour vous aider.