Cluster de transcodage FFMPEG distribué

Table of Contents

- Cluster de transcodage FFMPEG distribué
- Installation et configuration
  - Installation des noeuds
    - Méthode 1: Installation de ClusterHat
    - Méthode 2: Gadget Ethernet
  - Configuration des noeuds
    - Noeud de stockage
    - Configuration du controleur
    - Configuration des noeuds de calcul
    - Test
- Script dffmpeg.sh
  - Code source
  - Utilisation du script
  - Problèmes connus (à résoudre)

À partir de 2014, ffmpeg utilise un nombre optimal de threads pour améliorer les performances de transcodage:

1. threads 0 (optimal);
2. threads 1 (thread unique);
3. threads 2 (2 threads pour un processeur Intel Core 2 Duo, par exemple);
4. none (la valeur par défaut, également optimale).

Cela dépend du codec utilisé, de la version de ffmpeg et du nombre de cœurs du processeur. Parfois, il s'agit simplement d'un thread par noyau. Parfois, c'est plus complexe comme avec libx264, les cœurs x 1,5 pour les threads de trame et les cœurs x 1 pour les threads de tranches.

On peut le vérifier sur un ordinateur multicœur en examinant la charge du processeur (commande `top` sous Linux) avec différentes options de ffmpeg

Cependant le transcodage video peut être long, le script **dffmpeg.sh** présenté dans ce tutoriel permet d'utiliser un stockage distribué (NFS) et les nœuds de calculs d'un cluster pi pour le transcodage.

Le nœud de contrôle reçoit la requête, transfère le fichier vers le stockage partagé, calcule le nombre total d'images vidéo, puis envoie la commande au nœud de calcul. Différents nœuds traitent certaines images continues en fonction de leurs poids respectifs (réglage manuel, la fonction de test de performance peut être ajoutée) et la sortie Dans le stockage partagé, une fois que tous les nœuds ont été transcodés, le nœud de stockage sera fusionné et les fichiers temporaires partagés seront nettoyés, et enfin le nœud de contrôle renverra la vidéo transcodée reliée.

Installation et configuration

Il existe trois types de rôles de nœud de cluster : contrôle (un), calcul (au moins un) et stockage (un). Chaque nœud peut déployer un ou plusieurs rôles.

Dans l'architecture étudiée les rôles sont attribués comme suit:

Nœud	**Caractéristiques**	**adresse IP**
stockage		
contrôleur	Raspberry Pi 400	10.1.1.172
calcul	Banana Pi zero	10.1.1.3, 10.1.1.4, 10.1.1.5

Installation des noeuds

Méthode 1: Installation de ClusterHat

Télécharger une image Raspbian pour chaque Pi (Controller/P1/P2/P3/P4) sur le site ClusterHat, ensuite utiliser l'**imager** pour graver sur la carte TF, puis créer un fichier vide nommé **ssh** dans la partition de démarrage. On peut maintenant le brancher sur le Raspberry Pi et l'allumer.

Une fois le Raspberry Pi allumé, on peut voir l'adresse IP du Raspberry Pi sur la page du routeur, puis se connecter en ssh. Le nom d'utilisateur est pi et le mot de passe est **raspberry**.

Deskpi Pro est un kit matériel pour convertir une Raspberry PI 4 standard à partir d'un SBC nu, avec un stockage limité, en un mini PC complet avec un bouton d'alimentation, un refroidissement, de meilleurs ports et un Mass Storage via SATA USB3 ou M.2.
Pour l'installer sur Raspbian 64bit il faut installer git en premier `apt-get update && apt-get -y install git` puis procéder ainsi:

```
cd ~
git clone https://github.com/DeskPi-Team/deskpi.git
cd ~/deskpi/
./install.sh
```

Méthode 2: Gadget Ethernet

Le gadget Ethernet est un peu plus difficile à configurer, mais il est beaucoup plus puissant car on peut utiliser d'autres OS. Fondamentalement, on aura la possibilité de se connecter à la console ainsi que tout ce qu'on peut faire sur une connexion réseau

