Lecture de DCP pour le cinéma numérique avec le lecteur multimédia VLC et libav/ffmpeg

N. Bertrand[1,2], S.-M. Prodea[2], J.-D. Durou[2], C. Griwodz[2,3] et V. Charvillat[2]

[1]ISF - Indépendants, Solidaires et Fédérés
nicolas.bertrand@isf.cc

[2]IRIT - Institut de Recherche en Informatique de Toulouse
{durou, charvillat}@irit.fr

[3]Simula Research Laboratory, Oslo, Norway
griff@simula.no

Résumé

Les salles de cinéma sont passées à l'ère numérique. Pour la distribution des copies numériques de films, la Digital Cinema Initiative (DCI) a choisi d'encoder les films au format Digital Cinema Package (DCP). Notre activité est menée en collaboration avec les cinémas Utopia, qui forment un réseau de salles indépendantes. La principale exigence d'Utopia est de fournir des logiciels libres et gratuits pour le cinéma numérique. Le format DCP utilise la compression JPEG2000 pour la vidéo, en raison de son taux de compression élevé pour les grandes images. Réaliser une implémentation efficace de codage et de décodage de ce format est complexe. Néanmoins, nous proposons une implémentation améliorée du décodage pour la projection dans les salles obscures. Les équipements actuellement déployés dans les salles sont chers à l'achat et ont un coût de maintenance élevé, ce qui empêche les petites salles indépendantes de s'équiper. Notre but est de proposer une solution logicielle adaptée à la projection des DCP. Cet article présente une solution qui réalise le décodage en temps réel et la projection, en s'appuyant sur des outils multimédia libres et standards comme le lecteur VLC et les bibliothèques libav/ffmpeg. Nous présentons les améliorations implémentées dans VLC pour supporter les DCP, qui incluent la lecture des fichiers et la synchronisation entre audio et vidéo. Nous détaillons ensuite la réalisation du décodeur JPEG2000 dans libav/ffmpeg. Pour finir, nous évaluons les performances de lecture atteintes.

Mots clefs

Cinéma numérique, JPEG2000, décodeur, logiciel libre, VLC, libav, ffmpeg.

1 Introduction

Le cinéma est passé des copies 35 mm à l'ère numérique. La *Digital Cinema System Specification* [1], fournie par la DCI (*Digital Cinema Initiative*), est maintenant un standard international. Cette spécification décrit comment créer, distribuer, et projeter une copie numérique, appelée *Digital Cinema Package* (DCP). Elle requiert l'utilisation du JPEG2000 comme codec intra-trame pour la vidéo, du WAC/PCM pour l'audio et du XML pour les sous-titres. Les formats des images sont 2K ou 4K, avec 3 composantes couleur codées chacune sur 12 bits, séquencées à une fréquence d'au moins 24 images par seconde (fps). La taille du DCP varie entre 80 et 200 GB, selon la durée du film et son taux de compression. Notre activité de recherche est menée en collaboration avec les cinémas Utopia [1], qui constituent un réseau français de 5 cinémas indépendants. D'autres cinémas participent à cette activité au travers de l'association ISF [2] (Indépendants, Solidaires et Fédérés). Les cinémas qui ont démarré cette activité ont besoin de comprendre les implications liées à la transition vers le numérique. Ils s'inquiètent principalement de la dépendance à des entreprises technologiques, ce qui explique leur besoin de logiciels libres et ouverts pour le cinéma numérique.

Le format JPEG2000 [2] a été choisi par la DCI pour la compression vidéo. La compression est efficace mais l'implémentation d'un codeur/décodeur rapide est complexe.

Les équipements commerciaux déployés dans les cinémas utilisent une décompression matérielle (VLSI) pour atteindre la fréquence d'images requise. Ce type d'équipement est trop coûteux pour les petits exploitants indépendants. Notre but est de proposer un logiciel libre capable de lire les DCP sur des ordinateurs standards. Il existe des solutions logicielles, comme Kakadu [3] et EasyDCP [4], mais aucune n'est libre. Ces deux solutions sont dédiées à la lecture des DCP en temps réel, mais nous voulons fournir une solution libre, ouverte et multi-plateforme capable de lire aussi bien des DCP que d'autres formats vidéo, afin de

1. http ://www.cinemas-utopia.org
2. http ://www.isf.cc

lire tous les types de formats à partir d'un système unique. Nous ne voulons pas proposer un nouveau système de projection, mais en proposer un plus simple, qui est décrit dans [5]. Il consiste à remplacer le bloc média du DCI par un PC exécutant un lecteur multimédia.

