Guide de codage vidéo H.264

Table of Contents

L'objectif de cet article est de donner des indications sur les différents modes de la création d'une vidéo de haute qualité à l'aide du codeur x264.

Deux modes de contrôle du débit généralement recommandés pour un usage général: **facteur de débit constant (CRF)** ou **ABR à deux passes**. Le contrôle du débit décide combien de bits seront utilisés pour chaque image. Cela déterminera la taille du fichier et également la distribution de la qualité.

Facteur de taux constant (CRF)

Ce mode permet de conserver la meilleure qualité sans se soucier de la taille du fichier.

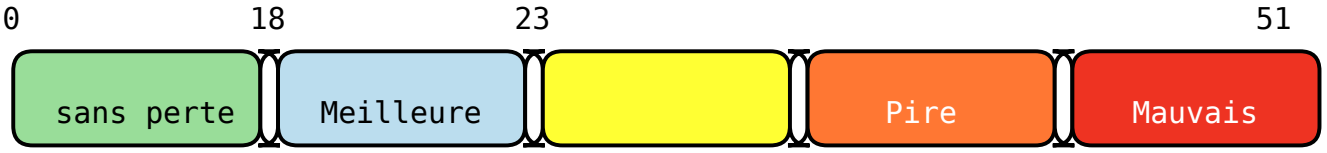
Cette méthode permet au codeur d’atteindre une certaine qualité de sortie pour l’ensemble du fichier

lorsque la taille du fichier de sortie est moins importante. Ceci fournit une efficacité de compression maximale avec un seul passage. En ajustant le quantificateur pour chaque image, il obtient le débit dont il a besoin pour conserver le niveau de qualité demandé. L'inconvénient est que l'on ne peut pas lui indiquer une taille de fichier spécifique ou de ne pas dépasser une taille ou un débit spécifique, ce qui signifie que cette méthode n'est pas recommandée pour l'encodage de vidéos en streaming.

Choisir une valeur de CRF

La plage de l'échelle CRF va de 0 à 51, 0 étant sans perte, 23 par défaut et 51 de qualité médiocre. Une valeur inférieure conduit généralement à une qualité supérieure et une plage subjectivement saine est comprise entre 17 et 28. CRF 17 ou 18 peuvent être considérés comme étant visuellement sans perte ou presque; il devrait ressembler ou presque à l'entrée, mais il n'est pas techniquement sans perte.

La plage est exponentielle. Par conséquent, si on augmente la valeur CRF de +6, on obtient environ la moitié du débit binaire/de la taille du fichier, tandis que -6 entraîne environ le double du débit.



Choisir la valeur CRF la plus élevée tout en offrant une qualité acceptable. Si la sortie semble bonne, ope peut essayer une valeur plus élevée. Si cela semble mauvais, il faudra choisir une valeur inférieure.

L'échelle de quantification **CRF 0–51** mentionnée sur cette page s'applique uniquement à 8 bits x264. Lorsqu'elle est compilée avec le support 10 bits, l'échelle de quantification de x264 est comprise entre 0 et 63. On peut voir quelle méthode est utilisée en se reportant à la sortie de la console ffmpeg lors de l’encodage (yuv420p ou similaire pour 8 bits et yuv420p10le ou similaire pour 10 bits). Le 8 bits est plus courant chez les distributeurs.

Choisir la vitesse d'encodage, une mise au point et un profil

Vitesse d'encodage

Un préréglage est un ensemble d’options qui fourniront une certaine vitesse de codage par rapport à la compression. Un préréglage plus lent fournira une meilleure compression (la compression est la qualité par taille de fichier). Cela signifie que, par exemple, si on cible une certaine taille de fichier ou un débit binaire constant, on obtiendra une meilleure qualité avec un préréglage plus lent. De même, pour un codage de qualité constante, on doit choisir un préréglage plus lent pour sauvegarder

simplement le débit.