- **Étape 0** Télécharger et installer la dernière version de Raspbian
- **Étape 1**: Configuration du nœud principal (mot de passe par défaut est raspberry)

```
ssh pi@192.168.2.18
sudo rpi-update
sudo reboot
sudo hostname pi0
# changer le nom d'hôte dans /etc/hostname
sudo vi /etc/hostname
# changer 'controle' □□en pi0 dans /etc/hosts
sudo vi /etc/hosts
# ajouter une configuration eth0 tout en bas du fichier /etc/dhcpcd.conf
#   interface eth0
#   static ip_address=10.0.0.1/8
#   static domain_name_servers=1.1.1.1,208.67.222.222
#   nolink
sudo vi /etc/dhcpcd.conf
# Configurer iptables pour permettre aux appareils sur le réseau 10.0.0.0/8
# d'accéder à Internet via l'interface eth0.
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o eth0 -j ACCEPT
# Pour rendre le partage internet permanent:
# - supprimer le #  devant la ligne #net.ipv4.ip_forward=1 dans le fichier
/etc/sysctl.conf
# - exécuter les commandes suivantes pour que les modifications d'iptables
se chargent à chaque fois :
sudo apt-get install iptables-persistent
# - Générer des clés ssh :
ssh-keygen -t rsa # accepter la valeur par défaut pour "Enter file in which
to save the key " et ne pas définir de phrase secrète
# Si le controleur est est aussi un nœud de calcul, il faut également copier
la clé publique
# en utilisant la commande:
ssh-copy-id -i ~/.ssh/id_rsa.pub 10.1.1.172
```

- **Étape 2**: Installer clusterctl

```

# Installer les dépendances :
sudo apt install python-pip python-gpiozero git
# Télécharger les fichiers **clusterctl**:
sudo git clone https://github.com/thagrol/clusterctl
# ou alors:
sudo wget
https://raw.githubusercontent.com/thagrol/clusterctl/master/clusterctl.py
sudo chmod a+x clusterctl.py
wget
https://raw.githubusercontent.com/thagrol/clusterctl/master/gpiosetup.sh
chmod a+x gpiosetup.sh
# Configurer **gpiosetup.sh** pour qu'il s'exécute à chaque démarrage en
ajoutant
# "/path/to/gpiosetup.sh &" immédiatement au-dessus de la ligne "exit(0)"
dans
# le fichier /etc/rc.local
```

```
# Éteindre le maître :
sudo poweroff
# Débrancher l'alimentation.
# - S'il n'est pas déjà connecté, attacher clusterhat et pi zéro(s).
# - Rebrancher l'alimentation et démarrer le maître.
# - Mettre le compute node sous tension :
# /path/to/clusterctl.py -n 1 on
```

- **Étape 3**: configuration des computes nodes

```

# Après avoir graver l'image sur la carte TF, mais avant de l'extraire
# Pour activer le device ethernet de Gadget USB (USB OTG),
# Sur Raspbian:
#  - ajouter "dtoverlay=dwc2" comme dernière ligne dans le fichier
`config.txt
#  - ajouter " modules-load=dwc2,g_ether" (avec une espace devant) dans le
#    fichier `cmdline.txt`, après **rootwait** (le dernier mot sur la
première ligne)
# Sur FreeBSD, ajouter ces lignes à /boot/loader.conf:
#   if_cdce_load="YES"
#   hw.usb.template=1
# Ajouter un fichier `ssh` à la racine de la partition boot(vide, c’est sans
importance).
# placer les cartes SD dans leur Raspberry Pis respectif.
ssh pi@raspberrypi.local
sudo rpi-update
sudo reboot
# changer le nom d'hôte des Pis:
sudo hostname pi1
sudo vi /etc/hostname # changer le nom d'hôte dans ce fichier
sudo vi /etc/hosts # changer 'contrôleur' □□en pi1
# ajouter une configuration eth0 tout en bas du fichier /etc/dhcpcd.conf
#   interface eth0
#   static ip_address=10.0.0.1/8
#   static domain_name_servers=1.1.1.1,208.67.222.222
#   nolink
sudo vi /etc/dhcpcd.conf# - Se connecter via ssh :
ssh p1.local
# Générer des clés ssh :
ssh-keygen -t rsa # accepter la valeur par défaut pour "Enter file in which
to save the key " et ne pas définir de phrase secrète
# Copier la clé publique du maître :
scp pi@controller.local:/home/pi/.ssh/idras.pub ~/.ssh/authorized_keys
# Se déconnecter, et de retour sur le contrôleur (maître), mettre l'esclave
hors tension :
/path/to/clusterctl.py -n 1 stop
```