Le choix du logiciel multimédia VLC (*VideoLAN Client*) découle logiquement de nos contraintes. VLC ne supportant pas encore le format DCP, nous avons créé un module spécifique pour le traiter. L'implémentation du décodeur JPEG2000 est codée dans libav/ffmpeg, qui consiste en un ensemble de bibliothèques multimédia utilisées par VLC.

Le point clef pour la lecture des DCP est d'atteindre une fréquence d'au moins 24 fps, qui garantisse la synchronisation entre l'affichage, l'audio et les sous-titres. Il est nécessaire de paralléliser le calcul pour surmonter ce problème du point de vue logiciel. Il existe actuellement trois approches dominantes : l'utilisation du *multi-threading* et de jeux d'instructions processeur spécifiques comme les *Streaming SIMD Extensions* (SSE), l'usage du *General Purpose computing on Graphical Processing Units* (GPGPU), et la combinaison des deux. Par exemple, [6] utilise du *multi-threading* et du SSE. Le logiciel breveté EasyDCP [7], qui repose sur du GPGPU et sur une méthode de type GOP (*Group Of Pictures*) fondée sur la similarité entre *codes-blocs* pour éviter la latence. Dans [8] les auteurs combinent du *multi-threading* avec du GPGPU. [9] et [10] développent des propositions pour paralléliser EBCOT avec du GPGPU. Taubman [6] montre qu'une décompression de 24 fps peut être atteinte avec du *multi-threading* et des instructions SSE. Éviter le GPGPU réduit la complexité du décodeur et la dépendance au matériel. La dépendance au matériel NVIDIA, liée à l'utilisation de CUDA, est une limitation des méthodes de [9] et [10]. Bien que OpenCL [8] soit prévu pour être indépendant du matériel, des réglages spécifiques sont nécessaires pour obtenir des performances élevées.

Après la décompression vidéo, pour garantir des couleurs correctes à l'affichage, nous avons implémenté un filtre GLSL de conversion du format CIE 1931 XYZ [3] vers le format colorimétrique requis, qui est généralement sRGB.

Dans le paragraphe 2, nous présentons le logiciel VLC et son architecture. Dans le paragraphe 3, le design du module DCP est décrit. L'implémentation du décodeur JPEG2000 est détaillée dans le paragraphe 4. Enfin, la validation et les performances de notre solution sont détaillées dans le paragraphe 5.

2 VLC

VLC est un lecteur de média libre. Le projet a été initié au milieu des années 90 par des étudiants de l'École Centrale de Paris. À l'heure actuelle, le projet n'est plus directement lié à cette école, même si les principaux développeurs en sont issus. Il est géré par une association (*VideoLAN*) à but non lucratif et n'ayant aucun employé. VLC est le logiciel français le plus répandu, téléchargé plus de 1,4 milliards de fois. Il est multi-plateforme, adapté par exemple à Windows, Mac OS X, GNU/Linux, BSD, Android et iOS.

2.1 Architecture de VLC

VLC est essentiellement un *framework* multimédia, où l'on peut dynamiquement charger des modules selon les entrées (fichiers, flux réseaux) et les sorties (audio ou vidéo, sur un écran ou sur le réseau). Le cœur du *framework* gère les opérations de bas niveau comme le *threading*, le *timing* et la synchronisation. Il a aussi pour rôle de transférer (par pipeline) les flux média de l'entrée vers la sortie. Chaque module est exécuté dans un *thread* différent. Le cœur connecte les modules entre eux, et permet la transmission de données entre les modules. Les modules (plus de 4000) sont utilisés pour traiter les média, ce qui inclut l'accès aux fichiers, le démultiplexage des conteneurs, le décodage et les sorties. Les modules sont choisis durant l'exécution suivant le type de machine et le type de média. Un exemple de chargement de modules dans le cas d'un DCP est présenté sur la figure 1.