Les préréglages disponibles par ordre décroissant de vitesse sont les suivants:

- **ultrafast**: ultra-rapide
- **superfast**: super rapide
- **veryfast**: très vite
- **faster**: plus rapide
- **fast**: rapide
- **medium**: préréglage par défaut
- **slow**: lent
- **slower**: plus lent
- **veryslow**: très lent
- **placebo**: ignorer ceci car ce n'est pas utile

Pour lister tous les préréglages et mélodies internes possibles:

```
ffmpeg -hide_banner -f lavfi -i nullsrc -c:v libx264 -preset help -f mp4 -
```

Les utilisateurs Windows peuvent avoir besoin d'utiliser NUL au lieu de - comme sortie.

Si le binaire x264 est installé, on peut également voir les paramètres exacts appliqués par ces préréglages en exécutant `x264 --fullhelp.`

Mise au point

On peut éventuellement utiliser `-tune` pour modifier les paramètres de mise au point en fonction des spécificités de l'entrée. Les réglages actuels incluent:

- **film**: utilisation pour un contenu de film de haute qualité; abaisse le déblocage
- **animation**: bon pour les dessins animés; utilise un déblocage plus élevé et plus de cadres de référence
- **grain**: préserve la structure du grain dans un vieux film granuleux
- **stillimage**: bon pour le contenu semblable à un diaporama
- **fastdecode**: permet un décodage plus rapide en désactivant certains filtres
- **zerolatency**: bon pour l'encodage rapide et le streaming à faible latence
- **psnr**: ignore ceci car il n'est utilisé que pour le développement de codecs
- **ssim**: ignore ceci car il n'est utilisé que pour le développement de codecs

Par exemple, si l'entrée est une animation, utiliser le réglage **animation** ou, si pour conserver le grain d'un film, utiliser le réglage **grain**. On peut omettre l'option `-tune` lorsqu'on ne sait pas quoi utiliser ou l'entrée ne correspond à aucun accord. On peut voir une liste des réglages actuels avec `-tune help` et les paramètres qu'ils appliquent avec `x264 --fullhelp`.

Profil

Un autre paramètre facultatif est `-profile:v` qui limitera la sortie à un profil H.264 spécifique. Ignorer ceci à moins que l'appareil cible ne prenne en charge qu'un certain profil (voir Compatibilité). Les profils actuels incluent: **baseline**, **main**, **high**, **high10**, **high422**, **high444**. L'utilisation de `-profile:v` est incompatible avec le codage sans perte.

Exemple de codage CRF

Cette commande code une vidéo de bonne qualité, en utilisant un préréglage plus lent pour obtenir une meilleure compression:

```
ffmpeg -i input.avi -c: v libx264 -preset slow -crf 22 -c: une copie
output.mkv
```

dans cet exemple, le flux audio du fichier d'entrée est simplement un flux copié sur la sortie et non ré-encodé.

Encodage ABR à deux passes

Cette méthode permet d'obtenir une taille de fichier de sortie spécifique, si la qualité de la sortie image par image est moins importante.

Calcul du débit

Pour une vidéo dure 10 minutes (600 secondes) Si on souhaite une sortie de 200 Mio `débit = taille du fichier / durée`:

```
(200 MiB * 8192 [convertit les MiB en kBit])/600 secondes=~2730 kBit/s
2730 - 128 kBit/s (débit binaire audio souhaité)=2602 kBit/s débit binaire
```

On peut également renoncer au calcul du débit lorsqu'on connait déjà le débit final (moyen) dont on a besoin.

Exemple à deux passages

Pour deux passes, il faut exécuter ffmpeg deux fois, avec presque les mêmes paramètres, à

l'exception de:

- Dans les passages 1 et 2, utiliser les options `-pass 1` et `-pass 2`, respectivement.
- Dans l'étape 1, spécifier la sortie dans un descripteur de fichier null, pas un fichier réel. (Cela générera un fichier journal dont ffmpeg a besoin pour la deuxième passe.)
- Dans l'étape 1, spécifier un format de sortie (avec -f) qui correspond au format de sortie que l'on utilisera dans l'étape 2.
- Dans le premier passage, on peut laisser la sortie audio en spécifiant -an.