Configuration des noeuds

Noeud de stockage

Dans un cluster, le stockage partagé est très important, il faut donc d'abord configurer le stockage:

```
# Installer NFS
sudo apt-get install nfs-kernel-server
sudo chmod 777 /var/lib/nfs/.etab.lock
sudo chmod 777 /var/lib/nfs
# Créer de nouveaux dossiers partagés pour les fichiers téléchargés avant le rendu, les fichiers fragmentés après le rendu et les fichiers complets après le rendu.
sudo mkdir -p /srv/distributed_ffmpeg_transcoding_shared_files
sudo chmod 777 /srv/distributed_ffmpeg_transcoding_shared_files
mkdir /srv/distributed_ffmpeg_transcoding_shared_files/upload
mkdir /srv/distributed_ffmpeg_transcoding_shared_files/tmp
mkdir /srv/distributed_ffmpeg_transcoding_shared_files/download
```

Modifier le fichier /etc/exports pour partager les répertoires:

- **upload**: le nœud de contrôle a des autorisations de lecture et d'écriture et le nœud de calcul a des autorisations de lecture seule
- **tmp**: le nœud de calcul a des autorisations de lecture et d'écriture
- **download** le nœud de calcul de stockage a des autorisations de lecture et d'écriture (puisqu'il n'y a qu'un seul nœud de stockage, il n'est pas nécessaire de partager)

```
# /etc/exports : la liste de contrôle d'accès pour les systèmes de fichiers qui peuvent être exportés
# aux clients NFS. Voir exports(5).
#
# Exemple pour NFSv2 et NFSv3 :
# /srv/homes hostname1(rw,sync,no_subtree_check)
hostname2(ro,sync,no_subtree_check)
#
# Exemple pour NFSv4 :
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/srv/distributed_ffmpeg_transcoding_shared_files/upload
10.1.1.3(ro,insecure)
/srv/distributed_ffmpeg_transcoding_shared_files/tmp 10.1.1.3(rw,insecure)
```

Il faut faire particulièrement attention à l'utilisation de **insecure**, sinon il échouera à monter et affichera l'accès refusé.

10.1.1.3 est l'adresse IP d'un nœud de calcul.
Quand on a trois nœuds de calcul, (IP 10.1.1.3, 10.1.1.4, 10.1.1.5), cela peut s'écrire comme suit :

```
/srv/distributed_ffmpeg_transcoding_shared_files/upload
10.1.1.3(ro,insecure) 10.1.1.4(ro,insecure) 10.1.1.5(ro,insecure)
/srv/distributed_ffmpeg_transcoding_shared_files/tmp
10.1.1.3(rw,insecure) 10.1.1.4(rw,insecure) 10.1.1.5(rw,insecure)
```

Si le nœud de stockage est également un nœud de calcul, il n'est pas nécessaire d'écrire le nœud de stockage. Parce que le nœud de stockage n'a pas besoin de monter NFS.

Une fois la modification terminée, exécuter la commande suivante pour que la configuration prenne effet :

```
exportfs -arv
```

On peut utiliser `showmount -e` pour afficher les exports.

Configuration du controleur

Télécharger le script de contrôle:

```
wget
https://raw.githubusercontent.com/chn-lee-yumi/distributed_ffmpeg_transcodin
g_cluster/master/dffmpeg.sh
chmod +x dffmpeg.sh
```

Ensuite, nous il faut modifier la configuration du script de contrôle en renseignant:

1. **storage_node**: l'adresse IP du nœud de stockage, ici 10.1.1.172.
2. **compute_node**: l'adresse IP du nœud de calcul, voici 10.1.1.172 et 10.1.1.3.
3. **compute_node_weight**: le poids de chaque nœud de calcul (plus les performances sont bonnes, plus le poids doit être élevé). Ici, on réglé le controleur sur 48 et les noeuds de calcul sur 36.