VLC dépend de nombreuses bibliothèques libres (45 pour une distribution GNU/Linux standard), incluant libav [4] /ffmpeg [5]. Les modules VLC encapsulent ces bibliothèques externes. Libav/ffmpeg met à disposition la bibliothèque libavcodec qui fournit des codeurs et des décodeurs pour les flux multimédia.

2.2 La problématique du DCP

Jusqu'à présent, VLC ne gérait pas les DCP. Nous avons comblé ce manque en ajoutant des modules cohérents avec l'architecture de VLC.

Le décodeur JPEG2000 décrit dans cet article est codé dans libavcodec (l'implémentation sera décrite plus en détail dans le paragraphe 4). Le module OpenGL a été légèrement modifié pour prendre en compte la conversion entre les espaces colorimétriques CIE XYZ et sRGB.

3 Le module DCP dans VLC

Un DCP est un ensemble de fichiers inclus dans un répertoire. Les métadonnées d'un DCP sont stockées dans des fichiers XML, tandis que les fichiers MXF contiennent les média (vidéo, audio et sous-titres). Il y a au moins un fichier MXF par média, mais plusieurs fichiers MXF audio et sous-titres peuvent être stockés dans le répertoire, afin de gérer plusieurs langues. La figure 2 représente un exemple de répertoire DCP.

Le premier problème lors de l'implémentation d'un module DCP dans VLC est le format lui-même. Un DCP est un conteneur (le répertoire) incluant tour à tour plusieurs autres conteneurs (les fichiers MXF). Dans VLC, il n'y a pas l'équivalent du conteneur DCP.

3. CIE 1931 XYZ est un espace colorimétrique de référence pour les systèmes de gestion de couleurs. Cet espace colorimétrique (requis par le DCI) est utilisé pour le codage des couleurs dans un DCP.

4. Libav – http ://libav.org

5. Ffmpeg – http ://ffmpeg.org

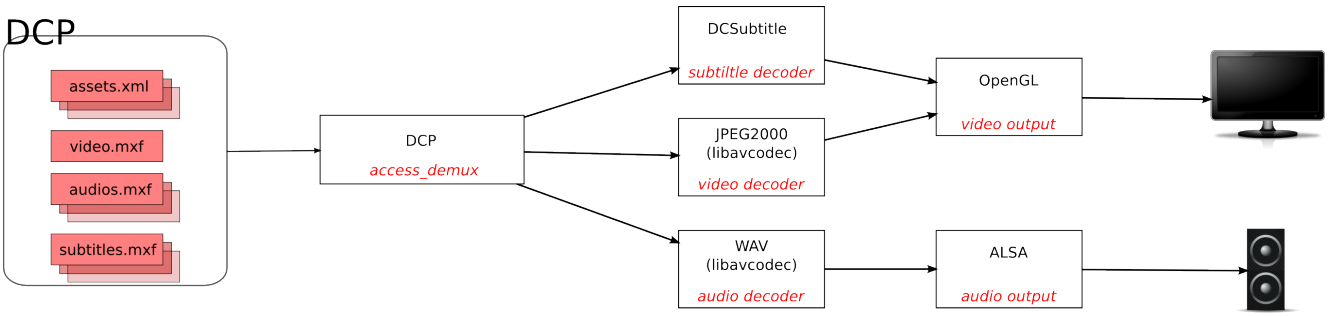

Figure 1 – *Lecture de DCP avec VLC et libav/ffmpeg. Chaque boîte représente un module VLC (son rôle est indiqué en rouge). Les flèches représentent le flux des média entre les différents modules.*

