Master 1 - Informatique
Systèmes Distribués

Éric Leclercq

Révision : 19 septembre 2022

Résumé

Ce document contient l'ensemble des exercices de travaux dirigés du module de systèmes distribués. Tous les exercices ne seront pas forcement traités durant les séances de travaux dirigés, certains seront prolongés en TP. Le fascicule de TD en ligne sera mis à jour assez régulièrement ainsi que le GitHub qui contient des squelettes de programme (https://github.com/EricLeclercq).

Table des matières

1 Notion Générales

Exercice 1. Lois d'Amdahl et de Gustafson

1. Représenter sur un graphique l'évolution du speedup $S(n)$ en fonction du nombre de processeurs pour différentes valeurs de f (en pourcentage : 0,5,10, 15,20).
2. Représenter sur un graphique l'évolution du scaling speedup $S_s(n)$ en fonction de la portion séquentielle (au moins pour les valeurs 0.2, 0.4, 0.6, 1.0) pour $n = 8$, $n = 16$, $n = 32$, $n = 64$.
3. Pour un programme donné, comment exploiter ces deux informations pour avoir une idée de sa qualité parallèle ?

Exercice 2. Architecture et limite de la lumière

On considère un système comportant 1 000 unité de calcul chacune ayant une puissance de 20 MIPS (20 millions d'instructions par seconde).

— Pour un processeur multicœurs quelle est la durée d'exécution d'une instruction ?
— On suppose que le processeur mesure 1,5cm, quelle est la durée pour parcourir cette distance à la vitesse de la lumière ?
— Qu'en est il pour 1 000 ordinateurs personnels actuels ayant une puissance de 6 000 GigaFLOPS (Floatting Point Operations Per Second) ? Consulter ensuite le top 500 [1], regarder la liste des super calculateurs, leur puissance, le type de processeur ainsi que le pays.

Exercice 3. Protocoles et transparence

1. Faire un schéma du fonctionnement du protocole HTTP incluant la partie serveur, les ressources nécessaires et la partie client.
2. Étudier les différents types de transparence par rapport au protocole.
3. Reprendre les deux questions ci-dessus pour le protocole NFS.

Exercice 4. Prise en charge des clients dans le modèle Client serveur

Rappeler les 3 stratégies vues en cours pour prendre en charge les demandes des clients dans une le modèle client serveur, pour chacune, écrire en pseudo code l'algorithme qui permet la communication du client et du serveur.

2 Modèles d'interactions

Exercice 5. Pattern bag of tasks et équilibrage de charge

1. Avec le langage Java créer une méthode qui renvoie `true` si un nombre passé en paramètre est premier, `false` sinon. Utiliser une algorithme qui teste les diviseurs jusqu'à la racine carrée du nombre donné. Énumérer les nombres premiers jusqu'à N, déterminer N pour avoir une exécution séquentielle de plusieurs minutes.
2. Utiliser plusieurs threads pour répartir le travail en appliquant le pattern *bag of task* : chaque thread vient chercher une tâche (un nombre à tester) dans un objet de la classe `BagOfTasks` en invoquant la méthode `getNext()`. Lorsqu'un thread a terminé le test de primalité d'une valeur, il va en rechercher une autre. Le distributeur

1. http://top500.org

de tâche peut retourner `-1` lorsque le travail global est terminé, c'est-à-dire lorsque les N nombres ont été testés. Cette méthode à l'avantage de répartir équitablement le travail entre tous les threads.

(a) Réaliser le programme.

(b) Mesurer le temps d'exécution pour un thread.

(c) En fonction du type de processeur du nombre de cœurs réels, lancer le programme avec n thread, mesurer le temps d'exécution et évaluer le speedup.

(d) Mesurer le speedup avec $n + 1$ threads (si n est le nombre de cœurs réels du processeur).

3. Généraliser le pattern *bag of task* pour l'utiliser avec plusieurs machines multi-cœurs [2]. Dans ce programme, utilisez des socket TCP ou UDP et ne pas utiliser Java RMI.