Il faut également installer le logiciel **bc** [1]:

```
sudo apt-get install bc
```

Configuration des noeuds de calcul

Un noeud de calcul n'a qu'un seul rôle informatique, et il y a très peu de choses à configurer. Il faut seulement installer ffmpeg et monter un stockage partagé:

```
# Installer ffmpeg
# Dans Jessie, il faut ajouter deb http://ftp.debian.org/debian jessie-backports main dans source.list pour trouver ce paquet.
sudo apt-get install ffmpeg
# Création des nouveaux répertoires
sudo mkdir -p /srv/distributed_ffmpeg_transcoding_shared_files
sudo chmod 777 /srv/distributed_ffmpeg_transcoding_shared_files
mkdir /srv/distributed_ffmpeg_transcoding_shared_files/upload
mkdir /srv/distributed_ffmpeg_transcoding_shared_files/tmp
# Montage NFS, seulement un montage temporaire, il faut modifier fstab ou le script de démarrage pour un montage automatique
sudo mount 10.1.1.172:/srv/distributed_ffmpeg_transcoding_shared_files/upload /srv/distributed_ffmpeg_transcoding_shared_files/upload
sudo mount 10.1.1.172:/srv/distributed_ffmpeg_transcoding_shared_files/tmp /srv/distributed_ffmpeg_transcoding_shared_files/tmp
```

10.1.1.172 est l'adresse IP du nœud de stockage, il faut renouveler cette opération sur chacun des noeuds.

Test

Le test est effectué du côté contrôleur. Télécharger d'abord une vidéo de test. On peut utiliser par exemple, celle disponible en ligne :

```
wget https://github.com/chn-lee-yumi/distributed_ffmpeg_transcoding_cluster/blob/master/test_video.mp4?raw=true
```

Le suffixe du nom de fichier après le téléchargement est erroné, le modifier : mv test*video.mp4?raw=true test*video.mp4 Ensuite, on peut tester transcodage FFMpeg distribué au format mpeg4, avec un débit de code vidéo est de 1Mbps.:

```
./dffmpeg.sh test_video.mp4 -c:v mpeg4 -b:v 1M
```

Un fichier mp4 sera produit dans le répertoire `upload`.

Si les serveurs sont répartis dans différents réseaux, le goulot d'étranglement est la vitesse de lecture

et d'écriture NFS, donc la vitesse de transcodage finale sera plus lente. Si le serveur met d'abord en cache le fichier à transcoder, la vitesse de transcodage finale est plus rapide qu'un transcodage de serveur.