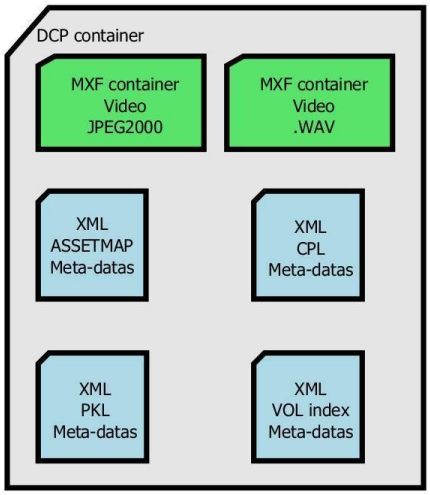

Figure 2 – *Conteneur DCP.*

Il existe deux grandes familles de conteneurs. Pour la première famille, qui comprend les formats de type MKV ou AVI, le conteneur est composé d'un seul fichier incluant tous les en-têtes et tous les média. Pour ce type de conteneur, VLC utilise des modules de type *access* pour accéder physiquement au fichier, et des modules *demux* pour extraire la vidéo et l'audio, avant de les envoyer aux décodeurs respectifs. La seconde famille, qui comprend les formats de type DVD ou Blu-ray, représente physiquement un répertoire contenant des fichiers d'en-têtes, de menus, et de média. Dans le cas d'un DVD, par exemple, l'audio et la vidéo sont contenus dans le même conteneur VOB. Le film complet est découpé en plusieurs fichiers VOB, pour des raisons de taille. Dans ce cas, VLC utilise des modules *access-demux*, et combine l'accès physique au démultiplexage.

Pour les DCP, nous avons décidé d'implémenter un module *access-demux*. Une autre approche consistant à écrire un module de type *access* et à utiliser libav/ffmpeg pour gérer le MXF au niveau *demux* n'est pas possible, car VLC attend un seul flux de données physique comme interface entre les modules *access* et *demux*. Même si nous avions surmonté cette contrainte, nous aurions dû lancer un module *demux* par fichier MXF, ce qui aurait augmenté la complexité de la synchronisation entre média. Au vu de ces considérations, nous avons jugé préférable d'utiliser un module de type *access-demux*. Les en-têtes des DCP sont écrits en XML. Nous utilisons un module VLC dédié pour analyser les fichiers XML et identifier les média. L'accès aux fichiers média MXF est effectué via une bibliothèque externe *asdcplib*. Nous avons choisi cette bibliothèque car elle permet de gérer les conteneurs MXF pour les DCP. Elle est diffusée sous licence libre, et est également très répandue dans le domaine du cinéma numérique. Une fois le contenu média extrait, des VLC *Elementary Streams (ES)* sont créés et envoyés aux décodeurs. Les ES, à raison d'un par média, forment les interfaces entre l'*access-demux* et les modules de décodage.

L'architecture de VLC est modulaire et extrêmement *multi-threadée*, et les modules *access-demux*, *decoder*, *video-output* et *audio-output* sont exécutés dans des *threads* séparés. Par conséquent, le démultiplexage, le décodage et le rendu audio et vidéo sont exécutés séparément. La synchronisation de l'audio, de la vidéo et des sous-titres sont réalisés par un mécanisme d'horodatage appelé *Presentation Time Stamps* (PTS). L'horodatage est intégré aux ES, au travers du module DCP. Les modules de sortie utilisent les PTS pour synchroniser l'affichage et le son.

4 Décodage JPEG2000 dans libav/ffmpeg

Les deux projets concurrents libav et ffmpeg proposent la même API. Ils se sont séparés en 2012 en raison de divergences entre développeurs. Ils fournissent des outils et des bibliothèques pour démultiplexer, coder et décoder un grand nombre de formats. VLC peut utiliser ces deux projets, la sélection se faisant selon l'environnement d'exécution ou au choix de l'utilisateur. Pour permettre une utilisation la plus large possible, nous maintenons le décodeur JPEG2000 dans les deux projets. C'est pour cela que nous utilisons la notation libav/ffmpeg tout au long de cet article. Comme VLC, libav et ffmpeg sont des projets libres