(a) Réaliser le programme avec des sockets UDP et des sockets TCP.

(b) Le tester avec une machine cliente et un serveur *bag of task* pour mesurer le temps d'éxecution et en déduire le coût induit par les communications et ainsi estimer la fraction de code non parallélisable.

(c) En utilisant ssh lancer le programme sur plusieurs machines. Évaluer le speedup réel et le speedup théorique. Évaluer à nouveau le speedup. Que peut-on dire sur l'impact du temps d'acheminement des valeurs entre les différentes machines.

(d) Comparer l'implémentation UDP et TCP.

(e) Quel sont les inconvénients de l'implémentation en TCP ? Utiliser les sélecteurs d'entrée-sortie pour améliorer cette version.

Exercice 6. Pattern bag of tasks généralisé

Cet exercice est la suite du précédent. Au lieux de distribuer une valeur, le *bag of task* distribue un objet qui possède une méthode `run()`. Ainsi, des clients peuvent soumettre n'importe qu'elle tâche dans le *bag of task* et un certain nombre de *workers* viennent chercher des objets tâches et exécutent la méthode `run()`.

1. Quelles classes sont nécessaires ?
2. Quel mécanisme de Java est essentiel pour implanter un *bag of task* générique ?
3. Comment retourner les valeurs à l'issue de l'exécution du `run()`, les transmettre au client ?
4. Implanter le programme

3 Topologies et architectures

Exercice 7. Topologie, plongement

1. Représenter un hypercube de dimension 3 puis 4, numéroter les nœuds en utilisant un code Gray.
2. Plonger un maillage de 16 nœuds dans un hypercube.
3. Plonger un arbre binaire de 25 éléments dans un maillage 8×8

2. en distribué, ce pattern se nomme Manager/Worker

4. Comment effectuer un *broadcast* sur un hypercube de dimension 3, sur un maillage ou un arbre binaire de profondeur 3 ?

Exercice 8. Routage

Soit 3 nœuds, A, B et C, un message composé de 4 paquets w,x,y,z doit être acheminé de A à C.

1. Représenter avec un tableau l'acheminement du message dans le cas des algorithmes *store-and-forward* et *cut-through*.
2. Conclure sur le temps d'acheminement, la taille des buffers utilisés.

4 Communication par passage de messages

Pour cette série d'exercices, en plus du cours, vous pouvez utiliser les ressources suivantes :

— https://www.mpich.org/documentation/guides/
— https://www.open-mpi.org/
— https://researchcomputing.princeton.edu/education/external-online-resources/mpi

Exercice 9. MPI, mise en route

1. Créer vos clés ssh si ce n'est pas déjà fait (se reporter à la dernière section du document si besoin). Avant d'utiliser plusieurs nœuds se connecter sur chacun avec ssh, vérifier qu'ils ne demandent pas de mot de passe (sinon les clés ssh ont mal été générées ou le fichier `authorized_keys` n'est pas bien renseigné.
2. Remplir un fichier contenant le nom des nœuds pour exécuter MPI sur ces différentes machines c'est-à-dire constituer le cluster qui servira lors du lancement de l'exécution (encadré 1). Remplir un fichier que vous pouvez nommer `myhost` qui contient la liste des nœuds sous le forme `MI103-04.iem slots=4` qui signifie que 4 coeurs sont alloué au nœud pour accepter des tâches MPI. La première ligne est la machine sur laquelle vous lancez `mpirun` [3].
3. Écrire et compiler le programme exemple du cours, l'essayer sur plusieurs nœuds (utiliser les commandes `mpicc` et `mpirun`).
4. Écrire un programme ping-pong qui effectue un échange de message n-fois entre deux nœuds. Tester différents modes d'envoi et réception. Mesurer le temps d'acheminement en local et entre 2 machines.

Exercice 10. MPI, communication en anneau

Réaliser un programme MPI qui pour n nœuds fait passer un message de l'un à l'autre, le dernier repasse le message au premier. Chaque nœud incrémente de 1 l'entier contenu dans le message. Ceci permet de constituer une communication selon une topologie d'anneau.