Script dffmpeg.sh

Code source

```
#!/bin/bash

# Transcodage FFMPEG distribué v1.3
# Prend en charge n'importe quel format vidéo en MP4
# Utilisation : dffmpeg.sh [fichier_d'entrée] [paramètre_de_sortie ffmpeg]
# Utilisation : dffmpeg.sh test.mp4
# Utilisation : dffmpeg.sh test.mp4 -c:v mpeg4 -b:v 1M
# A FAIRE : Corriger le bug étrange : ffmpeg est toujours en cours
d'exécution mais l'instruction tactile suivante a été exécutée

###Éléments de configuration
storage_node="hk.gcc.ac.cn"
storage_node_ssh_port=22
compute_node=("us.gcc.ac.cn" "hk.gcc.ac.cn" "cn.gcc.ac.cn") # nœud de calcul
compute_node_ssh_port=(10022 22 22) # port ssh du nœud de calcul
compute_node_weight=(10 50 15) # calcule le poids du nœud
nfs_path=/srv/distributed_ffmpeg_transcoding_shared_files #Répertoire
partagé
sync_wait_time=0 # Un paramètre défini pour résoudre l'étrange bug sur le
cluster Raspberry Pi (ffmpeg est toujours en cours d'exécution mais
l'instruction touch a été exécutée). Selon différents poids, différentes
valeurs doivent être définies et une mesure réelle est requise. Un réglage
raisonnable du poids peut réduire cette valeur. La valeur peut être 30.
###Fin de l'élément de configuration

upload_path=$nfs_path/upload
tmp_path=$nfs_path/tmp
download_path=$nfs_path/download
input_file=$1
ffmpeg_output_parameter=${@:2}

# fonction d'affichage, couleur de sortie
ECHO=`which echo`
display(){
    local type=$1
    local msg=${@:2}
    if [[ $type = "[Info]" ]]; then
        $ECHO -e "\\033[1;36;40m[Info] $msg \\033[0m"
    elif [[ $type = "[Error]" ]]; then
```

```
        $ECHO -e "\\033[1;31;40m[Error] $msg \\033[0m"
    elif [[ $type = "[Exec]" ]]; then
        $ECHO -e "\\033[1;33;40m[Exec] $msg \\033[0m"
    elif [[ $type = "[Success]" ]]; then
        $ECHO -e "\\033[1;32;40m[Success] $msg \\033[0m"
    else
        $ECHO -e $@
    fi
}

### Démarrer la surcharge de la fonction
# Recharger cp, enregistrer le journal
CP=`which cp`
cp(){
    local src=$1
    local dst=$2
    display [Exec] cp ${@:1}
    $CP ${@:1}
}
# Recharger rm, enregistrer le journal
RM=`which rm`
rm(){
    display [Exec] rm ${@:1}
    $RM ${@:1}
}
# Recharger ssh, enregistrer le journal
SSH=`which ssh`
ssh(){
    display [Exec] ssh ${@:1}
    $SSH ${@:1}
}
### Surcharge de fonction terminée

# Vérifiez le fichier d'entrée
if [ -f $input_file ]
then
   display [Info] Input: $input_file
   display [Info] FFmpeg output parameter: $ffmpeg_output_parameter
   filename=$(date +%s) # Utiliser l'horodatage comme nom de fichie
   display [Info] Téléchargement du fichier, veuillez patienter un moment.
Nom de fichier temporaire : $filename
   cp $input_file $upload_path/$filename
else
   display [Error] Input error!
   exit
fi

# Calculer le poids total du nœud de calcul
total_weight=0
for i in ${compute_node_weight[*]}
do
```

```
    total_weight=$[$total_weight + $i]
done
display [Info] Compute node total weight: $total_weight

# Tâche de distribution
video_length=$(ffprobe -show_format $input_file -loglevel error| grep
duration | awk -F = '{printf $2}')
part_start=0
part_end=0
node_number=${#compute_node[*]}
# for i in {0..${#compute_node[*]}} # Je ne sais pas pourquoi je ne peux pas
écrire comme ça
for ((i=0; i<$node_number; i++))
do
    # echo ${compute_node[$i]},${compute_node_weight[$i]} # Affiche le nœud
de calcul et son poids
    part_end=$(echo "scale=2; $part_start + $video_length *
${compute_node_weight[$i]} / $total_weight" | bc )
    display [Info] Compute node ["${compute_node[$i]}"] : start[$part_start]
, end[$part_end]
    # ssh ${compute_node[$i]} -p ${compute_node_ssh_port[$i]} "ffmpeg -ss
$part_start -i $upload_path/$filename -to $part_end $ffmpeg_output_parameter
$tmp_path/${filename}_$i.mp4 -loglevel error; touch
$tmp_path/${filename}_$i.txt" & # -ss à l'avant, l'entrée sera analysée à
l'aide d'images clés, ce qui est très rapide. Mais cela peut entraîner une
partie de la vidéo répété.
    ssh ${compute_node[$i]} -p ${compute_node_ssh_port[$i]} "ffmpeg -i
$upload_path/$filename -ss $part_start -to $part_end
$ffmpeg_output_parameter $tmp_path/${filename}_$i.mp4 -loglevel error; touch
$tmp_path/${filename}_$i.txt" & # -ss à l'arrière, la vitesse sera plus
lente, mais cela ne provoquera pas la répétition du clip vidéo
    part_start=$part_end
    echo "file '${filename}_$i.mp4'" >> $tmp_path/${filename}_filelist.txt
done

# Vérifie constamment si la tâche est terminée
sleep 1
display [Info] Checking if the tasks are completed.
while :
do
    for ((i=0; i<$node_number; i++))
    do
        if [ -f $tmp_path/${filename}_$i.txt ]
        then
            if [ $i==$[$node_number - 1] ] # Si tout est terminé
            then
                break 2
            else
                continue
            fi
        else
```

```
            break
        fi
    done
    sleep 1
    display ".\c"
done
display !

# Effectue l'assemblage vidéo
sleep $sync_wait_time
display [Info] Tasks all completed! Start to join them.
ssh $storage_node -p $storage_node_ssh_port "ffmpeg -f concat -i
$tmp_path/${filename}_filelist.txt -c copy $download_path/$filename.mp4 -
loglevel error"

# Efface les fichiers temporaires et les fichiers téléchargés
display [Info] Clean temporary files.
rm -r $tmp_path/${filename}*
rm $upload_path/${filename}

display [Finish] Mission complete! Output path:
[$download_path/$filename.mp4]


# ffmpeg commandes couramment utilisées. Référence
https://www.cnblogs.com/frost-yen/p/5848781.html

#ffprobe -v error -count_frames -select_streams v:0  -show_entries
stream=nb_read_frames -of default=nokey=1:noprint_wrappers=1 test.mp4 #
Calcule le nombre d'images, se reporter à
https://stackoverflow.com/questions/ 2017843/fetch -frame-count-with-ffmpeg

# Pour capturer la vidéo, se reporter à https://trac.ffmpeg.org/wiki/Seeking
#ffmpeg -ss 00:01:00 -i test.mp4 -to 00:02:00 -c copy cut.mp4
#ffmpeg -ss 1 -i test.mp4 -to 2 -c copy cut.mp4
#ffmpeg -ss 1 -i test.mp4 -t 1 -c copy -copyts cut.mp4
# Obtenir la longueur de la vidéo
#ffprobe -show_format test_video.mp4 | grep duration | awk -F = '{printf
$2}' > /tmp/1.txt

# Fusionner la vidéo
https://blog.csdn.net/doublefi123/article/details/47276739
```