Par exemple:

```
ffmpeg -y -i entrée -c:v libx264 -b:v 2600k -pass 1 -an -f mp4 /dev/null &&
\
ffmpeg -i entrée -c:v libx264 -b:v 2600k -pass 2 -c:aac -b:a 128k output.mp4
```

Les utilisateurs Windows doivent utiliser NUL au lieu de /dev/null et ^ au lieu de .

Comme avec CRF, choisir le paramètre le plus lent que possible et éventuellement appliquer un paramètre `-tune` et `profil:v`.

Informations complémentaires et astuces

H.264 sans perte

Utiliser `-crf 0` pour créer une vidéo sans perte. Deux préréglages utiles pour cela sont **ultrafast** ou **veryslow** car une vitesse de codage rapide ou une compression optimale sont généralement les facteurs les plus importants.

Exemple d'encodage ultrafast:

```
ffmpeg -i entrée -c:v libx264 -preset ultrafast -crf 0 sortie.mkv
```

Exemple d'encodage veryslow:

```
ffmpeg -i entrée -c:v libx264 -preset veryslow -crf 0 sortie.mkv
```

Produira une meilleure compression

Les fichiers de sortie sans perte seront probablement énormes, que la plupart des

lecteurs non basés sur FFmpeg ne pourront pas décoder sans perte. Par conséquent, si la compatibilité ou la taille du fichier pose problème, il ne faut pas utiliser cette méthode. Pour une sortie pratiquement “sans perte visuelle” (mais pas techniquement), il faut utiliser une valeur `-crf` d'environ **17** ou **18** (faire des essais pour voir quelle valeur est acceptable). Cela produira un résultat probablement impossible à distinguer de la source et ne produira pas un fichier énorme, peut-être incompatible, comme le véritable mode sans perte.

Remplacement des paramètres prédéfinis par défaut

`-preset` choisit les meilleurs paramètres possibles, mais on peut les écraser avec l'option `x264-params` ou en utilisant les options privées `libx264` (voir ffmpeg -h encoder = libx264). Exemple:

```
ffmpeg -i input -c:v libx264 -preset slow -crf 22 -x264-params
keyint=123:min-keyint=20 -c:a copy output.mkv
```

Les préréglages ont été créés par les développeurs x264 et modifier les valeurs pour obtenir un meilleur rendu est généralement une perte de temps. Ne pas utiliser l'option `x264opts`, elle sera éventuellement supprimée. Utiliser plutôt x264-params.

CBR (débit binaire constant)

Il n'y a pas de mode CBR natif ou réel, mais on peut “simuler” un réglage de débit constant en ajustant les paramètres d'un encodage à débit moyen à un passage:

```
ffmpeg -i input.mp4 -c:v libx264 -x264-params "nal-hrd = cbr" -b:v 1M -
minrate 1M -maxrate 1M -bufsize 2M output.ts
```

Dans l'exemple ci-dessus, `-bufsize` est le “tampon de contrôle du débit”. Il appliquera donc la “moyenne” demandée (1 Mbits/s dans ce cas) sur chaque tranche de 2 Mbits de vidéo. Ici, on suppose que le récepteur/lecteur va mettre en mémoire tampon beaucoup de données, ce qui signifie qu’une fluctuation dans cette plage est acceptable.

Les codages CBR sont généralement inefficaces si la vidéo est facile à coder (par exemple, des images vides ou noires).

Encodage contraint (VBV/débit maximal)

Utiliser ce mode pour contraindre le débit maximal utilisé ou conserver le débit du flux dans certaines

limites. Ceci est particulièrement utile pour la diffusion en ligne, où le client s'attend à un débit moyen, mais qu'on veut que l'encodeur règle le débit par image.