Exercice 11. MPI, somme partielle

3. `mpirun` encapsule le lancement des différentes tâches, il est similaire au scrit que nous avons écrit lors d'un TD précédent pour lancer les worker du bag of tasks (cf encadré 1). En MPI, ce genre de script n'est plus nécessaire.

1. Réaliser un programme MPI utilisant n nœuds qui calculent la somme partielle d'un tableau. Le tableau est réparti entre les $n-1$ nœuds à partir du nœud maître.
2. Réaliser le même programme en utilisant les fonctions `scatter` et `gather`.
3. Améliorer le programme précédent avec la fonction `reduce`.

Exercice 12. MPI, calcul de Pi

Une approximation de π peut calculée de différentes manière, la méthode de Monte Carlo est le suivante :

- définir un cercle de rayon r dans un carré dont le coté est de rayon $2r$
- l'aire du cercle est πr^2, celle du carré est $4r^2$
- le ration entre l'aire du cercle et l'aire du carré est : $\frac{\pi r^2}{4r^2} = \frac{\pi}{4}$
- si on génère N point dans le carré alors $M = N \times \frac{\pi}{4}$ de ces points seront dans le cercle
- ce qui permet d'approximer $\pi = \frac{4 \times M}{N}$

Ce qui conduit à l'algorithme séquentiel suivant exprimé en pseudo-code :

```
npoints = 10000
circle_count = 0
do j = 1,npoints
  generate 2 random numbers between 0 and 1
  xcoordinate = random1
  ycoordinate = random2
  if (xcoordinate, ycoordinate) inside circle
    then circle_count = circle_count + 1
end do
PI = 4.0*circle_count/npoints
```

1. Réaliser un programme séquentiel, régler N pour obtenir une exécution significative
2. Répartir le nombre de test N sur différents nœuds afin d'exploiter le parallélisme, le nœud maître doit rassembler les résultats et afficher l'approximation de π.

5 Paradigme Map-Reduce

Exercice 13. Cas d'utilisation des opérateurs algébriques

Pour les deux cas présentés ci dessous, l'objectif est définir ce que feront les opérations map et reduce.

1. On considère une table qui contient les liens établis par les URL (de la source vers la cible). Déterminer quelle expression algébrique permet de trouver les chemins de longueur 2 entre deux URLs.
2. On suppose que l'on dispose d'une table `Amis` qui recense tous les amis (attribut `FreindName` des utilisateurs spéficié par l'attribut `UserName`. Quelle expression permet de calculer le nombre d'amis d'une personne. Il existe en algèbre relationnelle un opération γ qui permet d'effectuer un regroupement et des opérations sur les groupes.

Exercice 14. Opérateurs algébriques et expression map-reduce

Pour chaque opération de l'algèbre relationnelle, définir les tâches map et reduce correspondante. Préciser la forme de la sortie du processus.

— $\sigma_C(R)$ où C est une expression conditionnelle
— $\pi(exp)R$, pensez à l'élimination des doubles
— $R \cup S$ et $R \cap S$
— $R - S$ rappeler la définition du résultat. Indice, garder l'information sur la source.
— $R \bowtie S$ pour deux relations $R(A, B)$ et $S(B, C)$

Exercice 15. Utilisation d'Hadoop

L'objectif de ce premier exercice est de configurer Hadoop en version autonome sur un seul nœud puis de compiler, exécuter quelques exemples.