Utilisation du script

Le script est utilisé comme suit :

```
Utilisation : dffmpeg.sh [fichier_d'entrée] [paramètre_de_sortie_ffmpeg]
```

Exemples d'utilisation:

```
dffmpeg.sh test.mp4
dffmpeg.sh test.mp4 -c:v mpeg4
dffmpeg.sh test.mp4 -c:v mpeg4 -b:v 1M
```

Problèmes connus (à résoudre)

Un bogue se produit au cours de ce processus : ffmpeg est toujours en cours d'exécution mais a exécuté l'instruction touch suivante. Cela entraînera des erreurs dans l'assemblage final de la vidéo. La solution temporaire consiste à modifier la configuration du script de contrôle itemssync*wait*time. La cause du bogue est inconnue, mais ce n'est pas à cause du problème d'utilisation de NFS, car le nœud de stockage n'utilise pas NFS. Ce problème reste à résoudre.

```
# ssh ${compute_node[$i]} -p ${compute_node_ssh_port[$i]} "ffmpeg -i
$upload_path/$filename -ss $part_start -to $part_end
$ffmpeg_output_parameter $tmp_path/${filename}_$i. mp4 - erreur loglevel ;
touch $tmp_path/${filename}_$i.txt"
```

```
ssh 10.1.1.172 -p 22 ffmpeg -i
/srv/distributed_ffmpeg_transcoding_shared_files/upload/1530104749 -ss 0 -to
8.76 -c:v mpeg4 -b:v 1M
/srv/distributed_ffmpeg_transcoding_shared_files/tmp/1530104749_shared_0.mp/
mp/153044749_0. 1530104749_0.txt
```

Il y a un code commenté `for i in {0..${#compute_node[*]}}` dans le script, mais il ne peut pas être utilisé de cette façon, alors la boucle `for ((i=0; i<$node_number; i++))` a été codé à la place.

Le code pour vérifier constamment si la tâche est terminée n'est pas très propre.

1)
abréviation de « basic calculator » (« calculatrice basique »), est un interpréteur de commandes Unix qui permet d'effectuer des calculs en arithmétique multiprécision. bc peut interpréter un script ou être invoqué en ligne de commande, de façon interactive