et multi-plateforme.
Libav/ffmpeg fournit 8 bibliothèques pour gérer les conteneurs, les codecs et les filtres audio/vidéo. Nous travaillons dans libavcodec, qui contient tous les codecs audio et vidéo de libav/ffmpeg.
Notre décodeur JPEG2000 n'est pas le seul disponible dans libavcodec. Libav/ffmpeg peut aussi utiliser la bibliothèque OpenJPEG pour le décodage de fichiers JPEG2000 [5]. OpenJPEG est une bibliothèque libre qui accepte tout type de profil JPEG2000. Elle est surtout destinée au codage/décodage d'images. En revanche, notre décodeur est destiné au format JPEG2000 pour la vidéo, et plus spécifiquement pour le cinéma numérique. Grâce à cette spécialisation, on peut utiliser des structures de données plus petites et un code plus simple, en évitant les structures internes complexes de OpenJPEG, qui sont nécessaires pour gérer toutes les options du JPEG2000.
Un autre avantage important de libav/ffmpeg est la partie liée aux tests. Le projet fournit des outils natifs pour la couverture de code, les tests de non régression et la mesure de performances. La relecture de code est aussi un point important pour la communauté libav/ffmpeg, qui permet d'atteindre une bonne qualité de code.
Comme le décodeur est intra-trame, nous avons décidé de *multi-threader* le décodeur au niveau de la trame, en utilisant les mécanismes fournis par les primitives de libavcodec.
De plus, nous avons augmenté les performances de décodage, en ne faisant qu'une seule fois l'analyse d'une partie des en-têtes des trames JPEG2000. Les contraintes DCI sur le JPEG2000 fixent certains paramètres comme par exemple la taille des code-blocs, qui sera donc toujours la même quelque soit la trame.
En effet, les profils JPEG2000 pour le cinéma numérique déterminent de nombreuses options (par exemple la taille des codes-blocs).
L'évaluation du décodeur et la comparaison avec OpenJPEG sont faites dans le prochain paragraphe.

5 Validation expérimentale

5.1 Évaluation du décodeur JPEG2000 dans libav/ffmpeg

Les tests sont faits sur deux machines. La machine 1 est un portable Dell Inspiron équipé d'un Intel Core i7-2820QM cadencé à 2,30 GHz. Le processeur a 4 cœurs pour un total de 8 SMT *threads* (*hyper-threads* dans la terminologie Intel). La machine 2 est un Dual Xeon E5-2620 cadencé à 2,0 GHz. Chaque *chip* a 6 cœurs physiques, avec au total 24 SMT *threads*. Tous les tests sont exécutés avec l'*hyper-threading* actif. Les tests sont faits en fonction du nombre de SMT *threads* et du nombre de niveaux de résolution (défini par le nombre de transformations en ondelettes). Le DCP testé est la bande-annonce du film *Moonrise Kingdom* durant les 10 premières secondes (240 trames). Le film a été encodé avec un taux de compression de 128 Mbits/s. Les images de *Moonrise Kingdom* ont une taille de 1998×1080 pixels, et possèdent 5 niveaux de résolution. Les nombres de niveaux de résolution testés sont 5 et 4.

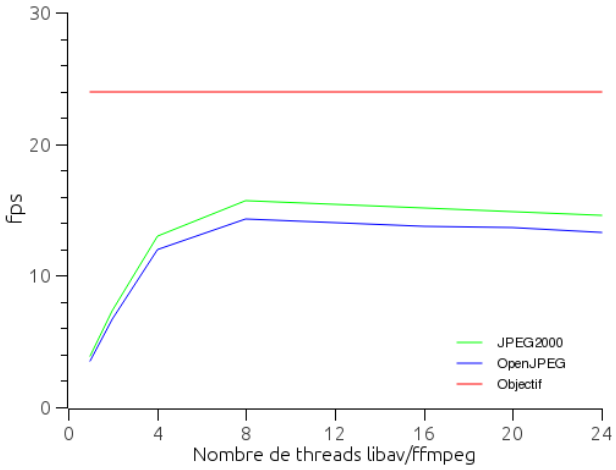

Figure 3 – *Performances du décodeur JPEG2000 sur la machine 1 en pleine résolution ($N = 5$).*

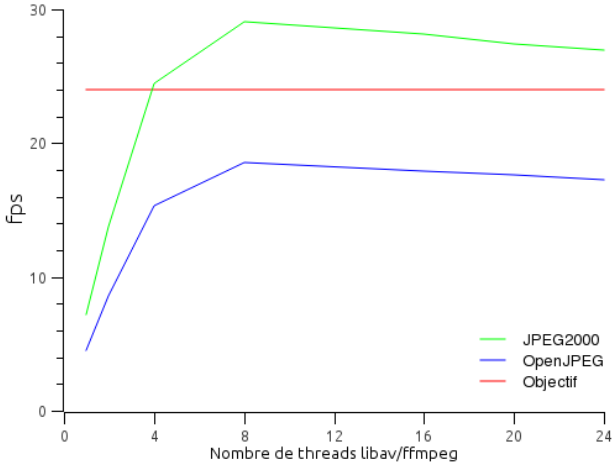

Figure 4 – *Performances du décodeur JPEG2000 sur la machine 1 en supprimant le niveau de résolution de plus haute fréquence ($N - 1 = 4$).*