On peut utiliser `-crf` ou `-b:v` avec un débit binaire maximal en spécifiant à la fois `-maxrate` et `-bufsize`:

```
ffmpeg -i entrée -c:v libx264 -crf 23 -maxrate 1M -bufsize 2M output.mp4
```

Cela va effectivement “cibler” **-crf 23**, mais si la sortie devait dépasser 1 Mbits/s, le codeur augmentera le CRF pour empêcher les pointes de débit. Cependant, libx264 ne contrôle pas strictement le débit maximal tel que spécifié (le débit maximal peut être bien supérieur à 1 M pour le fichier ci-dessus). Pour atteindre un débit binaire maximal parfait, utiliser deux passes:

```
ffmpeg -i entrée -c:v libx264 -b:v 1M -maxrate 1M -bufsize 2M -pass 1 -f mp4
/dev/null
ffmpeg -i entrée -c:v libx264 -b:v 1M -maxrate 1M -bufsize 2M -pass 2
sortie.mp4
```

Faible latence

x264 offre une option de réglage de zérolatence pour la diffusion en continu à faible latence.

Compatibilité tous dispositifs

Lorsqu'on veut que les vidéos soient parfaitement compatibles avec les appareils anciens (anciens téléphones Android, par exemple):

```
-profile:v baseline -level 3.0
```

Cela désactive certaines fonctionnalités avancées mais offre une meilleure compatibilité. En règle générale, on a pas besoin de ce paramètre (et éviter par conséquent d'utiliser `-profile:v` et `-level`), mais si on l'utilise, il peut augmenter le débit par rapport à ce qui est nécessaire pour obtenir la même qualité dans les profils plus élevés.

Compatibilité iOS

Options	niveau	périphériques	profil
Baseline	3.0	Tous les périphériques	-profile: v baseline -level 3.0
Baseline	3.1	Phone 3G et versions ultérieures, iPod touch 2ème génération et versions ultérieures	-profil:v baseline -level 3.1
Main	3.1	Ipad (toutes versions), Apple TV 2 et ultérieur, iPhone 4 et ultérieur	-profile:v main -level 3.1
Main	4.0	Apple TV 3 et versions ultérieures, iPad 2 et versions ultérieures, iPhone 4s et versions ultérieures	-profile:v main -level 4.0

Options	niveau	périphériques	profil
High	4.0	Apple TV 3 et versions ultérieures, iPad 2 et versions ultérieures, iPhone 4s et versions ultérieures	-profile:v high -level 4.0
High	4.1	iPad 2 et versions ultérieures, iPhone 4s et versions ultérieures, iPhone 5c et versions ultérieures	-profile:v high -level 4.1
High	4.2	iPad Air et versions ultérieures, iPhone 5s et versions ultérieures	-profile:v high -level 4.2

Compatibilité Quick Time

Bien que d'autres formats de pixels puissent être pris en charge, l'espace colorimétrique plan YUV avec sous-échantillonnage de la chrominance **4:2:0** est un format de pixel sécurisé pour les vidéos H.264. utiliser `-vf format=yuv420p` ou `-pix_fmt yuv420p`.

Pré-tester les paramètres

Encoder une section aléatoire au lieu de la vidéo entière avec les options `-ss` et `-t/-to` pour obtenir rapidement une idée générale de ce à quoi la sortie ressemblera.

- **-ss**: décalage par rapport au début. La valeur peut être exprimée en secondes ou au format **HH:MM:SS**.
- **-t**: durée. La valeur peut être exprimée en secondes ou au format **HH:MM:SS**.
- **-to**: arrête d'écrire la sortie à la position spécifiée. La valeur peut être exprimée en secondes ou au format **HH:MM:SS**.

Démarrage rapide pour la vidéo Web

On peut ajouter `-movflags + faststart` comme option de sortie si les vidéos vont être visionnées dans un navigateur. Cela déplacera certaines informations au début du fichier et permettra à la vidéo de commencer à être lue avant son téléchargement complet par le téléspectateur. Ce n'est pas nécessaire lorsqu'on veut utiliser un service vidéo tel que YouTube. YouTube recommande l'utilisation de `faststart` afin qu'ils puissent commencer le réencodage avant la fin du téléchargement.