1. Télécharger Hadoop sur le site officiel : https://hadoop.apache.org/releases.html prendre la version binaire 2.10.
2. Le décompresser dans un répertoire dédié, se reporter à la page suivante pour les opérations de configuration : https://hadoop.apache.org/docs/r2.10.1/hadoop-project-dist/hadoop-common/SingleCluster.html.
   — Lors de l'étape de configuration, attention à bien fixer les variables pour que le JDK pris en compte soit le 1.8. Vérifier aussi que la variable `JAVA_HOME` est bien positionnée dans le fichier `etc/hadoop/hadoop-env.sh`.
   — Modifier, selon les instructions de la documentation, les fichiers `core-site.xml` et `hdfs-site.xml`, notez que le facteur de réplication est fixé à 1 car nous réalisons une installation sur un seul nœud.
   — Formater le système de fichier, démarrer le gestionnaire du système de fichiers et l'ordonnanceur `yarn`.
   — Au moyen des commandes spécifiques, créer quelques répertoires et y déposer des fichiers.
3. Connectez vous sur l'interface Web et identifier les différents composants.
4. Où sont stockés les blocks constituants les fichier hébergés par HDFS ? Pour changer le répertoire de stockage, consulter l'encadré en fin de document.
5. À l'aide de la documentation pour les développeurs [4] compiler et lancer votre premier exemple (`WordCount`).
   — À l'issue de la compilation observer la génération des classes pour les mapper et les reducer. Qui effectue l'instanciation de ces objets ?
   — Créer un package avec ces éléments.
   — Lancer l'exécution et observer les informations fournies ensuite sur l'interface Web de yarn.
6. Utiliser un fichier plus important, comme par exemple des évaluation de films (https://grouplens.org/datasets/movielens/) et écrire un programme pour calculer le top-5 des films les plus populaires (les plus évalués).
7. Calculer sur ce jeu de données réaliser les programmes pour calculer :
   — le nombre moyen d'évaluations des films, la médiane, l'écart-type
   — le nombre moyen d'évaluations par utilisateur, la médiane, l'écart-type

4. https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

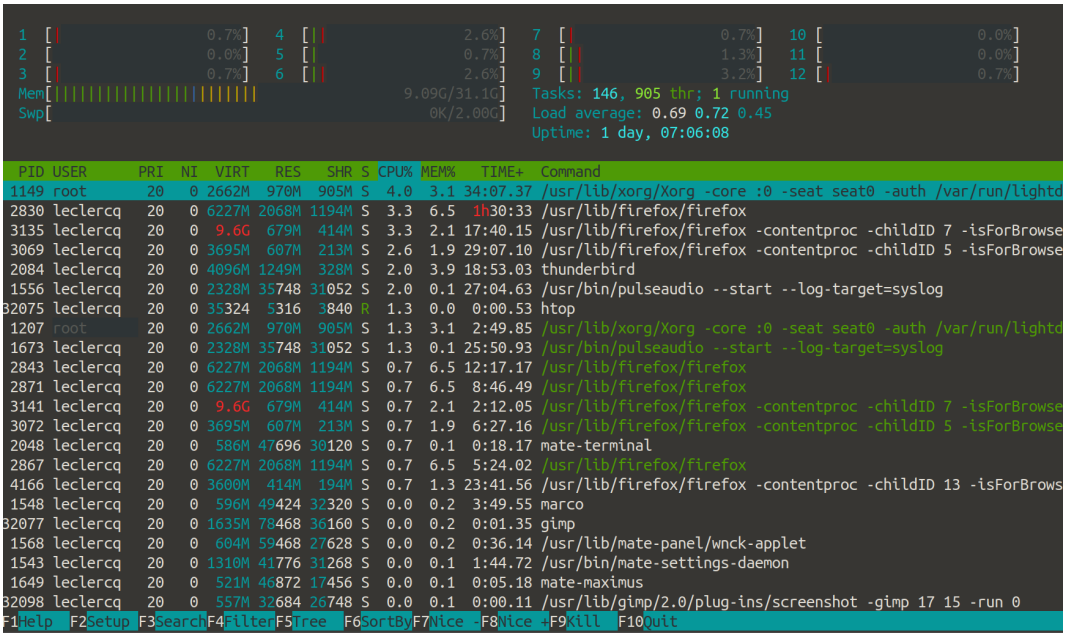

FIGURE 1 – Informations affichées par htop

6 Outils utiles pour les TD et les TP

6.1 Observer les exécutions, mesurer le temps

Pour observer les exécutions d'un programme sur une machine vous pouvez utiliser la commande `htop`. La figure 1 présente les informations affichées par la commande `htop`. Les premières lignes donnent l'activité des cœurs du CPU, les barres vertes correspondent à une exécution de programme utilisateur, les barres rouge à du code exécuté en mode noyau. Les trois lignes de droite donnent les information sur le nombre de processus et de thread et le nombre de processus s'exécutant dans les cœurs. La ligne suivante donne la charge moyenne à 1, 5 et 15 minutes.