Les figures 3 et 4 montrent la comparaison de notre décodeur JPEG2000 avec OpenJPEG sur la machine 1. Comme le processeur est limité à 8 SMT *threads*, les performances de décodage avec les deux décodeurs sont limitées à 8 libav/ffmpeg *threads*. On note une baisse des performances si l'on utilise un nombre de libav/ffmpeg *threads* supérieur au nombre de SMT *threads* du processeur. Avec $N = 5$ niveaux de résolution (figure 3), ni OpenJPEG ni notre décodeur n'atteignent l'objectif de 24 fps (courbe rouge), mais notre décodeur est au moins aussi rapide que OpenJPEG. Avec $N - 1 = 4$ niveaux de résolution (figure 4), notre

décodeur atteint l'objectif à partir de 4 *threads*.

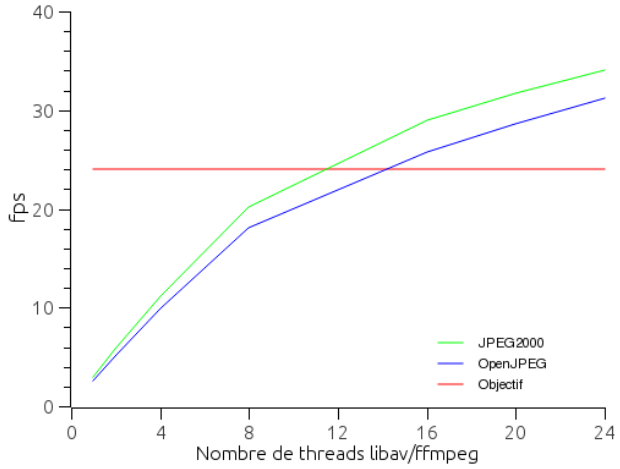

Figure 5 – *Performances du décodeur JPEG2000 sur la machine 2 en pleine résolution ($N = 5$).*

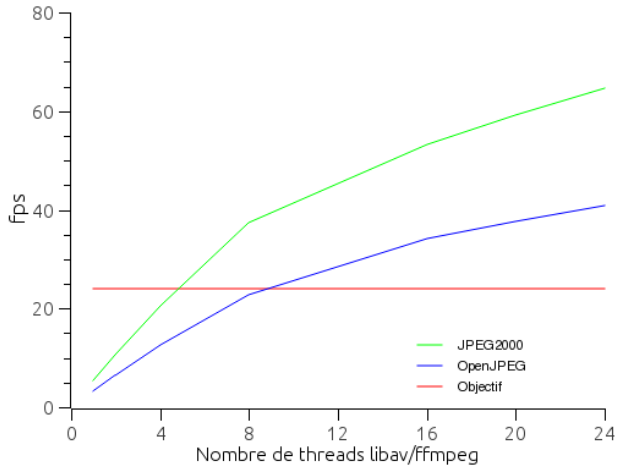

Figure 6 – *Performances du décodeur JPEG2000 sur la machine 2 en supprimant le niveau de résolution de plus haute fréquence ($N - 1 = 4$).*

Pour la machine 2, les résultats sont montés sur les figures 5 et 6. Les deux décodeurs atteignent l'objectif de 24 fps. Avec un décodage en pleine résolution ($N = 5$), notre décodeur atteint l'objectif à partir 12 libav/ffmpeg *threads* et OpenJPEG à partir de 16 libav/ffmpeg *threads*. Dans l'ensemble des résultats présentés, le gain en fps de notre décodeur est élevé entre les niveaux de résolution N et $N - 1$. Il y a deux raisons à cela. L'image à décoder est plus petite (4 fois plus petite qu'au niveau N), donc il y a moins de *code-blocs* à décoder. Nous ne pouvons néanmoins pas espérer un gain proche de 4 car les *code-blocs* non décodés sont ceux des hautes fréquences : EBCOT encode les *code-blocs* de hautes fréquences avec moins de bits que ceux de basses fréquences.

Film	Taux (MBits/s)	12 *threads* (fps)	24 *threads* (fps)
Spring Breakers	201	19,05	24,62
Django Unchained	151	19,58	24,64
Moonrise Kingdom	128	27,12	33,76
Jeux d'été	79	30,34	38,28

Tableau 1 – *Comparaison des fréquences (en fps) pour divers films avec des taux de compression différents. Tests effectués par notre décodeur JPEG200 sur la machine 2.*

L'autre raison est liée à l'utilisation de la mémoire et à la gestion des différents niveaux de cache des processeurs. Avec OpenJPEG, on atteint un gain de 1,5 entre les niveaux N et $N-1$, alors qu'en utilisant notre décodeur, on est plus proche de 2. La principale raison à cela est la gestion de la mémoire : nous avons des structures de données plus petites pour le flot de données de décodage, ce qui aboutit à une réduction des *cache misses* et *page faults* du processeur. Ceci a pu être mesuré grâce à l'outil de profilage *perf*[6] intégré dans le noyau Linux.

Le tableau 1 montre les valeurs de fps en fonction du taux de compression à l'encodage. Les films choisis sont des bandes-annonces projetées dans les salles de cinéma. Le décodeur est testé sur les 240 premières trames. Comme on pouvait s'y attendre, plus le taux de compression est faible, plus le décodage est rapide. Pour *Spring Breakers* et *Django Unchained*, nous obtenons des fréquences du même ordre, alors que le taux de compression est différent. La raison en est que dans *Django Unchained*, les images sont très colorées au début, alors que dans *Spring Breakers*, de nombreuses images sont noires. Les images noires (ou unies) sont les cas de décodage JPEG2000 les plus rapides.

5.2 Évaluation de la lecture d'un DCP

La lecture complète d'un DCP avec synchronisation de l'audio et de la vidéo, et avec affichage des couleurs dans l'espace colorimétrique voulu, est faite au niveau de VLC.

Il n'existe pas d'outil dans VLC pour mesurer la performance globale du décodage complet. Seuls des messages d'erreur peuvent être affichés, indiquant par exemple la perte de synchronisation entre l'audio et la vidéo, ou un décodage trop lent. Nous utilisons ces messages et l'inspection visuelle (visualisation du rendu sur écran et dans une salle de cinéma) pour évaluer le lecture d'un DCP. Les résultats de l'évaluation sont reportés dans le tableau 2.

Nous n'avons pas réussi à jouer un DCP en pleine résolution, même avec les performances de décodage présentées précédemment. Cela indique qu'il reste des améliorations à apporter à notre module VLC (mise en cache de la vidéo, et suppression de la limitation de VLC à 16 *threads* au maximum pour le décodage vidéo).

6. perf –https ://perf.wiki.kernel.org

Machine	N	$N-1$	$N-2$
1 (i7 4 cores)	Non	Non	Oui
2 (Dual Xeon)	Non	Oui	Oui

Tableau 2 – *Lecture de DCP à plusieurs niveaux de résolution : « Oui » indique que le DCP est lu sans désynchronisation.*