Pour pouvoir interpréter correctement les informations sur le nombre de cœurs, il faut connaître les spécification du processeur. Avec `cat /proc/cpuinfo` vous obtiendrez la description du ou des processeurs installés, utiliser ensuite la référence sur un moteur de recherche pour accéder à la fiche technique du site du constructeur.

Pour mesurer le temps d'exécution d'un programme utiliser la commande `time`. Il existe deux manières de la lancer :

— générique `time commande`
— spécifique à Java ou aux langages interprétés : `time java monprog`

6.2 Se connecter avec SSH en utilisant les clés

Avec ssh et son implémentation OpenSSH, il est possible de se connecter sur des machines distantes en utilisant la clé publique déposée dans le répertoire `.ssh` de la machien distante. Dans le cas des salles de TP il s'agit de votre *home* qui est partagé sur tous les postes.

La clé privé permet de prouver votre identité et donc d'éviter à avoir à saisir un mot de passe. De plus ssh permet d'exécuter directement une commande à la connexion, il suffit de la passer en paramètre. Il est donc possible et plutôt facile de lancer une commande sur

un ensemble de machines et ainsi d'administrer un cluster ou de lancer des programmes qui vont s'exécuter en parallèle et coopérer.

ENCADRÉ 1 – Configurer les clés ssh pour se connecter sur différentes machines

Créer les clés SSH sur une de machines :

```
# attention les clés et les authorisation sont dans le
# répertoire .ssh (et doivent rester là
cd $HOME/.ssh
ssh-keygen -t rsa
# ne pas mettre de passphrase
# les clés sont générées dans le répertoire .ssh de votre home
# la clé publique a l'extension .pub
cat id_ras.pub > authorized_keys
# le fichier authorized_keys contient la liste des clés
# autorisées a se connecter sans password
# tester avec la commande suite :
ssh MI104-04
# à la première connexion, il faut valider la source
```

Le principe de connexion a distance sans mot de passe en utilisant les clés ssh permet de lancer des tâches en séquence ou en parallèle sur plusieurs machines.

ENCADRÉ 2 – Comment lancer les mêmes commandes sur différents serveurs

Pour lancer des exécutions sur différentes machines on utilise un script shell qui exploite un fichier texte contenant l'ensemble des machines sur lesquelles on veut effectuer la commande passée en paramètre

```
for i in `cat myhost.txt`
do
 ssh $i  "ma␣commande"
done
```

Le fichier `myhost.txt` contenant la liste des machines est de la forme :

```
MI104-01
MI104-02
MI104-03
```

6.3 Configuration d'un cluster MPI simple

ENCADRÉ 3 – Configurer le liste des machines pour un cluster MPI

Pour constituer la liste des machines d'un cluster MPI on reprend le même format de fichier que celui de l'encadré précédent en ajoutant pour chaque nœud le nombre de cœurs disponibles séparé par un espace du nom de la machine. Par exemple `MI105-01 slots=6`.

Pour déterminer le nombre de cœurs utilisables il faut se référer à la fiche technique du processeur : `cat /proc/cpuinfo` puis reprendre le nom du processeur et faire une recherche sur le site Intel par exemple (`Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz`).

ENCADRÉ 4 – Modification du répertoire de stockage d'HDFS

Ficher `hdfs-site.xml`

```
<property>
 <name>dfs.replication</name>
 <value>1</value>
</property>

<property>
 <name>dfs.namenode.name.dir</name>
 <value>file:/home/hadoop_tmp/hdfs/namenode</value>
</property>

<property>
 <name>dfs.datanode.data.dir</name>
 <value>file:/home/hadoop_tmp/hdfs/datanode</value>
</property>
```