6 Conclusion et perspectives

Nous avons comparé les performances de notre décodeur avec celles d'OpenJPEG, qui constitue la référence et le plus rapide des décodeurs JPEG2000 libres. Notre décodeur est plus rapide, surtout quand nous ne décompressons pas le niveau de résolution de plus hautes fréquences.
La comparaison avec des logiciels non libres étant plus complexe, nous la ferons ultérieurement. Des résultats concernant les performances de Kakadu sont accessibles sur le web [7], mais une comparaison directe est impossible car la séquence d'images ayant servi aux mesures n'est pas disponible. La taille des images de la séquence de Kakadu est 20% plus petite, mais le taux de compression (244 Mbits/s) est deux fois supérieur à celui de notre séquence (*Moonrise Kingdom*). Sur un Dual Xeon, Kakadu atteint une fréquence de 24,24 fps, et de 35,08 fps avec le *Separate Speed Pack*. Sur notre Dual Xeon, la fréquence atteinte vaut 33,76 fps. Quant aux résultats de EasyDCP, ils ne sont pas accessibles.
Notre travail à venir sur la partie décodage consistera à implémenter du *multi-theading* au niveau des *code-blocs* et à aller plus loin dans la réduction de la taille des structures de données. Le but est d'avoir des fragments de code qui dépendent uniquement de données locales (c'est-à-dire sans référence à des données de plus haut niveau). Ainsi, les structures de données devraient rester au niveau des caches L1 et L2 durant l'exécution et permettre ainsi de réduire le temps d'accès aux données.
Vu que le nombre de cœurs croît dans les architectures, nous pensons pouvoir nous passer de l'usage de GPGPU pour le décodage. Cependant, dans le processus global de lecture des DCP, nous utiliserons du GPGPU pour du filtrage spécifique (changement d'espace colorimétrique et/ou correction des couleurs) après le décodage.
Au niveau de VLC, nous devons optimiser le module DCP pour gérer la synchronisation de l'audio, de la vidéo et des sous-titres en pleine résolution. Asdcplib permet de traiter des fichiers MXF cryptés, ce qui nous permettra de développer la lecture de DCP cryptés. La protection consiste a générer une clef unique par projecteur, utilisée pour crypter le film. La lecture de DCP crypté avec VLC sera également très utile pour les exploitants.
Le format DCP introduit une nouvelle organisation des conteneurs. La structure du DCP sera plus largement utilisée avec le nouveau format IMF [8] (*Inter-operable Master Format*), en cours de standardisation par le SMPTE. Le conteneur IMF utilise la même structure que celle du DCP et a pour vocation d'être un format de post-production unique, qui pourra être utilisé comme master pour tous les usages (DVD, Blu-ray, *streaming*, ...).
Nous ne sommes donc pas loin, du point de vue des performances et de la fonctionnalité, d'une solution libre complète pour la lecture des DCP, utilisable par les distributeurs de films pour prévisualiser les contenus, et par les exploitants pour la projection dans les salles obscures.

7. http ://www.kakadusoftware.com/documents/kakadu-decode-speed.pdf
8. IMF work group : http ://www.imfforum.com

Remerciements

Nous remercions les cinémas Utopia et les communautés VLC et libav/ffmpeg.

Références

[1] DCI. *Digital Cinema System Specification*. Digital Cinema Initiatives, 1.2 with errata édition, Mars 2012.

[2] ITU-T. *Information technology – JPEG 2000 image coding system – Part 1 : Core coding system*, Août 2002.

[3] UNSW. Kakadu software, Juillet 2013. http ://www.kakadusoftware.com.

[4] FraunHoffer IIS. EasyDCP software suite, Juillet 2013. http ://www.easydcp.com/.

[5] N. Bertrand, J.-D. Durou, et V. Charvillat. Vers un système de projection alternatif pour le cinéma numérique. Dans *Actes de la conférence CORESA'12*, volume 1, pages 185–190, Lille, Mai 2012.

[6] D. Taubman. Multithreaded processing paradigms for JPEG2000. Dans *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*, pages 164–169, 2012.

[7] V. Bruns, H. Sparenberg, et S. Fossel. Video decoder and methods for decoding a sequence of pictures, patent wo 2012004164 a1, Janvier 2012.

[8] R. Le, J.L. Mundy, et R.I. Bahar. High Performance Parallel JPEG2000 Streaming Decoder Using GPGPU-CPU Heterogeneous System. Dans *Application-Specific Systems, Architectures and Processors (ASAP), 2012 IEEE 23rd International Conference on*, pages 16–23, 2012.

[9] R. Le, I.R. Bahar, et J.L. Mundy. A novel parallel Tier-1 coder for JPEG2000 using GPUs. Dans *Application Specific Processors (SASP), 2011 IEEE 9th Symposium on*, pages 129–136, 2011.

[10] J. Matela, M. Šrom, et P. Holub. Low GPU Occupancy Approach to Fast Arithmetic Coding in JPEG2000. Dans Zdeněk Kotásek, Jan Bouda, Ivana Černá, Lukáš Sekanina, Tomáš Vojnar, et David Antoš, éditeurs, *Mathematical and Engineering Methods in Computer Science*, volume 7119 de *Lecture Notes in Computer Science*, pages 136–145. Springer Berlin Heidelberg, 2012.